Unit 1

Introduction [4 hrs]

* Review of sets, relation, function

1. Sets:
- collection of objects
- denoted by capital letters

eg:
$A = \{1, 2, 3\}$
$V = \{a, e, i, o, u\}$

Representation of element belongs to set
$1 \in A$
$e \in V$

Sets can be represented by
<i> listing its elements
Eg: $V = \{a, e, i, o, u\}$
<ii> by describing properties of elements
$V = \{x | x \text{ is a vowel letter}\}$
$N = \{n | n \text{ is a natural number}\}$

Types of sets:
<i> Singleton set
- has only one element
<ii> Empty set
- has no element denoted by ϕ.

CamScanner

<iii> Disjoint sets
- Two or more sets having no common elements.

<iv> Overlapping set
- having at least one common element

<v> Subset
- If every element of A is in B, then $A \subseteq B$

Additional term

<i> Cardinality
- number of elements in set
Eg: $A = \{1, 2, 3, 4\}$
$|A| = 4$

<ii> Cartesian Product
- Cartesian product of two sets A and B is ordered pair of set.
$A \times B = \{\{x, y\} \mid x \in A \text{ and } y \in B\}$

<iii> Power set
-

Eg: $A = \{1, 2\}$
$P(A) = \{\phi, 1, 2, \{1, 2\}\}$
↗
Power set

It is given by 2^n where
n = no. of elements

* Some set operations

(i) Union
- $A \cup B = \{ x : x \in A \text{ or } x \in B \}$

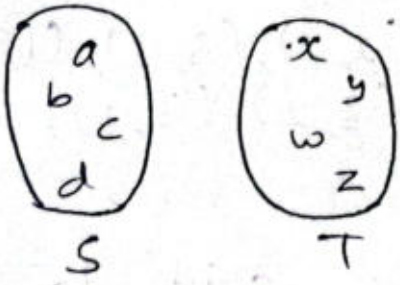
(ii) Intersection
- $A \cap B = \{ x : x \in A \text{ and } x \in B \}$

(iii) Set difference
- $A - B = \{ x : x \in A \text{ and } x \notin B \}$

(iv) Complement
- $\bar{A} = \{$ everything that is not in $A \}$

* Relation
- A Relation is an ordered pair of sets, such that to each element of domain, there is assigned one or more range.

Suppose S and T are two sets.



$R = \{ (a,x), (b,y), (c,w), (d,z) \}$

Types of Relation

(i) Reflexive relation
- A relation 'R' is reflexive, if every element of set 'A' is related to itself.

Eg:

$A = \{1, 2, 3\}$

$R = \{(1,1), (2,2), (3,3)\}$

(ii) Symmetric relation

- A relation 'R' is transitive if $(a,b) \in R$ whenever $(a,b) \in R$ and $(b,c) \in R$.

Eg:

$A = \{1, 2, 3\}$

$R = \{(1,2), (2,3), (1,3)\}$

* Equivalence relation

- A relation 'R' on set A is equivalent if it is reflexive, symmetric and transitive.

Eg:

$A = \{1, 2, 3\}$

$R = \{(1,1), (2,2), (3,3), (1,2), (2,1), (2,3), (1,3), (3,2), (3,1)\}$

* Function

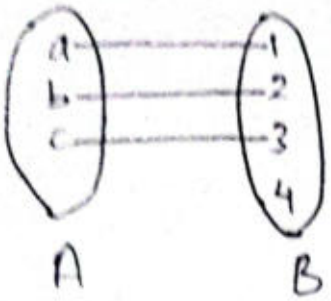
- A function is said to map an element in domain to an element in range.
- A function is denoted by $f(x)$.
- Function can be defined as $f: x \to y$ where, $f(x)$ is called image of x under f.

CamScanner

Types of Function

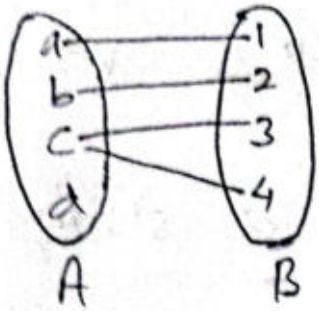
(i) One to One function (Injection)

- A function $f: A \to B$ is said to be one to one if every domain in A has different image in B.



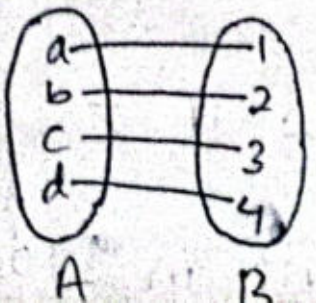
(ii) Onto function (Surjection)

- A function $f: A \to B$ is said to be onto function if each element of B has some image in A.



(iii) One to One Onto function (Bijection)

- A function $f: A \to B$ is said to be one to one onto function if each and every element of A is mapped to exactly one element of B, with no element left over.



* Alphabets and Languages

(i) Alphabet
- Symbols that can be letters or digits
- denoted by Σ

Eg:
$\Sigma = \{0, 1\}$
$\Sigma = \{a, b, c, \ldots\}$
$\Sigma = \{\#, \$, \ldots\}$

(ii) String or Word
- finite sequence of concatenated symbols over alphabet Σ.

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

Here, '1', '00', '11', '101', '1101', ... are strings over $\Sigma = \{0, 1\}$

(iii) Length of string
- no. of symbols in a string 'w'

Eg:
$\Sigma = \{0, 1\}$
$w = 0101$
$|w| = |0101| = 4$

(iv) Empty string
- string of zero length
- denoted by ϵ or ε, or $\wedge$ or λ.

(v) Substring
- Z is substring in w if Z appears consecutively in w.

Eg '01' is substring in '1011'

* Concatenation of strings

- Let w_1 and w_2 be two strings, then concatenation of w_1 and w_2 is

Here,

$w_1 = abc$

$w_2 = xyz$

$w_1 w_2 = abcxyz$

* Closure of alphabet

- defined as set of all strings over an alphabet (Σ) including empty string and denoted by Σ^*.

Eg:

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

$\Sigma^* = \{\epsilon, 0, 1, 00, 10, 11, 1011, \ldots\}$

* Language

- subset of all set of possible strings formed from given set of alphabet.
- Formal Language 'L' is subset of Σ^*.

Eg:

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

L = { set of all strings with equal no. of 0's and 1's }

i.e. $L = \{01, 10, 1100, 1010, \ldots\}$

* Grammar
- Grammar is a formal system for accepting or rejecting a string.

Proof of Techniques (New in syllabus/Imp)

1. Proof by contradiction
2. Pigeonhole principle
3. Induction
4. Diagonalization

1. Proof by contradiction
- given statement is either true or false but not both.
- mathematical proof techniques that determine the given statement / proposition is true by showing that assuming proposition is false.

steps involved in proof of contradiction

(i) Assume that the statement to be true be false.

(ii) In the next step, we show that our supposition leads to logically contradiction.

(iii) In next step, we conclude that our statement is true because our supposition is false.

Eg:

Theorem 1: Statement:
There is no greatest integer

→

In first step, assume the statement is false
i.e. There is a greatest integer

suppose, there is a greatest integer N.
Then $n \leq N$ for every integer n.

Let,
$$M = N + 1$$
Here, M is an integer, since, sum of two integers 'N' and '1' is also an integer.
Also, $M > N$ since $M = N + 1$
Thus, M is an integer that is greater than greatest integer N, which is an contradiction.
Hence, our assumption is not true and so there is no greatest integer.
i.e. There is no greatest integer is true.

CamScanner

statement :

" If n^2 is an even integer, then n is an even integer."

→

Suppose n^2 is an even integer, then n is an odd integer.

Hence,

$n = 2k+1$ for some integer k.

Taking square on both sides;

$n^2 = (2k+1)^2$

or $n^2 = 4k^2 + 4k + 1$

or, $n^2 = 2(2k^2 + 2k) + 1$ (taking 2 as comm

or, $n^2 = 2r + 1$ (general form of odd)

where,

$r = 2k^2 + 2k$

This shows that n^2 is odd which is contradiction towards our assumption.

i.e. " If n^2 is an even integer, then n^2 is an even integer" is true.

Prove that statement: "If n is an integer and n^3+5 is odd, then n is even" using contradiction method.

→

Suppose, If n is an integer and n^3+5 is odd, then n is odd.

Here,

$$n^3+5 = 2k+1 \quad \text{for any integer } k$$

or, $n^3 = 2k+1-5$.

or, $n^3 = 2k-4$

or, $n^3 = 2(k-2)$

Since $2(k-2)$ is product of 2 i.e. it is even which means n^3 is also even.

If n^3 is even, n is also even. as ~~product~~ cube of even numbers only is an even number.

It contradicts our initial assumption.

Therefore,

"If n is an integer and n^3+5 is odd, then n is even" is true.

2. Mathematical Induction

Steps:
(i) Inductive hypothesis (assumption)
(ii) Use $n=k+1$ in $P(n)$.

Proposition:
The sum of first positive integer is $\frac{n(n+1)}{2}$.

→

Initial step:
For $n=1$,

$\frac{1}{2} \times 1(1+1)$

$= \frac{1}{2} \times 1 \times 2$

$= 1$

Stage 1: Our assumption (inductive hypothesis) for $n=k$

$1+2+3+\ldots+k = \frac{1}{2} k(k+1)$

Stage 2:

$1+2+3+\ldots+k+1$

$= \frac{1}{2}(k+1)[(k+1)+1]$

$= \frac{1}{2}(k+1)(k+2)$

Stage 3:

$1+2+3+\ldots+k+1$

$= 1+2+3+\ldots+k+k+1$

$= \frac{(k+1)(k+2)}{2} \quad \frac{k(k+1)}{2} + (k+1)$

$= (k+1)\left(\frac{k}{2}+1\right)$

$= (k+1)\left(\frac{k+2}{2}\right)$

$= \frac{1}{2}(k+1)(k+2)$

This proves that statement holds true for $k+1$.
By principle of mathematical induction, the statement is true for all positive integer n.

Proposition:

For all $n \geq 1$, prove that

$$1^2+2^2+3^2+4^2+\ldots n^2 = \frac{n(n+1)(2n+1)}{6}$$

→ Initial step:

For $n=1$,

$$\frac{1(1+1)(2+1)}{6} = \frac{1 \cdot 2 \cdot 3}{6} = 1 = 1^2$$

Stage 1:

Our assumption (inductive hypothesis) for n = k,

$$1^2 + 2^2 + 3^2 + 4^2 + \ldots + k^2$$

$$= \frac{k(k+1)(2k+1)}{6}$$

Stage 2:

$$1^2 + 2^2 + 3^2 + 4^2 + \ldots + (k+1)^2$$

$$= \frac{(k+1)[(k+1)+1][2(k+1)+1]}{6}$$

$$= \frac{(k+1)(k+2)(2k+3)}{6}$$

Stage 3

$$1^2 + 2^2 + 3^2 + 4^2 + \ldots + k^2 + (k+1)^2$$

$$= \frac{k(k+1)(2k+1)}{6} + (k+1)^2$$

$$= (k+1)\left[\frac{k(2k+1)}{6} + (k+1)\right]$$

$$= (k+1)\left[\frac{2k^2+k+6k+6}{6}\right]$$

$$= \frac{(k+1)(2k^2+7k+6)}{6}$$

$$= \frac{(k+1)(2k^2+4k+3k+6)}{6}$$

$$= \frac{(K+1)[2K(K+2)+3(K+2)]}{6}$$

$$= \frac{(K+1)(K+2)(2K+3)}{6}$$

6. Prove that $1\cdot2\cdot3+2\cdot3\cdot4+\ldots\ldots+n(n+1)(n+2) = \frac{n(n+1)(n+2)(n+3)}{4}$

Solution:

Initial step:

For $n=1$,

$$\frac{1(1+1)(1+2)(1+3)}{4}$$

$$= \frac{1\cdot2\cdot3\cdot4}{4}$$

$$= 1\cdot2\cdot3$$

stage 1:

Assume (inductive hypothesis) for $n=K$,

$$1\cdot2\cdot3+2\cdot3\cdot4+\ldots\ldots+K(K+1)(K+2)$$

$$= \frac{K(K+1)(K+2)(K+3)}{4}$$

stage 2

$$1\cdot2\cdot3+2\cdot3\cdot4+\ldots\ldots+(K+1)(K+2)(K+3)$$

$$= \frac{(K+1)(K+1+1)(K+1+2)(K+1+3)}{4}$$

$$= \frac{(K+1)(K+2)(K+3)(K+4)}{4}$$

CamScanner

Stage 3

$1 \cdot 2 \cdot 3 + 2 \cdot 3 \cdot 4 + \ldots\ldots + (K+1)(K+2)(K+$

$= 1 \cdot 2 \cdot 3 + 2 \cdot 3 \cdot 4 + \ldots\ldots + K(K+1)(K+2)$

$+ (K+1)(K+2)(K+3)$

$= \frac{K(K+1)(K+2)(K+3)}{4} + (K+1)(K+2)(K+3)$

$= (K+1)(K+2)(K+3)\left[\frac{K}{4} + 1\right]$

$= (K+1)(K+2)(K+3)\left(\frac{K+4}{4}\right)$

$= \frac{1}{4}(K+1)(K+2)(K+3)(K+4)$

This proves that statement holds true for $K+1$.

By principle of mathematical induction, the statement is true for all positive integers n.

Q. For every positive integer n, prove that $7^n - 3^n$ is divisible by 4.

→

We have,
$P(n) = 7^n - 3^n$, divisible by 4

For $n = 1$,
$P(1) = 7^1 - 3^1 = 4$ which is divisible by 4

Thus, $P(n)$ is true for $n = 1$.

Let $P(K)$ is true for some positive integer k.
i.e. $P(K) = 7^K - 3^K$ which is divisible by 4

So,
$$P(K) = 7^K - 3^K$$
$$= 4d, \quad d \in N.$$

Now, for some positive integer $K+1$,
$$P(K+1) = 7^{K+1} - 3^{K+1}$$
$$= 7^{K+1} - 7 \cdot 3^K + 7 \cdot 3^K - 3^{K+1}$$
$$= 7(7^K - 3^K) + 3^K(7 - 3)$$
$$= 7 \cdot 4d + 3^K \cdot 4 \quad [\because 7^K - 3^K = 4d]$$
$$= 4(7d + 3^K)$$

which is divisible by 4.

Thus, given statement $P(n) = 7^n - 3^n$ is true for positive integer n.

Q. Prove that for all $n \in N$, $n(n+1)(n+5)$ is multiple of 3.

→

We have,

$P(n) = n(n+1)(n+5)$ multiple of 3.

For $n = 1$,

$P(1) = 1(1+1)(1+5) = 12$ which is multiple of 3.

Thus, $P(n)$ is true for $n=1$

Let $P(k)$ is true for some positive integer k.

i.e. $P(k) = k(k+1)(k+5)$ which is multiple of 3.

so, $P(k) = k(k+1)(k+5)$

$= 3d$, $d \in N$

Now, for some positive integer $k+1$,

$P(k+1) = \cancel{k(k+1)(k+5)}$

$= (k+1)(k+2)(k+6)$

$= (k+1)[k^2+6k+2k+12]$

$= (k+1)[k^2+8k+12)$

$= (k+1)(k^2+5k+3k+12)$

$= (k+1)[k(k+5)+3(k+4)]$

$= k(k+1)(k+5)+3(k+1)(k+4)$

$= 3d + 3(k+1)(k+4)$

$= 3(d+(k+1)(k+4))$

which is multiple of 3.

Pigeonhole Principle:

→ If $k+1$ objects is to be placed in k boxes, then there is at least one box that has more than one object.

Generalized:
If N objects are to be placed in k boxes, then there is at least one box containing $\frac{N}{k}$ object.

Theorem:
A function f from a set with $k+1$ element to a set with k elements is not one to one.

Proof:

Using pigeonhole principle,

(a) Create a box for each y in the codomain f.

(b) Put in the box for y of all the elements of x such that $f(x)=y$.

(c) Since, there are $k+1$ elements and only k boxes, atleast one box has 2 or more elements.

Hence, the function is not one to one.

Generalized pigeon principle

If there are n pigeon holes and kn+1 or more pigeons then at least one pigeonhole is occupied by k+1 or more pigeon.

eg:

Q. How many students are there in a class among which at least four of them are born in same month.

Soln:

Total no. of month (n) = 12 (pigeon hole)

i.e. $k+1 = 4 = 3+1$

$\therefore k = 3$

Total no. of students (pigeons)

$= kn+1$

$= 3 \times 12+1$

$= 37$

Q. 25 crates of apples are delivered to a store. The apples are of three different sorts. and all the apples in each crates are of the same sort. Show that among these crates, there are at least 9 containing the same sort. of apples?

Soln:

Diagonalization Principle

Statement:

Let R be a binary relation on a set A and ~~set~~ B let D, the diagonal set for R, be $\{a : a \in A$ and $(a,a) \notin R\}$. For each $a \in A$, let $R_a = \{b : b \in A$ and $(a,b) \in R\}$. Then, D is distinct from each R_a.

In short, Diagonalization principle states that the complement of diagonal is different from each row.

For example,

Let $S = \{a, b, c, d\}$

$R = \{(a,a), (b,c), (b,d), (c,a), (c,c), (c,d), (d,a), (d,b)\}$

Matrix form

	a	b	c	d
a	x			
b			x	x
c	x		x	x
d	x	x		

From figure,

$R_a = \{a\}$

$R_b = \{c, d\}$

$R_c = \{a, c, d\}$

$R_d = \{a, b\}$

Complement of diagonal is,

$D = \{b, d\}$

i.e. if we compare each of the set R_a, R_b, R_c and R_d with diagonal set (D), we can see D is different from each row. Thus, complement of diagonal is different from each row.

Chomsky Hierarchy

→ Hierarchy of grammar

→ defined by Noam chomsky in 1956.

→ Grammar is defined by four tuples

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

↳ rules to define/generate language.

where, V = non Terminals or Variable (capital letter)

T = Terminals (small letter)

P = Production rule

S = starting symbol

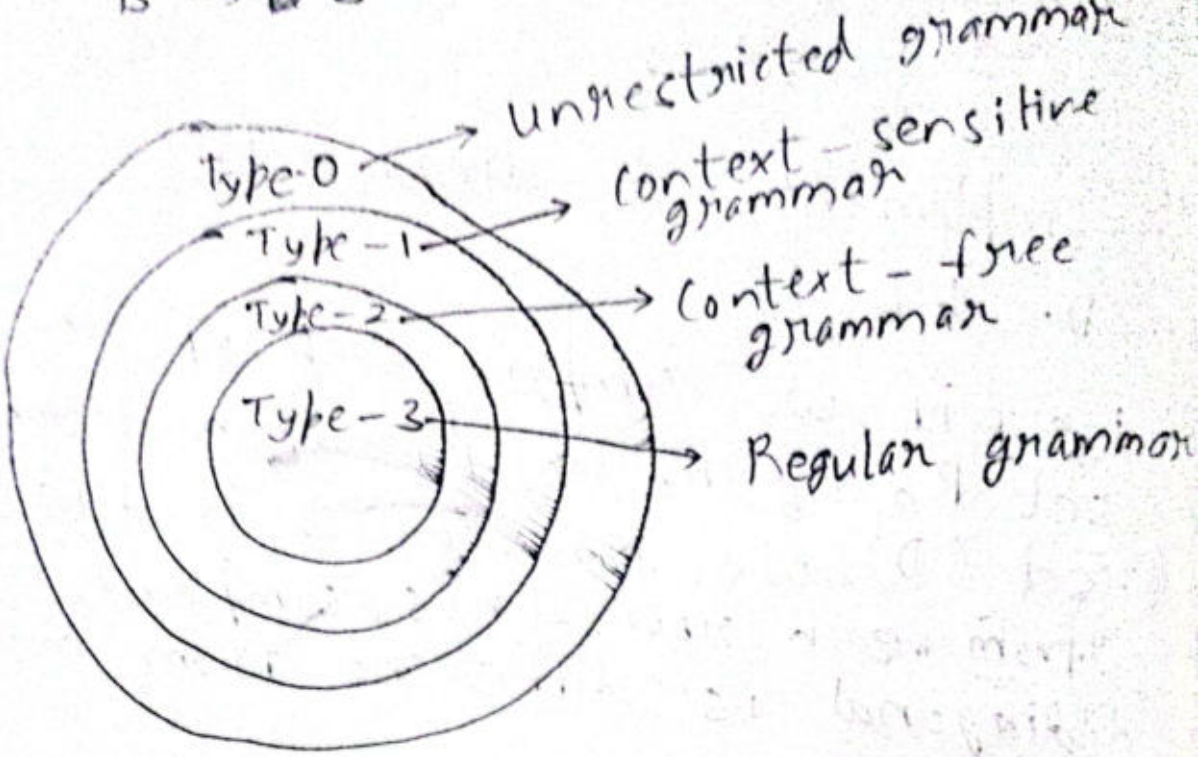
Eg:

Production rule

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$ L



- More restricted the grammar, easier the language. and vice versa.

#* Languages generated by different grammar

- Unrestricted grammar
  ↳ Restricted enumerable Language (REL)
- Context sensitive grammar
  ↳ Context sensitive Language (CSL)
- Context free grammar
  ↳ Context free Language (CFL)
- Regular grammar
  ↳ Regular Language (RL)

(i) Type - 0 grammar

→ include all formal grammar
→ all language are recognized by Turing Machine.
→ describe syntax for programming language
→ grammar has rules of form

$$\alpha \rightarrow \beta$$

where, α is non terminal
β is terminal or non terminal.

(ii) Type - 1 grammar

→ generate context sensitive language.
→ all language are recognized by LBA (Linear Bound Absolute)
→ Grammar has rules of form, with restriction that length of $|\alpha| \leq |\beta|$

$$\alpha \rightarrow \beta$$

eg:

$AB \rightarrow A$ *
$A \rightarrow aB$ ✓
$B \rightarrow a$ ✓

(iii) Type – 2 Grammar

→ generate context free language.

→ all language are recognized by PDA (Push down Automata)

→ Grammar has rules of form

$A \rightarrow \alpha$

where, A is a non terminal

α is terminal or non terminal.

→ There will be no context on the left and right of non terminal.

Eg:

$A \rightarrow BCD$ ✓

$A \rightarrow aBC$ ✓

$a \rightarrow Abcx$

(iv) Type – 3 Grammar

→ generate Regular language

→ all languages are recognized by FA (Finite Automata)

→ grammar has rules of form

$\alpha \rightarrow \beta$

α: non-terminal

β: terminal or non terminal.

Eg:

$S \rightarrow ab \mid S$

$A \rightarrow b$

$B \rightarrow \varepsilon$

Chapter - 2

Finite Automata and Regular Language (10 hrs)

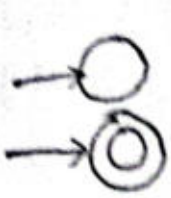
Finite Automata : (self-acting)

→ set of states and it's control moves from state in response to external input
→ simplest model of computing
→ recognized model of computing

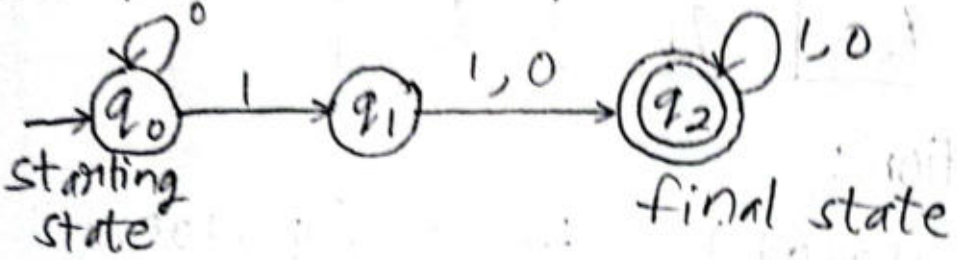
Formal Definition:
Finite Automata has 5 types defined as,
$M = \{ Q, \Sigma, \delta, q_0, F \}$
where
Q = set of finite state
Σ = input symbol
δ = transition function,
$Q \times \Sigma = Q$
q_0 = initial state
F = set of final states / accepted states

Types:
1. Deterministic FA (DFA):
→ depend on current states and input.

2. Non deterministic FA (NFA/NDFA):
→ particularly depend on current state and input.

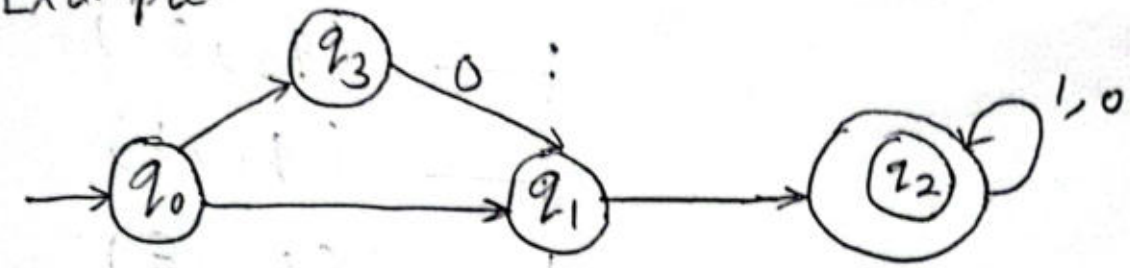


Example: Deterministic Finite Automata.



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

Example: Non-deterministic Finite Automa



Instantaneous Description (ID)

Q Given string is 110101. Check whether the string is accepted or not when transition function (δ) are.

State Table:

state	Input	
	0	1
→ (q_0)	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_2
q_3	q_1	q_2

Solution:

$(q_0, 110101) \vdash (q_1, 10101)$
$\vdash (q_0, 0101)$
$\vdash (q_2, 101)$
$\vdash (q_3, 01)$
$\vdash (q_1, 1)$
$\vdash (q_0, \varepsilon)$

∴ sequence of states
$= (q_0, q_1, q_0, q_2, q_3, q_1, q_0)$

Since q_0 is final state.
So, string is accepted.

Q. Consider finite state automata in which

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

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

$q_0 = \{q_0\}$, initial state

$F = \{q_0\}$, final state

δ is given by,

state	Input	
	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

a. Write a entire sequence of state for input string 101101.

b. Check whether string is accepted or not.

i) 11111

ii) 000000

iii) 101101

Solution

(a)

$(q_0, 101101) \vdash (q_1, 01101)$

$\vdash (q_3, 1101)$

$\vdash (q_2, 101)$

$\vdash (q_3, 01)$

$\vdash (q_1, 1)$

$\vdash (q_0, \epsilon)$

Here, q_0 is final state. so it is accepted.

CamScanner

The sequence of state is
$\{q_0, q_1, q_3, q_2, q_3, q_1, q_0\}$

(b)

<i> $(q_0, 11111) \vdash (q_1, 1111)$
$\vdash (q_0, 111)$
$\vdash (q_1, 11)$
$\vdash (q_0, 1)$
$\vdash (q_1, \varepsilon)$

Here q_1 is final state so, it is not accepted.

<ii> $(q_0, 000000) \vdash (q_2, 00000)$
$\vdash (q_0, 0000)$
$\vdash (q_2, 000)$
$\vdash (q_0, 00)$
$\vdash (q_2, 0)$
$\vdash (q_0, \varepsilon)$

Here q_0 is final state, so it is accepted.

CamScanner

(iii)

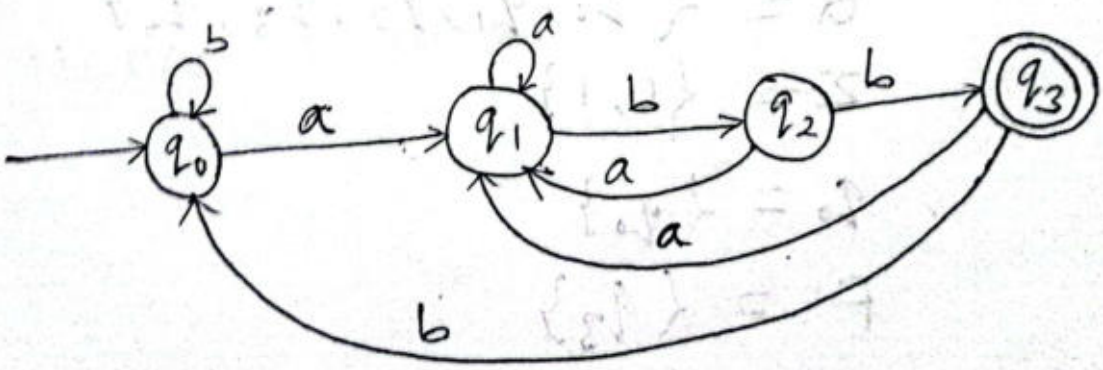
$(q_0, 101101) \vdash (q_1, 01101)$
$\vdash (q_3, 1101)$
$\vdash (q_2, 101)$
$\vdash (q_3, 01)$
$\vdash (q_1, 1)$
$\vdash (q_0, \varepsilon)$

Here, q_0 is final string so it is accepted.

Q. Design a DFA to accept the string of a's and b's ending with 'abb' over $\Sigma = \{a, b\}$.

→

Given, substring = abb.
length of substring |abb| = 3
no. of states = 3+1 = 4



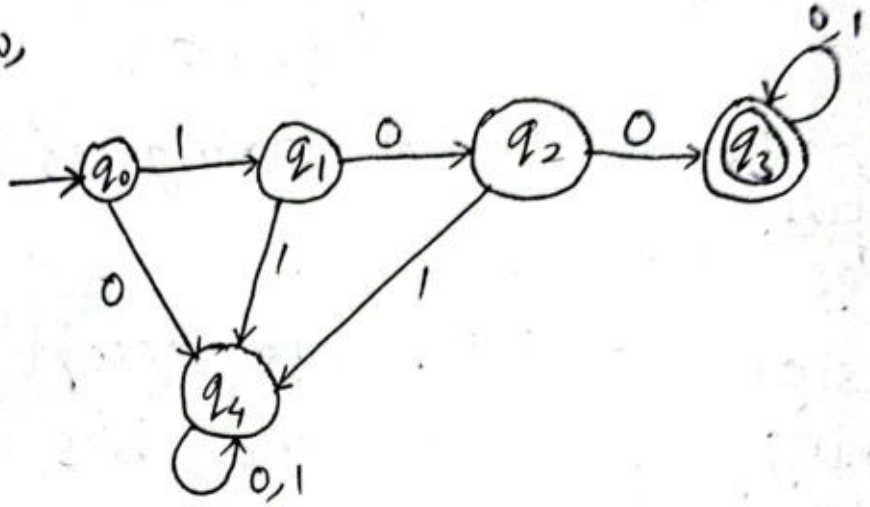
Q. Design a DFA that accepts string with '100' over $\Sigma = \{0, 1\}$

→

Length of substring, $|100| = 3$

no. of state $= 3 + 1 = 4$

Now,



Here, DFA, M is defined as

$M = \{Q, \Sigma, \delta, q_0, F\}$

where,

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ is given by,

State	Input	
	0	1
→ q_0	q_4	q_1
q_1	q_2	q_4
q_2	q_3	q_4
(q_3)	q_3	q_3
q_4	q_4	q_4

Now, Processing DFA for '1001010' as

$(q_0, 1001010) \vdash (q_1, 001010)$
$\vdash (q_2, 01010)$
$\vdash (q_3, 1010)$
$\vdash (q_3, 010)$
$\vdash (q_3, 10)$
$\vdash (q_3, 0)$
$\vdash (q_3, \varepsilon)$

Hence, accepted.

Q. Design a DFA that accepts Language L which has set of strings in $(a,b)^*$ which have even no. of b's.

→ Possible language, L = {bb, abb, ...}
Length of substring, |bb| = 2
∴ no. of state = 2+1 = 3

Now, state transition diagram can be drown as.

