

Introduction

1/10

Date _____
Page _____

Review of sets, relation, function

① Sets: collection of objects

→ denoted by capital letters

Eg:

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

$$B = \{a, e, i, o, u\}$$

$$a \in B$$

$$1 \in A \quad |A| = 3$$

→ sets can be represented by

① listing its element

Eg: $V = \{a, e, i, o, u\}$

② by describing properties of element

$$V = \{x \mid x \text{ is a vowel letter}\}$$

$$N = \{n \mid n \text{ is natural number}\}$$

Types of sets

① Singleton sets: has only one element

② Empty sets: has no element, denoted by ϕ

③ Disjoint set: Two sets having no common elements

④ Overlapping set: having at least one common element

① Subset: if every element of A is in B then
 $A \subseteq B$

Additional term

① Cardinality: no. of element in set
Eg: $A = \{1, 2, 3\}$
 $|A| = 3$

② Cartesian Product:

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

③ Power set:

Eg: $A = \{1, 2\}$
 $P(A) = \{ \emptyset, \{1\}, \{2\}, \{1, 2\} \}$
Power set
 $2^n = 2^2 = 4$
 $n = \text{no. of element}$

Some Set Operations

① Union

$$A \cup B = \{x : x \in A \text{ or } x \in B\}$$

② Intersection

$$A \cap B = \{x : x \in A \text{ and } x \in B\}$$

③ Set Difference

$$A - B = \{x : x \in A \text{ and } x \notin B\}$$

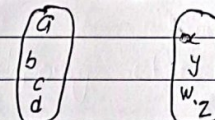
④ Complement

$$\bar{A} = \{\text{everything that is not in } A\}$$

Relation

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

Suppose S and T are two sets



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

Types of Relation:

1) Reflexive relation:

A relation 'R' is reflexive, if every element of set A is related to itself

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

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

2) Symmetric relation:

A relation 'R' is symmetric if $(a,b) \in R$ whenever $(b,a) \in R$

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

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

3) Transitive relation:

A relation 'R' is transitive if $(a,c) \in R$ 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 equivalence if it is reflexive, symmetric and transitive.

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

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

* Function

A function is said to map an element in domain to an element in range.

→ A function is denoted by $f(x)$

→ A function can be defined as

$f(x)$ or $f: x \rightarrow y$

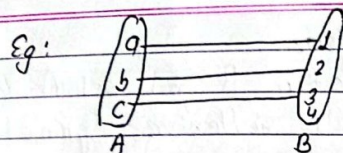
where $f(x)$ is called image of x under f .

* Types of function

(i) One to one function

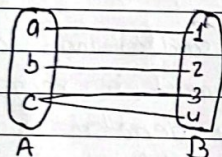
A function $f: A \rightarrow B$ is said to be one to one if every domain in A has different image in B.

↳ also known as injection function



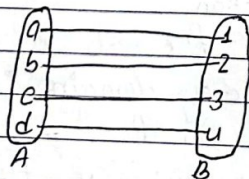
② Onto / ^{Sur}jection Function

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



③ One to One onto function

A function $f: A \rightarrow B$ 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.



also known as bijection function

* Alphabet and Languages

1) Alphabet:

→ symbol that can be letters or digits

→ denoted by Σ

Eg:

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

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

$$\Sigma = \{\#, *\}$$

2) String or Word:

→ finite sequence of concatenated symbol over alphabet Σ .

Eg:

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

Here '00', '1011', '1110', --- are strings over

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

3) Length of string

→ no. of symbol in a string w

Eg:

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

$$w = 0101$$

$$|w| = |0101| = 4$$

4) Empty string

→ string of zero length

→ denoted by ϵ or ε or λ or Λ (ε) (ϵ)

5) Substring

z is substring in w if z appears consecutively in w

Eg:

'01' is substring in '1011'

Concatenation of string

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

$\Sigma^* = \{\epsilon, 0, 1, 00, 10, 11, 01, \dots\}$

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

$L = \{\text{set of all string with equal no. of 0's and 1's}\}$

i.e. $L = \{01, 10, 0011, 1100, \dots\}$

Grammar

A grammar is a formal system for accepting or rejecting a string.

Proof of Techniques

1. proof by contradiction
2. ~~pigeon~~ pigeon hole 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/property is true by showing that assuming proposition is false.

Steps involve in proof of contradiction:

- a) Assume that statement to be proof be false
- b) In the next step we show that our supposition leads to logically contradiction [i.e. our supposition is false]
- c) In next step, we conclude that our statement is true because our supposition is false.

Eg:

Theorem: There is no greatest integer

In first steps assume the statement is false
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 integer.

Also, $M > N$ since $M = N + 1$

Thus M is an integer that is greater than the greatest integer N , which is a contradiction.

Hence our assume statement /supposition is not true and so there is no greatest integer.

Eg:

Give a proof by contradiction 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 squaring on both side

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

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

$$n^2 = 2(2k^2 + 2k) + 1 \text{ [Taking '2' as common]}$$

$$n^2 = 2r + 1 \text{ [general form of odd]}$$

$$\text{where } r = 2k^2 + 2k$$

This shows that n^2 is odd, which is a contradiction to our assumption or supposition.

Eg: If n is an integer and $n^3 + 5$ is odd then n is even using contradiction method

Suppose $n^3 + 5$ is odd then n is an odd integer

Here ~~$n^3 + 5 = 2k + 1$~~ $n = 2k + 1$

Now,

$$\begin{aligned} n^3 + 5 &= (2k+1)^3 + 5 \\ &= 2(4k^3 + 6k^2 + 3k + 3) \end{aligned}$$

Since this expression is multiple of 2 it is even.

However this given informat that $n^3 + 5$ is odd. Therefore our initial assumption that n is odd must be incorrect. Hence n must be even.

Assignment

Date _____
Page _____

1. IF n and m are odd integer, then nm is even integer by using contradiction

Suppose n and m are odd integers but nm is not even if n and m are odd they can be represented as $n = 2k+1$, $m = 2l+1$

Now lets suppose that nm is odd as well
 $nm = (2k+1)(2l+1)$
 $= 2(kl+k+l+1)$

But this implies that nm is also even, which contradicts our assumption

Therefore, if n and m are odd integer, then nm must be even.

2) $\sqrt{2}$ is irrational

Assume $\sqrt{2}$ is rational if can be expressed as a fraction $\frac{a}{b}$

$$\text{so } \sqrt{2} = \frac{a}{b}$$

Squaring both side we

$$2 = \frac{a^2}{b^2}$$

Assignment

Date _____
Page _____

$$a^2 = 2b^2$$

Since, a^2 is even (because it is divisible by 2)

a must be even as well

$$\text{let } a = 2k$$

Substitute $a = 2k$ into $a^2 = 2b^2$ we get

$$(2k^2) = 2b^2$$

$$4k^2 = 2b^2$$

divide by 2

$$2k^2 = b^2$$

This implies b^2 is even. So b must be even

However if both a and b are even then they have a common factor of 2 which contradicts the assumption that $\frac{a}{b}$ is in it's simplest form.

Therefore our assumption that $\sqrt{2}$ is rational must be false. Thus $\sqrt{2}$ is irrational.

3. If $3n+2$ is odd, then n is odd.

Assume that $3n+2$ is odd, but n is not odd.

If n is not odd then it must be even. So, we can represent n as $n=2k$ where k is an integer.

Substituting $n=2k$, we get $3(2k)+2=6k+2$.

Now analyze $6k+2$. This expression is even because it can be written as $2(3k+1)$. But

we assumed that $3n+2$ is odd. This contradicts our finding $6k+2$ is even.