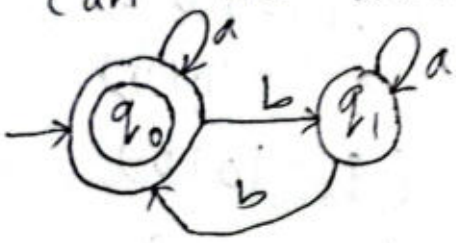


Fig: State Transition Diagram

DFA M is defined as.

$M = \{Q, \Sigma, \delta, q_0, F\}$

where,

$Q = \{q_0, q_1\}$

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

$q_0 = \{q_0\}$

$F = \{q_0\}$

δ is given by

State	Input	
	a	b
→ q_0	q_0	q_1
q_1	q_1	q_0

Fig: State Transition Diagram.

Now, Processing DFA for 'aabbbb' as

~~(aabbbb)~~

$\delta(q_0, aabbbb)$

$\vdash (q_0, abbbb)$

$\vdash (q_0, bbbb)$

$\vdash (q_1, bbb)$

$\vdash (q_0, bb)$

$\vdash (q_1, b)$

$\vdash (q_0, \varepsilon)$

Since, q_0 is final state.

Hence, it is accepted.

Q. Design a DFA that accepts language $L = \{ a^n b \; ; \; n \geq 0 \}$.

→ possible language, $L = \{ b, ab, aab, \ldots \}$

Length of substring $|b| = 1$

∴ no. of state $= 1 + 1 = 2$

Now, drawing state transition diagram as,

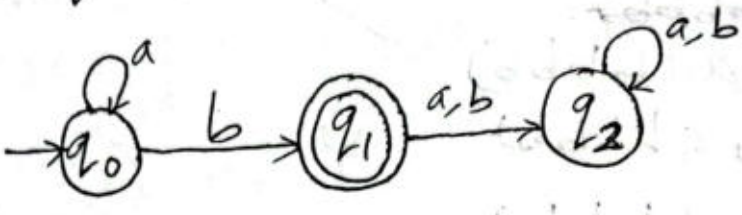


Fig: state Transition Diagram

DFA, M is defined as,

$M = \{ Q, \Sigma, \delta, q_0, F \}$

where,

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

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

$q_0 = \{ q_0 \}$

$F = \{ q_1 \}$

δ is given by

state	Input	
	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

Fig: state Transition Table

Now, processing DFA for 'aaab' as,

$\delta(q_0, aaab)$
$\vdash (q_0, aab)$
$\vdash (q_0, ab)$
$\vdash (q_0, b)$
$\vdash (q_1, \varepsilon)$

Since q_1 is final state
Hence it is accepted.

Q. Design a DFA that accepts string containing three consecutive b's over $\Sigma = \{a, b\}$

→

Possible language, $L = \{bbb, abbb, \ldots\}$

Length of substring, $|bbb| = 3$

no. of state $= 3 + 1 = 4$

Now, state transition diagram can be drawn as,

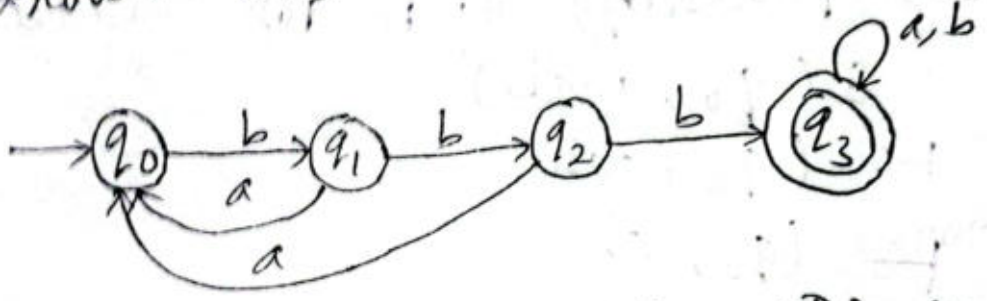


Fig: state Transition Diagram

DFA M is defined as,

$M = \{Q, \Sigma, \delta, q_0, F\}$

where,

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

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ is given by,

State	Input a	Input b
→ q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_1
(q_3)	q_3	q_3

Fig: state transition Table

Now, Processing DFA for 'abbbab' as

$\delta(q_0, abbbab)$
$\vdash (q_0, bbbab)$
$\vdash (q_1, bbab)$
$\vdash (q_2, bab)$
$\vdash (q_3, ab)$
$\vdash (q_3, b)$
$\vdash (q_3, \varepsilon)$

Since q_3 is final state.
Hence, given string is accepted.

Q. Design a DFA that accepts language $L = \{ b^m a b^n ; m, n \geq 0 \}$.

→

Possible Language, $L = \{ bab, bbab, \ldots \}$

Length of substring $|bab| = 3$

no. of state $= 3 + 1 = 4$

Now, drawing state transition diagram as,

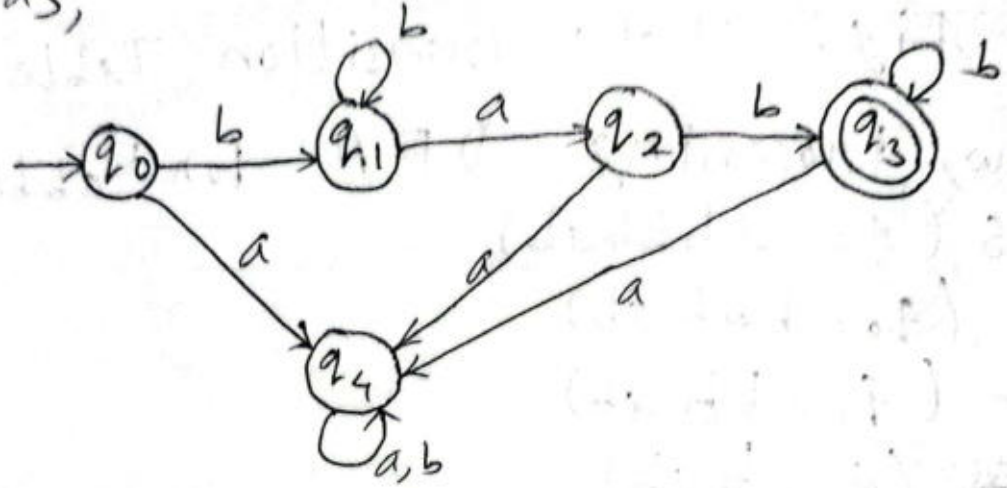


Fig: state Transition Diagram

DFA, M is defined as

$M = \{Q, \Sigma, \delta, q_0, F\}$

where,

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ is given by,

State	Input	
	a	b
→ q_0	q_4	q_1
q_1	q_2	q_1
q_2	q_4	q_3
(q_3)	q_4	q_3
q_4	q_4	q_4

Fig: State Transition Table

Now, Processing DFA for 'bbbabb'

$\delta(q_0, bbbabb)$
$\vdash (q_1, bbabb)$
$\vdash (q_1, babb)$
$\vdash (q_1, abb)$
$\vdash (q_2, bb)$
$\vdash (q_3, \varepsilon)$

since q_3 is final state.
So, this string is accepted.

Q. Design a DFA that accepts all string that doesnot have thrice consecutive b's over $\Sigma = \{a, b\}$

→ Possible language, $L = \{ bb, abb, \ldots \}$

Now, state transition diagram can be drawn as,

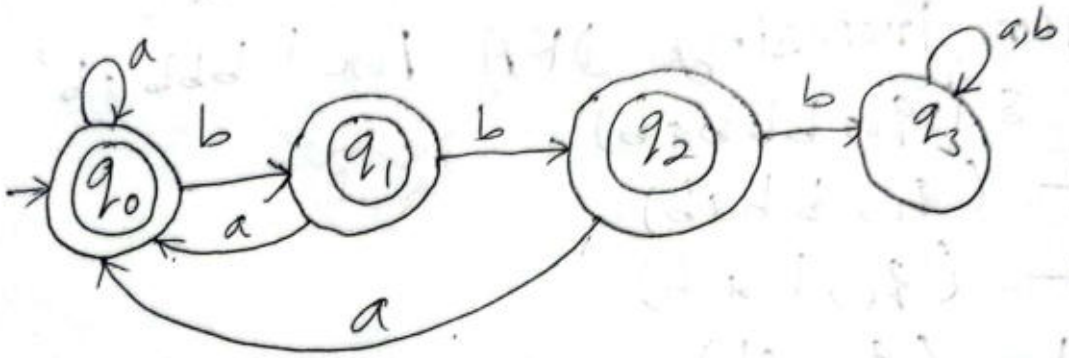


Fig: state Transition Diagram

DFA, M is defined as

$M = \{ Q, \Sigma, \delta, q_0, F \}$

where,

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

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

$q_0 = \{ q_0 \}$

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

s is given by,

state	Input	
	a	b
→ q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_3
q_3	q_3	q_3

Fig: State Transition Table

Now, Processing DFA for 'abba'

$\delta(q_0, abba)$
$\vdash (q_0, bba)$
$\vdash (q_1, ba)$
$\vdash (q_1, a)$
$\vdash (q_0, \varepsilon)$

Since, q_0 is final state.
So, it is accepted.

CamScanner

Q. Design DFA that accepts all strings not having more than two a's over $\Sigma = \{a, b\}$

→

Possible Language, $L = \{aa, baa, \ldots\}$

Now, drawing state transition diagram as,

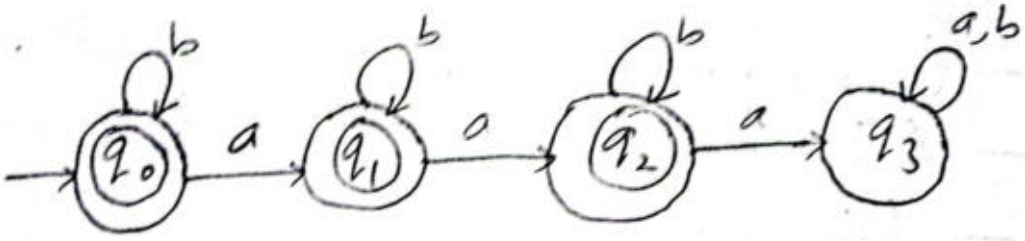


Fig: state Transition Diagram

DFA, M is defined as.

$M = \{Q, \Sigma, \delta, q_0, F\}$

where,

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

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

$q_0 = \{q_0\}$

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

δ is given by:

state	Input	
	a	b
→ q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_2
q_3	q_3	q_3

Fig: state Transition Table

Now, Processing DFA for 'abbab' as

$\delta(q_0, abbab)$
$\vdash (q_1, bbab)$
$\vdash (q_1, bab)$
$\vdash (q_1, ab)$
$\vdash (q_2, b)$
$\vdash (q_2, \varepsilon)$

Since q_2 is final state.

So, it is accepted.

Equivalence of DFA and NFA

Construct the DFA equivalent to NFA,

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

and state table is

State	Input	
	0	1
→ q_0	q_0	q_1
q_1	q_1	$q_0 q_1$

Solution:

For NFA,

Here,

initial state = q_0

final state = q_0

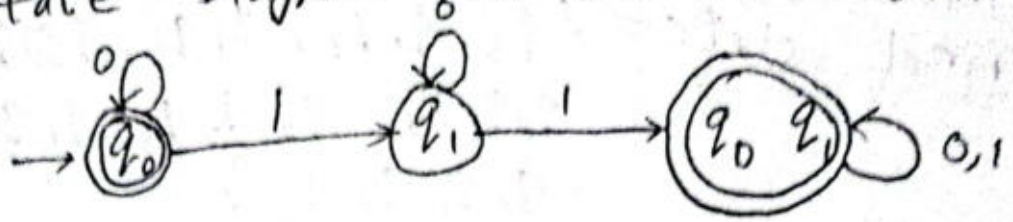
Now, for DFA, we construct transition function δ as,

State	Input	
	0	1
q_0	q_0	q_1
q_1	q_1	$q_0 q_1$
$q_0 q_1$	$q_0 q_1$	$q_0 q_1$

CamScanner

Note: Initial state of NDFA is same as DFA.

So, state Diagram for DFA is,



Here, final state of DFA is,

$$F' = 2^{Q-1} = 2^{2-1} = 2$$

$$F' = \{q_0, q_0q_1\}$$

Q. Construct a DFA equivalent to $M' = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, \{q_0\}, \{q_3\})$ and δ is

state	Input	
	0	1
q_0	q_0q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_2
q_3	q_3	q_3

Soln:

Here, for NFA

initial state = q_0

final state = q_3.

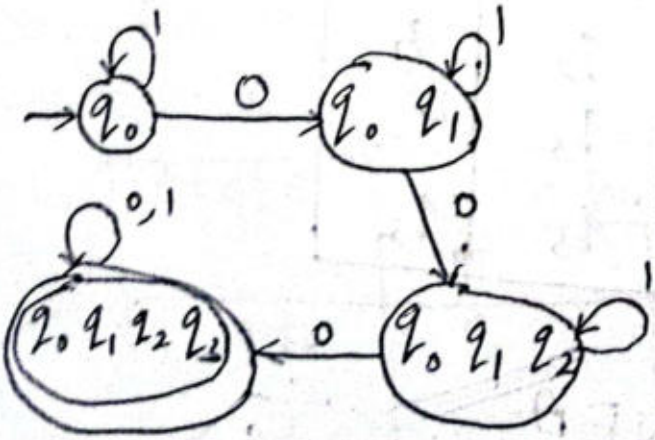
Now, for DFA, we construct transition table δ,

initial state = q_0

final state = { q_3, q_0q_3, q_1q_3, q_2q_3, $q_0q_1q_3$, $q_0q_2q_3$, $q_1q_2q_3$, $q_0q_1q_2q_3$ }

state	Input	
	0	1
q_0	q_0q_1	q_0
q_0q_1	$q_0q_1q_2$	q_0q_1
$q_0q_1q_2$	$q_0q_1q_2q_3$	$q_0q_1q_2$
$q_0q_1q_2q_3$	$q_0q_1q_2q_3$	$q_0q_1q_2q_3$

So, State Diagram for DFA is,



Q. Construct a DFA equivalent to $M' = (\{q_1, q_2, q_3\}, \{0, 1\}, \delta, \{q_1\}, \{q_3\})$ and δ is given by,

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

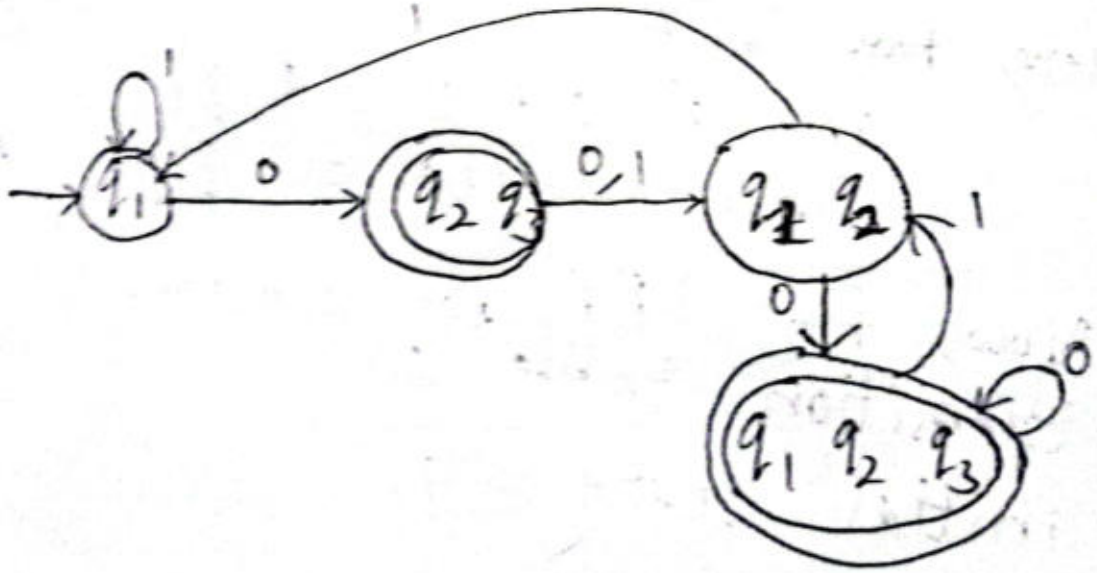
Soln:

Here, for NFA,
initial state = q_1
final state = q_3

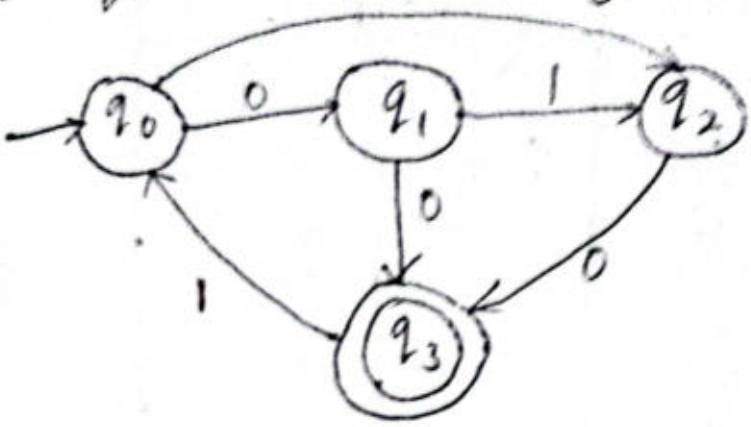
Now, for DFA, we construct transition table δ,
initial state = q_1

state	Input	
	0	1
q_1	$q_2\ q_3$	q_1
$q_2\ q_3$	$q_1\ q_2$	$q_1\ q_2$
$q_1\ q_2$	$q_1\ q_2\ q_3$	q_1
$q_1\ q_2\ q_3$	$q_1\ q_2\ q_3$	$q_1\ q_2$

So, state diagram for DFA is



Q. Construct a given DFA equivalent to given NFA.



Here, for NFA,

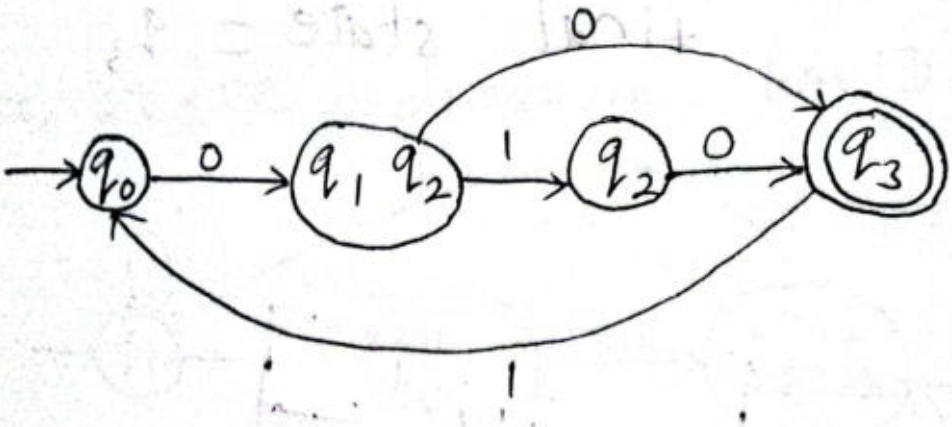
initial state = q_0

final state = q_3

State	Input	
	0	1
q		

state	Input	
	0	1
q_0	$q_1 q_2$	$\emptyset$
$q_1 q_2$	q_3	q_2
q_3	$\emptyset$	q_0
q_2	q_3	$\emptyset$

So, state diagram for DFA is



Minimization of DFA

Eg:

Current

Minimization of DFA

Eg:

Current state	Input symbol a	Input symbol b
→ q_0	q_5	q_1
q_1	q_2	q_6
* q_2	q_2	q_0
q_4	q_5	q_7
q_5	q_6	q_2
q_6	q_4	q_6
q_7	q_2	q_6
q_3	q_6	q_2

Step 1: Eliminate state that can't be reached from starting state.

Here q_3 is not reachable, so remove it.

we get

Current state	Input symbol a	Input symbol b
→ q_0	q_5	q_1
q_1	q_2	q_6
* q_2	q_2	q_0
q_4	q_5	q_7
q_5	q_6	q_2
q_6	q_4	q_6
q_7	q_2	q_6
q_8		

Step 2: Now Divide the table into two sets as

(a) set 1: Containing rows which start from non-final state.

Current State	Input symbol	
	a	b
→ q_0	q_5	q_1
q_1	q_2	q_6
q_4	q_5	q_7
q_5	q_6	q_2
q_6	q_4	q_6
q_7	q_2	q_6

(b) set 2: Containing rows which starts from final state.

Current state	Input symbol	
	a	b
* q_2	q_2	q_0

Step 3: Considering set 1,

Here, Row 2 and Row 6 transit to same states on input 'a' and b. So, we remove one of them (for instance q_7) and replace q_7 with q_1

Current State	Input symbol a	Input symbol b	
→ q_0	q_3	q_1	Row 1
q_1	q_2	q_6	Row 2
q_4	q_3	q_1	Row 3
q_5	q_6	q_2	Row 4
q_6	q_4	q_6	Row 5

Here, Row 1 and Row 3 is same, so, we remove q_4 and replace it with q_6.

Current State	Input symbol a	Input symbol b	
→ q_0	q_3	q_1	
q_1	q_2	q_6	
q_3	q_6	q_2	
q_6	q_0	q_6	

Now, there are no similar rows.
considering set 2,

Current State	Input symbol a	Input symbol b
* q_2	q_2	q_0

Here, there is only one row, so, it is already minimized.

Step 4: Now, combine minimized set 1 and set 2 as,

Current state	Input Symbol	
	a	b
→ q_0	q_5	q_1
q_1	q_2	q_6
* q_2	q_2	q_0
q_5	q_6	q_2
q_6	q_0	q_6

Minimized

Q. Minimize the DFA. (Assignment)

Input state	Input symbol	
	a	b
→ q_0	q_1	q_2
q_1	q_0	q_3
q_2	q_1	q_4
* q_3	q_5	q_5
q_4	q_3	q_3
* q_5	q_5	q_5

Solution:

Step 1: Eliminate state that can't be reached from starting state.

Here, q_2 is not reachable. So, remove it.

We get

Input state	Input symbol	
	a	b
→ q_0	q_1	q_3
q_1	q_0	q_3
* q_3	q_5	q_5
q_4	q_3	q_3
* q_5	q_5	q_5

Again, q_4 is not reachable. So, remove it.

Input state	Input symbol	
	a	b
→ q_0	q_1	q_3
q_1	q_0	q_3
* q_3	q_5	q_5
* q_5	q_5	q_5

Step 2: Now, divide the table into two sets as:

(a) set 1: Containing rows which start from non final state.

Input state	Input symbol	
	a	b
→ q_0	q_1	q_3
q_1	q_0	q_3

(b) set 2: Containing rows which start from final state.

Input state	Input symbol		
	a	b	
* q_3	q_5	q_5	Row 1
* q_5	q_5	q_5	Row 2

Step 3: Considering set 2,

Here, row 1 and row 2 transit to some state on input a and b.

So, we remove one of them (for instance q_5) and replace q_5 with q_3.

~~Considering set 1,~~

~~There are no s~~

Input state	Input Symbol	
	a	b
$*q_3$	q_3	q_3

considering set 1,

Input state	Input Symbol	
	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3

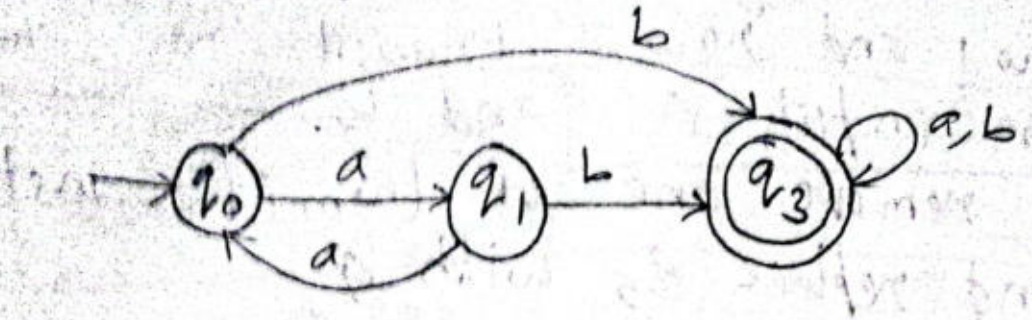
There are no common rows, so already minimized.

Step 4: Now combine minimized set 1 and set 2 as,

Input state	Input Symbol	
	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
$*q_3$	q_3	q_3

CamScanner

State Diagram can be drawn as;



Regular Expression

↳ Language accepted by Finite Automata can be represented as expression called regular expression.

↳ sequence of pattern that defines string.

↳ Formal definition over Σ,

1. Any terminal symbol (null: ε and empty set : $\emptyset$) are regular expression.
2. The union of two regular expression R_1 and R_2 written as $R_1 + R_2$ is also a regular language.
3. Concatenation of two regular expression $R_1 \cdot R_2$ is also a regular expression.
4. The iteration (Kleene Closure) of regular expression written as $R_1 * R_2$ is also a regular expression.

$L = \{ a, aa, aaa, aaaa, \ldots \}$
R.E. = a^+
$L = \{ \varepsilon, a, aa, aaa, \ldots \}$
R.E. = a^*

Note: $\varepsilon, \Lambda, e, \lambda$ are all same.

* → include ε too
+ → donot include ε

Eg: Let $\Sigma = \{a, b\}$, Find regular expression for language L whose length is exactly 2.

$\Rightarrow$

Here, $L = \{aa, ab, ba, bb\}$

Now, Union of all string.

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

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

Eg: For at least length.

$L = \{aa, ab, ba, bb, aaa, aab, bab, \ldots\}$

For exactly length 2,

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

Now, for at least 2,

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

Eg:

Q. Write R.E. for language accepting all strings containing any no. of a's and b's.

→

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

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

Q. Write R.E for string over $\Sigma = \{0, 1\}$ starting with 1 and ending with 0.

→

$L = \{ 10, 100, 110, \ldots \}$

$R.E. = \{1 (0+1)^* 0\}$

Q. Write R.E. for string over $\Sigma = \{0, 1\}$ having even length of string.

→

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

$R.E. = \{ (0+1)(0+1)\}^*$

$= \{ (0+1)^2\}^*$

$= (0+1)^{2*} = (0+1)^{2n} \; ; n \geq 0$

Q Write R.E. for string over $\Sigma = \{0, 1\}$ that do not contain substring 01.

→

$L = \{\varepsilon, 0, 10, 11, 1000, \ldots\}$

$R.E. = \{1^* 0^*\}$

Q. Write R.E. for language accepting all strings starting and ending with a and having any combination of b's in between.

→

$L = \{aa, aba, abba, \ldots\}$

R.E. $= ab^*a$

Identities of R.E.

1. $\phi + R = R$
2. $\phi R + R\phi = \phi$
3. $\varepsilon R = R\varepsilon = R$
4. $\varepsilon^* = \varepsilon$ and $\phi^* = \varepsilon$
5. $R + R = R$
6. $R^* R^* = R^*$
7. $RR^* = R^* R$
8. $(R^*)^* = R^*$
9. $\varepsilon + RR^* = \varepsilon + R^* R = R^*$
10. $(PQ)^* Q = P(PQ)^*$
11. $(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$
12. $(P+Q)R = PR + QR$

    $R(P+Q) = RP + RQ$

Example to show use (basic example not shown)

Prove that

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

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

Soln:

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

$= (1+00^*1)\{\varepsilon+(0+10^*1)^*(0+10^*1)\}$

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

$[\because \varepsilon + R^*R = R^*]$

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

$= 0^*1(0+10^*1)^*$ $[\because \varepsilon+R^*R=R^*]$

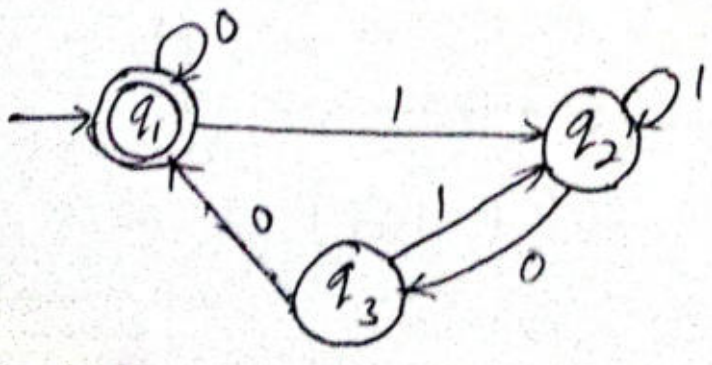
$\equiv$ (Proved)

Imp. # Arden's Theorem

If P, Q, R are R.E. and $R = Q+RP$.

Then, $\boxed{R = QP^*}$

Q. Find R.E. from following state diagram



Soln:

Here q_1 is initial

$q_1 = q_1 0 + q_3 0 + \varepsilon$ ..... ①

$q_2 = q_1 1 + q_2 1 + q_3 1$ ..... ②

$q_3 = q_2 0$ ..... ③

Solving equation ① and ③,

$q_1 = q_1 0 + q_2 00 + \varepsilon$ ..... ④

Solving equation ② and ③,

$q_2 = q_1 1 + q_2 1 + q_2 01$

or, $q_2 = q_1 1 + q_2 (1 + 01)$

Now, using Arden's theorem, we get

$q_2 = q_1 1 (1 + 01)^*$ ..... ⑤ [∵ If $R = Q + RP$ ⇒ $R = QP^*$]

Solving ④ and ⑤,

$q_1 = q_1 0 + q_1 1 (1 + 01)^* 00 + \varepsilon$

$q_1 = q_1 (0 + 1 (1 + 01)^* 00) + \varepsilon$

$\underset{R}{q_1} = \underset{Q}{\varepsilon} + \underset{R}{q_1} \underbrace{(0 + 1(1 + 01)^* 00)}_{P}$

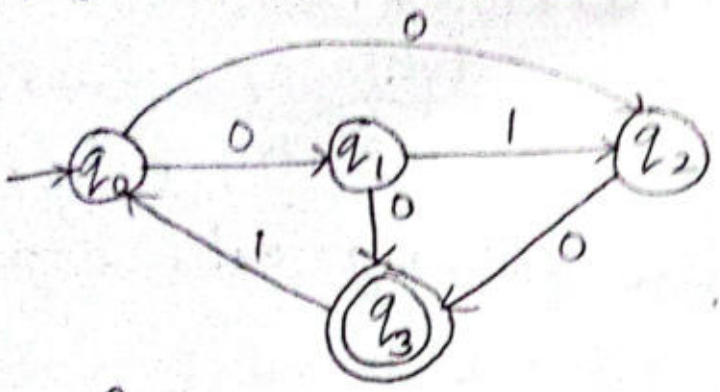
Using Arden's theorem,

$q_1 = \varepsilon (0 + 1(1 + 01)^* 00)^*$

$\therefore q_1 = (0 + 1(1 + 01)^* 00)^*$ [∵ $\varepsilon R = R\varepsilon = R$]

So, R.E. is $(0 + 1(1 + 01)^* 00)^*$

Q. Find R.E. for



Soln:

$q_0 = q_3 1 + \varepsilon$ — ①

$q_1 = q_0 0$ — ②

$q_2 = q_0 0 + q_1 1$ — ③

$q_3 = q_1 0 + q_2 0$ — ④

solving equation ① and ③,

$q_2 = (q_3 1 + \varepsilon) 0 + q_1 1$ — ⑤

solving equation ① and ②,

$q_1 = (q_3 1 + \varepsilon) 0$ — ⑥

solving ⑤ and ⑥,

$q_2 = (q_3 1 + \varepsilon) 0 + (q_3 1 + \varepsilon) 01$

~~or, $q_2 = (q_3 1 + \varepsilon) 0 \{\varepsilon + 1\}$~~

or, $q_2 = q_3 10 + \varepsilon 0 + (q_3 10 + \varepsilon 0) 1$

or, $q_2 = q_3 10 + 0 + (q_3 10 + 0) 1$

or $q_2 = q_3 10 (\varepsilon + 1) 0 + 0 + 01$ — ⑦

Now, solving ④, ⑥ and ⑦,

$q_3 = (q_3 10 + \varepsilon 0) 0 + \{q_3 10(\varepsilon + 1) + 0 + 01\} 0$

or, $q_3 = q_3 100 + 00 + q_3 100 (\varepsilon + 1) + 00 + 010$

or, $q_3 = q_3 100 + 00 + q_3 100 + q_3 1001 + 00 + 010$

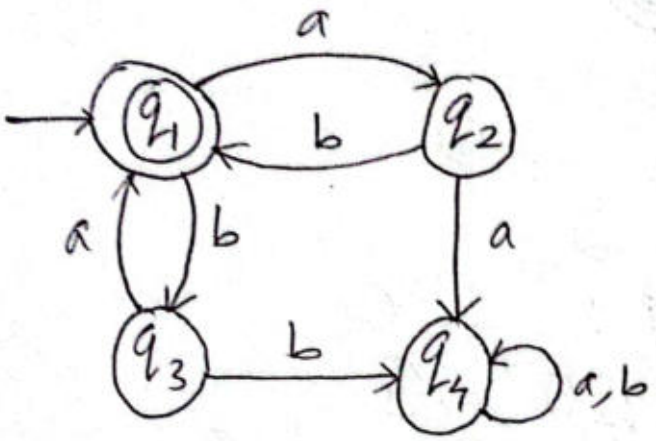
or, $q_3 = q_3 (100 + 100 + 1001) + (00 + 00 + 010)$

q_3 = R, q_3 = R, $(100 + 100 + 1001)$ = P, $(00 + 00 + 010)$ = Q

$q_3 = (00+00+010)(100+100+1001)^*$

(∵ If $R = Q + RP$
Then, $R = QP^*$)

Q. Find R.E. for following state diagram.



solution:

$q_1 = q_2 b + q_3 a + \varepsilon$ — (I)

$q_2 = q_1 a$ — (II)

$q_3 = q_1 b$ — (III)

$q_4 = q_2 a + q_3 b + q_4 a + q_4 b$ — (iv)

Solving (I) and (II),

$q_1 = q_1 ab + q_3 a + \varepsilon$ — (V)

Solving (III) and (IV) (V),

$q_1 = q_1 ab + q_1 ba + \varepsilon$

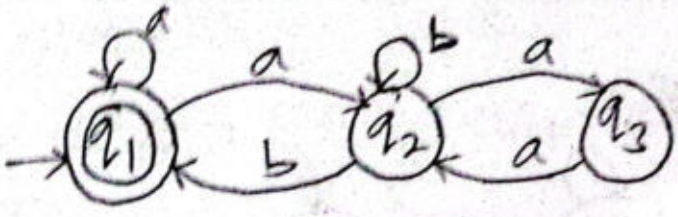
or, $q_1 = q_1 (ab + ba) + \varepsilon$

or, $q_1 = \varepsilon (ab + ba)^*$

$q_1 = (ab + ba)^*$

Assignment

Q.



Solution:

$q_1 = q_1 a + q_2 b + \varepsilon$ — (I)

$q_2 = q_2 b + q_3 a + q_1 a$ — (II)

$q_3 = q_2 a$ — (III)

Solving (II) and (III),

$q_2 = q_2 b + q_2 aa + q_1 a$

or, $q_2 = q_2 (b + aa) + q_1 a$

$q_2 = q_1 a (b + aa)^*$ — (IV)

Now, from (I) and (IV),

$q_1 = q_1 a + [q_1 a (b+aa)^*] b + \varepsilon$

or, $q_1 = q_1 a + q_1 ab(b+aa)^* b + \varepsilon$

or, $q_1 = q_1 (a + ab(b+aa)^* b) + \varepsilon$

or, $q_1 = \varepsilon (a + a(b+aa)^* b)^*$

$q_1 = (a + a(b+aa)^* b)^*$

~~Also,~~ Hence,

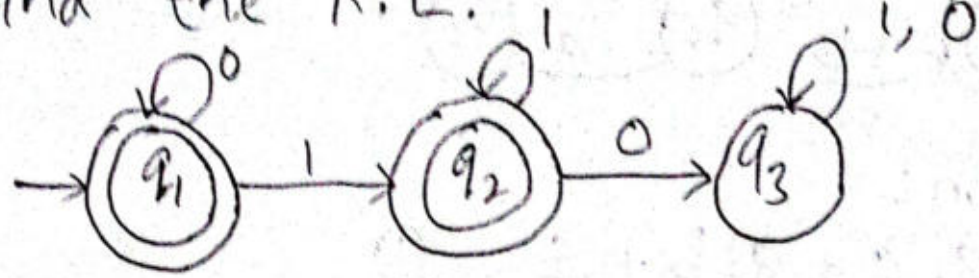
$q_1 = (a + a(b+aa)^* b)^*$

$q_2 = (a + a(b+aa)^* b)^* a (b+aa)^*$

$q_3 = (a + a(b+aa)^* b)^* a (b+aa)^*$

Assignment:

Find the R.E.



Solution:

$$q_1 = q_1 0 + \varepsilon \quad -(I)$$

$$q_2 = q_1 1 + q_2 1 \quad -(II)$$

$$q_3 = q_3 1 + q_3 0 + q_2 1 \quad -(III)$$

From eqn (I),

$$q_1 = q_1 0 + \varepsilon$$

or, $q_1 = \varepsilon 0^*$

$\therefore q_1 = 0^*$

From (ii),

$$q_2 = 0^* 1 + q_2 1$$

or $q_2 = q_2 1 + 0^* 1$

or $q_2 = 0^* 1 1^*$

Now, from (III),

$$q_3 = 0^* 1 1^* 1 (1+0)^*$$

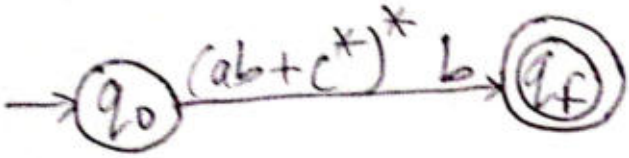
Assignment:

Q. Construct FA equivalent to RE

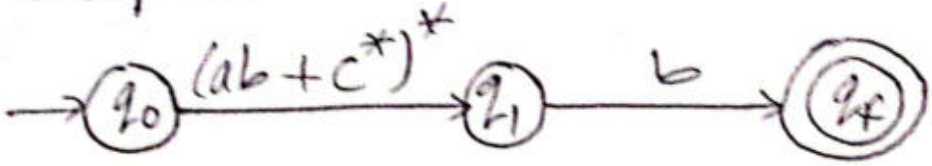
$(ab + c^*)^* b$

solution:

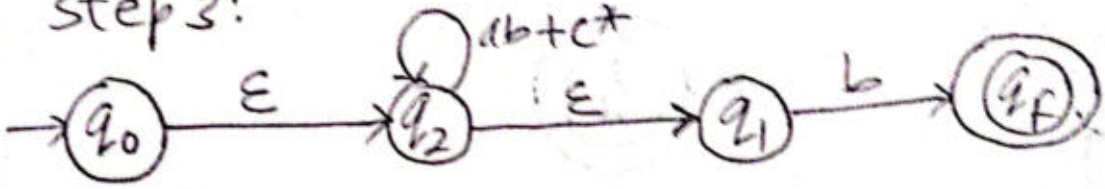
step 1:



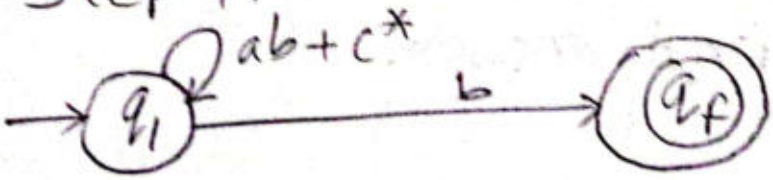
step 2:



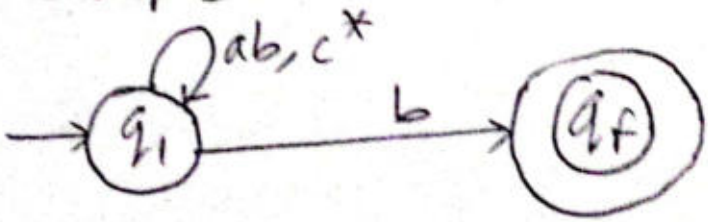
step 3:



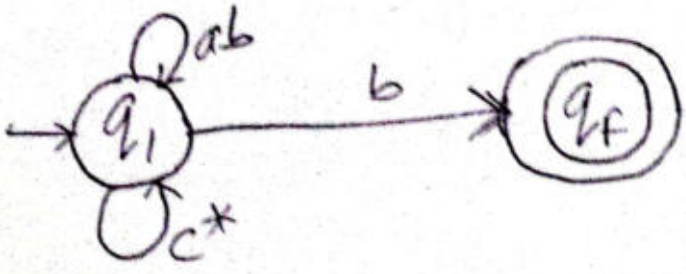
step 4:



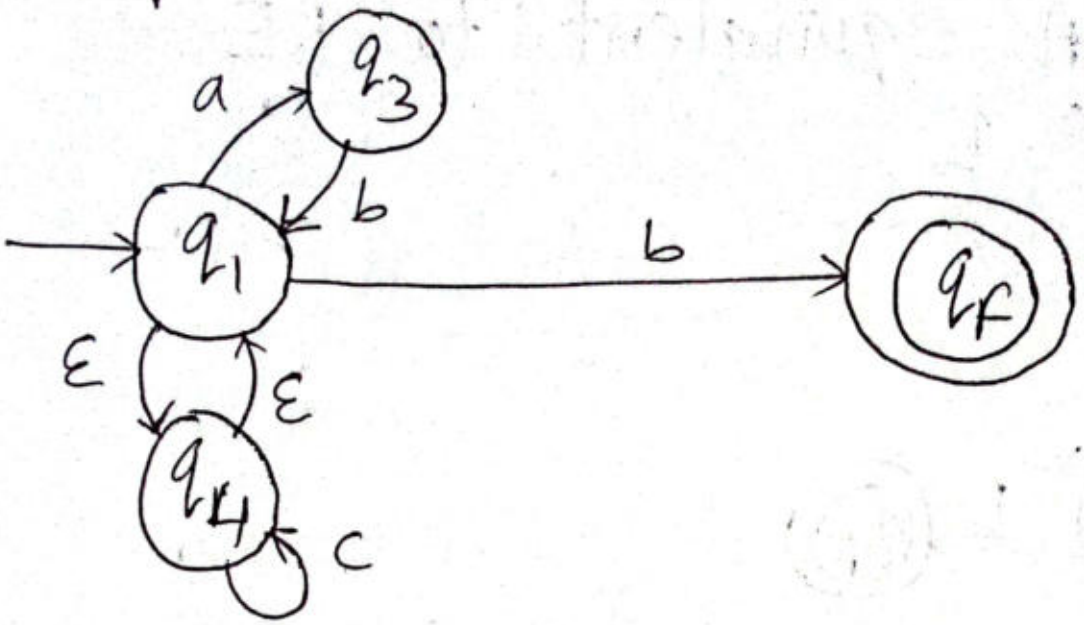
step 5:



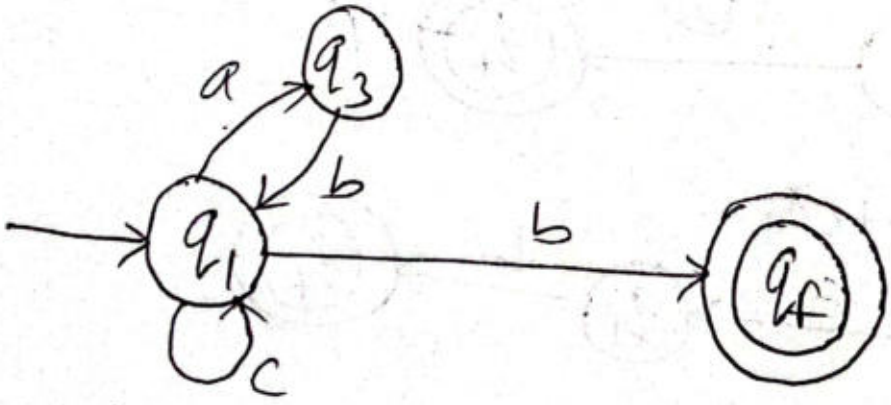
step 6:



Step 7:



Step 8:

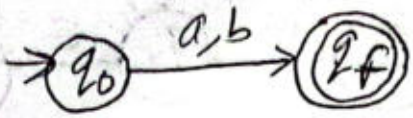


This is the final answer.

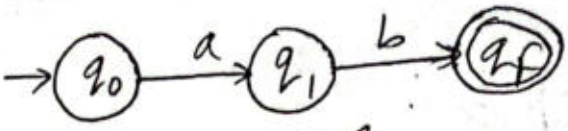
Construction of FA from R.E.

R.E. | FA

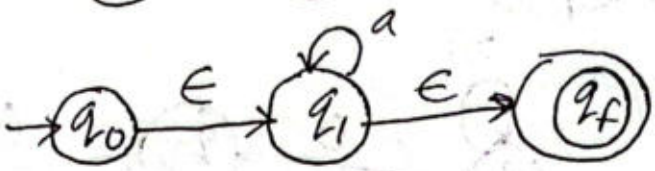
1. $a+b$



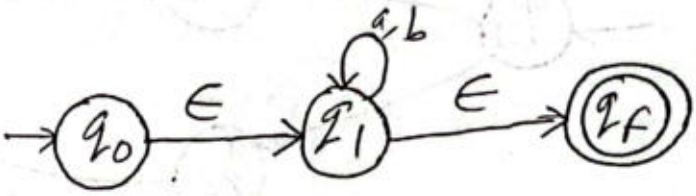
2. ab



3. a^*



4. $(a+b)^*$

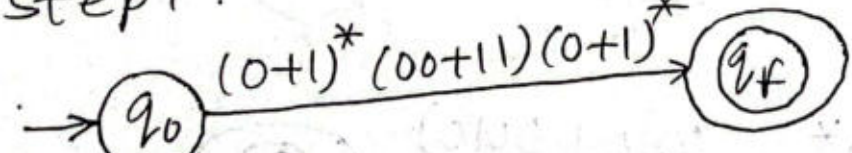


Q. Construct FA equivalent to R.E.

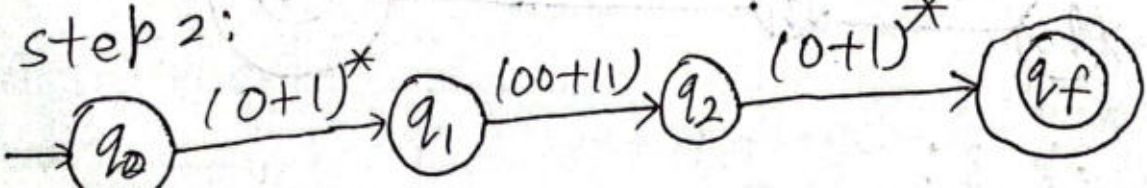
$(0+1)^*(00+11)(0+1)^*$

solution:

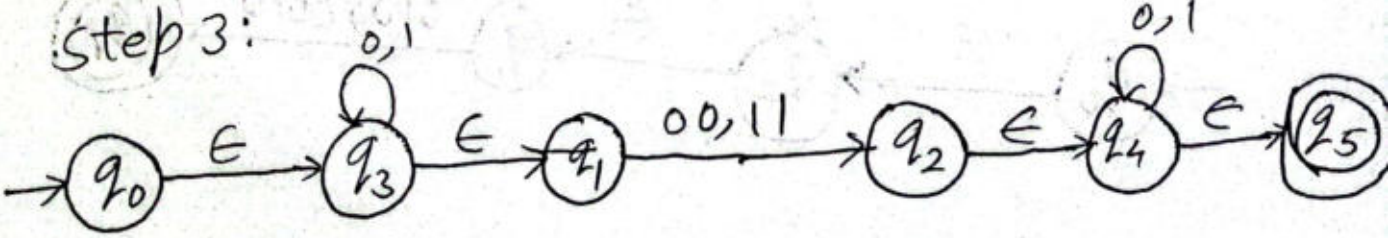
step 1:



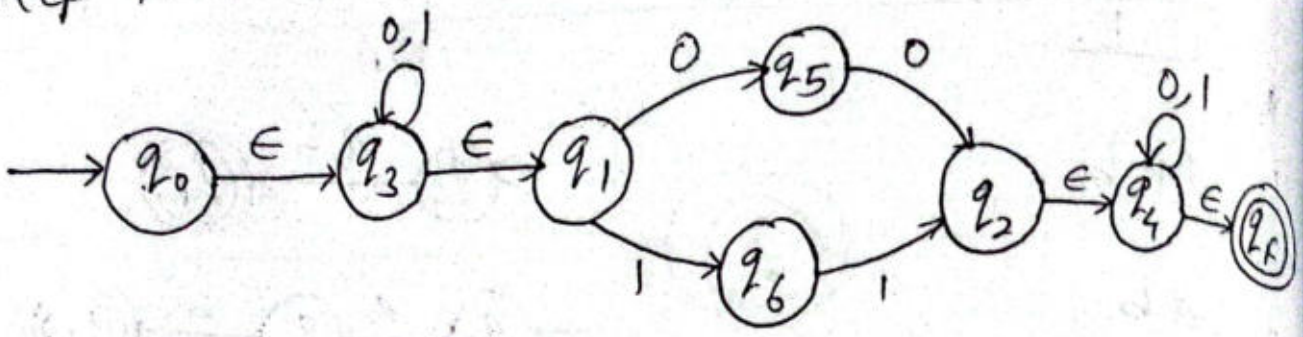
step 2:



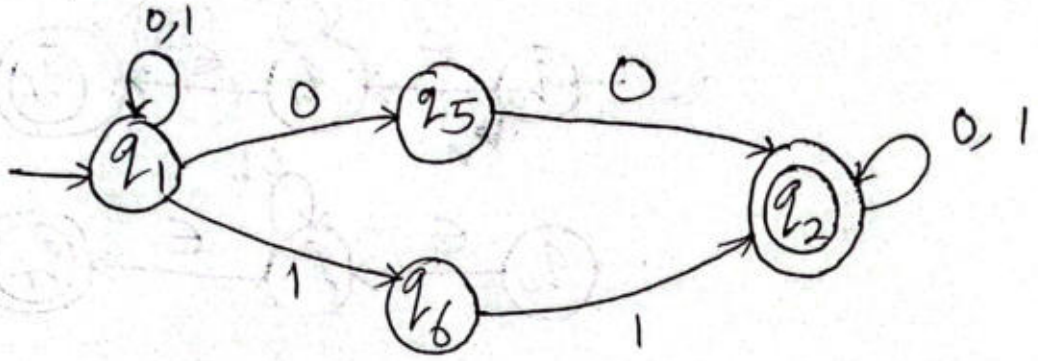
step 3:



step 4:



step 5:

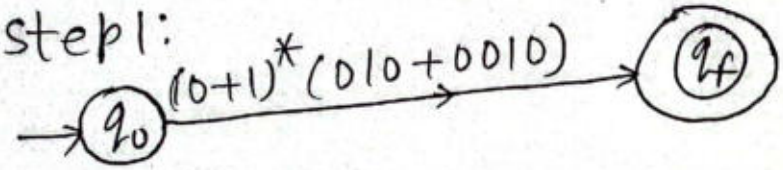


Q. Construct FA equivalent to R.E.

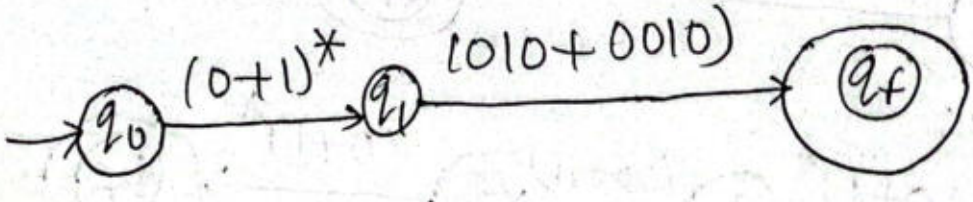
$(0+1)^*(010+0010)$

solution:

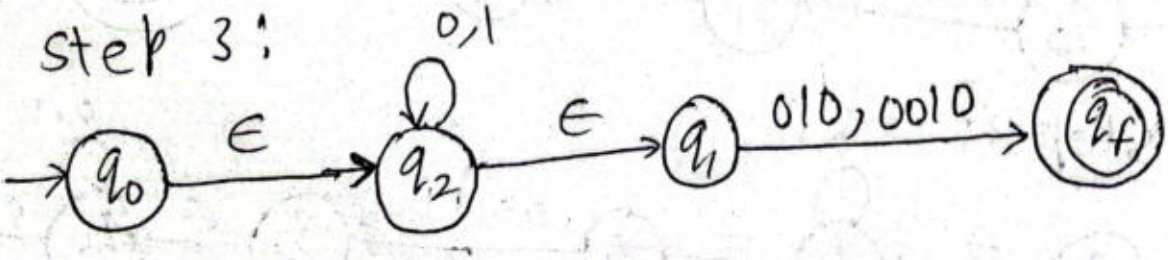
step 1:



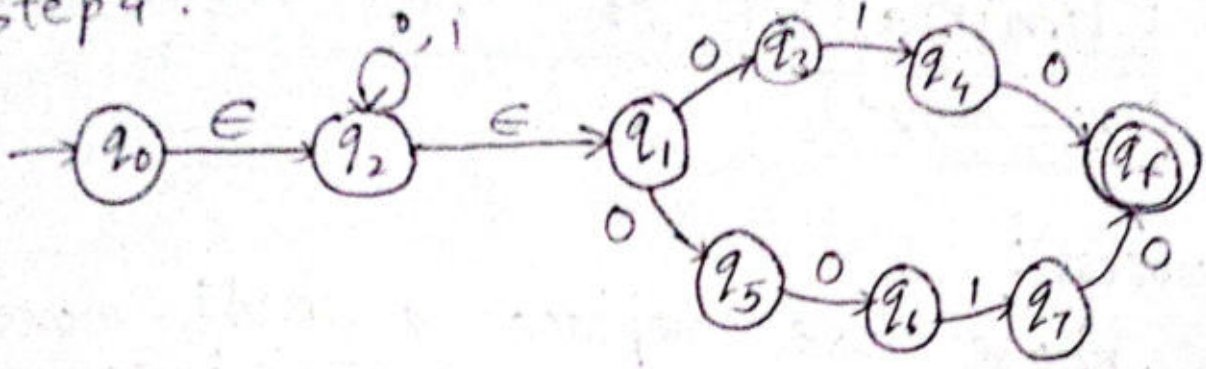
step 2:



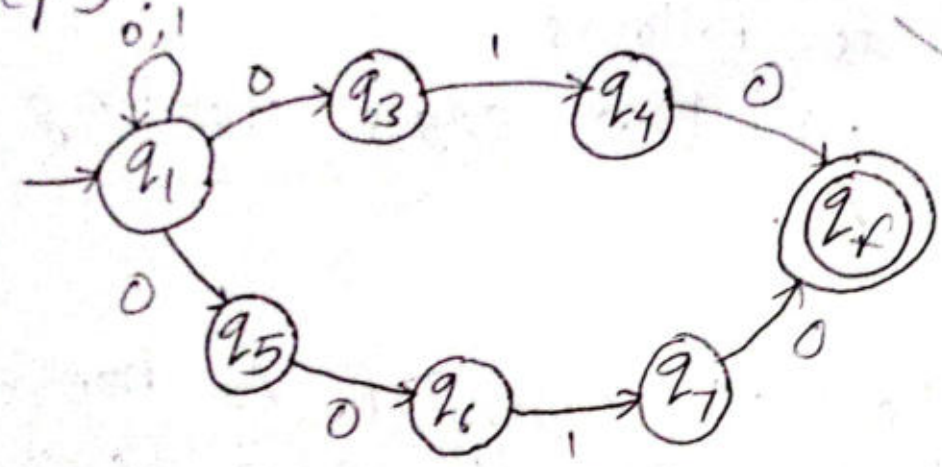
step 3:



step 4:

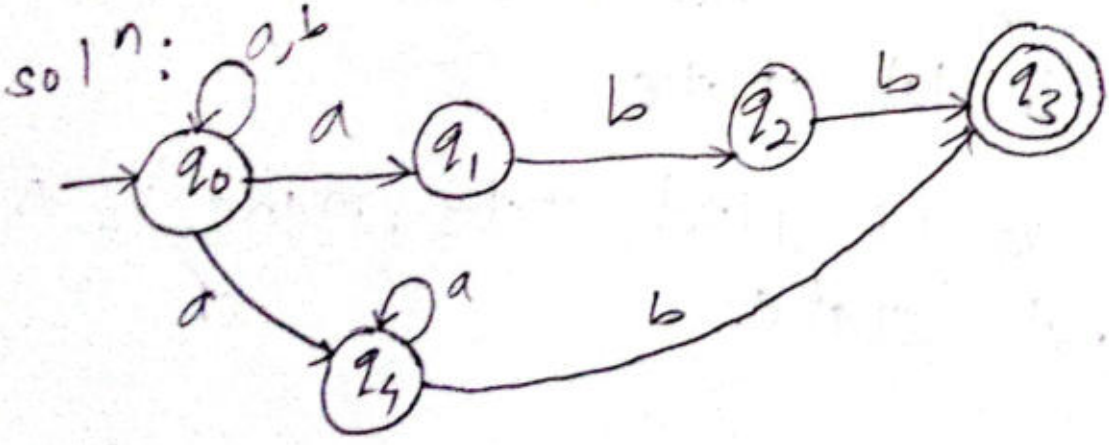


step 5:



Q. $(a/b)\ (abb/a^+b)$

soln:



Assignment:

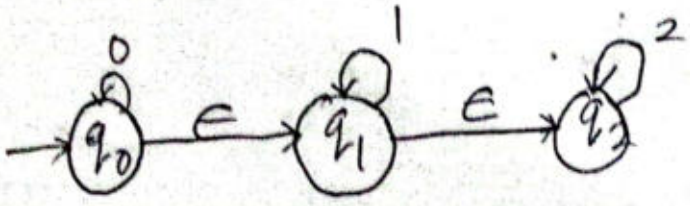
Q. Construct FA equivalent to RE $(ab + c^*)^*\ b$.

Elimination of Null move (ϵ-move)

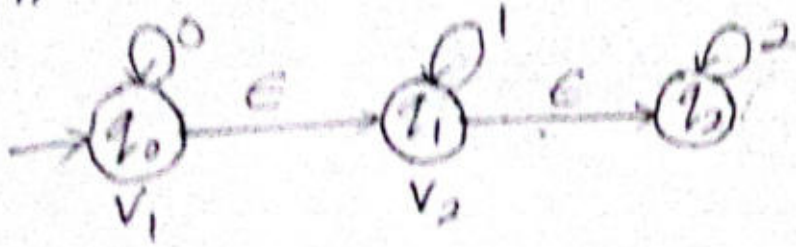
Process:
Suppose we replace a null move from a vector V_1 and V_2, then we process as follows:

1. Find all the edges starting from V_2.
2. Duplicate all edges of V_2 from V_1 without changing the edge level.
3. If V_1 is initial state, make V_2 as initial state.
4. If V_2 is final state, make V_1 as final state.

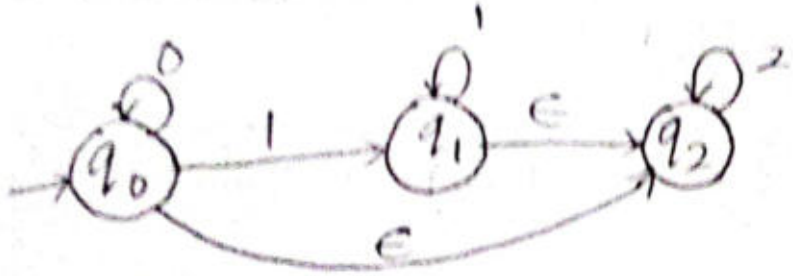
Q. Construct a FA without ϵ-move equivalent to given FA with ϵ-move



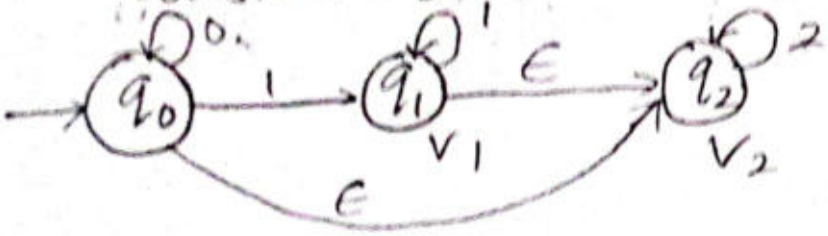
step 1:
Suppose ϵ move between q_0 and q_1.



step 2:

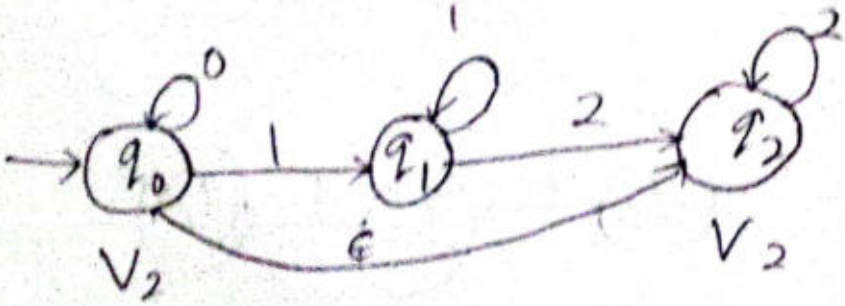


step 3:
considering ϵ move between q_1 and q_2.