Therefore, our initial assumption that n is not odd must be false. Thus if $3n+2$ is odd then n must be odd as well.

ii) Mathematical Induction

Steps:

1) Inductive hypothesis (assumption) $\rightarrow k$

2) Use $n=k+1$ in $P(n)$

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

Initial steps: For $n=1$, $\frac{1}{2} \times 1(1+1) = 1$

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

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

Stage 2:

For $n=k+1$,

$$\begin{aligned} 1+2+3+4+\dots+k+1 &= \frac{1}{2}(k+1)[k+1+1] \\ &= \frac{1}{2}(k+1)(k+2) \end{aligned}$$

Stage 3:

$$\begin{aligned} 1+2+3+4+\dots+k+1 &= 1+2+3+4+\dots+k+k+1 \\ &= \frac{1}{2}k(k+1) + (k+1) \\ &= (k+1) \left[\frac{1}{2}k+1 \right] \end{aligned}$$

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

2) Proposition: For all $n \geq 1$, prove that
 $1^2 + 2^2 + 3^2 + 4^2 + \dots + n^2 = \frac{n(n+1)(2n+1)}{6}$

Initial step: For $n=1$ then

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

$$= \frac{1 \cdot 2 \cdot 3}{6} = 1$$

Stage 1: Our assumption for $n=k$
 $1^2 + 2^2 + 3^2 + 4^2 + \dots + k^2 = \frac{k(k+1)(2k+1)}{6}$

Stage 2: For $n=k+1$

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

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

Stage 3: For

$$1^2 + 2^2 + 3^2 + \dots + (k+1)^2 = 1^2 + 2^2 + 3^2 + \dots + k^2 + (k+1)^2$$

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

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

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

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

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

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

Q. Prove that $1 \cdot 2 \cdot 3 + 2 \cdot 3 \cdot 4 + \dots + n(n+1)(n+2)$
 $= \frac{n(n+1)(n+2)(n+3)}{4}$

Initial step:

$$\text{For } n=1, \frac{n(n+1)(n+2)(n+3)}{4} = \frac{1 \cdot (1+1) \cdot (1+2) \cdot (1+3)}{4}$$

$$= 6$$

Stage 1: Our assumption for $n=k$

$$1 \cdot 2 \cdot 3 + 2 \cdot 3 \cdot 4 + \dots + k(k+1)(k+2) = \frac{k(k+1)(k+2)(k+3)}{4}$$

Assignment

Date _____
Page _____

Stage 2: For $n = k+1$,

$$1 \cdot 2 \cdot 3 + 2 \cdot 3 \cdot 4 + \dots + (k+1)(k+1+1)(k+1+2) \\ = \frac{(k+1)(k+1+1)(k+1+2)(k+1+3)}{4}$$

Stage 3:

$$1 \cdot 2 \cdot 3 + 2 \cdot 3 \cdot 4 + \dots + k(k+1)(k+2) + (k+1)(k+1+1)(k+1+2)$$

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

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

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

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

Thus $k+1$ is true whenever the statement is true for all natural number.

4/16

Date _____
Page _____

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 $n=1$,

$$P(1) = 7^1 - 3^1 = 4 \text{ which is divisible by 4}$$

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,

$$P(k) = 7^k - 3^k$$

$$= 4d, \quad d \in \mathbb{N}$$

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

$$P(k+1) = 7^{k+1} - 3^{k+1}$$

$$= 7^{k+1} - 7 \cdot 3^k + 7 \cdot 3^k - 3^{k+1}$$

$$= 7(7^k - 3^k) + 3^k(7 - 3)$$

$$= 7(7^k - 3^k) + 3^k \cdot 4$$

$$= 7 \cdot 4d + 3^k \cdot 4 \quad [7^k - 3^k = 4d]$$

$$= 4(7d + 3^k)$$

which is divisible by 4

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

1/16

Date
Page

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

$$\text{We have } P(n) = n(n+1)(n+5)$$

for $n=1$,

$$P(1) = 1(1+1)(1+5) = 12 \text{ which is the multiple of 3}$$

Thus $P(1)$ 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,

$$P(k) = k(k+1)(k+5) = 3d, \quad d \in \mathbb{N}$$

Now for some positive integer $k+1$

$$\begin{aligned} P(k+1) &= (k+1)(k+1+1)(k+1+5) \\ &= (k+1)(k+2)(k+6) \\ &= (k+1)(k^2+6k+2k+12) \\ &= (k+1)(k^2+8k+12) \\ &= (k+1)(k^2+5k+3k+12) \\ &= (k+1)(k(k+5)+3(k+4)) \\ &= k(k+1)(k+5) + 3(k+4)(k+1) \\ &= 3d + 3(k+4)(k+1) \end{aligned}$$

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

which is divisible multiple of 3

Thus given statement $P(n) = n(n+1)(n+5)$ is true for all positive $n \in \mathbb{N}$.

Date
Page

Pigeonhole principle:

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

→ If N object are to be placed in k boxes then there is at least one box containing N/k object.

Theorem: A function f from a set with $k+1$ element to a set with k element 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 element of x such that $f(x) = y$.
- (c) Since there are $k+1$ element and only k boxes, at least one box has two or more element.

Hence the function is not one to one.

Generalized: pigeon principle:-

If There are n pigeon holes and $k \cdot n + 1$ or more pigeons, then at least one pigeon hole is occupied by $k+1$ or more pigeon.

Example: How many students are there in a class among which at least four of them are born in same months.

$$\Rightarrow \begin{aligned} \text{Total no. of month (n)} &= 12 \quad (\text{Pigeon hole}) \\ \text{i.e. } k+1 &= 4 = 3+1 \\ \therefore k &= 3 \end{aligned}$$

$$\begin{aligned} \text{Total no. of student (pigeon)} &= k \cdot n + 1 \\ &= 3 \times 12 + 1 \\ &= 37 \end{aligned}$$

Q) 25 crates of apples are delivered to a store. The apples are of 3 different sorts and all the apple in each crates are of the same sort. So that among these crates there are at least 9 containing the same sorts of apples.

No. of different creates $(n) = 3$
 Total no. of creates $(kn+1) = 25$
 i.e. $k \cdot 3 + 1 = 25$
 $k = \frac{24}{3} = 8$

To Now, $k+1 = 8+1 = 9$

Thus there are at least 9 containing the same sorts of apples.

Diagonalization principle

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

→ use to prove undecidability
 (Halting problem)

In short, diagonalization principle state that the complement of diagonal is different from each row

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 of R

	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 , 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 1955
- Grammar is defined by four tuples
 $G = \{V, T, P, S\}$
 rule to define / generate language.

where

V = non-terminal or variables (capital letter)

T = Terminal (small letter)

P = Production rule

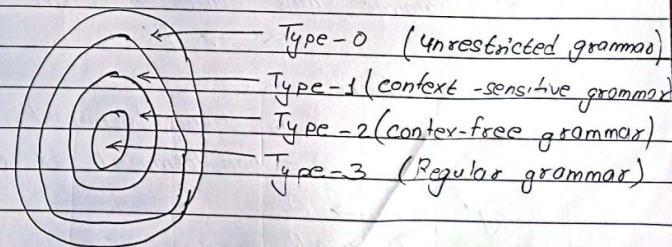
S = Starting Symbol

Eg: Production rule

$$S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow b$$



① unrestricted grammar generate recursive enumerable language (REL)

② Context sensitive grammar generate context sensitive language (CSL)

③ Context free grammar generate context free language

④ Regular grammar generate regular language (RL)

① Type-0 grammar:

→ include all formal grammar

→ all language are recognized by TM

→ describe syntax for programming language

→ Grammar has rule of form

$$\alpha \rightarrow \beta$$

where

α is non-terminal

β = non-terminal or terminal

Eg: $AB \rightarrow A$

$$A \rightarrow aB$$

$$B \rightarrow a$$

② Type 1 Grammar

→ generate context sensitive language

→ all language are recognized by LBA
(Linear Bound Automata)

→ grammar has rule of form with restriction
that length of $|\alpha| \leq |\beta|$

$$\alpha \rightarrow \beta$$

Eg: $AB \rightarrow A$ (not acceptable) ✗

$$A \rightarrow aB$$
 (acceptable) ✓

$$B \rightarrow a$$
 (acceptable) ✓

③ Type 2 Grammar

→ generate context free language

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

→ grammar has rule of form $A \rightarrow \alpha$
where

A = non-terminal

α = non-terminal or terminal

[There will be no context on the left & right
on the non-terminal]

Eg: $A \rightarrow ABC$ ✓

$$A \rightarrow BCD$$
 ✓

$$A \rightarrow Abc$$
 ✗

⑨ Type-3 grammar

- generate regular language
- all language are recognized by FA (Finite Automata)
- grammar has rule of form $\alpha \rightarrow \beta$

where α = non-terminal

β = non-terminal / terminal or ϵ

Eg: $S \rightarrow ab | \epsilon$

$A \rightarrow b$

$B \rightarrow \epsilon$

Finite Automata & Regular Expression

Finite Automata : (self-acting)

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

Formal definition:

FA has 5 types defined as

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

where

Q = set of finite state

Σ = input symbol

δ = transition function, $Q \times \Sigma \rightarrow Q$

q_0 = initial state

F = set of final states / accepted states

Types:-

- ① Deterministic FA (DFA): does not
- ② Non-deterministic

Types:

① Deterministic FA (DFA):

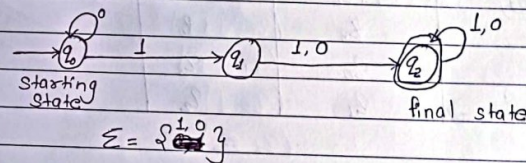
depend on current state and i/p

② Non-deterministic FA (NFA / NDA):
particular depend on current state and i/p

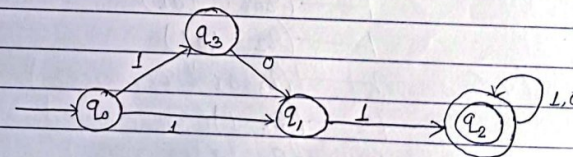
→ ○ initial state

→ ⊙ final state

Example: Deterministic FA



Example: Non-deterministic FA



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
$\rightarrow (q_0)$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Solution:

$q_0, 110101$ $\rightarrow (q_1, 10101)$
 $\rightarrow (q_0, 0101)$
 $\rightarrow (q_2, 101)$
 $\rightarrow (q_3, 01)$
 $\rightarrow (q_1, 1)$
 $\rightarrow (q_0, \epsilon)$

Since q_0 is final state
So string is accepted.

Assignment

Consider finite state Automata in which

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

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

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

$$F = \{q_2\} \text{ final state}$$

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

- Write a entire sequence of state for the input string 101101
- Check whether string is accepted or not
 - 11111
 - 00000
 - 101101

Solution

$a) (q_0, 101101) \rightarrow (q_1, 01101)$
 $\rightarrow (q_3, 1101)$
 $\rightarrow (q_2, 101)$

Assignment

Date _____
Page _____

└── (q₃, 01)
└── (q₁, 1)
└── (q₀, ε)

Here q₀ is final state so it is accepted and

the sequence of state is
{q₀, q₁, q₃, q₂, q₁, q₀}

b) i) (q₀, 11111) └── (q₁, 1111)
└── (q₀, 111)
└── (q₁, 11)
└── (q₀, 1)
└── (q₁, ε)

Here q₁ is final state so it is not accepted.

ii) (q₀, 000000) └── (q₂, 00000)
└── (q₀, 0000)
└── (q₂, 000)
└── (q₀, 00)
└── (q₂, 0)
└── (q₀, ε)

Here, q₀ is final state so it is accepted.

Assignment

Date _____
Page _____

iii) (q₀, 101101) └── (q₁, 01101)
└── (q₃, 1101)
└── (q₂, 101)
└── (q₃, 01)
└── (q₁, 1)
└── (q₀, ε)

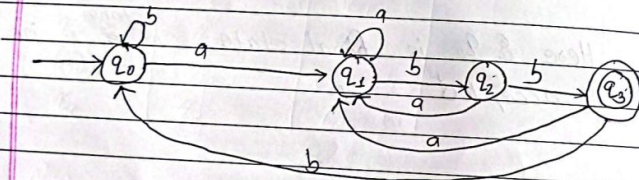
Here q₀ is final state so ^{string} is an accepted

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 = $3 + 1 = 4$

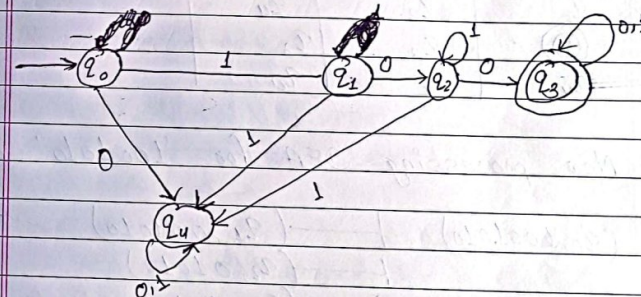


Q. Design a DFA that accept string of 0's and 1's starting with 100 over $\Sigma = \{0, 1\}$

given substring = 100

length of substring $|100| = 3$

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



Hence 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
$\rightarrow (q_0)$	q_1	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

$(q_0, 1001010)$	$(q_1, 001010)$
	$(q_2, 01010)$
	$(q_3, 1010)$
	$(q_3, 010)$
	$(q_3, 10)$
	$(q_3, 0)$
	(q_3, ϵ)

Hence, the string is accepted.

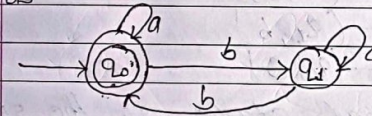
Q. Design a DFA that accepts language which set string in $(a, b)^*$ that have even no. of b's

possible language of even b's - $\{bb, bab, aabb, babababa\}$

length of substring $|bb| = 2$

no. of state = $2+1 = 3$

Now state transition diagram can be drawn as



Here

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

Now processing DFA for 'aabb' as

$\delta(q_0, aabb) \vdash (q_0, abb)$
 $\vdash (q_0, bb)$
 $\vdash (q_1, b)$
 $\vdash (q_0, \epsilon)$

Here q_0 is final state so the string is accepted.

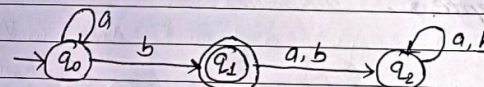
Q. Design a DFA that accept language :
 $L = \{a^n b ; n \geq 0\}$

Possible language, $L = \{b, ab, aab, \dots\}$

Length of string = $|b| = 1$

no. of state = $1+1 = 2$

State transition diagram:



Here 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

Assignment

Date _____
Page _____

Now, processing DFA for 'aaab' as

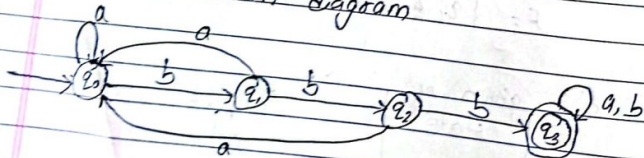
$(q_0, aaab) \rightarrow (q_0, aab)$
 $\rightarrow (q_0, ab)$
 $\rightarrow (q_0, b)$
 $\rightarrow (q_1, \epsilon)$

Since q_1 is final state so it is accepted.