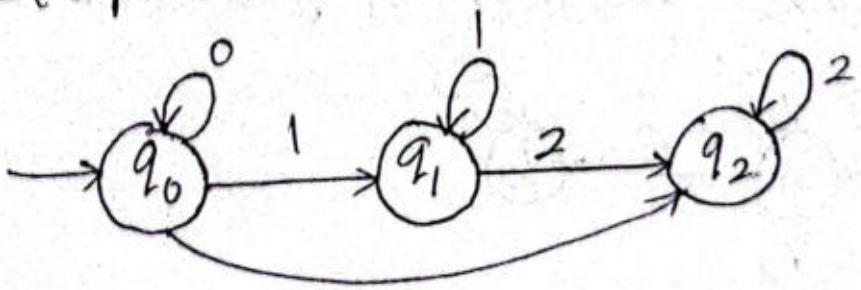


step 4:

step 5:

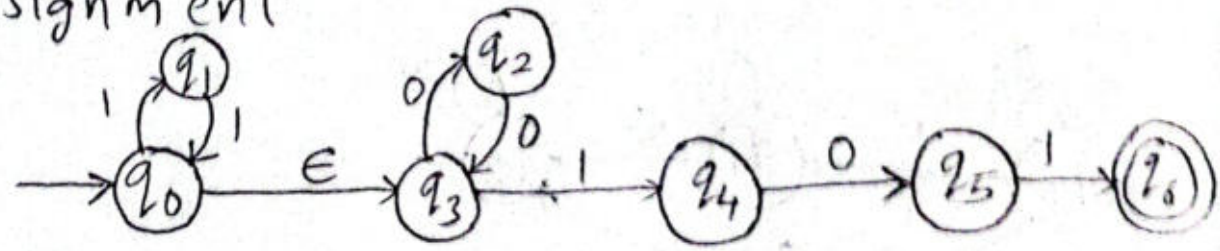


Step 6:

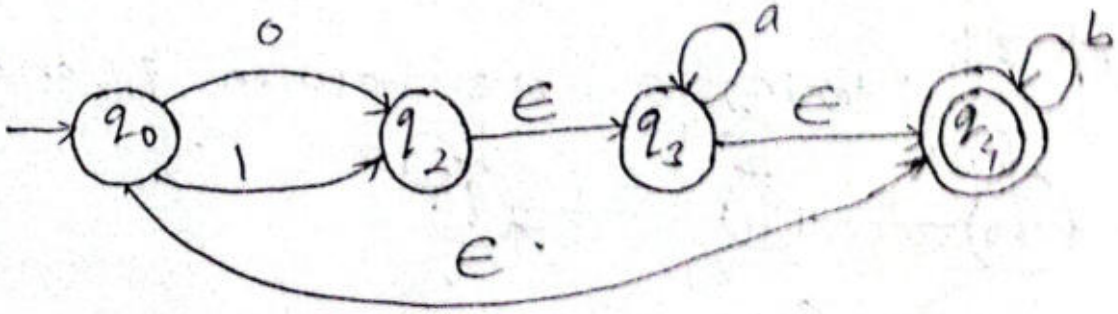


Assignment

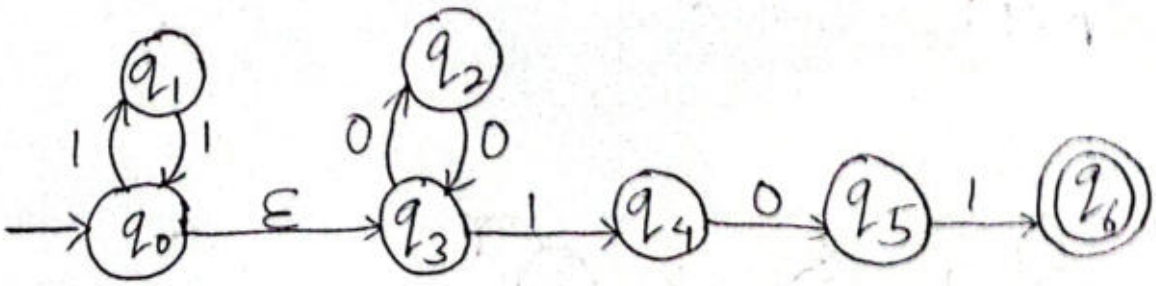
1.



2.



Q.

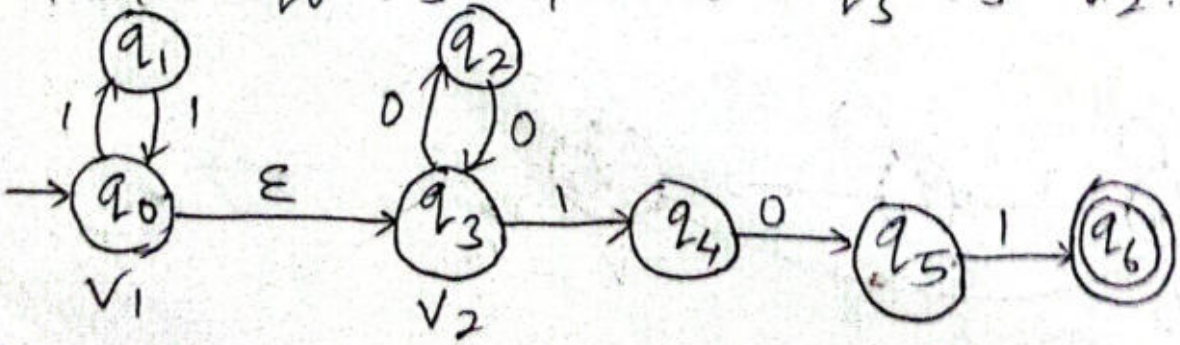


Solution:

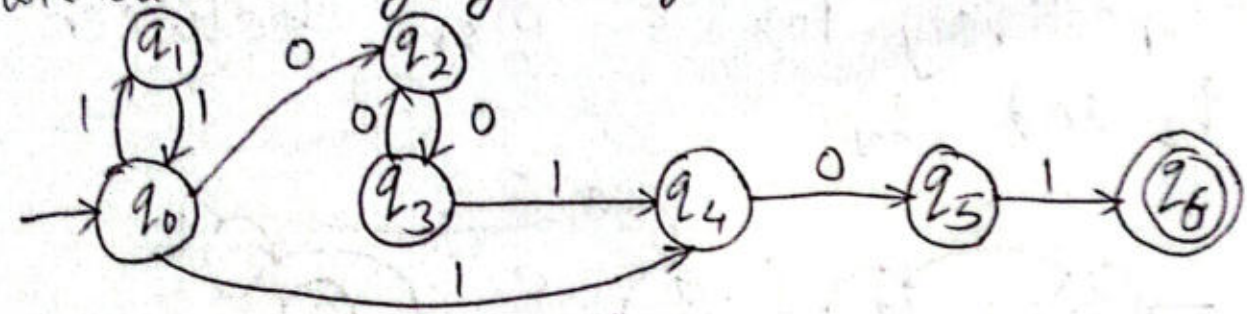
Step 1:

Suppose ϵ-move between q_0 and q_3.

Make q_0 as V_1 and q_3 as V_2.



step 2:
Duplicate all edges of V_2 from V_1 without changing edge level.



step 3:
Since, V_1 i.e. q_0 is initial state. make V_2 i.e. q_3 as initial state too. But since V_2 is not final state, so V_1 is also not final state.

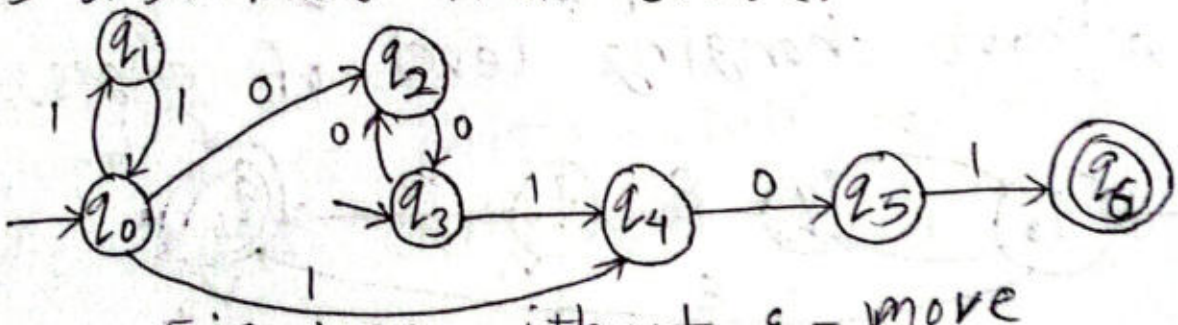
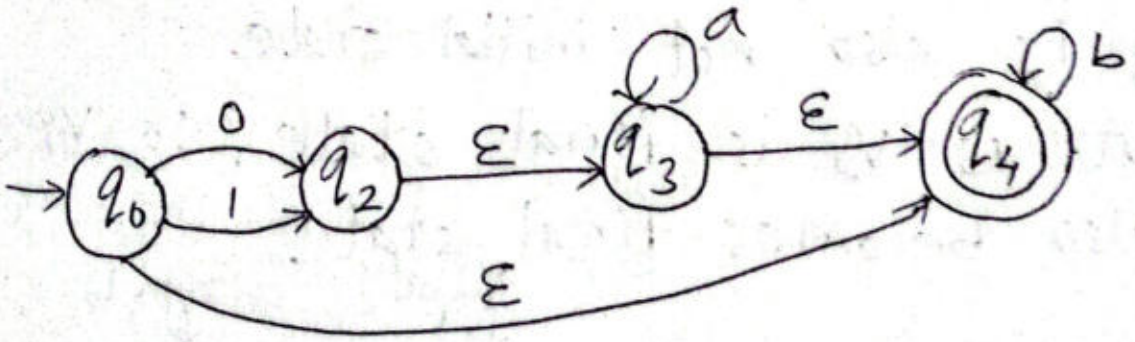


Fig: NFA without ε - move

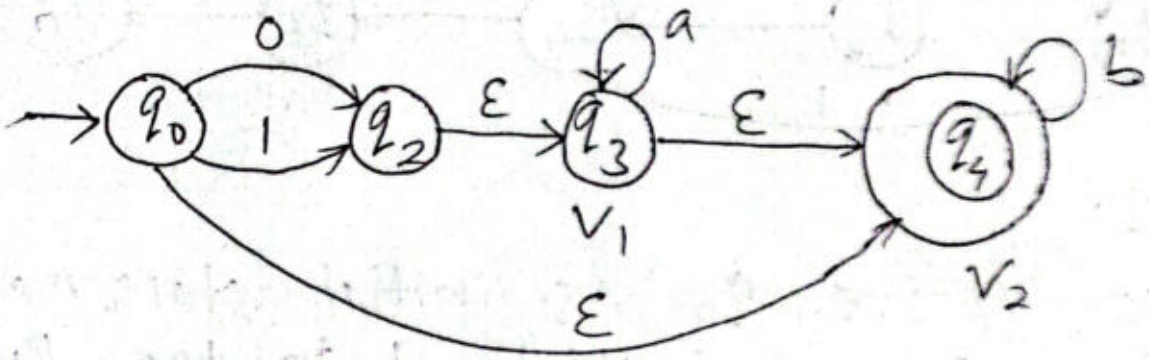
This is the final answer.

Q.

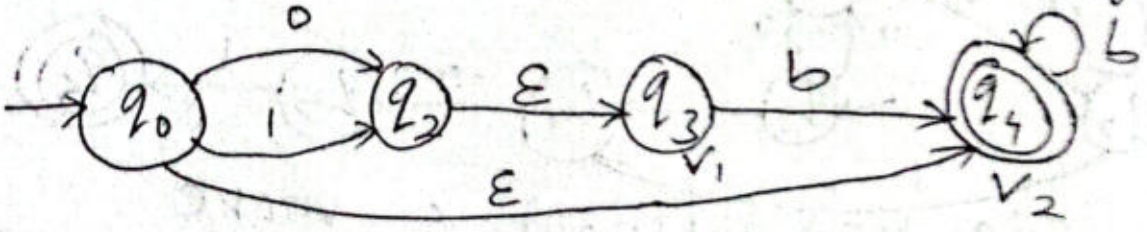


Solution:

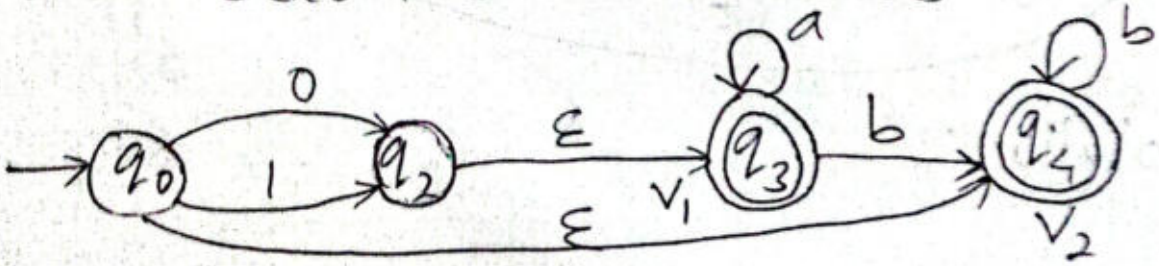
Step 1:
Make q_3 and q_4 as v_1 and v_2 respectively for ε- move between q_3 and q_4.



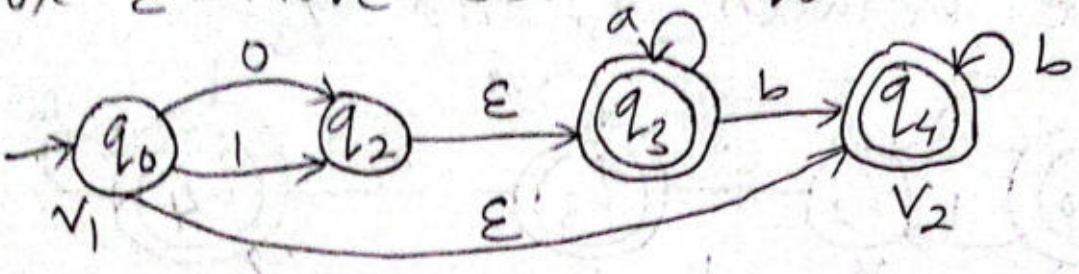
Step 2:
Duplicate all edges of v_2 from v_1 without changing level of edges.



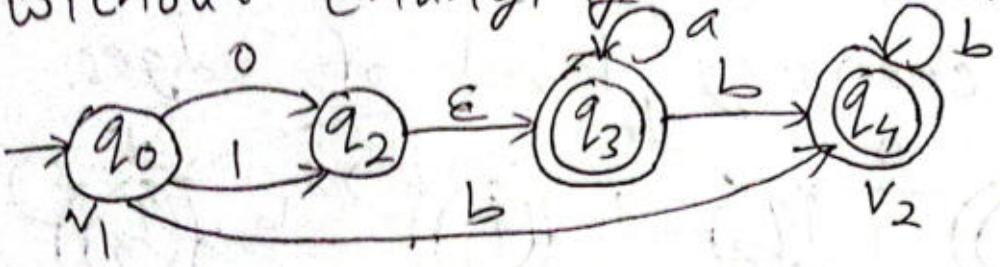
Step 3:
Since, v_1 is not initial state. so v_2 is also not initial state.
But as v_2 is final state. so, v_1 also becomes final state.



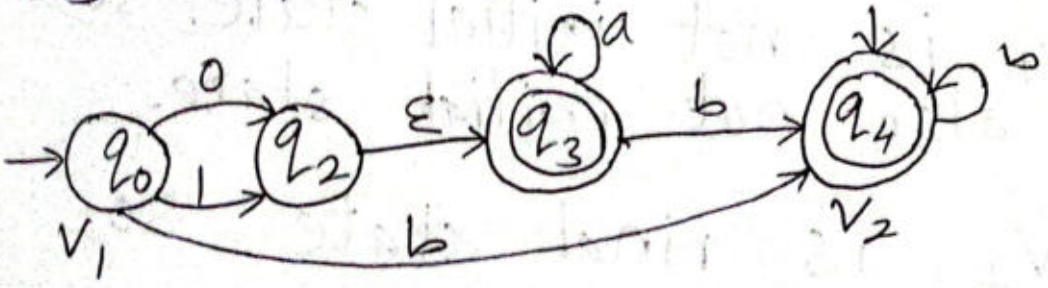
Step 4:
Make q_0 and q_4 as V_1 and V_2 respectively for ε-move between q_0 and q_4.



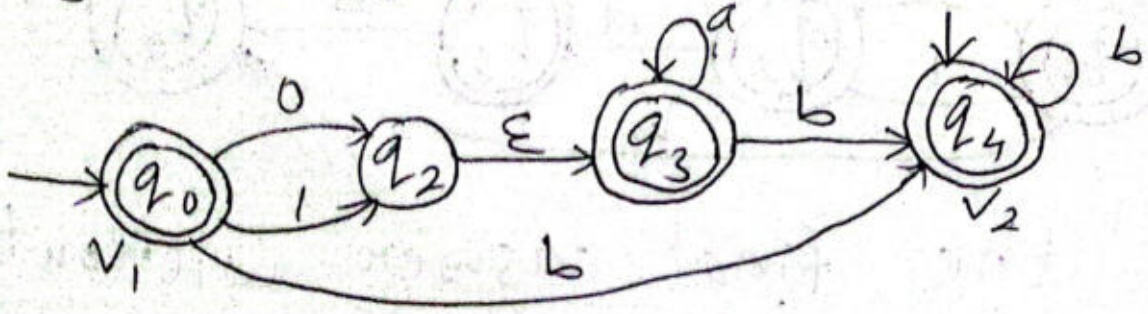
Step 5:
Duplicate all edges of V_2 from V_1 without changing level of edges.



Step 6:
Since, V_1 is initial state. So, make V_2 as initial state too.

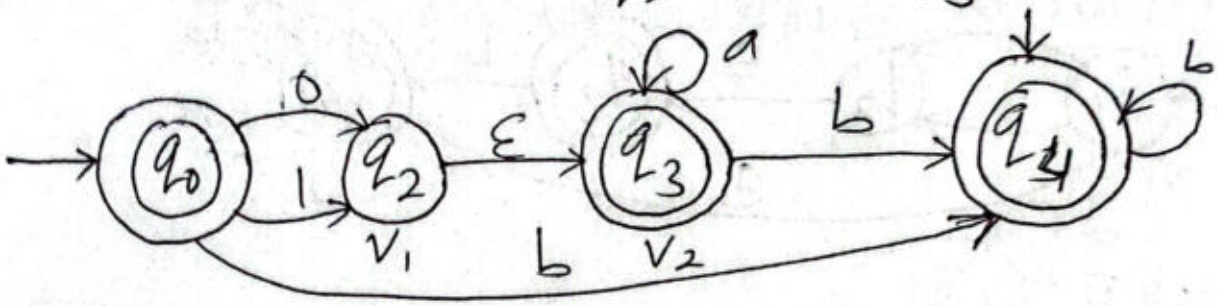


Step 7:
since, V_2 is final state make V_1 as final state too.



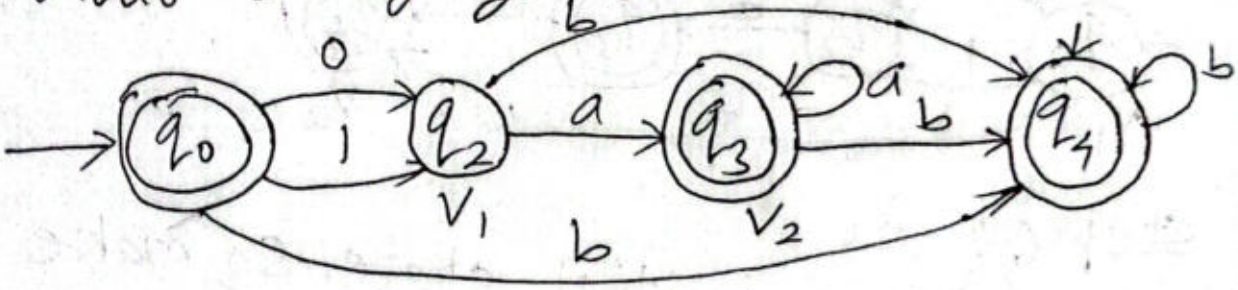
Step 8:

Make q_2 as v_1 and q_3 as v_2 for ε-move between q_2 and q_3.



Step 9:

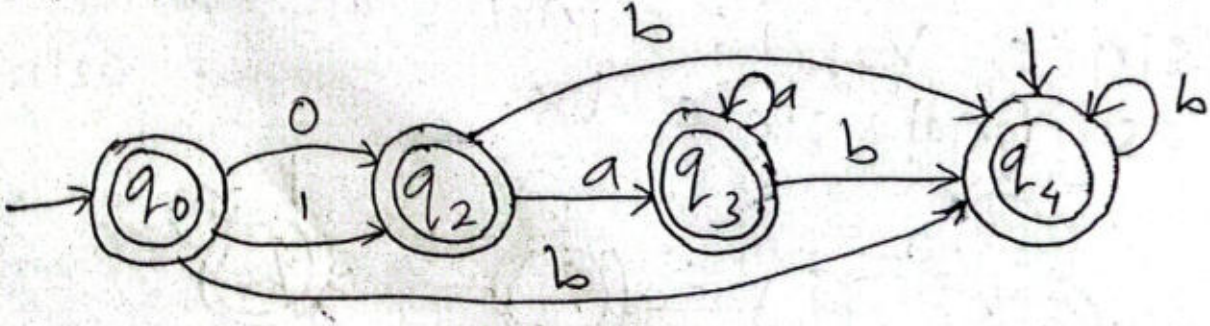
Duplicate all edges of v_2 from v_1 without changing level of edges.



Step 10:

Since v_1 is not initial state. So, v_2 is also not initial state.

since v_2 is final state. So, v_1 is also final state



So, the final answer without ε-move is,

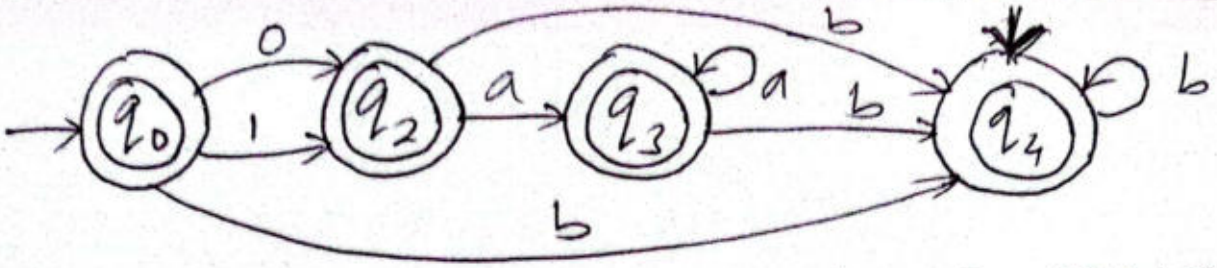


Fig: NFA without ε-move

Q. Reduce the given NFA with Null to without Null.

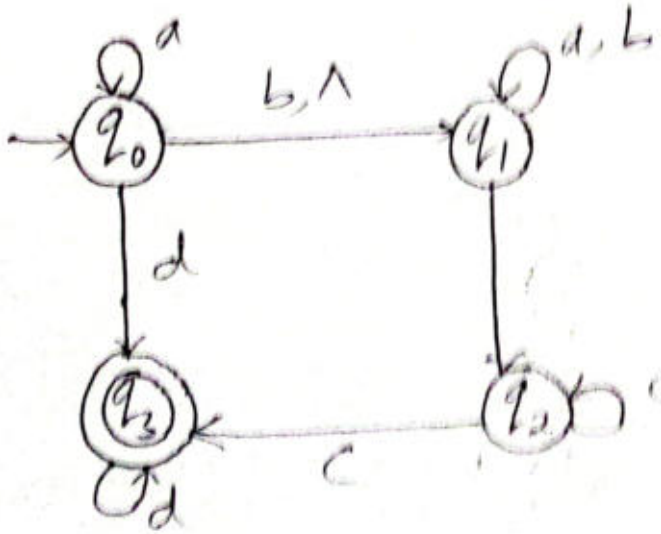
$M = \{ Q, \{a, b, c, d\}, \delta, q_0, \{q_3\} \}$

state Table

State	Input				
	a	b	c	d	$\wedge$
q_0	q_0	q_1	ϕ	q_3	q_3
q_1	q_1	q_1	ϕ	ϕ	q_2
q_2	ϕ	ϕ	$\{q_2, q_3\}$	ϕ	ϕ
q_3	ϕ	ϕ	ϕ	q_3	ϕ

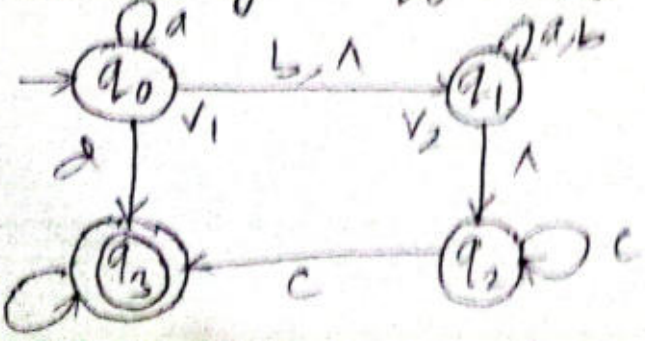
where, $Q = \{ q_0, q_1, q_2, q_3 \}$

Solution:



Step 1:

Considering q_0 and q_1,



Q. Reduce the given NFA with Null to without Null.

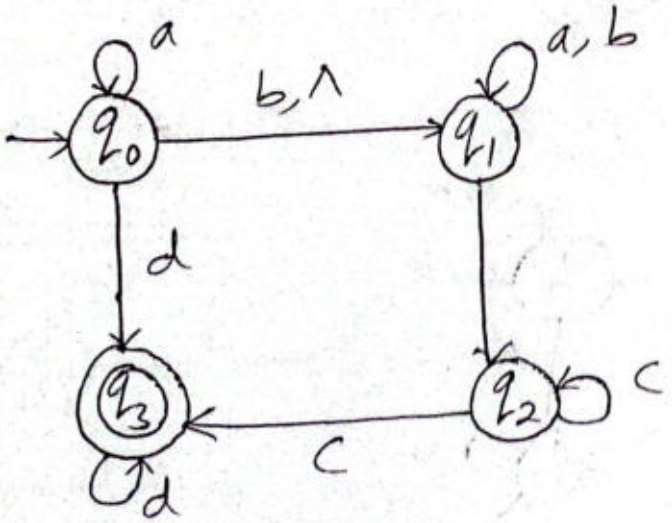
$M = \{ Q, \{a, b, c, d\}, \delta, q_0, \{q_3\} \}$

state Table

State	Input				
	a	b	c	d	^
q_0	q_0	q_1	$\emptyset$	q_3	q_3
q_1	q_1	q_1	$\emptyset$	$\emptyset$	q_2
q_2	$\emptyset$	$\emptyset$	$[q_2 q_3]$	$\emptyset$	$\emptyset$
q_3	$\emptyset$	$\emptyset$	$\emptyset$	q_3	$\emptyset$

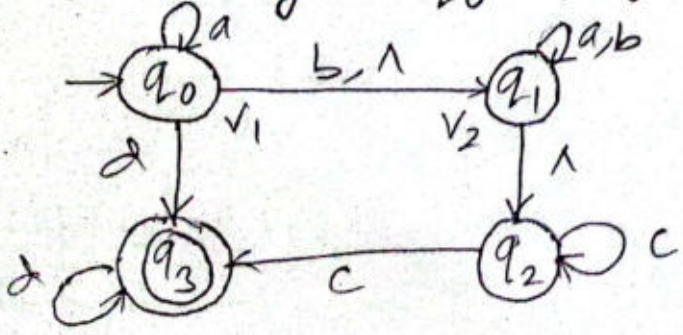
where, $Q = \{ q_0, q_1, q_2, q_3 \}$

solution:

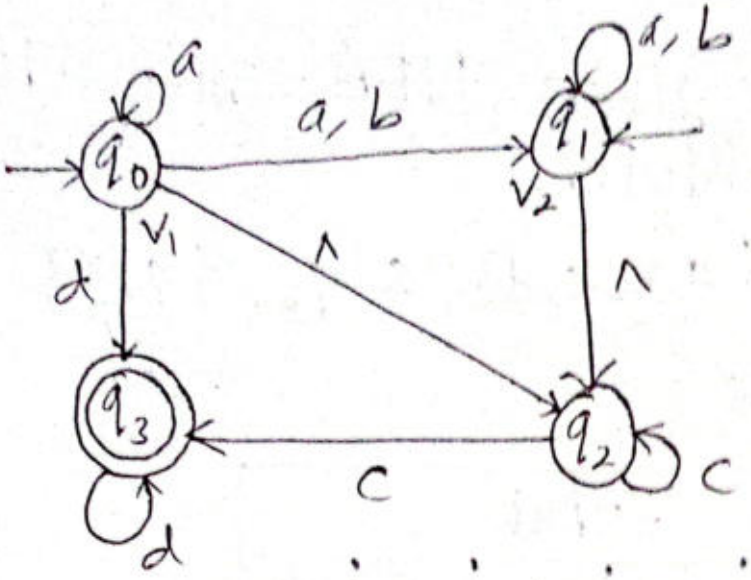


Step 1:

Considering q_0 and q_1,

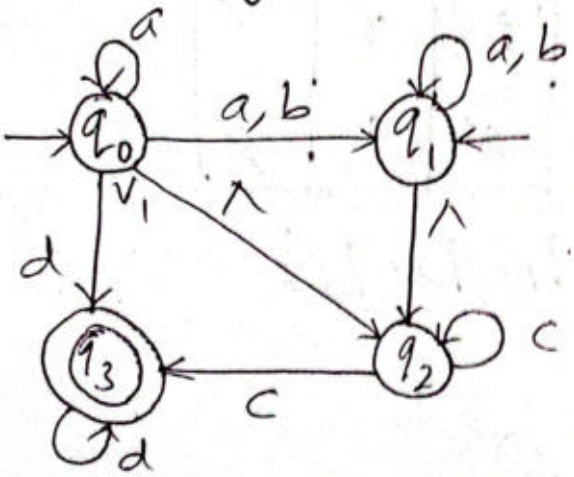


Step 2:

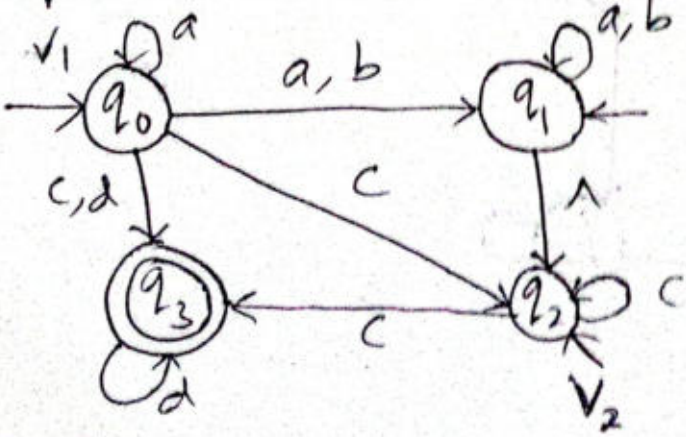


Step 3:

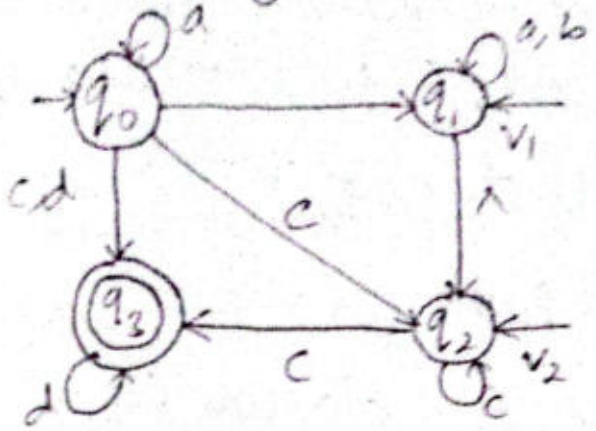
considering q_0 and q_2,



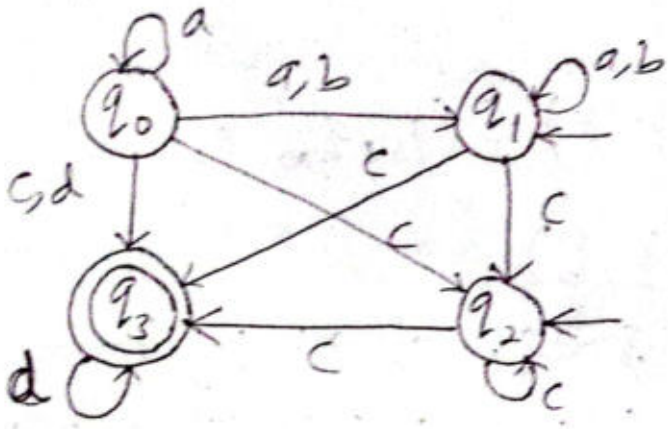
Step 4:



Step 5:
Considering q_1 and q_2.



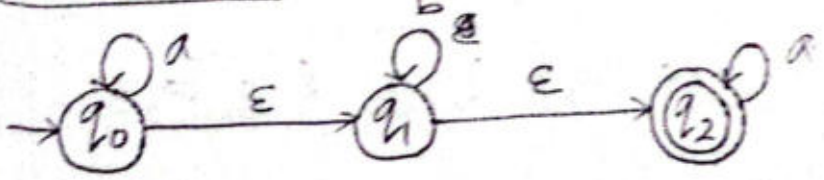
Step 6:



NFA with No Null (final Answer)

Conversion of NFA with ε to DFA

Eg:



Note:
ε-closure(∅) = ∅

Solution:

ε-closure (q_0)
$= \{q_0, q_1, q_2\}$

ε-closure (q_1)
$= \{q_1, q_2\}$

ε-closure (q_2)
$= \{q_2\}$

Now, DFA transition function δ'

(a) $\delta'(\{q_0, q_1, q_2\}, a) = \varepsilon\text{-closure}(\delta\{(q_0, q_1, q_2), a\})$

$= \varepsilon\text{-closure}(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a))$

$= \varepsilon\text{-closure}(q_0 \cup \emptyset \cup q_2)$

$= \varepsilon\text{-closure}(q_0 \cup q_2)$

$= \varepsilon\text{-closure}(q_0) \cup \varepsilon\text{-closure}(q_2)$

$= (q_0, q_1, q_2) \cup (q_2)$

$= (q_0 . q_1\ q_2)$

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

(b) $\delta'(\{q_0, q_1, q_2\}, b)$

$= \varepsilon\text{-closure}(\delta\{(q_0, q_1, q_2), b\})$

$= \varepsilon\text{-closure}(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b))$

$= \varepsilon\text{-closure}(\emptyset \cup q_1 \cup \emptyset)$

$= \varepsilon\text{-closure}(q_1)$

$= \{q_1, q_2\}$ [New state]

$\therefore \delta'(\{q_0, q_1, q_2\}, b) = \{q_1 . q_2\}$

(c) $\delta'(\{q_1, q_2\}, a)$

$= \varepsilon\text{-closure}(\delta\{(q_1, q_2), a\})$

$= \varepsilon\text{-closure}\{\delta(q_1, a) \cup \delta(q_2, a)\}$

$= \varepsilon\text{-closure}(\emptyset \cup q_2)$

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

$= q_2$ [New state]

(d) $\delta'(\{q_1, q_2\}, b)$

$= \varepsilon$ - closure $(\delta\{(q_1, q_2), b\})$

$= \varepsilon$ - closure $(\delta(q_1, b) \cup \delta(q_2, b))$

$= \varepsilon$ - closure $(q_1 \cup \emptyset)$

$= \varepsilon$ - closure (q_1)

$= \{q_1, q_2\}$.

(e) $\delta'(\{q_2\}, a)$

$= \varepsilon$ - closure $(\delta\{(q_2), a\})$

$= \varepsilon$ - closure (q_2)

$= q_2$

(f) $\delta'(\{q_2\}, b)$

$= \varepsilon$ - closure $(\delta\{(q_2), b\})$

$= \varepsilon$ - closure $(\emptyset)$

$= \emptyset$

DFA state Table

State	Input	
	a	b
$\rightarrow q_0\, q_1\, q_2$	self	$q_1\, q_2$
$q_1\, q_2$	q_2	self
q_2	q_2	$\emptyset$

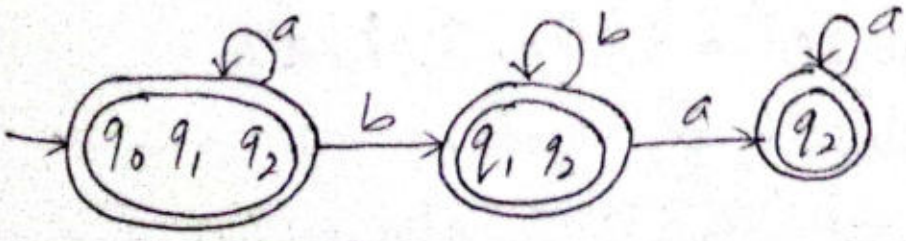


Fig: state Diagram

Q. Transition table for NFA, δ is

State	Input		
	a	b	ε
→ Q_0	Q_0	$\{Q_1, Q_2\}$	Q_1
Q_1	Q_4	$\{Q_0, Q_4\}$	∅
Q_2	Q_3	∅	∅
(Q_3)	∅	∅	Q_4
(Q_4)	Q_3	∅	∅

solution:

ε - closure (Q_0) = $\{Q_0, Q_1\}$

ε - closure (Q_1) = $\{Q_1\}$

ε - closure (Q_2) = $\{Q_2\}$

ε - closure (Q_3) = $\{Q_3, Q_4\}$

ε - closure (Q_4) = $\{Q_4\}$

Now, transition function δ' for DFA,

① $\delta'(\{Q_0, Q_1\}, a)$

Note: Q_0 is start state of NFA, so we take ε-closure of Q_0 as starting state of DFA.

= ε - closure ($\delta\{(Q_0, Q_1), a\}$)

= ε - closure ($Q_0 \cup Q_4$)

= ε - closure (Q_0) ∪ ε - closure (Q_4)

= $\{Q_0, Q_1\} \cup \{Q_4\}$

= $\{Q_0, Q_1, Q_4\}$ [new state]

③ $\delta'(\{q_0, q_1\}, b)$

$= \varepsilon$-closure $(\delta'(\{q_0, q_1\}, b))$

$= \varepsilon$-closure $(\delta(q_0, b) \cup \delta(q_1, b))$

$= \varepsilon$-closure $(\{q_0, q_2\} \cup \{q_2, q_4\})$

$= \varepsilon$-closure $(q_0) \cup \varepsilon$-closure (q_2)
$\cup \varepsilon$-closure $(q_2) \cup \varepsilon$-closure q_4.

$= \{q_0, q_1\} \cup \{q_2\} \cup \{q_2\} \cup \{q_4\}$

$= \{q_0, q_1, q_2, q_4\}$ [new state]

④ $\delta'(\{q_0, q_1, q_4\}, a)$

$= \varepsilon$-closure $(\delta'(\{q_0, q_1, q_4\}, a))$

$= \varepsilon$-closure $(q_0 \cup q_4 \cup q_3)$

$= \{q_0, q_1\} \cup \{q_4\} \cup \{q_3, q_4\}$

$= \{q_0, q_1, q_2, q_4\}$ [new state]

⑤ $\delta'(\{q_0, q_1, q_4\}, b)$

$= \varepsilon$-closure $(\delta'(\{q_0, q_1, q_4\}, b))$

$= \varepsilon$-closure $(\{q_0, q_2\} \cup \{q_2, q_4\} \cup \emptyset)$

$= \{q_0\ q_1\ q_2\ q_4\}$

⑥ $\delta'(\{q_0, q_1, q_2, q_4\}, a)$

$= \varepsilon$-closure $(\delta'(\{q_0, q_1, q_2, q_4\}, a))$

$= \varepsilon$-closure $(q_0 \cup q_4 \cup q_3 \cup q_3)$

$= \{q_0, q_1\} \cup \{q_4\} \cup \{q_3, q_4\} \cup \{q_3, q_4\}$

$= \{q_0, q_1, q_3, q_4\}$

⑥ $\delta'(\{Q_0, Q_1, Q_2, Q_4\}, b)$

$= \varepsilon\text{-closure}(\delta\{(Q_0, Q_1, Q_2, Q_4), b\})$

$= \varepsilon\text{-closure}(\{Q_0, Q_2\} \cup \{Q_4\} \cup \emptyset \cup \emptyset)$

$= \{Q_0\ Q_1\ Q_2\ Q_4\}$

⑦ $\delta'(\{Q_0, Q_1, Q_3, Q_4\}, a)$

$= \varepsilon\text{-closure}(\delta\{(Q_0, Q_1, Q_3, Q_4), a\})$

$= \varepsilon\text{-closure}(\{Q_0\} \cup \{Q_4\} \cup \emptyset \cup Q_3)$

$= \{Q_0\ Q_1\ Q_3\ Q_4\}$

⑧ $\delta'(\{Q_0\ Q_1\ Q_3\ Q_4\}, b)$

$= \varepsilon\text{-closure}(\delta\{(Q_0, Q_1, Q_3, Q_4), b\})$

$= \varepsilon\text{-closure}(\{Q_0, Q_2\} \cup \{Q_4\} \cup \emptyset \cup \emptyset)$

$= \{Q_0\ Q_1\ Q_2\ Q_4\}$

Transition table for DFA, δ' is

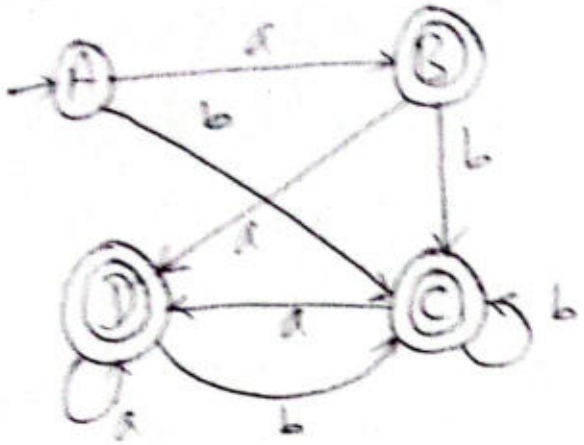
state	Input	
	a	b
$\rightarrow\{Q_0 Q_1\}$	$\{Q_0 Q_1 Q_4\}$	$\{Q_0 Q_1 Q_2 Q_4\}$
$*\{Q_0 Q_1 Q_4\}$	$\{Q_0 Q_1 Q_3 Q_4\}$	$\{Q_0 Q_1 Q_2 Q_4\}$
$*\{Q_0 Q_1 Q_2 Q_4\}$	$\{Q_0 Q_1 Q_3 Q_4\}$	$\{Q_0 Q_1 Q_2 Q_4\}$
$*\{Q_0 Q_1 Q_3 Q_4\}$	$\{Q_0 Q_1 Q_3 Q_4\}$	$\{Q_0 Q_1 Q_2 Q_4\}$

Let $\{Q_0 Q_1\} = A$, $\{Q_0\ Q_1\ Q_4\} = B$,

$\{Q_0 Q_1 Q_2\ Q_4\} = C$ and

$\{Q_0 Q_1 Q_3 Q_4\} = D$

) FA is



Imp. The Pumping lemma for regular sets

repitition

→ to prove not regular

Theorem:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automata with 'n' state that accept regular language 'L'. Let w be the string such that $w \in L$. and $|w| \geq m$. If $m \geq n$, then there exists x, y, z (substrings) such that

i) $w = xyz$

ii) $y \neq \wedge$ (y cannot be null).

iii) xy^iz ~~bbb~~ $\in L$ for each $i \geq 0$

Assignment

* Properties of regular expression or language
1. Commutative
2. Associativity
3. Identities
4. Annihilator
5. Idempotent law
6. Distributive law
7. Law of closure

Algebric Rules/laws for regular expression

1. Commutativity
- Commutative law of operator means we can switch the order of its operands and get the same result.
The union of regular expressions is commutative but concatenation of regular expression is not commutative. i.e.
If r and s are regular expressions representing like languages $L(r)$ and $L(s)$, then $r+s = s+r$ i.e. $r \cup s = s \cup r$.
but $r.s \neq s.r$

2. Associativity
- The union as well as concatenation of regular expressions are assoeiative i.e if t, r, s are regular expressions representing regular languages

L(t), L(r) and L(s), then

$t+(r+s) = (t+r)+s$

and, $t.(r.s) = (t.r).s$

3. Distributive Law

- For any regular expression r, s, t representing regular language L(r), L(s) and L(t) then

$r(s+t) = rs+rt$ ------ left distribution

$(s+t)r = sr+tr$ ------ right distribution

4. Identity law

- $\emptyset$ is identity for union i.e. for any regular expression representing regular expression L(r).

$r+\emptyset = \emptyset+r = r$ i.e. $\emptyset \cup r = r$.

ε is identity for concatenation i.e.

$\varepsilon.r = r = r.\varepsilon$

5. Annihilator

- An annihilator for an operator is a value such that when the operator is applied to the annihilator and some other value, the result is annihilator, $\emptyset$ is annihilator for concatenation.

i.e. $\emptyset.r = r.\emptyset = \emptyset$

6. Idempotent law of union

- For any regular expression r representing the regular language L(r), $r+r = r$. This is idempotent law of union.

7. Law of closure

- For any regular expression r, representing the regular language $L(r)$, then
  - $(r^*)^* = r^*$
  - Closure of $\phi = \phi^* = \varepsilon$
  - Closure of $\varepsilon = \varepsilon^* = \varepsilon$
  - Positive closure of r, $r^+ = rr^*$

Proof:

Let $w = a_1, a_2, \ldots, a_m$, $m \geq n$

$\delta(q_0, a_1 a_2 \ldots a_m) = q_i$, for $i = 1, 2, 3, \ldots m$

$Q = (q_0, q_1, q_2, \ldots q_m)$

Here, Q is the sequence of states in the path with path value $w = a_1 a_2 \ldots a_m$

Now, input string w can be decomposed into substring as,

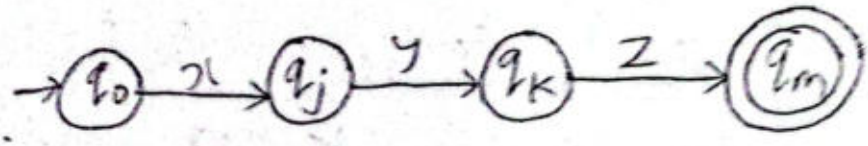
$x = a_1 a_2 \ldots a_j$

$y = a_{j+1} \ldots a_k$

$z = a_{k+1} \ldots a_m$

Since we know that for any regular expression, there exists a path from initial state to final state with path value in the given regular expression.

For w, the finite automata can be constructed as shown in figure

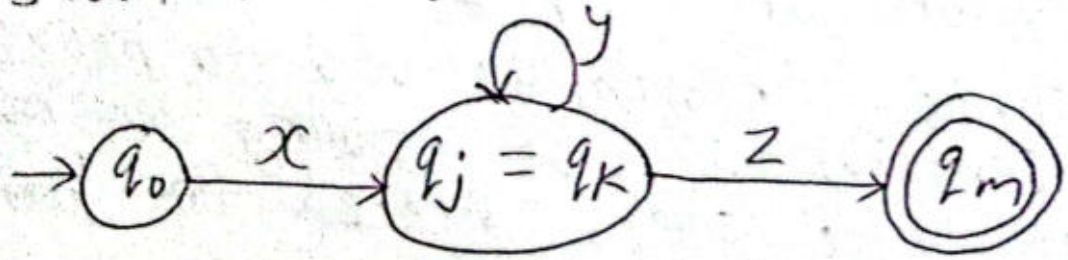


Here, length of string, $|w| = 3$ i.e. $m = 3$,
no. of states, $n = 4$

On reading, the string xyz, a new state are added on existing states, But by definition of pumping lemma, we have,

$|w| \geq m$ and $m \geq n$.

Thus, by using pigeonhole principle, There must be at least two state in Q. As there are only n distinct state defined, but on applying the input string, the state becomes n+1. Thus among various pair of repeated states, we take first as q_j and q_k for merging and hence the path with w in the transition diagram of FA is shown in figure below.



Here, length of string, $|w| = 3$ i.e. $m = 3$.
no. of ~~state~~ state, $n = 3$
i.e. $m \geq n$ (True)

Since,
$$w = xyz$$
$$= a_1 a_2 \ldots a_j a_{j+1} \ldots a_k a_{k+1} \ldots a_n$$
so, we can write $0 \leq j \leq k \leq n$ and this implies that $|xy| \leq n$

Here, $xy^iz \in L$ for each $i \geq 0$.

* When, $i = 0$, $w = xz$ (accepted by FA)
* When, $i = 1$, $w = xyz$ (accepted by FA)
* When, $i = 2$, $w = xy^2z$
$= xyyz$ (accepted by FA)

And so on.

Hence, we can conclude that this condition i.e $xy^iz \in L$ for each $i \geq 0$ is valid for all regular expression.

Thus, the string of y can pump in several number of time to generate large strings. Hence, $xy^iz \in L$. As every state in Q is obtained by applying an input symbol, $y \neq \varepsilon$.

Application of Pumping Lemma for regular sets

This theorem is used to prove that certain set of string are not regular. The steps needed for proving that a given set is not regular are:

step1: Assume that L is regular. Let n be no. of states in FA.

step2: Choose a string w such that $|w| \geq n$. Use pumping lemma to write, $w = xyz$, $|xy| \leq n$ and $|y| > 0$ or, $|y| \geq 1$.

step3: Find a suitable i for which $xy^iz \notin L$.

This contradicts our assumption. So, L is not regular.

Q. Show that $L = \{ww \mid w \in (a,b)^*\}$ is not regular.

sol^n:

step 1: Assume L is regular.

step 2: Let $ww = \overbrace{a^n b}^{w} \overbrace{a^n b}^{w}$. Then,

By pumping lemma, we can write

$ww = xyz$, $|xy| \leq n$, $|y| > 0$.

suppose, $n = 7$.

$ww = a^7 b a^7 b$

$= aaaaaaa b aaaaaaa b$

Case 1: y has only parts of a.

$\underbrace{aa}_{x} \underbrace{aaaaa}_{y} \underbrace{b\ aaaaaaab}_{z}$

Case 2: y has only b's

$\underbrace{aaaaaaa}_{x} \underbrace{b}_{y} \underbrace{aaaaaaab}_{z}$

Case 3: y has both a and b.

$\underbrace{aaaaaaa}_{x}\ \underbrace{b}_{y}\ \underbrace{aaaaaaab}_{z}$

step 3: We want to find i such that $xy^iz \notin L$.

Take $i = 2$,

In case 1, xy^2z

i.e. $xy^2z = xyyz$

$= aaaaaaaaaaab\ aaaaaaab$

$= a^{12}\ b\ a^7\ b$

This is not in pattern.

Thus, $xy^iz \notin L$ for $i = 2$.

Also, in case 1, $|xy| = 7$, $n = 7$, $|xy| \leq n$ (True)

Take $i = 3$,

In case 2, xy^3z

$= xyyyyz$

$= aaaaaaaa\ b\ bb\ aaaaaaaab$

$= a^7\ b^3\ a^7\ b$

This is not in pattern.

Thus, $xy^iz \notin L$.

Also,

In case 2,

$|xy| = 8$, $n = 7$, $|xy| \leq n$ (false)

Show that $L = \{0^n 1^n \mid n \geq 1\}$ is not regular.

solution:

step 1: Assume L is regular.

step 2:

Let $w = 0^n 1^n$.

Then, by pumping lemma, we can write

$w = xyz$ with $|xy| \leq n, |y| > 0$

suppose, $n = 5$,

$w = 0^5 1^5$

$= 0000011111$

Case I: y has 0's parts.

i.e. $\underbrace{0}_{x}\underbrace{0000}_{y}\underbrace{11111}_{z}$

Case 2: y has only 1's.

i.e. $\underbrace{00000}_{x}\underbrace{1}_{y}\underbrace{1111}_{z}$

Case 3: y has both 0's and 1's.

i.e. $\underbrace{0000}_{x}\underbrace{01}_{y}\underbrace{1111}_{z}$

step 3: We want to find i, such that $xy^iz \notin L$.

Take $i = 2$,

In case 1, xy^2z

$= xyyz$

$= 00000000011111$

$= 0^8 1^5$

Here, no. of 0's and 1's are not equal i.e. not in pattern.
So, $xy^iz \notin L$ for $i=2$.

Also, $|xy| = 5 \leq 5$ (true)

~~Take $i=3$,~~

~~In case 2, xy^3z~~

~~$= xyyyz$~~

~~$= 00000111111111$~~

~~$= 0^5 1^9$~~

~~Here not in pattern.~~

~~So, $xy^iz \notin L$ for $i=2$.~~

Take $i=2$,

In case 2, xy^2z

$= xyyz$

$= 000001111111$

$= 0^5 1^7$

not in pattern.

So, $xy^iz \notin L$ for $i=2$.

$|xy| = 7, n = 5$ (false)

Take $i=2$,

In case 3; xy^2z

$= xyyz$

$= 00000101111$

$= 0^6 1^6$

not in pattern. So, $xy^iz \notin L$ for $i=3$

$|xy| = 6, n = 5$ (false)

Thus, in all cases, we get contradiction.
so, given language
$L = \{ 0^n 1^n \mid n \geq 1 \}$ is not regular.

Assignment

Q. show that $L = \{ a^p \mid p \text{ is prime} \}$ is not regular.

solution:

step 1:
Assume 'L' is a regular language.

step 2:
Let $w = a^p$, then by Pumping Lemma,
we can write
$w = xyz, |xy| \leq n, |y| > 0$

suppose
$n = 7$
$w = a^7$
$= aaaaaaa$

case 1:
y has only a i.e. aaaaaaa

step 3:
We want to find i such that $xy^iz \notin L$
For $i = 2$,
In case 1, $xy^2z = a\,aaaaaaaaaaaaa$
$= a^{14}$
This is not prime. So not in pattern.
$\therefore xy^iz \notin L$ also $|xy| = 7, n = 7, |xy| \leq n$ (true)

(Imp) Closure properties of regular set language

1. If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

Proof:

Let $M_1 = (Q_1, \Sigma_1, s_1, q_1, F_1)$ be NFA that accept regular language, L_1, $L_1 = L_1(M_1)$ and,

$M_2 = (Q_2, \Sigma_2, s_2, q_2, F_2)$ be another NFA that accept regular language L_2, $L_2 = L_2(M_2)$

Here, assume Q_1 and Q_2 are disjoint sets.

Now, we construct NFA, $m = (Q, \Sigma, s, q, F)$ such that it can accept $L = L_1(m_1) \cup L_2(m_2)$ where q is new state not in Q_1 and Q_2 where;

$$Q = Q_1 \cup Q_2 \cup \{q\}$$

$$\Sigma = \Sigma_1 \cup \Sigma_2 \cup \{\varepsilon\}$$

$$s_b = s_1 \cup s_2 \cup [\delta(q, n) q_1, \delta(q, \Lambda) \to q_2]$$

m begins any computation by non deterministically, choosing to either initial state of M_1 or the initial state of M_2 and M initiates either M_1 or M_2

If w belongs to $w \in \Sigma^*$, then $(q,w) \xrightarrow{*} (P,\varepsilon)$
For some $P \in F$ if and only if
$(q,w) \xrightarrow[M_1]{*} (P,\varepsilon)$ for some $P \in F_1$

Or,
$(q,w) \xrightarrow[M_2]{*} (P,\varepsilon)$ for some $P \in F_2$

Hence, $L_1 \cup L_2$ is regular.
$L(M) = L_1(M_1) \cup L_2(M_2)$.

2. If L_1 and L_2 are regular language, then $L_1 - L_2$ is also regular language.

Proof:
Let $M_1 = (Q_1, \Sigma_1, s_1, q_1, F)$ be NFA that accepts language L_1, i.e.
$L_1 = L_1(M_1)$ and
$M_2 = (Q_2, \Sigma_2, s_2, q_2, F_2)$ be another NFA that accept language L_2, i.e. $L_2 = L_2(M)$.
Now, we construct NFA, $M = (Q, \Sigma, s, q, F)$ such that it accept $L_1 \cdot L_2$, i.e.
$L(M) = L_1(M_1) \cdot L_2(M_2)$.
where, $Q = Q_1 \cup Q_2$.
$\Sigma_1 = \Sigma_1 \cup \Sigma_2 \cup \{\wedge\}$
$\delta = \delta_1 \cup \delta_2 \cup [(F_2, \wedge) \to q_2]$
$q = q_1$
$F = F_2$

Formally,
$(q,w) \xrightarrow[M]{x} (P,\varepsilon)$ for some $P \in F$

if and only if, there exist $w_1, w_2 \in \Sigma^*$ and $p_1 \in F_1, p_2 \in F_2$ such that
$w = w_1 \cdot w_2$, $(q_1, w_1) \xrightarrow[M_1]{*} (p_1, \varepsilon)$
and, $(q_2, w_2) \xrightarrow[M_2]{*} (p_2, \varepsilon)$.

3. If L_1 and L_2 are regular, then $L_1 \cap L_2$ is also regular.

Proof:

Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ be DFA that accepts language L_1 i.e. $L_1 = L_1(M_1)$ and $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ be another DFA that accept language L_2 i.e. $L_2 = L_2(M_2)$.

We assume that alphabets of both automata are same and Σ is union of alphabets of L_1 and L_2, if they are different.

Now, we construct DFA, $M = (Q, \Sigma, s, q, F)$ such that it accept $L_1 \cap L_2$, i.e. $L(M) = L_1(M_1) \cap L_2(M_2)$.

where,

$$Q = Q_1 \times Q_2$$
$$\Sigma = \Sigma_1 \cup \Sigma_2$$
$$s = (s_1, s_2)$$
$$q = (q_1, q_2)$$
$$F = F_1 \times F_2$$

If M_1 goes from state p to s on reading x, and M_2 goes from state q to t on reading a. Then M will go from state (P, q) to (S, t) on reading a.

If $(p, a) \xrightarrow[M_1]{*} (s, q)$ and $(q, a) \xrightarrow[M_2]{*} (t, \varepsilon)$

then, $(\{P, q\}, a) \xrightarrow[M]{*} (\{s, t\}, \varepsilon)$

i.e. $\delta(\{P, q\}, a) = (\delta_1(P, a), \delta_2(q, a))$.

Thus, string is accepted by M if and only if w is accepted by M_1 and M_2.

4. If L is a regular language over Σ then $\bar{L} = \Sigma^* \setminus L$ is also regular.

Proof:

Let L be recognized by DFA,

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

Then,

$\bar{L} = L(B)$ where, B is a DFA.

$B = (Q, \Sigma, \delta, q_0, Q \setminus F)$

i.e. B is exactly like A but final state of A have become non final state of B and vice versa.

Then w is in L(B) if and only if $\delta(q_0, w)$ is in Q\F which occurs if and only if w is not in L(A).

* The closure operation (Kleene closure) on a regular language is also regular.

Proof:

Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ be NFA that accept a regular language $L = L(M_1)$.

Now, we construct NFA, $M = (Q, \Sigma, s, q, F)$ such that it can accept $L(M) = L(M_1)^*$

where,

$Q = Q_1 \cup \{q\}$

$\Sigma = \Sigma_1 \cup \{\Lambda\}$

$\delta = \delta_1 \cup [(q, \Lambda) \to q_2, (q_2, \Lambda) \to F_1, (F_1, \Lambda) \to q]$

q = starting state = q

$F = F_1 \cup \{q\}$

Here, M consists of the states of M_1 and all transition of M_1 and also any final state of M_1

is also final state of M. The new initial state is also final state. so that ε or $\wedge$ is accepted.

It follows the inspection of M that if $\omega \in L(M)$ then either $\omega = \wedge$ or ε (Null) or $\omega = \omega_1 \omega_2 \omega_3 \ldots \omega_k$ for $k \geq i$, $i = 1, 2, \ldots, k$, there is $F_i \in F$ such that $(q_1, \omega) \xrightarrow[M_2]{*} (F_1, \wedge)$.

* If L and M are regular, then difference L/M is also regular.

Here,
$$L/M = L \cap \bar{M}$$
We already know that regular languages are closed under compliment and intersection.

Decision Algorithm for regular sets / language

↳ algorithm for regular language that tells whether or not following property holds.

1. The Membership problem
2. The Emptiness problem
3. The infinitiness problem

1. The membership problem

→ ~~Each~~ Is string w in regular language?
→ Assume L is represented by a DFA M.
→ simulate the action of M. on the sequence of input symbol forming w.
→ If processing the w, M reaches the final state then w is member of L otherwise not.

2. The Emptiness problem

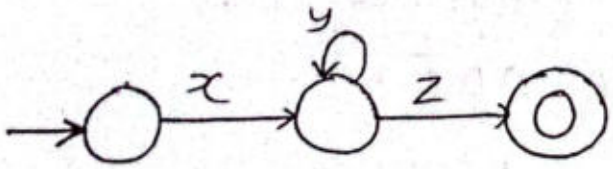
→ Given a regular Language, does the language contain any string at all

→ Assume a DFA for language L.
→ Construct a transition graph
→ Compute the set of states reachable from the start state.
→ If for any string, the final state is reachable, then the string is present in L, else no.

3. The infinitiness problem

→ Is a given regular language infinite?
→ Start with a DFA for the language L.
→ If the DFA has 'n' states, and the language contains any string of length 'n' or more, then the language is ~~finite~~. infinite otherwise the language is finite.
→ If n-state DFA accept a string w. of length n or more, then there must be a state that appears twice on the path labelled 'w' from start to final state.

→



Here, $w = xyz$

Then, $xy^iz \in L$ for $i \geq 0$.

→ Since, y is not null, there is an infinite no. of string in L.

Unit 3

Context-Free Language and Pushdown Automata

[13 hrs]

* **Context Free Language**
  - ↳ Language generated by context free grammar
  - ↳ Language accepted by PDA

* **Context - Free Grammar**
  - ↳ A context-free grammar G is a quadruple (4-tuples) defined as

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

where

V = set of variables or non terminal symbols

Σ = set of terminal symbols

P = Production Rule

S = starting Symbol

A CFG has production rule in the form,

$$A \rightarrow \alpha$$

where, $\alpha = \{ V \cup \Sigma \}^*$ and $A \in V$

or,

non-terminal →	non-terminal
non-terminal →	terminal

Capital letters.

represented by small letters / digits / symbols.

Why Context free?

Eg:

Consider $S \rightarrow aaAb$

A is non terminal and a, b are terminals.

sentence → Noun Verb

Noun → Ram / Sita

verb → goes / writes / sings.

Eg: construct a CFG for the language having any number of a's over

$\Sigma = \{a\}$

soln:

As we know, R.E for this

R.E = a^*

Now, production rule

$S \rightarrow \varepsilon$
$S \rightarrow aS$

If we want 'aaaa' string to be derived, we can start with start symbol.