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

Possible language, $L = \{bbb, abbb, \dots\}$
 length of substring, $|bbb| = 3$
 no. of state = $3+1 = 4$

State transition diagram



Assignment

Date _____
Page _____

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

where

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

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

State Transition table (δ)

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

Now, processing DFA for 'abbbab' as

$(q_0, abbbab) \rightarrow (q_0, bbbab)$
 $\rightarrow (q_1, bbbab)$
 $\rightarrow (q_2, bbbab)$
 $\rightarrow (q_3, ab)$
 $\rightarrow (q_3, b)$
 $\rightarrow (q_3, \epsilon)$

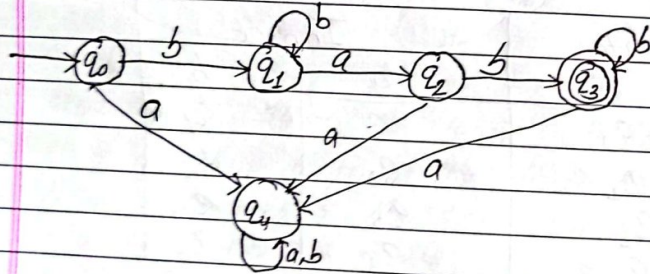
Since q_3 is final state so, the string is accepted!

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

Solution:

Possible language, $L = \{ab, bbaab, \dots\}$
 Length of substring $|bab| = 3$
 no. of state $= 3 + 1 = 4$

State Transition Diagram:



We know, M is defined as

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

where,

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

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

State transition table (S):

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

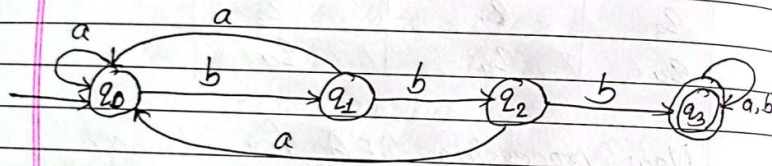
Now processing DFA for 'bbbab'

$(q_0, bbbab) \rightarrow (q_1, bbab)$
 $\rightarrow (q_1, bab)$
 $\rightarrow (q_1, ab)$
 $\rightarrow (q_2, b)$
 $\rightarrow (q_3, \epsilon)$

Since q_3 is final state
 so this string is accepted.

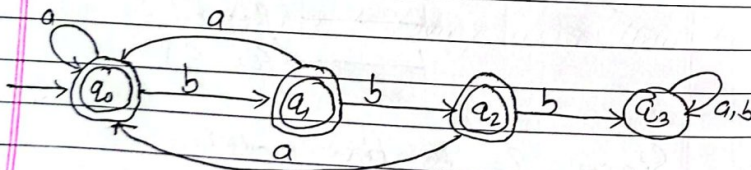
Q. Design DFA that accept all string that does not have three consecutive b's over $\Sigma = \{a, b\}$

firstly state transition diagram for having three consecutive b's



Now,

state transition diagram for does not have three consecutive b's



Here DFA, M is defined as

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

where,

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

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

$$q_0 = \{q_0\}$$

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

State transition table (δ):

State	Input	
	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_0	q_3
q_3	q_3	q_3

Now checking for DFA 'abababb'

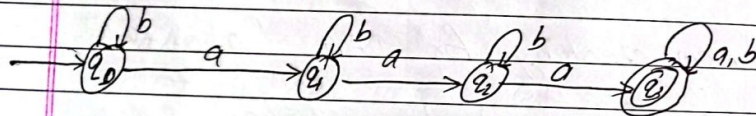
$(q_0, abababb) \xrightarrow{a} (q_0, bababb)$
 $\xrightarrow{b} (q_1, ababb)$
 $\xrightarrow{a} (q_0, babb)$
 $\xrightarrow{b} (q_1, abb)$
 $\xrightarrow{a} (q_0, bb)$
 $\xrightarrow{b} (q_1, b)$
 $\xrightarrow{a} (q_2, \epsilon)$

Here q_2 is final state so, the string is accepted.

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

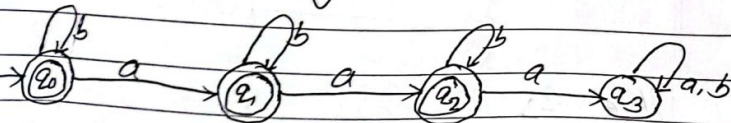
Firstly for ~~more~~ string have more than two or possible substring, $L = \{aa, baa, baab, \dots\}$
length of substring $|aa| = 2$
no. of state = $2+1 = 3$

Transition state diagram:



Now For string not having more than two a's

State transition diagram:



Here, M is defined as
 $M = \{Q, \Sigma, q_0, F\}$

where

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

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

$q_0 = \{q_0\}$

$F = \{q_3, q_1, q_2\}$

State transition table (δ)

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

Now, processing for DFA 'abbaba'

$(q_0, abbaba)$	_____	$(q_1, bbaba)$
	_____	$(q_1, baba)$
	_____	(q_1, aba)
	_____	(q_2, ba)
	_____	(q_2, a)
	_____	(q_3, ϵ)

Here q_3 is final state so the string is acceptable

Equivalence of DFA and NFA

- Q. Construct 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

For NFA,

Here initial state = q_0

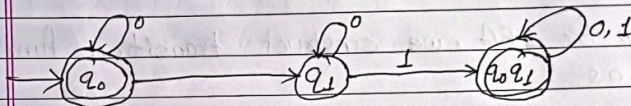
final state = q_0

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

Note: Initial state of NFA is same as DFA

So, state diagram for DFA is



Here final state of DFA is

$$F' = 2^2 - 1 = 2^2 - 1 = 2$$

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

- 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_0, q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_2
q_3	q_3	q_3

Here for NFA

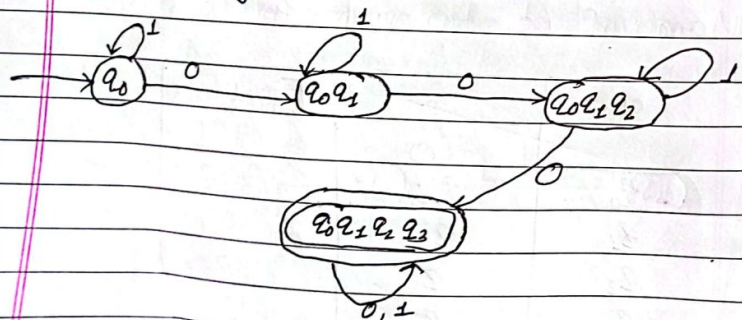
initial state = q_0

final state = q_3

Now for DFA, we construct transition function δ as

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



$\therefore$ final state of DFA is $\{q_0q_1q_2q_3\}$

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

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

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

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

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

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

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

Solution:

For NFA,

initial state = q_1

final state = q_3

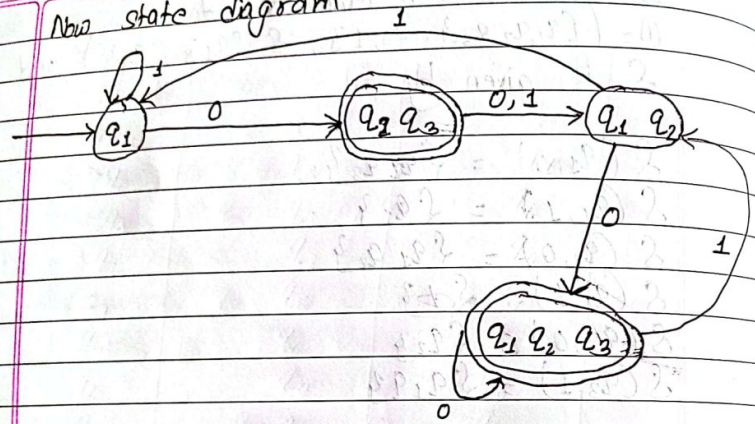
For DFA,

initial state = q_1 (same as NFA)

State transition table (δ) for DFA is

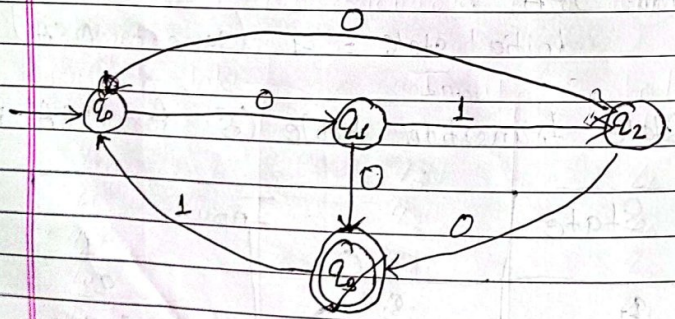
State	Input	
	0	1
q_1	q_2, q_3	q_1
q_2, q_3	q_1, q_2	q_1, q_2
q_1, q_2	q_1, q_2, q_3	q_1
q_1, q_2, q_3	q_1, q_2, q_3	q_1, q_2

New state diagram



∴ The final state for DFA: $q_2q_3, q_1q_2q_3$

2. Construct a DFA equivalent to given NFA



Solution For NFA,

initial state = q_0

final state = q_3

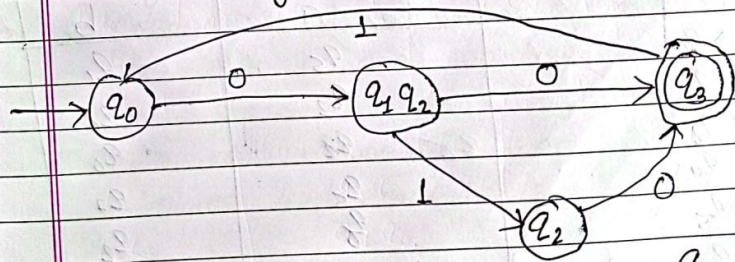
For DFA

initial state = q_0 (same as NFA)

New state transition for DFA is

State	Input	
	0	1
q_0	q_1q_2	ϕ
q_1q_2	q_3	q_2
q_3	ϕ	q_0
q_2	q_3	ϕ

New state diagram:



∴ The final state for DFA is q_3 .

Minimization of DFA

Example:

Current State	Input Symbol	
	a	b
(Initial) $\rightarrow q_0$	q_5	q_1
q_1	q_2	q_6
(Final) $*q_2$	q_1	q_0
q_4	q_5	q_7
q_5	q_6	q_1
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	b
$\rightarrow q_0$	q_5	q_1
q_1	q_2	q_6
$*q_2$	q_1	q_0
q_4	q_5	q_7
q_5	q_6	q_1
q_6	q_4	q_6
q_7	q_2	q_6

Step 2: Now divide the table into two sets as

a) set 1: containing row which start from non-final state.

$\rightarrow q_0$	q_5	q_1	Row 1
q_1	q_2	q_6	Row 2
q_4	q_5	q_7	Row 3
q_5	q_6	q_1	Row 4
q_6	q_4	q_6	Row 5
q_7	q_2	q_6	Row 6

Set 1

b) set 2: containing row which start from final state.

$*q_2$	q_1	q_0
--------	-------	-------

Step 3: Considering set 1

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

$\rightarrow q_0$ q_5 q_1
 q_3 q_2 q_6
 q_0 q_5 q_1
 q_5 q_5 q_2
 q_5 q_0 q_6

Again row 1 and row 3 is same.
So we remove q_4 and replace with q_0

$\rightarrow q_0$ q_5 q_1
 q_1 q_2 q_6
 q_5 q_6 q_2
 q_6 q_0 q_6

Now there are no similar rows

Consider set 2

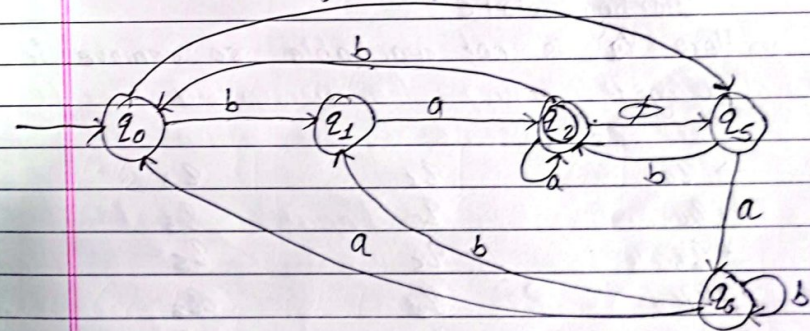
Here there is only one row. so it is minimized

* q_2 q_2 q_0

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

Current State	Input Symbol	
	a	b
$\rightarrow q_0$	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

Now, State diagram:



Q. Minimize the DFA

Current State	Input a	Symbol b
→ q ₀	q ₁	q ₃
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

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

Here q₂ is not reachable so remove it we get

→ 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 q₄ is not reachable so also remove it

→ 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 set as

a) Set 1: containing row which start from non-final state

→ q ₀	q ₁	q ₃
q ₁	q ₀	q ₃

b) Set 2: containing row which start from final state

* q ₃	q ₅	q ₅
* q ₅	q ₅	q ₅

Step 3: Considering set 1
Here the row of set 1 is already
minimized

→ q_0	q_1	q_3
q_1	q_0	q_3

Considering set 2

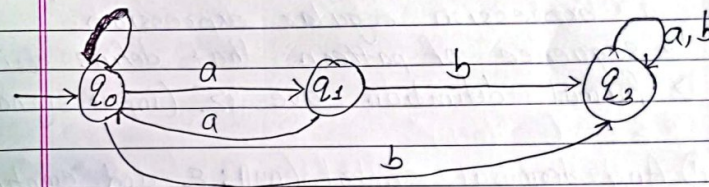
Here row 1 and row 2 are same so
we remove q_1 and replace with q_3

* q_3	q_3	q_3
---------	-------	-------

Step 4: Now combining the minimized set 1
and set 2 as

Current State	Input symbol	
	a	b
→ q_0	q_1	q_3
q_1	q_0	q_3
* q_3	q_3	q_3

State diagram:



Regular Expression

- ↳ language accepted by FA can be represented as expression regular expression
- ↳ sequence of pattern that define string
- ↳ formal definition over Σ (input signal),

- Any terminal symbol (null: ϵ and empty set: ϕ) are regular expression.
- The union of two regular expression R_1 and R_2 written as $R_1 + R_2$ is also a regular expression.
- Concatination of two regular expression R_1, R_2 is also a regular expression.
- The iteration (Kleene closure) of regular expression written as R_1^* is also a regular expression.

Eg: $\Sigma = \{a\} = a^* = a$ all repetition
 $L = \{a^*\}$
 $L = \{\epsilon, a, aa, aaa, \dots\}$
 $R.E = a^*$

$*$ $\rightarrow \epsilon$ include Σ^0
 $+$ $\rightarrow \epsilon$ include Σ^1
 $L = \{a, aa, aaa, \dots\}$
 $R.E = a^+$

Example 1: Let $\Sigma = \{a, b\}$. Find regular expression for language L whose length is exactly 2.