$S \rightarrow aS$

$\rightarrow aaS$ $(S \rightarrow aS)$

$\rightarrow aaaS$ $(S \rightarrow aS)$

$\rightarrow aaaaS$ $(S \rightarrow aS)$

$\rightarrow aaaa\varepsilon$ $(S \rightarrow \varepsilon)$

$\rightarrow aaaa$

CFG for given problem is,

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

where, $V = \{S\}$

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

$S = \{S\}$

P is, $S \rightarrow \varepsilon \mid aS$

2. Construct a CFG for R.E. $(0+1)^*$

Soln:

Here, R.E. $= (0+1)^*$

Possible strings are $\{\varepsilon, 0, 1, 01, 10, \ldots\}$

Production rule is

$S \rightarrow 0S \mid 1S \mid \varepsilon$

3. Construct a CFG for string having at least two a's over $\Sigma = \{a, b\}$

solution:

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

Possible strings are $\{aa, aba, baaa, \ldots\}$

Now,

Production rule

$S \rightarrow AaAaA$

$A \rightarrow \varepsilon \mid aA \mid bA$

4. Construct a CFG for string having different first and last symbol over $\Sigma = \{0, 1\}$

Soln:

Here,

$R.E. = 0(0+1)^* 1$

& $R.E. = 1(0+1)^* 0$

Possible strings are $\{01, 10, 001, 110, \ldots\}$

Thus, Production rule,

$S \rightarrow 0A1$

$S \rightarrow 1A0$

$A \rightarrow \varepsilon \mid 0A \mid 1A$

5. Construct a CFG for language $L = w c w^T$, $w \in (a, b)^*$

$w^T \Rightarrow$ reverse

Solution:

Production rule:

$S \rightarrow c \mid aSa \mid bSb$

6. Construct a CFG for $L = \{a^n b^n \mid n \geq 1\}$

Solution:

Possible strings are $L = \{ab, aabb, abb, \ldots\}$

Now, Production Rules,

$A \rightarrow a \mid aA$

$B \rightarrow b \mid bB$

$S \rightarrow AB$

7. Construct a CFG for $L = \{a^n b^m \mid n \geq 1, m \geq 0\}$

Solution:

Possible strings are, $L = \{a, ab, aab, aabb, \ldots\}$

Now, Production Rules,

$A \rightarrow a \mid aA$

$B \rightarrow \varepsilon \mid bB$

$S \rightarrow AB$

8. Construct a CFG for $L = \{a^n b^m \mid (n+m) \text{ is even}\}$

solution

As we know,

$a^* = \{\varepsilon, a, aa, aaa, \ldots\}$ [not all even]

$(aa)^* = \{\varepsilon, aa, aaaa, \ldots\}$ [all even]

$a(aa)^* = \{a, aaa, aaaaa, \ldots\}$ [all odd]

Also,

even + even = even

odd + odd = even

R.E. for $L = \{a^n b^m \mid n+m \text{ is even}\}$ is

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

Production Rule for $(aa)^*$ is

$A \rightarrow \varepsilon \mid aaA$

Production rule for $(bb)^*$ is

$B \rightarrow \varepsilon \mid bbB$

Now, CFG For $L = \{a^m b^m \mid n+m \text{ is odd}\}$ is

$S \rightarrow AbB \mid aAB$

$A \rightarrow \varepsilon \mid aaA$

$B \rightarrow \varepsilon \mid bbB$

9. Construct a CFG for
$L = \{a^n b^m \mid n+m \text{ is odd}\}$

Solution:

~~R.E. for $L = \{a^n b^m \mid n+m \text{ is odd}\}$~~

even + odd = odd
odd + even = odd

R.E. for $L = \{a^n b^m \mid n+m \text{ is odd}\}$ is
$(aa)^* b (bb)^* + a(aa)^* (bb)^*$

Now, production rule for $(aa)^*$ is
$A \to \varepsilon \mid aaA$

Production rule for $(bb)^*$ is
$B \to \varepsilon \mid bbB$

Now, CFG for $L = \{a^n b^m \mid n+m \text{ is odd}\}$ is
$S \to AbB \mid aAB$

$A \to \varepsilon \mid aaA$

$B \to \varepsilon \mid bbB$

Assignment

Q. Construct CFG for string whose length is odd over $\Sigma = \{a, b\}$

Q. Construct CFG for string having same starting and ending symbol over $\Sigma = \{a, b\}$

Q. Construct CFG for string where length is odd over $\Sigma = \{a, b\}$

→

$S \rightarrow AX \mid bX$

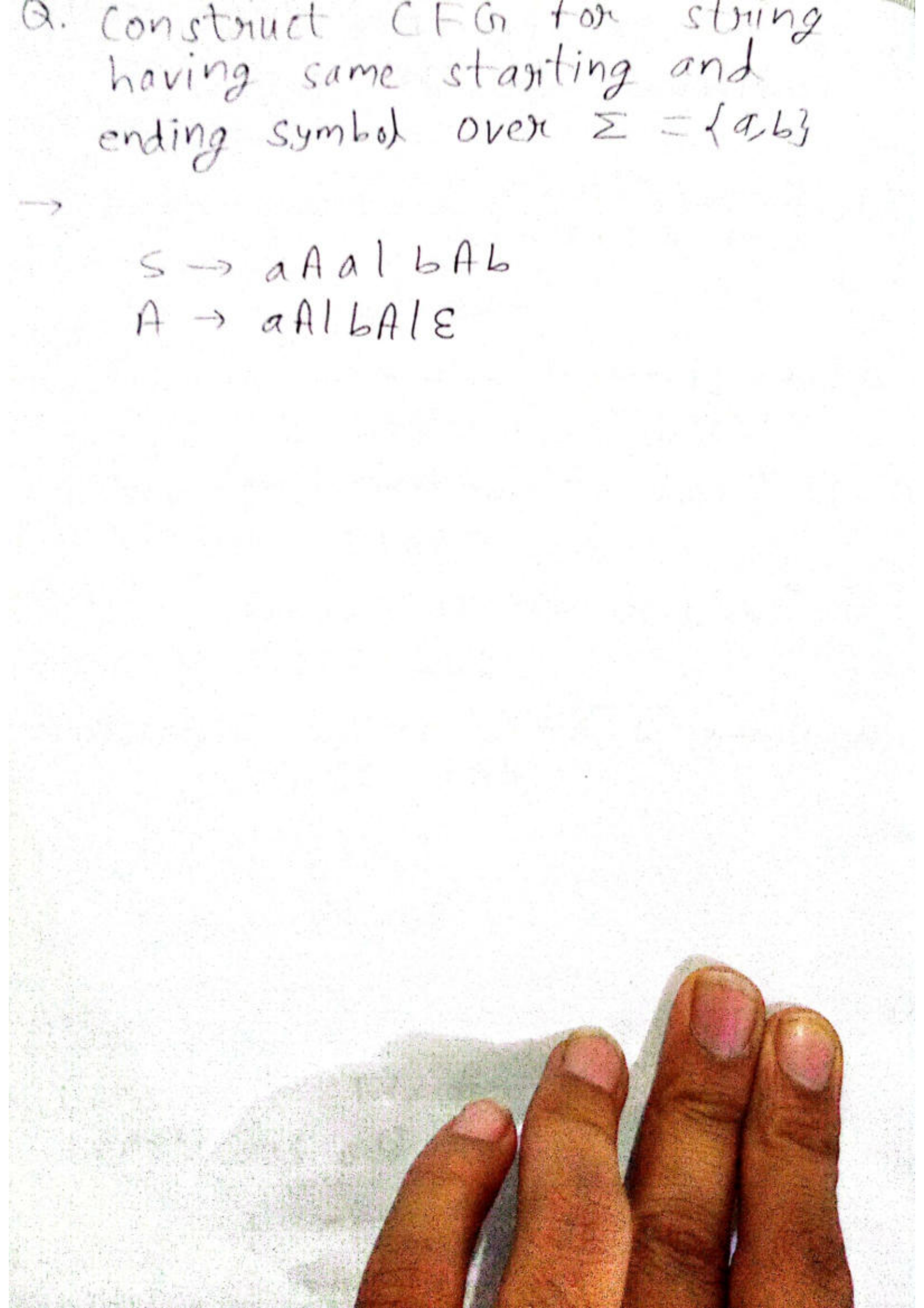
$X \rightarrow aS \mid bS \mid \varepsilon$

Q. Construct CFG for string having same starting and ending symbol over $\Sigma = \{a, b\}$

→

$S \rightarrow aAa \mid bAb$

$A \rightarrow aA \mid bA \mid \varepsilon$



Derivation of CFG:

↳ process of deriving strings from grammar.

Eg: Derive a^4 from grammar.

$S \rightarrow aS \mid \varepsilon$

solution:

$S \rightarrow aS$
$\rightarrow aaS \quad [S \rightarrow aS]$
$\rightarrow aaaS \quad [S \rightarrow aS]$
$\rightarrow aaaaS \quad [S \rightarrow aS]$
$\rightarrow aaaa\varepsilon \quad [S \rightarrow \varepsilon]$
$\rightarrow aaaa$

Hence, string a^4 is accepted by grammar.

The language accepted by the grammar is $L = \{a^n \mid n \geq 0\}$

Q. Let $G = \{(S, C), (a, b), P, S\}$, where P is

$S \rightarrow aCa$
$C \rightarrow aCa \mid b$

Find $L(G)$ i.e. language accepted by grammar (CFL).

Solution:

$S \rightarrow aCa$
$\rightarrow aba \quad [C \rightarrow b]$.

$\therefore aba \in L(G)$

Also, $S \to \cancel{aCA}\ aCa$

$\to aaCaa \quad [C \to aCa]$

$\to aabaa \quad [C \to b]$

$\therefore aabaa \in L(G_1)$

similarly, $S \xrightarrow{*} aaabaaa \in L(G_1)$

$S \xrightarrow{*} aaaabaaaa \in L(G_1)$

and so on.

so, the language is

$L = \{a^n b a^n \mid n \geq 1\}$

Q. Find $L(G_1)$ for $S \to aS|bS|a|b$

solution:

$S \to a$

Also, $S \to b$

$S \to aS$

$\to aa \quad [S \to a]$

$S \to bS$

$\to bb \quad [S \to b]$

similarly,

$S \xrightarrow{*} aba$

$S \xrightarrow{*} aaba$

and so on.

so, the language is

$L = \{a^n b^m \mid (n+m) \geq 1\}$

Assignment:

Q. Find the language and derive 'abbaaba' from grammar.

$S \rightarrow XaaX$

$X \rightarrow aX \mid bX \mid \varepsilon$

A. Solution:

Q. If G is a grammar

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

$V = \{A, B, S\}$

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

$S = \{S\}$

and, production rule 'P' is

$S \rightarrow AaB$

$A \rightarrow baA \mid a$

$B \rightarrow aBB \mid b$

Check whether the string is accepted or not

(i) baaab

(ii) aab

(iii) baaabb

Q. Find the language and derive 'abbaaaba' from grammar.

$S \to XaaX$

$X \to aX \mid bX \mid \varepsilon$

Solution:

$S \to XaaX$ [$S \to XaaX$]

$\to aXaabX$ [$X \to aX, X \to bX$]

$\to a\varepsilon aabX$ [$X \to \varepsilon, X \to \varepsilon$]

$\to aaab$

$\therefore aaab \in L(G)$

So, the language is,

$L = \{(a+b)^* a^n (a+b)^* \mid n = 2\}$

Again, we have to derive 'abbaaba'

$S \to XaaX$ [$S \to XaaX$]

$\to aXaabX$ [$X \to aX, X \to bX$]

$\to abXaabaX$ [$X \to bX, X \to aX$]

$\to abb\varepsilon aaba\varepsilon$ [$X \to \varepsilon, X \to \varepsilon$]

$\to abbaaba$

$\therefore abbaaba \in L(G)$.

Q. If G is grammar,

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

$V = \{A, B, S\}$

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

$S \to \{S\}$

and, production rule P is:

$S \rightarrow AaB$
$A \rightarrow baA \mid a$
$B \rightarrow abB \mid b$

check whether the string is accepted or not.

(i) baaab
(ii) aab
(iii) baaabb

Solution:

$S \rightarrow AaB$
$A \rightarrow baA \mid a$
$B \rightarrow abB \mid b$

We have,

$S \rightarrow AaB$
$\rightarrow baAab \quad [A \rightarrow baA, B \rightarrow b]$
$\rightarrow baaab \quad [A \rightarrow a]$

So, the language L is,

$\{(a+b)^* a (a+b)^*\}$

Now, for a string,

(i) baaab

$S \rightarrow AaB \quad [S \rightarrow AaB]$
$\rightarrow baAab \quad [A \rightarrow baA, B \rightarrow b]$
$\rightarrow baaab \quad [A \rightarrow a]$

$\therefore$ The string is accepted.

(ii) aab

$S \rightarrow AaB$ [$S \rightarrow AaB$]

$\rightarrow aaBb$ [$A \rightarrow a, B \rightarrow b$]

∴ The string is accepted.

(iii) baaabb

$S \rightarrow AaB$ [$S \rightarrow AaB$]

$\rightarrow baAaabB$ [$A \rightarrow baA, B \rightarrow abB$]

$\rightarrow baaaaabb$ [$A \rightarrow a, B \rightarrow b$]

∴ the string is not accepted.

Derivation can be classified as:

1. Left most Derivation
- If we apply rule at left most variable every time.

2. Right most Derivation
- If we apply rule at right most variable every time.

3. Mixed Derivation
- combination of left most and right most.

Derivation Tree

→ representation of derivation using tree
→ also called parse tree.

Definition:

A derivation tree for CFG, $G = \{V, \Sigma, P, S\}$ is a tree satisfying following condition:

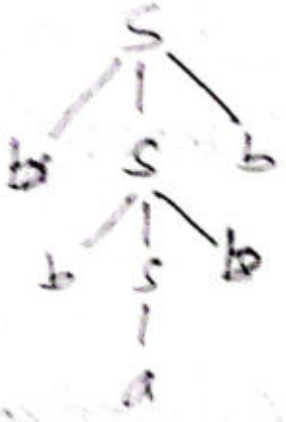
1. Every vertex has a label which is a variable or a terminal or null (ε or $\wedge$).
2. The root has label S.
3. The label of an internal vertex is a variable.
4. If the vertices $n_1, n_2, \ldots n_k$ written with labels $X_1, X_2, \ldots X_k$

are the children nodes of vertex n with label A, then

$A \rightarrow X_1 X_2 \ldots X_k$ is a production rule.

For example

$S \rightarrow bSb \mid a \mid b$



$[S \rightarrow bSb]$

$[S \rightarrow bSb]$

$[S \rightarrow a]$

Fig: Derivation Tree

Q. Draw a derivation tree for the string "abaaba" for the Grammar G where

$P = \{ S \rightarrow aSa \mid bSb \mid a \mid b \mid \varepsilon \}$

solution:

$S \rightarrow aSa$	$[S \rightarrow aSa]$
$\rightarrow abSba$	$[S \rightarrow bSb]$
$\rightarrow abaSaba$	$[S \rightarrow aSa]$
$\rightarrow abaεaba$	$[S \rightarrow \varepsilon]$
$\rightarrow abaaba$	

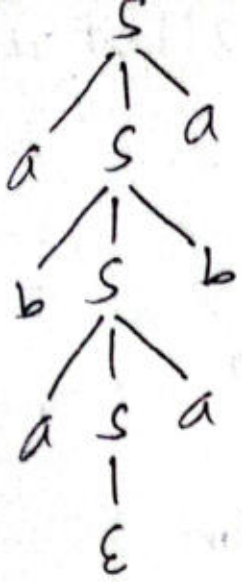


Fig: Parse tree

Assignment

Q. Consider a grammar G which has production rules as

$$S \rightarrow aAS \mid a$$
$$A \rightarrow SbA \mid SS \mid ba$$

Construct a parse tree which yeild "aabbaa" where, S is starting symbol.

Q. Derive the string "aabbabba" for left most derivation for a CFG given by,

$$S \rightarrow aB \mid bA$$
$$A \rightarrow a \mid aS \mid bAA$$
$$B \rightarrow b \mid bS \mid aBB$$

1. Left most derivation
2. Right most derivation
3. Mixed derivation
4. Make parse tree of each derivation.

Q. Derive the string "00110101"

$S \rightarrow 0B \mid 1A$

$A \rightarrow 0 \mid 0S \mid 1AA$

$B \rightarrow 1 \mid 1S \mid 0BB$

by (i) Left most derivation
(ii) Right most derivation
(iii) Mixed ~~part~~ derivation
(iv) Make parse tree of each derivation

Solution:

(i) Left most derivation

$S \rightarrow 0B$ [S → 0B]

$\rightarrow 00BB$ [B → 0BB]

$\rightarrow 001SB$ [B → 1S]

$\rightarrow 0011AB$ [S → 1A]

$\rightarrow 00110SB$ [A → 0S]

$\rightarrow 001101AB$ [S → 1A]

$\rightarrow 00110101B$ [A → 0]

$\rightarrow 00110101$ [B → 1]

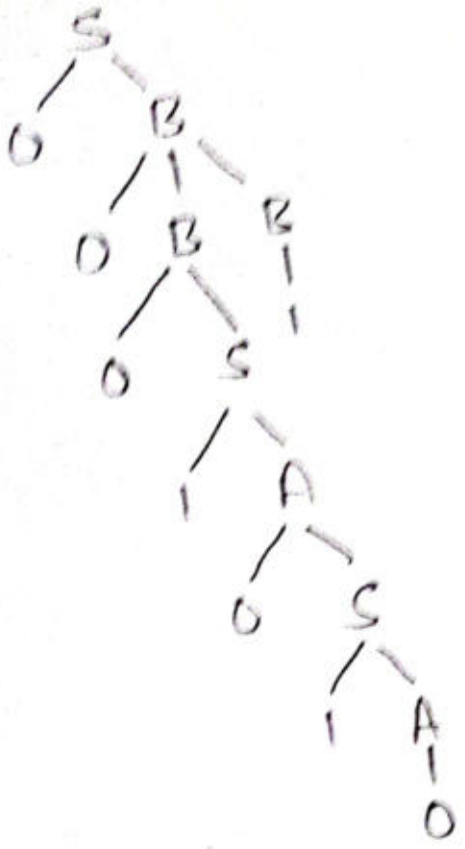


Fig: Parse tree

(ii) Right most derivation

$S \rightarrow 0B$ [S → 0B]
$\rightarrow 00BB$ [B → 0BB]
$\rightarrow 00B1S$ [B → 1S]
$\rightarrow 00B10B$ [S → 0B]
$\rightarrow 00B101S$ [B → 1S]
$\rightarrow 00B1010B$ [S → 0B]
$\rightarrow 00B10101$ [B → 1]
$\rightarrow 00110101$ [B → 1]

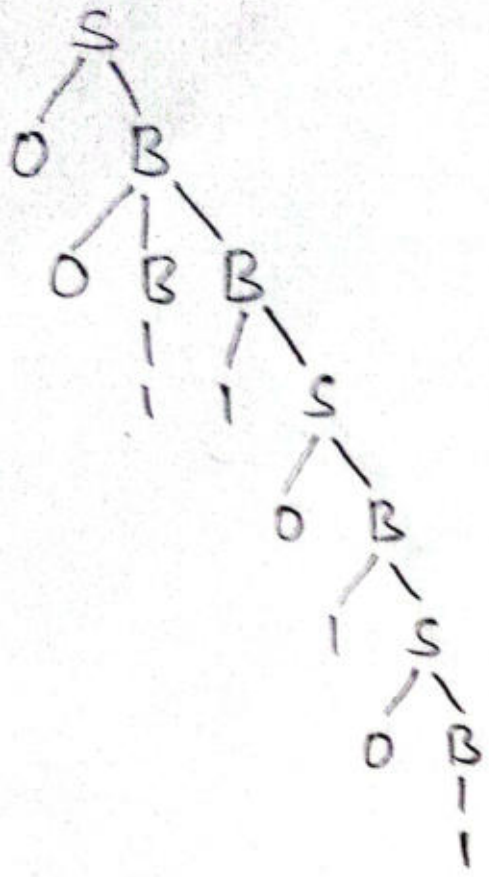


Figure: Parse tree

(iii) Mixed derivation

$S \rightarrow 0B$ [$S \rightarrow 0B$]

$\rightarrow 00BB$ [$B \rightarrow 0BB$]

$\rightarrow 0011S$ [$B \rightarrow 1$, $B \rightarrow 1S$]

$\rightarrow 00110B$ [$S \rightarrow 0B$]

$\rightarrow 001101S$ [$B \rightarrow 1S$]

$\rightarrow 0011010B$ [$S \rightarrow 0B$]

$\rightarrow 00110101$ [$B \rightarrow 1$]

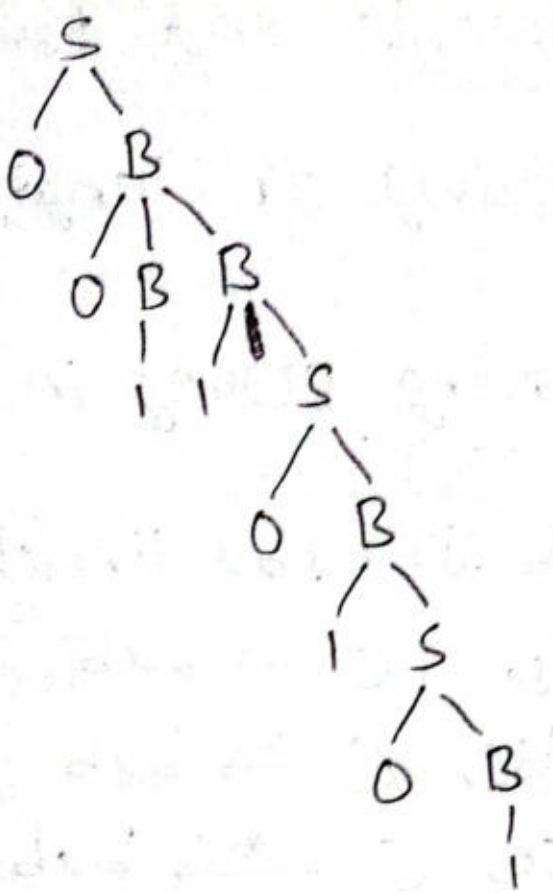


Fig: Parse Tree

Assignment:

Q. Derive the string "aaabbabbba" by

$S \rightarrow aB \mid bA$

$A \rightarrow aS \mid bAA \mid a$

$B \rightarrow bS \mid aBB \mid b$

1. Left most derivation
2. Right most derivation
3. Mixed derivation
4. Make parse tree of each derivation

Q. Consider a CFG with following production.

$S \rightarrow ASA \mid B$

$B \rightarrow aCb \mid bCa$

$C \rightarrow ACA \mid A$

$A \rightarrow a \mid b$

(a) What are variable and terminal in G?

(b) Give the string of length '7' in L(G).

(c) Are the following string in language of G i.e. L(G)?
(i) aaa (ii) bbb (iii) aba (iv) abb

(d) True or False $C \rightarrow bab$

(e) True or False, $C \xrightarrow{*} bab$

(f) True or False, $C \xrightarrow{*} bab$.

V.V. Imp.

Ambiguity in CFG

→ A terminal string $w \in L(G)$ is ambigious if there exist more than one derivation tree for same string i.e. w.

Eg: Show that the grammar

$S \rightarrow a \mid abSb \mid aAb$

$A \rightarrow bS \mid aAAb$ is ambigious.

solution:

1st method

$S \rightarrow abSb$ [$S \rightarrow abSb$]

$\rightarrow ababb$ [$S \rightarrow a$]

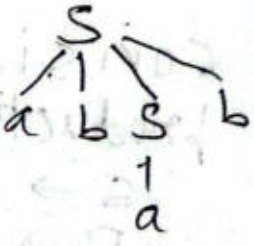


Fig: derivation tree

2nd method

$S \to aAb$ [$S \to aAb$]

$\to abSb$ [$A \to bS$]

$\to abab$ [$S \to a$]

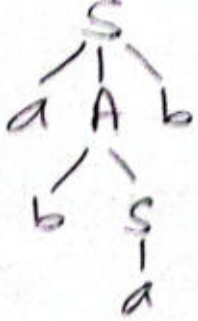


Fig: derivation tree

since more than one derivation for "abab".

so, grammar is ambigious.

Q. show that grammar G: is ambigious.

$S \to aB \mid ab$

$B \to ABb \mid b$

$A \to aAB \mid a$

Solution:

1st method.

$S \to aB$ ab [$S \to ab$]

Fig: derived tree

2nd method

$S \to aB$ [$S \to aB$]

$\to ab$ [$B \to b$]

Fig: derived tree

Since, more than one derivation for "ab". so, grammar is ambigious.

Q. If G is the grammar,

$S \rightarrow SbS \mid a$

show that grammar is ambigious for "abababa" string.

Solution:

1st method

$S \rightarrow SbS$ [$S \rightarrow SbS$]

$\rightarrow aSbS$ [$S \rightarrow a$, $S \rightarrow SbS$]

$\rightarrow abaSbS$ [$S \rightarrow a$, $S \rightarrow SbS$]

$\rightarrow ababaSba$ [$S \rightarrow a$, $S \rightarrow a$]

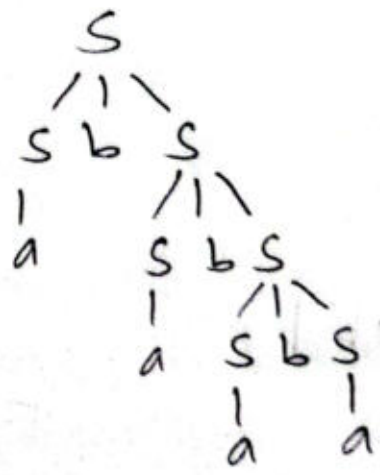


Fig: Parse Tree

2nd method.

$S \rightarrow SbS$ [$S \rightarrow SbS$]

$\rightarrow SbSba$ [$S \rightarrow SbS$, $S \rightarrow a$]

$\rightarrow SbSbaba$ [$S \rightarrow SbS$, $S \rightarrow a$]

$\rightarrow abababa$ [$S \rightarrow a$, $S \rightarrow a$]

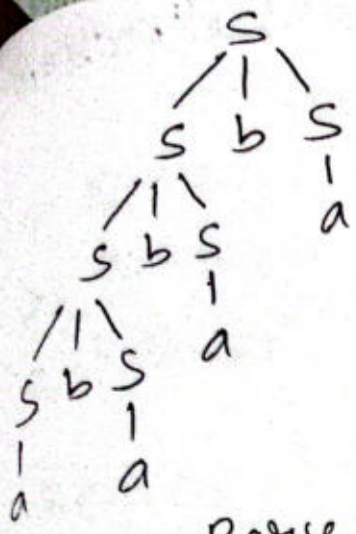


Fig: Parse Tree

since, different parse tree. So, ambigious.

Q. show that grammar G is ambigious.

$S \rightarrow aS \mid aSbS \mid X$

$X \rightarrow Xa \mid a$

Solution:

1st method

$S \rightarrow aS \quad [S \rightarrow aS]$

$\rightarrow aX \quad [S \rightarrow X]$

$\rightarrow aa \quad [X \rightarrow a]$

Fig: Derivation Tree

2nd method,

$S \rightarrow Xa \quad [S \rightarrow Xa]$

$\rightarrow aa \quad [X \rightarrow a]$

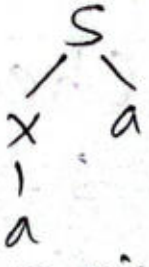


Fig: Derivation Tree

since, two different methods to derive "aa" string. So, ambigious.

Q. If G is grammar for Palindrome
$S \to aSa|bSb|a|b|\varepsilon$
check for ambiguity.

Solution:

1st method,

$S \to aSa$ [$S \to aSa$]
$\to aba$ [$S \to b$]

~~2nd method~~
~~$S \to aSa$~~

Not ambigious as it can't be derived in any other way.

Assignment

Q. show that grammar G is ambigious
$S \to aS|SaSa$

Q. Check for ambiguity
$S \to iCtS|iCtSeS|a$
$C \to b$

Simplification of Grammar

↳ removing all unnecessary symbols

1. Removal of useless symbols
2. Removal of NULL symbols
3. Removal of Unit Symbol

1. Removal of Useless symbol (Reduction of CFG)

→ CFG are reduced in two phases

Phase 1:

Derivation of CFG, G' from G such that each variable derive some terminal string.

Procedure.

step 1: Include all symbols w_i that derives some terminal and $i = 1$.

step 2: Include all symbols W_{i+1} that derives w_i

step 3: Increment i and repeat step 2 until $W_{i+1} = w_i$

step 4: Include all production rule that have w_i in it.

Phase 2:

Derivation of CFG, G'' from G', such that each symbol appears in sentential form (derives from start symbol).

Step 1: Include the start symbol in Y_i and $i=1$

Step 2: Include all symbols Y_{i+1} that can be derived from Y_i and include all production rule that have been applied.

Step 3: Increment 'i' and repeat step 2 until $Y_{i+1} = Y_i$

Q. Find a reduced grammar equivalent to G having production rule P as

$S \to AC \mid B,\ A \to a,\ C \to c \mid BC,\ E \to aA \mid e$

Solution:

Phase 1:

$T = \{a, c, e\}$

$W_1 = \{A, C, E\}$

$W_2 = \{A, C, E, S\}$

$W_3 = \{A, C, E, S\}$

$G' = \{(A, C, E, S), \{a, c, e\}, P, S\}$

where,

P: $S \to AC$, $A \to a$, $C \to c$, $E \to aAle$

Phase 2:

$Y_1 = \{S\}$

$Y_2 = \{S, A, C\}$

$Y_3 = \{S, A, C, a, c\}$

$Y_4 = \{S, A, C, a, c\}$

$G'' = \{(S, A, C), (a, c), P, S\}$

where,

P: $S \to AC$, $A \to a$, $C \to c$

so, this is the reduced grammar.

Q. Find a reduced grammar equivalent to G having production rule P as,

$S \to ABa \mid BC$, $A \to aC \mid BCC$, $C \to a$, $B \to bcc$

$D \to E$, $E \to d$, $F \to e$

Solution:

Phase1

$T = \{a, b, c, d, e\}$

$W_1 = \{S, A, C, B, E, F\}$

$W_2 = \{S, A, C, B, E, F, D\}$

$W_3 = \{S, A, B, C, D, E, F\}$

$G' = \{(S, A, C, B, E, F, D), (a, b, c, d, e), P, S\}$

where

P: $S \to ABa \mid BC$, $A \to aC \mid BCC$, $C \to a$,
$B \to bcc$, $D \to E$, $E \to d$, $F \to E$

Phase 2:

$Y_1 = \{S\}$

$Y_2 = \{S, A, B, C, a\}$

$Y_3 = \{S, A, B, C, a, b, c\}$

$G'' = \{(S, A, B, C), (a, b, c), P, S\}$

where

P: $S \to ABa \mid BC$, $A \to aC \mid BCC$, $C \to a$
$B \to bcc$

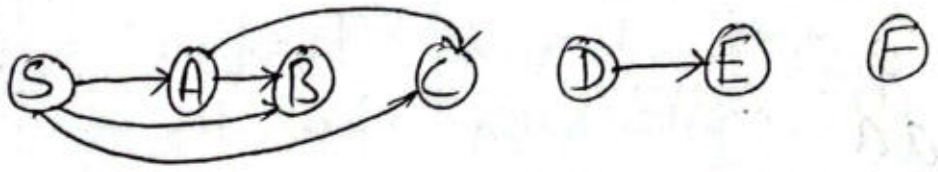
So, this is the reduced grammar.

Alternate method

Step 1: Eliminate non-generating symbols.
Here, all symbols are generating.

Step 2: Eliminate non-reachable symbols (variables)

Draw dependency graph



Here, D, E and F are non reachable from S.

After removing useless symbols, we get

P : $S \to ABa \mid BC$, $A \to aC \mid BCC$, $C \to a$,
$B \to bcc$

Assignment

Q. Find a reduced grammar equivalent to G having production rule P as,

$S \to AB \mid CA$, $A \to a$
$S \to BC \mid AB$, $C \to aB \mid b$

Q. Find a reduced grammar equivalent to G having production rule. P as

$S \to aAa$
$B \to ab$
$A \to bBB$
$C \to ab$

Solution:

Here, C is non reachable from S. so, remove it.

$S \to aAa$
$A \to bBB$
$B \to ab$

Q. $S \rightarrow aS \mid A \mid BC$
$A \rightarrow a$
$B \rightarrow aa$
$C \rightarrow aCb$

solution:

Here, C is useless as it does not derive any string.
After removing C, we get
$S \rightarrow aS \mid A$
$A \rightarrow a$
$B \rightarrow aa$

Here, B is non reachable from S.
So, after removing it, we get
$S \rightarrow aS \mid A$
$A \rightarrow a$

2. Removal of Unit production ($A \rightarrow B$)

Any production rule of the form $A \rightarrow B$ where $A, B \in$ non_terminal are called Unit production.

Procedure:

Step 1: To remove production $A \rightarrow B$, add $A \rightarrow x$ to the grammar rule whenever $B \rightarrow x$
($x \in$ terminal or $x \in$ null)

step 2: Remove $A \to B$ from grammar.
step 3: Repeat step 1 and step 2 until all unit production are removed.

Q. Remove unit production from given grammar,
P: $S \to XY$, $X \to a$, $Y \to Z \mid b$, $Z \to M$, $M \to N$, $N \to a$.

Solution:
Here, Unit production are
$Y \to Z$, $Z \to M$, $M \to N$

1. Since, $N \to a$, add $M \to a$, we get
P: $S \to XY$, $X \to a$, $Y \to Z \mid b$, $Z \to M$, $M \to a$, $N \to a$

2. Since, $M \to a$, add $Z \to a$, we get
P: $S \to XY$, $X \to a$, $Y \to Z \mid b$, $Z \to a$, $M \to a$, $N \to a$

3. Since, $Z \to a$, add $Y \to a$, we get
P: $S \to XY$, $X \to a$, $Y \to a \mid b$, $Z \to a$, $M \to a$, $N \to a$

Also, remove non reachable symbols,
Here, M, N and Z are non reachable
So, reduced grammar is,
P: $S \to XY$
$X \to a$
$Y \to a \mid b$

Assignment:

Q. Remove unit production from given grammar,

P: $S \to A \mid bb$
$A \to B \mid b$
$B \to s \mid a$

Q. Eliminate Unit production from grammar,

$S \to Aa \mid B$, $B \to A \mid bb$, $A \to a \mid bc \mid B$

Solution:

Here, Unit productions are,

$S \to B$, $B \to A$, $A \to B$

1. Since $A \to a \mid bc$, add $B \to a \mid bc$
(Remove $B \to A$)

$B \to a \mid bc \mid bb$

2. Since $B \to a \mid bc \mid bb$, add $A \to a \mid bc \mid bb$
(Remove $A \to B$)

$A \to a \mid bc \mid bb$

3. Since $B \to a \mid bc \mid bb$, add $S \to a \mid bc \mid bb$
(Remove $S \to B$)

$S \to Aa \mid a \mid bc \mid bb$

Here, B is non reachable. So, remove B.

we get

$S \to Aa \mid a \mid bc \mid bb$

$A \to a \mid bc \mid bb$

Elimination of NULL production

In CFG, a non terminal symbol 'A' is null production if $A \to \varepsilon$ (or $A \to \wedge$) or a production starting at 'A' leads to ε or $\wedge$ (i.e. $A \to \to \ldots . \varepsilon$ or $A \xrightarrow{*} \varepsilon$)

Procedure

step 1: To remove $A \to \varepsilon$, look for all productions whose right hand side contain A.

step 2: Replace each occurence of A in each of these productions with ε.

step 3: Add the resultant productions to the Grammar.

Q. Eliminate null production from grammar,

$S \to ABAC$, $A \to aA \mid \varepsilon$, $B \to bB \mid \varepsilon$, $C \to c$

Solution:

Here null production: $A \to \varepsilon$, $B \to \varepsilon$

1. Remove $A \to \varepsilon$

(i) $S \to ABAC$
$\to \varepsilon BAC$
$\to BAC$

(ii) $S \to ABAC$
$\to ABEC$
$\to ABC$

(iii) $S \to ABAC$
$\to \varepsilon B \varepsilon C$
$\to BC$

(iv) $A \to aA$
$\to a\varepsilon$
$\to a$

Now, new production rule are,

$S \to ABAC \mid BAC \mid ABC \mid BC$

$A \to aA \mid a, B \to bB \mid \varepsilon, C \to c$

2. ~~Remove $B \to \varepsilon$~~

(i) ~~$S \to ABAC$~~
~~$\to A\varepsilon AC$~~
~~$\to AAC$~~

(ii) ~~$B \to bB$~~
~~$\to b\varepsilon$~~
~~$\to b$~~

~~Now, new production rule are,~~

~~$S \to ABAC \mid A$~~

2. Remove $B \to \varepsilon$

(i) $S \to ABAC$
$\to A\varepsilon AC$
$\to AAC$

(ii) $C \to BAC$
$\to \varepsilon AC$
$\to AC$

(iii) $S \to ABC$
$\to A\varepsilon C$
$\to AC$

(iv) $S \to BC$
$\to \varepsilon C$
$\to C$

(iv) $B \to bB$
$\to b\varepsilon$
$\to b$

New production rule,

$S \to ABAC \mid BAC \mid ABC \mid BC \mid AAC \mid AC \mid C$

$A \to aA \mid a$

$B \to bB \mid b$

$C \to c$

4. Eliminate null production from grammar.

$S \to ABaC, A \to BC, B \to b \mid \varepsilon, C \to D \mid \varepsilon, D \to d$

5. Eliminate null production from grammar.

$S \to A$
$A \to BB$
$B \to aBb \mid \varepsilon$

1. Remove $B \to \varepsilon$

(i) $A \to BB$
$\to \varepsilon B$
$\to B$

(ii) $A \to BB$
$\to B\varepsilon$
$\to B$

(iii) $A \to BB$
$\to \varepsilon\varepsilon$
$\to \varepsilon$

(iv) $B \to aBb$
$\to a\varepsilon b$
$\to ab$

New production rule,

$S \to aA$
$A \to BB \mid B \mid \varepsilon$
$B \to aBb \mid ab$

2. Remove $A \to \varepsilon$,

(i) $S \to aA$
$\to a\varepsilon$
$\to a$

Now, new production rule,

$S \to aA \mid a$
$A \to BB \mid B$
$B \to aBb \mid ab$

8. Simplify the grammar

$S \rightarrow aA \mid aBB$

$A \rightarrow aAA \mid \varepsilon$

$B \rightarrow bB \mid bbC$

$C \rightarrow B$

Solution:

Remove

① Null

② Unit

③ Useless

1. Eliminating ε - production gives resulting grammar as,

$S \rightarrow aA \mid aBB \mid a$

$A \rightarrow aAA \mid aA \mid a$

$B \rightarrow bB \mid bbC$

$C \rightarrow B$

2. Removing unit production gives resulting grammar as,

$S \rightarrow aA \mid aBB \mid a$

$A \rightarrow aAA \mid aA \mid a$

$B \rightarrow bB \mid bbC$

$C \rightarrow bB \mid bbC$

3. Removing useless symbol gives resulting grammar as,

Here, B and C are useless because B and C do not derive any string.

After removing useless symbol we get

$S \rightarrow aAla$

$A \rightarrow aAAlaAla$

This is simplified grammar.

Normal Forms

↳ production rule that has certain restriction.

Types

1. Chomsky Normal Form (CNF)
2. Greibach Normal Form (GNF)

1. Chomsky Normal Form (CNF)

↳ A CFG is in CNF if production are of the following form:

$A \rightarrow a$

$A \rightarrow BC$

where, A, B and C are variables and a is terminal.

Procedure for converting into CNF

Step 1: If the start symbol S occurs on some right side, create a new start symbol S' and a new production

$S' \rightarrow S$

Step 2: Remove null production.
Step 3: Remove unit production
Step 4: Remove string of terminals on right hand side of production if it exceeds as follows:

Suppose, we have production

$$S \rightarrow a_1 a_2 a_3$$

Then, introduce non-terminal C_{a_i} as,

$$C_{a_1} \rightarrow a_1, C_{a_2} \rightarrow a_2, C_{a_3} \rightarrow a_3$$

Step 5: To restrict number of variables on the right hand side, introduce new variable and separate them as follows:

Suppose, we have production with n, order non terminals as shown below with 5 non terminals:

$$Y \rightarrow X_1 X_2 X_3 X_4 X_5$$

Add $n-2$ new production rule using $n-2$ new non terminals. and Modify the production as below:

$$Y \rightarrow X_1 R_1$$
$$R_1 \rightarrow X_2 R_2$$
$$R_2 \rightarrow X_3 R_3$$
$$R_3 \rightarrow X_4 X_5$$

step 6: If the right side of any production is in the form

$A \rightarrow aB$, where a is terminal

then, the production is replaced by,

$A \rightarrow XB$, $X \rightarrow a$.

Q. Convert the following CFG into CNF.

$P: S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \varepsilon$

Solution:

Since S appears in right hand side, introduce new start symbol S' and add $S' \rightarrow S$ to production rule.

$P: S' \rightarrow S$, $S \rightarrow ASA \mid aB$, $A \rightarrow B \mid S$, $B \rightarrow b \mid \varepsilon$

2. Remove null production: $B \rightarrow \varepsilon$

$P: S' \rightarrow S$, $S \rightarrow ASA \mid aB \mid a$, $A \rightarrow \varepsilon \mid S$, $B \rightarrow b$

After removing $A \rightarrow \varepsilon$

$P: S' \rightarrow S$, $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S$, $A \rightarrow S$,

$B \rightarrow b$

3. Remove unit production

$S' \rightarrow S$, $S \rightarrow S$, $A \rightarrow S$

After removing : $S \to S$

P: $S' \to S$
$S \to ASA \mid aB \mid a \mid SA \mid AS$
$A \to S \mid B$, $B \to b$

After removing : $S' \to S$

P: $S' \to aB \mid a \mid ASA \mid SA \mid AS$
$S \to ASA \mid aB \mid a \mid SA \mid AS$
$A \to S \mid B$, $B \to b$

After removing $A \to S$

P: $S' \to ASA \mid aB \mid a \mid SA \mid AS$
$S \to ASA \mid aB \mid a \mid SA \mid AS$
$A \to ASA \mid aB \mid a \mid SA \mid AS \mid B$
$B \to b$

After removing $A \to B$

P: $S' \to ASA \mid aB \mid a \mid SA \mid AS$
$S \to ASA \mid aB \mid a \mid SA \mid AS$
$A \to ASA \mid aB \mid a \mid SA \mid AS \mid b$
$B \to b$

4. Now, check for more than two variables on R.H.S.,
$S' \to ASA$, $S \to ASA$, $A \to ASA$
Removing these
$S' \to ASA$
$\to AX \; [X \to SA]$

$S \rightarrow ASA$
$\rightarrow AX \quad [X \rightarrow SA]$

$A \rightarrow ASA$
$\rightarrow AX \quad [X \rightarrow SA]$

Now, we get
P: $S' \rightarrow AX \mid aB \mid a \mid sA \mid AS$
$S \rightarrow AX \mid aB \mid a \mid sA \mid AS$
$A \rightarrow AX \mid aB \mid a \mid sA \mid AS \mid b$
$B \rightarrow b$
$X \rightarrow SA$

5. Change production $S' \rightarrow aB$, $S \rightarrow aB$, $A \rightarrow aB$ as:

$S' \rightarrow aB$.
$\rightarrow YB$, where, $Y \rightarrow a$

Similarly,
$S \rightarrow YB$ and $A \rightarrow YB$

Finally we get
P: $S' \rightarrow AX \mid YB \mid a \mid sA \mid AS$
$S \rightarrow AX \mid YB \mid a \mid sA \mid AS$
$A \rightarrow AX \mid YB \mid a \mid sA \mid AS \mid b$
$B \rightarrow b$
$X \rightarrow SA$
$Y \rightarrow a$

Assignment:

Q. Convert following CFG to CNF;

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

---

2. Convert into CNF

$S \rightarrow AB \mid aB$

$A \rightarrow aab \mid \varepsilon$

$B \rightarrow bbA$

Solution:

1. Since S doesnot appear in right hand side. So, no need to introduce new symbol S'.

2. Since, there is null production.

$A \rightarrow \varepsilon$, We need to remove it

Removal of $A \rightarrow \varepsilon$

(i) $S \rightarrow AB$
$\rightarrow \varepsilon B$
$\rightarrow B$

(ii) $B \rightarrow bbA$
$\rightarrow bb\varepsilon$
$\rightarrow bb$

New production rule is;

P: $S \rightarrow AB \mid aB \mid B$

$A \rightarrow aab$

$B \rightarrow bbA \mid bb$

3. Since there is unit production $S \to B$, we need to remove it.
After removal of $S \to B$, new production rule is:

P: $S \to AB \mid aB \mid bbA \mid bb$
A: aab
B: $bbA \mid bb$

4. There is not more than two variables on both sides.

5. Change production $S \to aB$,
$S \to bbA$, $S \to bb$, $A \to aab$,
$B \to bbA$, $B \to bb$ & $S \to aB$

$S \to bbA$ [$S \to bbA$]
$\to X_1 A$ [$X_1 \to bb$]

$S \to bb$ [$S \to bb$]
$\to X_2 X_2$ [$X_2 \to b$, $X_2 \to b$]

$A \to aab$ [$A \to aab$]
$\to X_3 bX_2$ [$X_3 \to aa$, $X_2 \to b$]

$B \to bbA$ [$B \to bbA$]
$\to X_1 A$ [$X_1 \to bb$]

$B \to bb$ [$B \to bb$]
$\to X_2 X_2$ [$X_2 \to b$, $X_2 \to b$]

New production rule,
$S \to aB$ [$S \to aB$]
$\to X_4 B$ [$X_4 \to a$]

New production rule is:

P: $S \to AB \mid X_4B \mid X_1A \mid X_2X_2$

$A \to X_3X_2$

$B \to X_1A \mid X_2X_2$

$X_1 \to bX_2X_2$

$X_2 \to b$

$X_3 \to aX_4X_4$

$X_4 \to a$

Greibach normal form (GNF)

A CFG is in GNF if the production rules are of following forms:

$A \to b$

$A \to bC_1C_2 \ldots\ldots C_n$

where, $A, C_1, C_2, \ldots\ldots C_n$ are non terminals and b is terminal.

eg: $A \to BCD$ (invalid) ✗

$A \to aBCD$ (valid) ✓

* Steps to convert into GNF

step 1: Check CFG have any null or unit production, if any remove it.

step 2: Check if CFG is in CNF, if not convert it.

step 3: Change the non terminal symbols into A_i in ascending order of i.

step 4: Alter the rules so that non terminals are in ascending order such that, if the production is of the form $A_i \rightarrow A_j X$, then $i < j$ and should never be $i \geq j$.

step 5: Remove left recursion, if there are any.

Eg:

$S \rightarrow CA \mid BB$

$B \rightarrow b \mid SB$

$C \rightarrow b$

$A \rightarrow a$

Solution:

1. No null or unit production
2. Already in CNF
3. Here, we replace non-terminals with A_i,

S with A_1	$A_1 \rightarrow A_2 A_3 \mid A_4 A_4$
C with A_2	$A_4 \rightarrow b \mid A_1 A_4$
A with A_3	$A_2 \rightarrow b$
B with A_4	$A_3 \rightarrow a$

4. Here, we have

$A_4 \rightarrow b \mid A_1 A_4$ [$i \rightarrow j$]

or $A_4 \rightarrow b \mid A_2 A_3 A_4 \mid A_4 A_4 A_4$

[$A_1 \rightarrow A_2 A_3 \mid A_4 A_4$]

or, $A_4 \rightarrow b \mid bA_3 A_4 \mid A_4 A_4 A_4$

[$A_2 \rightarrow b$]

Here, we get left recursion.

To remove left recursion introduce a new non-terminals as

$A_4 \rightarrow b \mid bA_3 A_4 \mid A_4 A_4 A_4$

$Z \rightarrow A_4 A_4 Z \mid A_4 A_4$

Now,

$A_4 \rightarrow b \mid bA_3 A_4 \mid b Z \mid bA_3 A_4 Z$

The grammar is,

$A \rightarrow A_2 A_3 \mid A_4 A_4$

$A_4 \rightarrow b \mid bA_3 A_4 \mid bZ \mid bA_3 A_4 Z$

$A_2 \rightarrow b$

$A_3 \rightarrow a$

$Z \rightarrow A_4 A_4 Z \mid A_4 A_4$

Now, convert it into GNF as.

$A_1 \rightarrow bA_2 | bA_4 | bA_3A_4A_4 | bZA_4 | bA_2A_4ZA_4$

$A_4 \rightarrow b | bA_2A_4 | bZ | bA_3A_4Z$

$A_3 \rightarrow b$

$A_2 \rightarrow a$

$Z \rightarrow bA_4Z | bA_3A_4A_4Z | bZA_4Z | bA_2A_4ZA_4Z$
$| bA_4 | bA_2A_4A_4 | bZA_4 | bA_2A_4ZA_4$

This is required GNF.

Assignment

Convert the CFG to GNF.

$S \rightarrow AA | a$

$A \rightarrow SS | b$

Pushdown Automata (PDA)

↳ a PDA is a machine that is designed to implement context-free language.

↳ CFL is accepted by PDA.

$L = \{a^n b^n | n \geq 1\}$ is not regular

CFL

↳

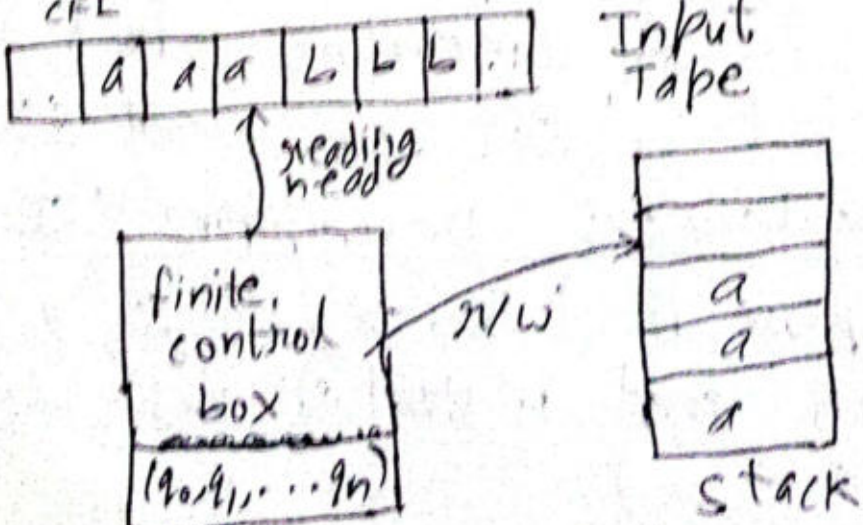


Fig: Pushdown Automata

↳ Similar to FA, but it has memory called pushdown stack.
↳ element can be read-write.

Formally,
A PDA consists of seven tuples, defined as,

$M = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\}$

where,
Q = set of finite state
Σ = i/p symbol or alphabet (that can be null)
Γ = symbols in stack
δ = transition function,
$\delta(q, a, X)$
where q is state in Q.
a is i/p symbol.
X is member of Γ (i.e. element in stack)
q_0 = initial state
Z_0 = top of stack (member of Γ)
F = set of final states

* Instantaneous Description (ID)

Suppose, $A = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\}$ is PDA and instantaneous description is (q, x, α)

For eg:

$(q_0, a_1 a_2 \ldots a_n, z_0 z_1 \ldots z_n)$ is ID.

This describe when PDA is in current state q_0 and input symbol to be processed is $a_1 a_2 \ldots a_n$ and PDS has $z_0 z_1 \ldots z_n$ where z_0 is top element, z_1 is second top element and z_n is the lowest element.

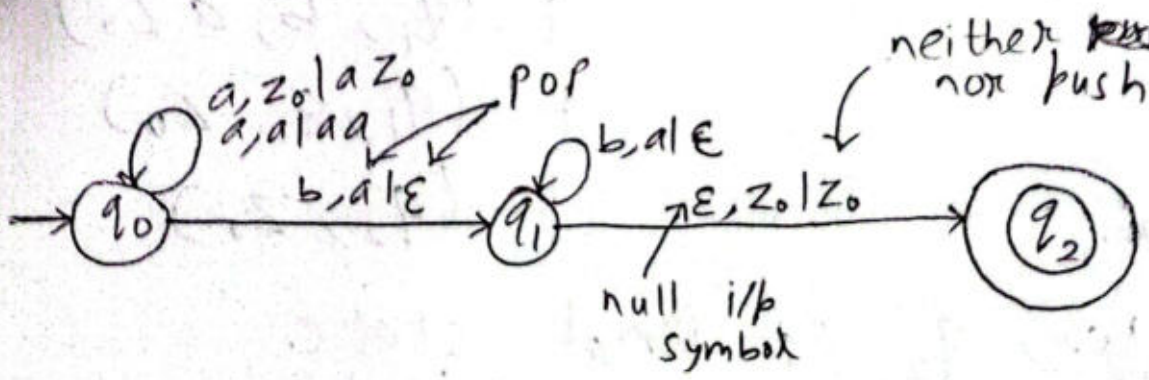
Q. Design a PDA for $L = \{a^n b^n / n \geq 1\}$

solution:

Concept:

→ Push every 'a' in the stack.

→ Then by reading each 'b', remove each 'a' from the stack.



Thus PDA is,

$A = \{Q, \Sigma, \Gamma, \delta, q_0, \Sigma, F\}$

where,

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

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

$\Gamma = \{a, z_0\}$

$q_0 = \{q_0\}$

$z_0 =$ top of stack

$F = \{q_2\}$

δ can be written as,

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ // Push 'a'

$\delta(q_0, a, a) = (q_0, aa)$ // Push 'a'

$\delta(q_0, b, a) = (q_1, \varepsilon)$ // Pop 'a'

$\delta(q_1, b, a) = (q_1, \varepsilon)$ // Pop 'a'

$\delta(q_1, \varepsilon, Z_0) = (q_2, Z_0)$

Now, we will simulate PDA for string "aaabbb" as,

$(q_0, aaabbb, Z_0) \vdash (q_0, aabbb, aZ_0)$

$\vdash (q_0, abbb, aaZ_0)$

$\vdash (q_0, bbb, aaaZ_0)$

$\vdash (q_1, bb, aaZ_0)$

$\vdash (q_1, b, aZ_0)$

$\vdash (q_1, \varepsilon, Z_0)$

$\vdash (q_2, Z_0)$

Here, q_2 is final state.
So, string is accepted.

Assignment:

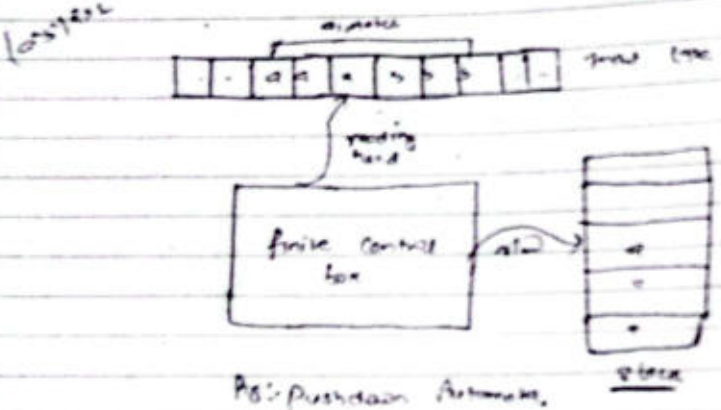
Q. Design a PDA for

1. $L = \{a^n b^{2n} \mid n \geq 1\}$
2. $L = \{a^n b^n c^m \mid n, m \geq 1\}$

Unit 3

Pushdown Automata (PDA)

- a PDA is a machine that is designed to implement context free language
- CFL is accepted by PDA.



- Similar to FA, but has memory called pushdown stack.
- elements can be read, write on stack.

* $L = \{a^n b^n \mid n \geq 1\}$ is not regular.
  - PDA can accept it, so, CFL is.

PDA

formally, a PDA consist of seven tuples, defined by

$M = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$

where;

Q = set of finite state
Σ = i/p symbol or alphabet (that can be read)
Γ = symbols in stack
δ = transition function, $\delta(q, a, x)$

where,
$q \rightarrow$ is state in Q
$a \rightarrow$ is i/p symbol
$x \rightarrow$ is member of Γ (i.e element in stack).

q_0 = initialization
z_0 = top of stack (member of Γ)
F = set of final states.

* Instantaneous Description (ID) :-

System $A = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$ is a PDA and instantaneous description is (q, w, α).

or $(q_0, a_1 a_2 \ldots a_n, z_0 z_1 \ldots z_n)$ is ID.

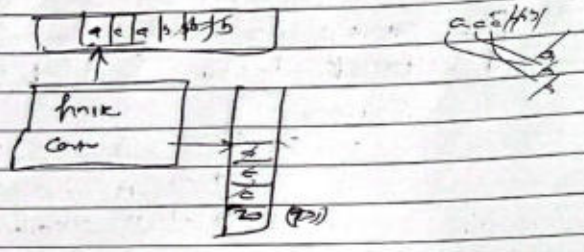
This describe when PDA is in current state (q_0) and i/p symbol to be processed is $a_1 a_2 \ldots a_n$ and PDA has $z_0 z_1 \ldots z_n$ where z_0 is top of stack, z_1 is second top element & z_n is bottom element

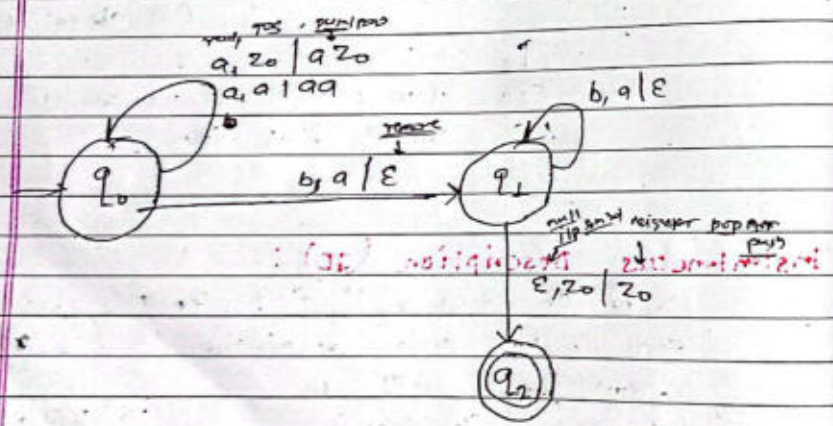
Q Design a PDA for $L = \{a^n b^n \mid n \geq 1\}$

Concept :-

→ push every 'a' in stack

→ Then, reading each 'b', remove each 'a' from stack.





Thus, PDA is

$A = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\}$

where

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

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

$\Gamma = \{a, b, Z_0\}$

$q_0 = \{q_0\}$

Z_0 = top of stack

$F = \{q_2\}$

δ can be written as :-

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ // push 'a'

$\delta(q_0, a, a) = (q_0, aa)$ // push 'a'

$\delta(q_0, b, a) = (q_1, \varepsilon)$ // pop 'a'

$\delta(q_1, b, a) = (q_1, \varepsilon)$ // "

$\delta(q_1, \varepsilon, Z_0) = (q_2, Z_0)$

Now we will simulate PDA for string "aaabbb" as,

$(q_0, aaabbb, Z_0) \vdash (q_0, aabbb, aZ_0)$

$\vdash (q_0, abbb, aaZ_0)$

$\vdash (q_0, bbb, aaaZ_0)$

$\vdash (q_1, bb, aaZ_0)$

$\vdash (q_1, b, aZ_0)$

$\vdash (q_1, \varepsilon, Z_0)$

$\vdash (q_2, Z_0)$

Here q_2 is final state

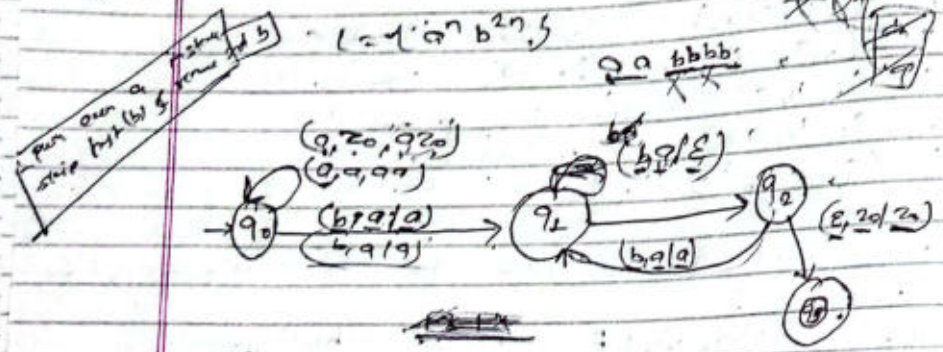
So string is accepted.

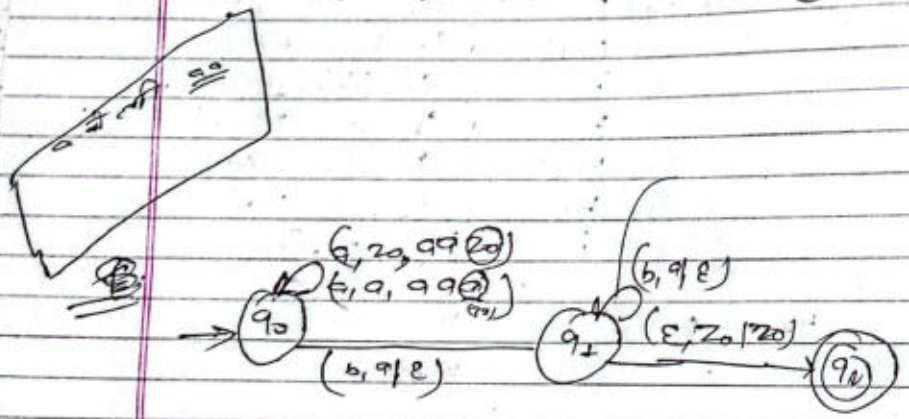
Design a PDA for (1) $L = \{$

	TOS	Input	Action
R	Z	a	push
R	Z	ε	pop
R	Tos	Tos	Skip

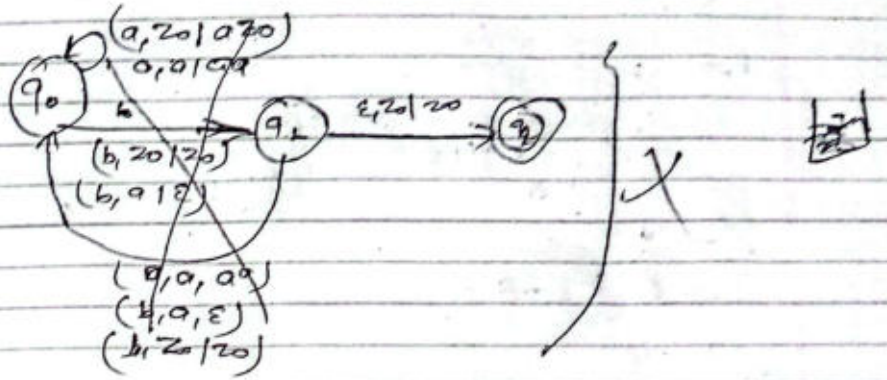
Q. Design a PDA for $L = \{a^n b^{2n} \mid n \geq 1\}$

$L = \{a^n b^{2n}, \}$





Q. Design PDA to accept $L = \{w \mid w \in (a+b)^* \text{ and } n_a(w) = n_b(w)\}$

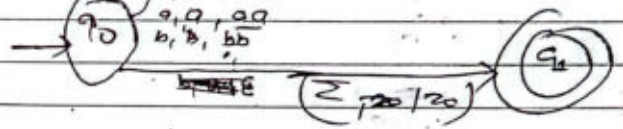


Concept :-

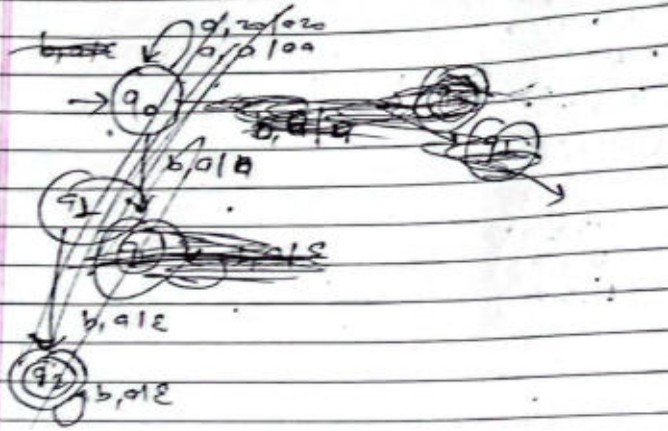
→ Initially, when stack is empty, whatever we read, push in stack.

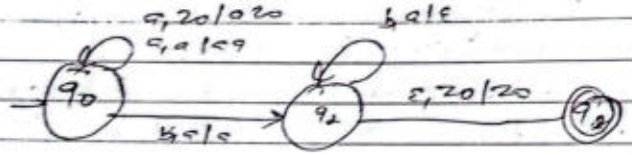
→ If we read 'a' & TOS is 'b' pop 'b'

→ if we read 'b' & TOS is 'a', pop 'a'

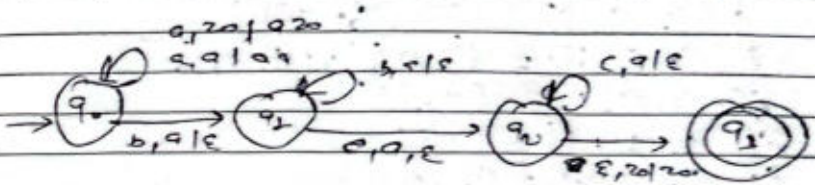


Q. Design PDA for $L = \{a^n b^n \mid n \geq 0\}$





Q. Design PDA for $L = \{a^n b^m c^n \mid n,m \geq 1\}$



Equivalence of CFG & PDA

* Construction of PDA from given CFG.

Ex: Given a grammar,
$S \to OBB$, $B \to OS \mid 1S \mid 0$
find PDA.

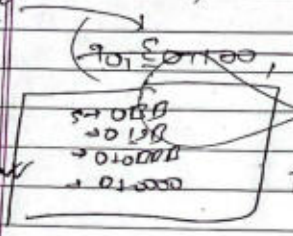
Sol: We define PDA as :

$P = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\}$

now, defined transition function δ as,

$\delta(q_0, \varepsilon, z_0) = (q_1, Sz_0)$
$\delta(q_1, \varepsilon, S) = (q_1, 0BB)$
$\delta(q_1, \varepsilon, B) = (q_1, 0S), (q_1, 1)$
$\delta(q_1, 0, 0) = (q_1, \varepsilon)$
$\delta(q_1, 1, 1) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, z_0) = (q_f, z_0)$

Example, for "01000".



$(q_0, 01000, z_0) \vdash (q_1, 01000, Sz_0)$
$\vdash (q_1, 01000, 0BBz_0)$
$\vdash (q_1, 1000, BBz_0)$
$\vdash (q_1, 000, Bz_0)$
$\vdash (q_1, 00, z_0)$
$\vdash (q_1, 0, z_0)$
$\vdash (q_1, \varepsilon, z_0)$
$\vdash (q_f, z_0)$

∴ So accepted

Construct PDA for $S \to aSa \mid bSb \mid c$.

Given grammar
$S \to aSa \mid bSb \mid c$

We define PDA as

$P = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$

We define δ as

$\delta(q_0, \varepsilon, z_0) = (q_1, Sz_0)$
$\delta(q_1, \varepsilon, S) = (q_1, aSa), (q_1, bSb), (q_1, c)$
$\delta(q_1, a, a) = (q_1, \varepsilon)$
$\delta(q_1, b, b) = (q_1, \varepsilon)$
$\delta(q_1, c, c) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, z_0) = (q_f, z_0)$

Now we simulate PDA for 'abbcbba',

$\delta(q_0, abbcbba, z_0) \vdash (q_1, abbcbba, Sz_0)$
$\vdash (q_1, abbcbba, aSa z_0)$
$\vdash (q_1, bbcbba, Sa z_0)$
$\vdash (q_1, bbcbba, bSba z_0)$
$\vdash (q_1, bcbba, Sba z_0)$
$\vdash (q_1, bcbba, bSbba z_0)$
$\vdash (q_1, cbba, Sbba z_0)$
$\vdash (q_1, bba, cbba z_0)$
$\vdash (q_1, ba, bba z_0)$
$\vdash (q_1, a, ba z_0)$
$\vdash (q_1, \varepsilon, a z_0)$
$\vdash (q_f, a z_0)$

CFG बनाउने PDA को ।

Construction of CFG from PDA.

We have,

PDA = $\{ Q, \Sigma, \Gamma, \delta, q_0, z_0, F \}$

CFG = $\{ V, \Sigma, P, S \}$

Terminal.

Terminal $T = \{ \Sigma \}$

starting symbol $S = \{ S \}$.

Suppose,

$Q = \{ q_0, q_1 \}$; Γ = a of a PDA

now, variable 'V' can be defined as,

$[Q \cdot \Gamma \cdot Q]$

$V = \{ S, [q_0, q q_1], [q_0 a q_0], [q_1 a q_0], [q_1 a q_1] \}$

Suppose, we have transition function δ as,

① $\delta(q_0, a, z_0) = (q_0, \varepsilon)$

$[q_0, z_0, q_0] \rightarrow a$ (pop)

② $\delta(q_0, a, z_0) = (q_0, q z_0)$ push

$[q_0, z_0, q_0] \rightarrow a [q_0, q, q_0] [q_0, z, q_0]$

some state

some state

③ $\delta(q_0, b, q) = (q_1, q)$ ⇒ similar.

$[q_0, q, q_1] \rightarrow b [q_1, q, q_1]$

some state

④ $\delta(q_0, q_1, z_0) = (q_0, a x x)$

$[q_0, z_0, q_1] \rightarrow a [q, q, q_1] [x, q, q_1] [x, q, q_1]$

Construct CFG from PDA:-

Q. Given PDA, $A = (Q, q_0, q_1, \{a, b\}, \{q, z_0\}, \delta, q_0, z_0, F)$ and δ is given by

$\delta(q_0, a, z_0) = (q_0, q, z_0)$
$\delta(q_0, a, q) = (q_0, qq)$
$\delta(q_0, b, q) = (q_1, \varepsilon)$
$\delta(q_1, b, q) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$

Sol^n (1) for $\delta(q_0, a, z_0) = (q_0, q z_0)$

~~$[q_0\ z_0]$~~

P1: $[q_0\ z_0\ q_0] \to a\ [q_0\ q\ q_0]\ [q_0\ z_0\ q_0]$

P2: $[q_0\ z_0\ q_0] \to a\ [q_0\ q\ q_1]\ [q_1\ z_0\ q_0]$
P3: $[q_0\ z_0\ q_1] \to a\ [q_0\ q\ q_0]\ [q_0\ z_0\ q_1]$
P4: $[q_0\ z_0\ q_1] \to a\ [q_0\ q\ q_1]\ [q_1\ z_0\ q_1]$

(2) for $\delta(q_0, a, q) = (q_0, qq)$

P5: $[q_0\ q\ q_0] \to a\ [q_0\ q\ q_0]\ [q_0\ q\ q_0]$
P6: $[q_0\ q\ q_0] \to a\ [q_0\ q\ q_1]\ [q_1\ q\ q_0]$
P7: $[q_0\ q\ q_1] \to a\ [q_0\ q\ q_0]\ [q_0\ q\ q_1]$
P8: $[q_0\ q\ q_1] \to a\ [q_0\ q\ q_1]\ [q_1\ q\ q_1]$

(3) $\delta(q_0, b, q) = (q_1, \varepsilon)$

P9: $[q_0\ q\ q_1] \to b$

(4) $\delta(q_1, b, q) = (q_1, \varepsilon)$

$[q_1\ q\ q_1] \to b$

(5) $\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$

$[q_1\ z_0\ q_1] \to \varepsilon$

(1) $S \to [q_0\ z_0\ q_0] \mid [q_0\ z_0\ q_1]$

q_0 initial state, start state

now, we reduce useless symbol (non reachable or non generating symbols)

Here, we remove (P2) & (P6) bcz $[q_1\ z_0\ q_0]$ & $[q_1\ q\ q_0]$ are non-generating symbols.

Remove करने के लिए पहले check करो non-generating symbols → फिर remove करो

Remove (P_5) b/c it is generating itself (so, it cannot derive a string)

Remove (P_1), (P_3), (P_7) b/c $[q_0\ a\ q_0]$ doesn't generate any symbol.

Remove $[q_0\ z_0\ q_0]$ from (3) b/c it doesn't generate any symbol. (non-generating symbol).

So, we have CFG,

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

where,

$V = \{S, [q_0, z_0, q_1], [q_0, a, q_1], [q_0\ z_0\ q_1]\ [q_1\ a\ q_1]\}$

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

$S = \{S\}$

P is given by

$S \to [q_0\ z_0\ q_1]$

$[q_0\ z_0\ q_1] \to a[q_0\ a\ q_1][q_1\ z_0\ q_1]$

$[q_0\ a\ q_1] \to a[q_0\ a\ q_1][q_1\ a\ q_1]$

$[q_0\ a\ q_1] \to b$

$[q_1\ a\ q_1] \to b$

$[q_1\ z_0\ q_1] \to \varepsilon$

Closure properties of CFL.

(1) Closed under union:

Let L_1 & L_2 be two CFL (Context free Lang) generated by the CFG (Context free Grammar) $G_1 = \{V_1, \Sigma_1, R_1, S_1\}$ and $G_2 = \{V_2, \Sigma_2, R_2, S_2\}$ respectively.

Now, we construct a new language $L(G)$ using the grammar $G = \{V, \Sigma, R, S\}$ such that it can accept $L(G_1) \cup L(G_2)$, where

$V = V_1 \cup V_2 \cup \{S\}$

$\Sigma = \Sigma_1 \cup \Sigma_2$

$R = R_1 \cup R_2 \cup \{S \to S_1 \mid S_2\}$

S = start state

Now

let us choose a string $\omega \in \{\Sigma_1 \cup \Sigma_2\}^*$

If $S_1 \Rightarrow \omega$ or $S_2 \Rightarrow \omega$

and, in our grammar
$S \to S_1 | S_2$, so (S) will lead the string (w).

ie $S \xrightarrow{*} w$.

Hence, $L(G) = L(G_1) \cup L(G_2)$ proved.

(2) closure under concatenation

let (L_1) & (L_2) be two CFL generated by $G_1 = \{V_1, \Sigma_1, R_1, S_1\}$ & $G_2 = \{V_2, \Sigma_2, R_2, S_2\}$ respectively.

now we construct new language $L(G)$ using the Grammar (G)
$G = \{V, \Sigma, R, S\}$ such that it can accept $L(G_1).L(G_2)$
where, $V = V_1 \cup V_2 \cup \{S\}$
$\Sigma = \Sigma_1 \cup \Sigma_2$
$R = R_1 \cup R_2 \cup \{S \to S_1 S_2\}$
S = start Symbol

now, let us choose a string $w_1 \in \Sigma_1$ and $w_2 \in \Sigma_2$.
we know, that $S_1 \xrightarrow{*} w_1$ and $S_2 \xrightarrow{*} w_2$,
but in the above grammar (G),
$S \to S_1 S_2$.

So, (S) will Read the concatenation of the string (w_1) and (w_2) i.e. (w_1, w_2). and the language will be (L_1, L_2).

Hence, (L_1, L_2) is also a Context Free Lang. (CF

(3) closed under Kleen star :

let (L_1) be a context free language by CFG (context free Grammar) $G_1 = \{V_1, \Sigma_1, R_1, S_1\}$. now, we construct a new language $L(G_2)$ using the Grammar (G) $G = \{V, \Sigma, R, S\}$ such that it can accept kleene star of the language (L_1) ie L_1^*.
where $V = V_1 \cup \{S\}$
$\Sigma = \Sigma_1 \cup \{\lambda\}$
$R = R_1 \cup \{S \to \lambda, S \to S_1 S\}$
S = start Symbol,

CamScanner

Here, R follows all the properties of CFG & R_2 is a production of given CFG L

$S \to \wedge$, $S \to SS$ also fulfill the requirement so, we say that G is a CFG that can generate CFL (L^*)

(3) Context free languages are not closed under intersection.

We know that, $L_1 = \{a^n b^n c^i \mid n \geq 1, i \geq 0\}$ and $L_2 = \{a^j b^n c^n \mid n \geq 1, j \geq 0\}$ are context free language.

Now, $L_1 \cap L_2 =$

$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$

Say $L = L_1 \cap L_2$

$\therefore L = \{a^n b^n c^n\}$, is not CFL

bcz PDA नहीं बनेगा।

Hence, $L_1 \cap L_2$ is not context free.

(5) Context free language are not closed under complement.

Suppose L_1 & L_2 are CFL.

Assume $\overline{L_1}$ & $\overline{L_2}$ are context free language.

now using De-morgan's - law

$L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$ - (2)

(not CFL) (CFL) (CFL)

We know that CFL is closed under union and by our Assumption, we have CFL closed under complement. But CFL are not closed under intersection.

(intersection ≠ ∪)

Hence by contradiction, CFL is not closed under complement.

Decision properties of CFL. (Algo)

Deciding whether a CFL is finite.

Construct a non-redundant CFG (G) generating L-{}. (null ∉ or epsilon ∉)
we draw a directed graph whose vertices are variables in (G).
If A → BC is a production, then there are directed edges from A to B and A to C.
(L) is finite if and only if directed graph has no cycle.

Deciding whether a CFL is empty.

Given any CFL (L), there is a CFG (G) to generate it.
we can determine, using the construction described in the context of elimination of useless symbols, whether the start symbol is useless;
if so, then language L(G) = ϕ (empty set);
otherwise, not.

3. Deciding whether a string 'w' can be generated by same CFG. { any string is member of CFL }

→ To determine whether a particular string (w) is in Language of (G), we will inspect all derivation trees of height at most length of string (w).
If string (w) is found in any of the derivation tree, then
w ∈ L.

Pumping lemma for CFL.

(Repeated if string or symbol)

statement:-
Let L be a context free language and 'n' be the length of string or pumping length such that

(1) Every z ∈ L and z ≥ n can be written as uvwxy for some u, v, w, x, y.
(2) |vx| ≥ 1
(3) |vwx| ≤ n . i.e. if u, y are 1, then |vwx| =
(4) uv^kwx^ky ∈ L for all k ≥ 0 (i.e generate infinite number of strings by setting any value for k)

Ans

Proof:

To proof the theorem we consider a CFG whose production are given as

$S \to AB$

$A \to aBb \mid a$

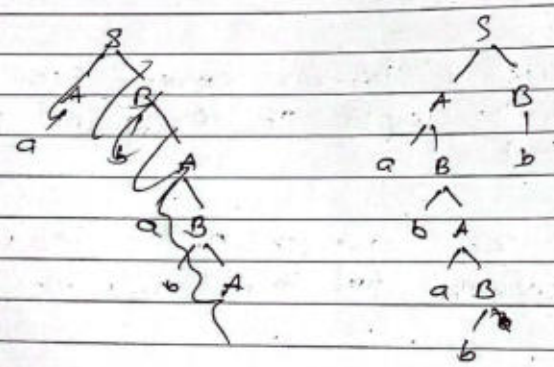
$B \to bA \mid b$

now, let any string $z = abbabb$ such that $z \in L$

Thus, we decompose z as

$u = a,\ v = bb,\ w = b,\ x = \Lambda,\ y = b.$

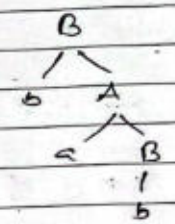
now, draw a parse tree for the given string z



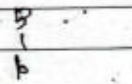
$\therefore T_1 : z = abbabb$

$|vwx| \leq n$

$|vx| \geq 1$



$\therefore T_1 : z_1 = bab$



$T_2 : z_2 = b$

as, z of z_1 are derivation of (T) as a proper sub-tree T_1 of T,

so, we can write

$z = uvwxy$

$z = uz_1y$

as, z_1 and z_2 are derivation of T_1 of a proper sub tree T_2 of T_1,

so we can write

$z_1 = vz_2x$

$\Rightarrow (\because z_2 = b)\ (\because vwx)$

also, we have $|vwx| > |w|$

so, we can write $|vx| \geq 1$

Thus, we have $z = uvwxy$

with

$|vwx| \leq n$ & $|vx| \geq 1$

∴ Here, "n" is length of string (z).

as, T is S-tree (starting from S)

T_1 and T_2 are B-tree (starting from symbol 'B')

we get

$z = u z_1 y$

$= u B y$

$z_1 = v z_2 x$

$= v w x$

$= v B x$

$S \xrightarrow{*} uBy$

$B \xrightarrow{*} vBx$

$B \rightarrow w$.

now,

$S \rightarrow uBy$

$\rightarrow uvBxy$ [$B \rightarrow vBx$]

$\rightarrow uvwxy$ [$B \rightarrow w$]

so, $S \rightarrow uv^k wx^k y$ where $k = 1$,

IIIly,

$S \rightarrow uBy$

$\rightarrow uvBxy$ [$B \rightarrow vBx$]

$\rightarrow uv\,vBx\,xy$ ["]

$\rightarrow uv^2Bx^2y$

$\rightarrow uv^2wx^2y$ ($B \rightarrow w$)

and, so on,

Thus, $uv^k \cdot wx^k y \in L$ for $k \geq 0$ ($k \geq 0$)

Hence proved

Procedure to prove given language is not CFL. (By contradiction)

step1:
Assume L is context free. let, 'n' be pumping length.

step2:
Choose $z \in L$ so that $|z| \geq n$ (length of z). write $z = uvwxy$ using the pumping lemma.

step3:
Find a suitable (k) so that $uv^k wx^k y \notin L$

This a contradiction so, L is not CFL.

#

CamScanner

Q. Show that $L = \{a^n b^n c^n \mid n \geq 1\}$ is not CFL.

Sol: Assume L is CFL, let, k be pumping length.

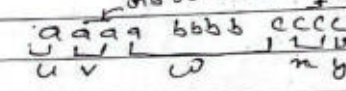
Step 1:- let, $z = a^n b^n c^n$, then, $|z| \geq 3n \geq n$.
write $z = uvwxy$, where $|vx| \geq 1$

Step 2:- Here, $z = uvwxy$
so, $uvwxy = a^n b^n c^n$

As $1 \leq |vx| \leq n$, v or x contain all three symbols.

Suppose, $n = 4$, so, $z = a^4 b^4 c^4$

case (I) :- v & x, each contain only one type of symbol

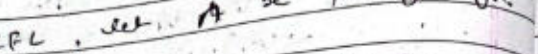


we have,
$uv^k wx^k y$

for $k = 2$
$uv^2 wx^2 y$
$= aaaaa\ abbbbc\ cccc$
$= a^6 b^4 c^6$

Here, no of a, b & c are not equal
so, $uv^k wx^k y \notin L$ for $k \neq 1$.

case (II) :- either v or x can have more than one symbol.



we have, $uv^k wx^k y$
for $k = 2$
$= uv^2 wx^2 y$
$= aa\ aabb\ aabb\ bbc\ cccc\ c$
$=$ [illegible] is pattern [illegible]

Since, it is not in pattern. so,
$uv^k wx^k y \notin L$ for $k = 2$.

Here, in both case we get contradiction.
So given language $L = \{a^n b^n c^n \mid n \geq 1\}$ is not CF

Unit 4

Turing machine

Turing machine :-

→ mathematical model of general computer.
→ TM can solve any problem that can be solved by any computer machine.
→ developed / designed by "Alan Turing".

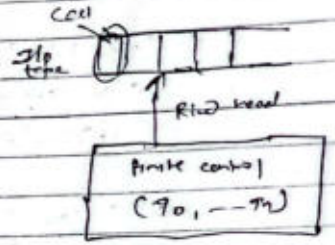


fig :- TM

→ TM has one ilp tape which is divided into number of cell, each cell can store only one symbol.
→ In one move, the machine read the present symbol under the read/write (R/W) head on the tape & the present state of finite control to determine :
- A new symbol to be written on the tape under the R/W head
- a motion of R/W head along the tape either the head move one cell left or one cell right.
- The next state of Automata.
- whether to ~~to halt or not~~ halt (stop) or not

Mathematically, TM has 7 tuples :-

$$M = \{ Q, \Sigma, \Gamma, \delta, q_0, b, F \}$$

where,
Q = set of finite state
Σ = ilp symbols
Γ = set of tape symbols
δ = transition function
q_0 = initial state
b = blank symbol ($b_1 B_1 \Delta, \#$)
F = set of final state.

← Transition process ~~of~~ describe as :-

Instantaneous Description (ID) :-
→ ID of Turing Machine 'M' is a string $(\alpha \beta \gamma)$, here
(β) is the present state of (M), the entire ilp string is split by (α) and (γ), The first symbol of (γ) is the current symbol under the R/W head of (γ) has all the symbol of ilp string, and the string (α) is the substring of ilp string formed by all the symbols to the

left of the present state.

present state	Tape symbol (input + blank symbol)		
	b	0	1
→ q_0	1 L q_2	0 R q_1	—
q_1	b R q_3	0 R q_2	1 L q_2
q_2	—	b R q_4	b R q_5
q_3	0 R q_5	0 R q_4	1 R q_4
q_4	0 L q_2	—	—
q_5			

Consider above TM, show computation of i/p string "001".

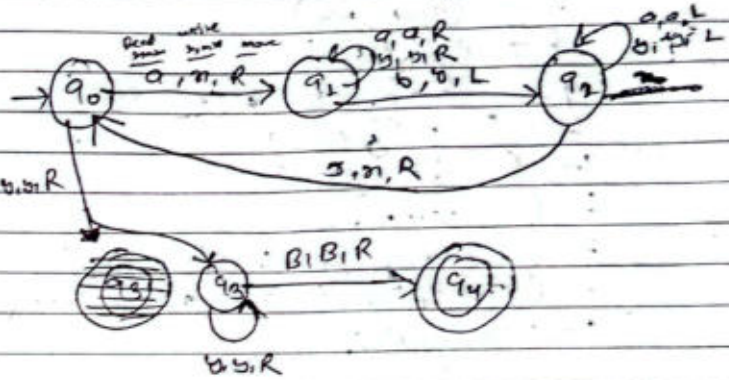
⇒ $(q_0, 001) \vdash 0q_1 01 \vdash 01q_0 1 \vdash 010q_2 b$
$\vdash 01q_3 0 1 \vdash 010q_5 b$.
$\vdash 01q_2 01 \vdash 010q_2 1 \vdash 01q_4 00$
$\vdash 010q_5 b \vdash 01q_2 001 \vdash 010q_2 100$
$\vdash 01q_3 0100 \vdash 010q_3 100 \vdash 01q_4 0100$
$\vdash 010q_4 100 \vdash 01q_2 0100 \vdash 010q_2 b00$
$\vdash 01q_4 0100 \vdash 010q_4 00 \vdash 01q_2 0000$
$\vdash 010q_2 b000 \vdash 01q_4 0000 \vdash 010q_4 1000$
$\vdash 01q_2 0000 \vdash 010q_3 1000 \vdash 01q_5 0000$

Since, q_5 is final state

∴ i/p accepted.

Design a TM :-

Q. Design a TM that accept $L = \{a^n b^n \mid n \geq 1\}$



Hence, Required TM, M is

$M = \{Q, \Sigma, \Gamma, \delta, q_0, b, F\}$
$Q = \{q_0, q_1, q_2, q_3, q_4\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, B, x, y\}$
$q_0 = \{q_0\}$
$F = \{q_4\}$

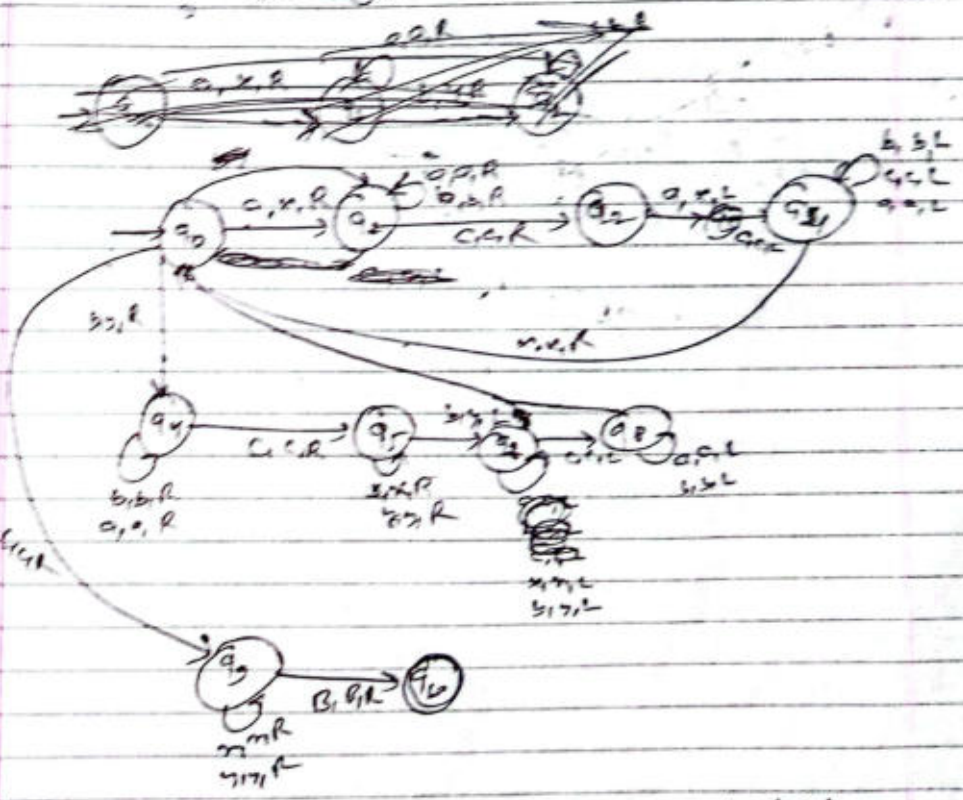
and δ is

State / Symbol	a	b	B	x	y
→ q_0	x R q_1	—	—	—	y R q_3
q_1	a R q_1	y L q_2	—	—	y R q_1
q_2	a L q_2	—	—	x R q_0	y L q_2
q_3	—	—	B R q_4	—	y R q_3
q_4 *	—	—	—	—	—

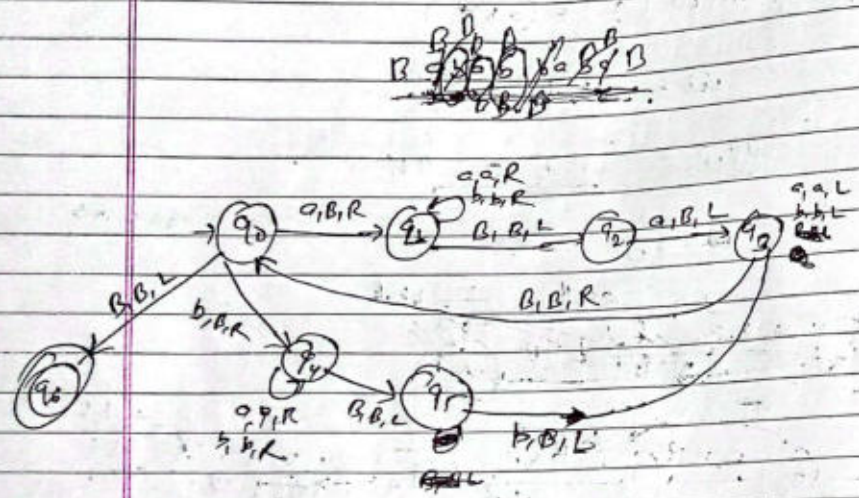
$L = \{ ww^R \mid w \in (a,b)^* \}$

$w = a^n b \ldots$

$ww^R = a^n b \ldots b a^n$



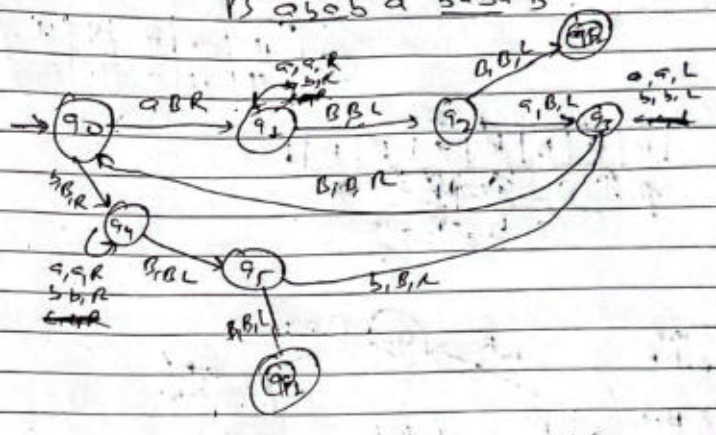
Q. Design a TM to check a string $w \in (a,b)^*$ is palindrome.



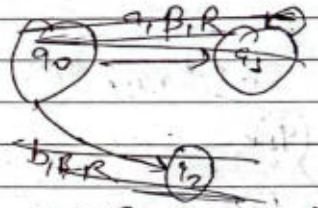
$(q_0, baab) \vdash$

Q. Design TM to check $w \in (a,b)^*$ is odd palindrome

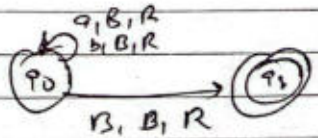
B a b a b a b a B



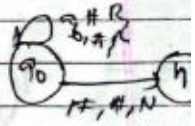
B a b a B



(1) Design Turing machine of string over Σ {a,b}



or

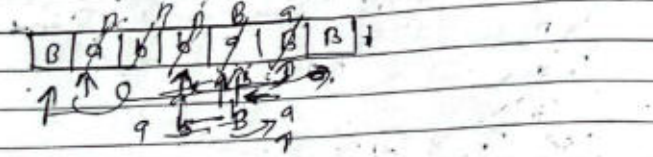


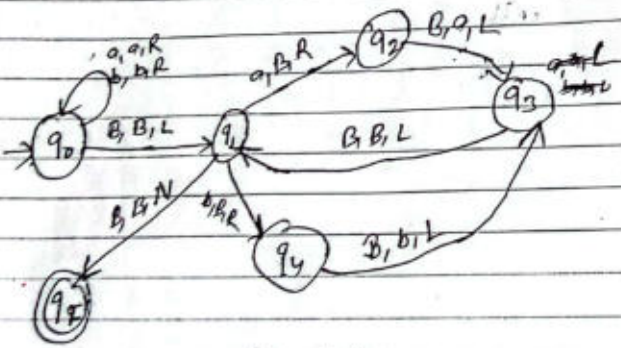
Blank Symbols

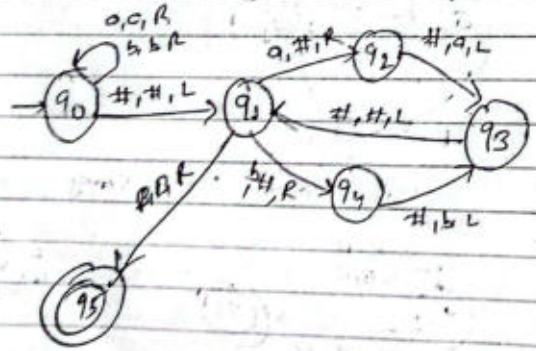
Ans

Q. Design a TM as a right shift machine that transform #w# to ##w# with $\Sigma = \{a, b\}$

	#	a	b	H	H
q_0	#	#	a	b	#







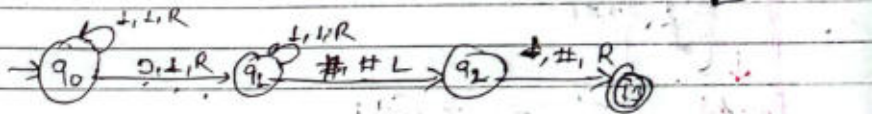
Computing function :-

A function $f(x) = y$ is said to be computable by turing machine if $(q_0, B x B) \vdash^{*} (q_0, B y B)$ (Blank Symbol)

It means, that if we input (x) to T.M, it gives output (y) as a string. If it compute function $f(x) = y$

T.M for addition of two number.

$f(n) = n + 3$, n_1, n_2, m th



Turing machine for (TM : m+n)

$(q_0, \$ 110 | 11 \# H) \vdash (\perp q_0 1 0 | 11 \# H)$

$\vdash (\perp 1 q_0 0 | 11 \# H)$

$\vdash (\perp 1 1 q_1 | 11 \# H)$

$\vdash (\perp 1 1 | q_1 1 1 \# H)$

$\vdash (\perp 1 1 | 1 q_1 1 \# H)$

$\vdash (\perp 1 1 | 1 1 q_1 \# H)$

$\vdash (\perp 1 1 | 1 1 \# q_2 H)$

$\vdash (\perp 1 1 | 1 1 \# H q_3)$

* TM for subtraction of two numbers:

$$f(m,n) = \begin{cases} m-n, & m \geq n \\ 0, & m < n \end{cases}$$

Eg:-