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

union of all string
 $= aa + ab + ba + bb$
 $= a(a+b) + b(a+b)$
 $= (a+b)(a+b)$

Example 2: For at least length 2

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

For exactly length 2
 $R.E = (a+b)(a+b)$

Now for at least

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

Regular Expression

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

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

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

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

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

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

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

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

$$\begin{aligned} R.E &= \{(0+1)(0+1)\}^* \\ &= \{(0+1)^2\}^* \\ &= \{(0+1)^{2*}\} = (0+1)^{2n}, n \geq 0 \end{aligned}$$

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

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

Identities of R.E

$$1. \phi + R = R$$

$$2. \phi R + R \phi = \phi$$

$$3. \epsilon R = R \epsilon = R$$

$$4. \epsilon^* = \epsilon \text{ and } \phi^* = \epsilon$$

$$5. R + R = R$$

$$6. R^* R^* = R^*$$

$$7. R R^* = R^* R$$

$$8. (R^*)^* = R^*$$

$$9. \epsilon + R R^* = \epsilon + R^* R = R^*$$

$$10. (PQ)^* P = P(PQ)^*$$

$$11. (P+Q)^* = (P^*Q^*)^* = (P^*+Q^*)^*$$

$$12. (P+Q)R = PR + QR$$

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

Eg: Prove that $(1+00^*1) + (1+00^*1)(0+10^*1)^*$
 $(0+10^*1) = 0^*1(0+10^*1)^*$

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

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

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

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

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

$$= RHS \quad \#$$

Imp

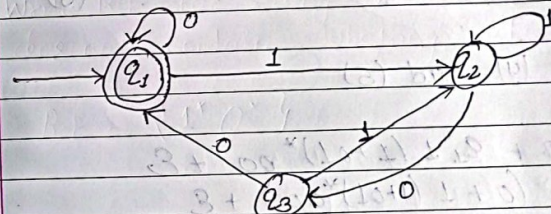
Arden's Theorem

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

then

$$R = QP^*$$

Q. Find R.E from the following state diagram



Soln

Here,

$$q_1 = q_10 + q_20 + \epsilon \quad \text{--- (1)}$$

$$q_3 = q_20 \quad \text{--- (3)}$$

$$q_2 = q_10 + q_21 + q_31 \quad \text{--- (2)}$$

Solving (1) and (3)

$$q_1 = q_10 + q_200 + \epsilon \quad \text{--- (4)}$$

Solving eqⁿ (2) and (3)

$$Q_2 = Q_1 1 + Q_2 1 + Q_2 01$$

$$\underbrace{Q_2}_R = \underbrace{Q_1 1}_Q + \underbrace{Q_2 1}_R + \underbrace{Q_2 01}_P$$

Now using arden's theorem, we get

$$Q_2 = Q_1 1 (1+01)^* \quad (5)$$

Solving (4) and (5)

$$Q_1 = Q_1 0 + Q_1 1 (1+01)^* 00 + \epsilon$$

$$Q_1 = Q_1 (0 + 1 (1+01)^* 00) + \epsilon$$

$$\underbrace{Q_1}_R = \underbrace{\epsilon}_Q + \underbrace{Q_1 (0 + 1 (1+01)^* 00)}_P$$

Using arden's theorem

$$Q_1 = \epsilon (0 + 1 (1+01)^* 00)^*$$

$$Q_1 = (0 + 1 (1+01)^* 00)^*$$

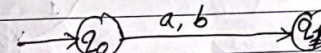
So the RE is $(0 + 1 (1+01)^* 00)^*$

Construction of FA from R.E

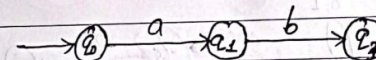
R.E

F.A

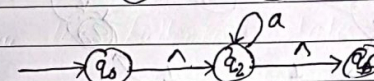
1. $a+b$



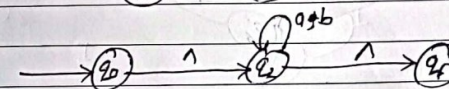
2. ab



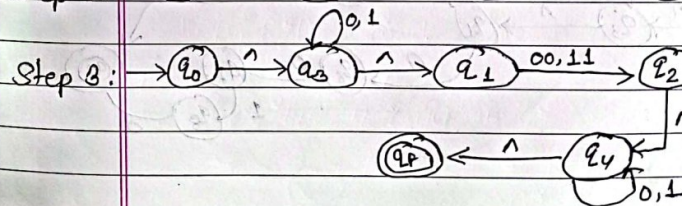
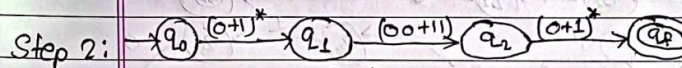
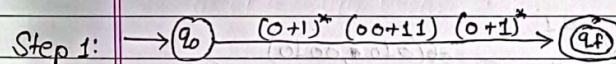
3. a^*



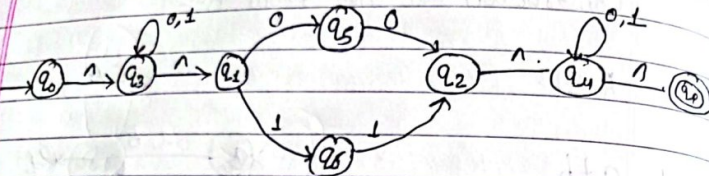
4. $(a+b)^*$



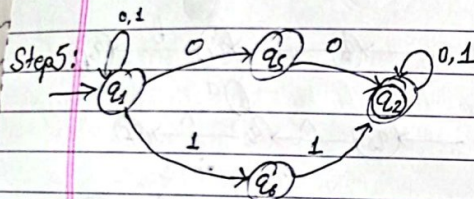
Q. Construct FA equivalent to RE:
 $(0+1)^* (00+11) (0+1)^*$



Step 4:

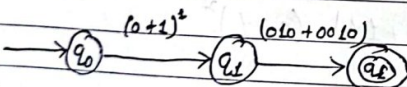


Step 5:

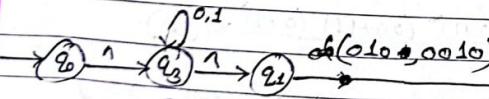


Q- Construct FA equivalent RE
 $(0+1)^* (010 + 0010)$

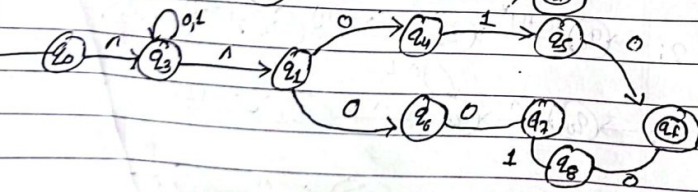
Step 1:



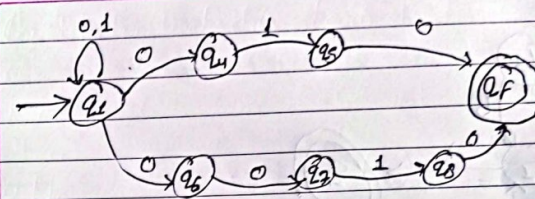
Step 2:



Step 3:



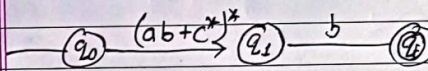
Step 4:



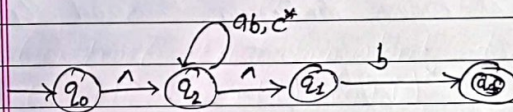
Assignment

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

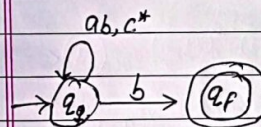
Step 1:



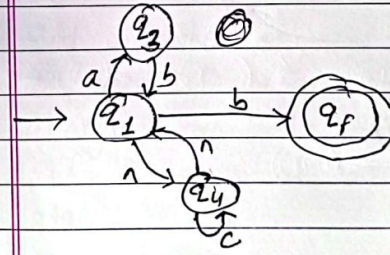
Step 2:

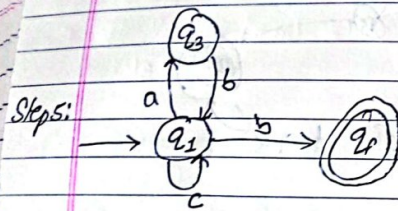


Step 3:



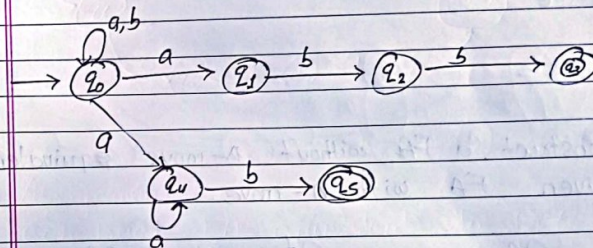
Step 4:





→ or

Q) (a/b) $(abb/a+b)$



Elimination of null move (n-move)

Suppose we replace a null move from a vector v_1 and v_2 then we process as follows:

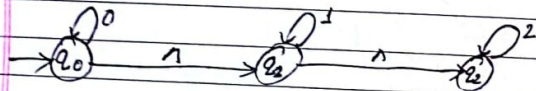
Process:

Replace

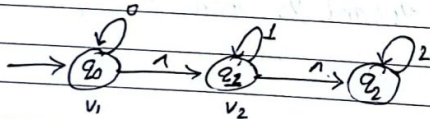
- 1) Find all the edges starting from v_1
- 2) Duplicate all edges of v_2 from v_1 without changing the edge label.
3. If v_1 is initial state make v_2 as initial state.

4. If v_2 is final state make v_1 as final state

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



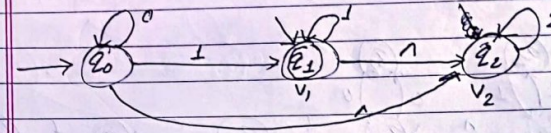
Step 1: Suppose Λ -move between q_0 and q_1



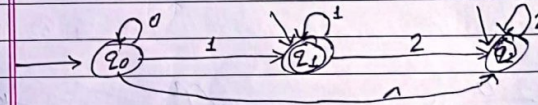
Step 2:



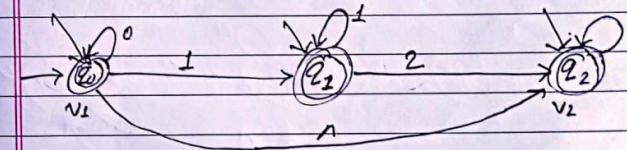
Step 3: Considering Λ -move between q_1 and q_2



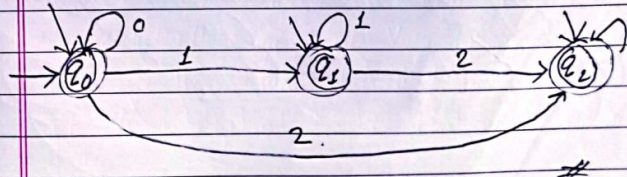
Step 4:



Step 5: Suppose Λ -move between q_0 and q_2



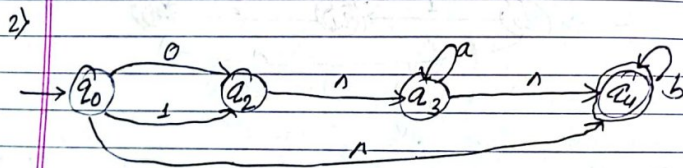
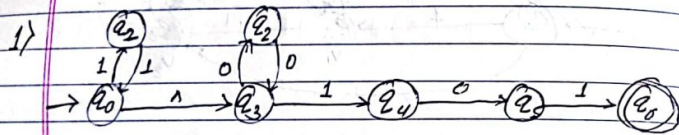
Step 6:



Assignment

Date _____
Page _____

Q1) Construct a FA without ϵ -move equivalent to given FA with ϵ -move



Q2) Reduce the given NFA with NULL to without null move.

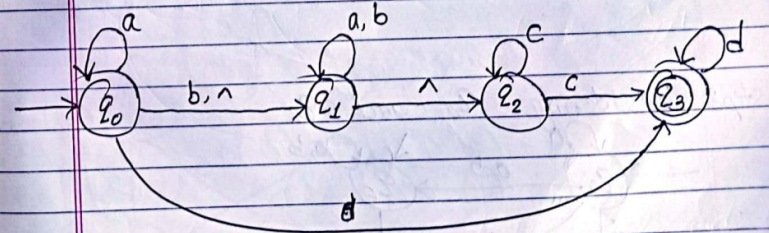
$M = \{Q, \{a, b, c, d\}, \delta, q_0, \{q_3\}\}$
state table

State	Input				
	a	b	c	d	ϵ
q_0	q_0	q_1	ϕ	q_3	q_1
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

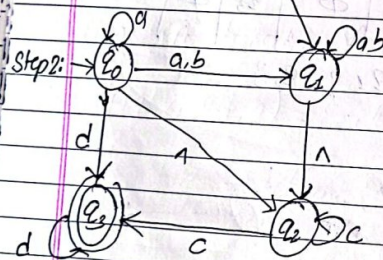
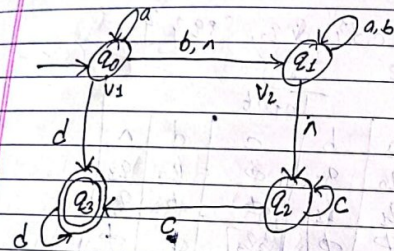
where $Q = \{q_0, q_1, q_2, q_3\}$

Solution

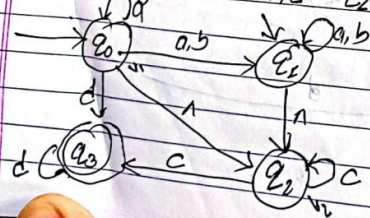
State diagram:



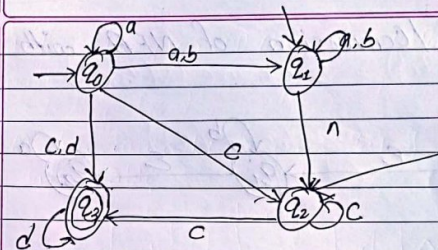
Step 1: Considering q_0 and q_1



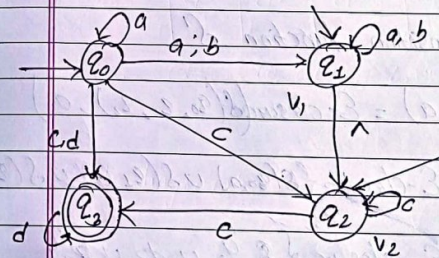
Step 3: Considering q_0 and q_2



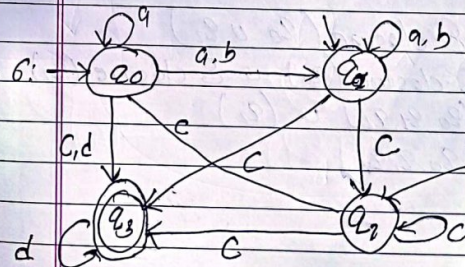
Step 4:



Step 5: Considering q_1 and q_2



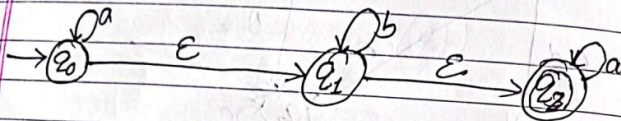
Step 6:



Note: ϵ -closure(ϕ) = ϕ

Date _____
Page _____

Consider / Conversion of NFA with ϵ to DFA.



$$\epsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}$$

$$\epsilon\text{-closure}(q_1) = \{q_1, q_2\}$$

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

Now, DFA transition function δ' ,

$$\begin{aligned} 1. \delta'(\{q_0, q_1, q_2\}, a) &= \epsilon\text{-closure}(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)) \\ &= \epsilon\text{-closure}(\{q_0, a\} \cup \{q_1, b\} \cup \{q_2, a\}) \\ &= \epsilon\text{-closure}(\{q_0 \cup \phi \cup q_2\}) \\ &= \epsilon\text{-closure}(q_0 \cup q_2) \\ &= \epsilon\text{-closure}(q_0) \cup \epsilon\text{-closure}(q_2) \\ &= \{q_0, q_1, q_2\} \cup \{q_2\} \\ &= \{q_0, q_1, q_2\} \end{aligned}$$

$$\begin{aligned} 2. \delta'(\{q_0, q_1, q_2\}, b) &= \epsilon\text{-closure}(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b)) \\ &= \epsilon\text{-closure}(\{q_0, b\} \cup \{q_1, b\} \cup \{q_2, b\}) \\ &= \epsilon\text{-closure}(\{q_1 \cup q_2\}) \\ &= \epsilon\text{-closure}(q_1) \\ &= \{q_1, q_2\} \quad [\text{New state}] \end{aligned}$$

$$\begin{aligned} 3. \delta'(\{q_1, q_2\}, a) &= \epsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a)) \\ &= \epsilon\text{-closure}(\{q_1, b\} \cup \{q_2, a\}) \\ &= \epsilon\text{-closure}(q_2) \\ &= \{q_2\} \quad [\text{New state}] \end{aligned}$$

$$\begin{aligned} 4. \delta'(\{q_1, q_2\}, b) &= \epsilon\text{-closure}(\delta(q_1, b) \cup \delta(q_2, b)) \\ &= \epsilon\text{-closure}(\{q_1 \cup \phi\}) \\ &= \epsilon\text{-closure}(q_1) \\ &= \{q_1, q_2\} \end{aligned}$$

$$5. \delta'(\{q_2\}, a) = \epsilon\text{-closure}(\delta(q_2, a))$$

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

$$= \{q_2\}$$

$$6. \delta'(\{q_2\}, b) = \epsilon\text{-closure}(\delta(q_2, b))$$

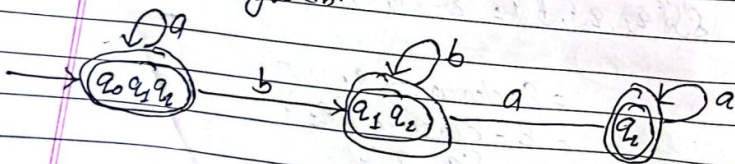
$$= \epsilon\text{-closure}(\emptyset)$$

$$= \emptyset$$

Now transition state

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

$\therefore$ state diagram:



Conversion of NFA with ϵ to DFA

Q. Transition table for NFA, δ is

	Input		
State	a	b	ϵ
$\rightarrow q_0$	q_0	$\{q_0, q_2\}$	q_1
q_1	q_1	$\{q_0, q_2\}$	$\emptyset$
q_2	q_2	$\emptyset$	$\emptyset$
(q_3)	$\emptyset$	$\emptyset$	q_4
(q_4)	q_3	$\emptyset$	$\emptyset$

Solⁿ

$$\epsilon\text{-closure}(q_0) = \{q_0, q_1\}$$

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

$$(q_2) = \{q_2\}$$

$$(q_3) = \{q_3, q_4\}$$

$$(q_4) = \{q_4\}$$

Now transition function δ' for DFA

$$1) \delta'(\{q_0, q_1\}, a)$$

Note: q_0 is start of NFA so we take $\epsilon\text{-closure}$ of q_0 as starting state of DFA

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

$$= \epsilon\text{-closure}(q_0 \cup q_1)$$

$$= \epsilon\text{-closure}(q_0) \cup \epsilon\text{-closure}(q_1)$$

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

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

[new state]

$$\textcircled{2} \delta'(\{Q_0, Q_1\}, b)$$

$$= E\text{-closure}(\delta(\{Q_0, Q_1\}, b))$$

$$= E\text{-closure}(\delta(Q_0, b) \cup \delta(Q_1, b))$$

$$= E\text{-closure}(\{Q_0, Q_2\} \cup \{Q_1, Q_4\})$$

$$= E\text{-closure}(Q_0) \cup E\text{-closure}(Q_2) \cup E\text{-closure}(Q_1) \cup E\text{-closure}(Q_4)$$

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

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

[new state]

$$\textcircled{3} \delta'(\{Q_0, Q_1, Q_4\}, a)$$

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

$$= E\text{-closure}(Q_0 \cup Q_4 \cup Q_2)$$

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

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

[new state]

$$\textcircled{4} \delta'(\{Q_0, Q_1, Q_4\}, b)$$

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

$$= E\text{-closure}(\{Q_0, Q_2\} \cup \{Q_1, Q_4\} \cup \emptyset)$$

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

$$\textcircled{5} \delta'(\{Q_0, Q_1, Q_2, Q_4\}, a)$$

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

$$= E\text{-closure}(Q_0 \cup Q_4 \cup Q_2 \cup Q_1)$$

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

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

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

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

$$= E\text{-closure}(\{Q_0, Q_2\} \cup \{Q_1, Q_4\} \cup \emptyset \cup \emptyset)$$

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

$$\textcircled{7} \delta'(\{Q_0, Q_1, Q_2, Q_4\}, a)$$

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

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

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

$$\textcircled{8} \delta'(\{Q_0, Q_1, Q_2, Q_4\}, b)$$

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

$$= E\text{-closure}(\{Q_0, Q_2\} \cup \{Q_1, Q_4\} \cup \emptyset \cup \emptyset)$$

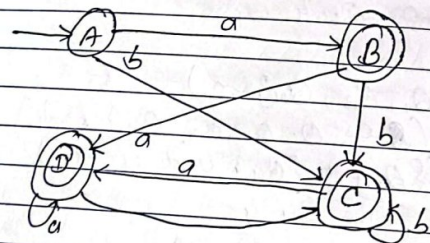
$$= \{Q_0, Q_1, Q_2, Q_4\}$$

Transition table for DFA, δ' is

State	Input	
	a	b
$\rightarrow \{Q_0, Q_1\}$	$\{Q_0, Q_1, Q_4\}$	$\{Q_0, Q_1, Q_2, Q_4\}$
* $\{Q_0, Q_4\}$	$\{Q_0, Q_1, Q_2, Q_4\}$	$\{Q_0, Q_1, Q_2, Q_4\}$
* $\{Q_0, Q_2, Q_4\}$	$\{Q_0, Q_1, Q_2, Q_4\}$	$\{Q_0, Q_1, Q_2, Q_4\}$
* $\{Q_0, Q_1, Q_2, Q_4\}$	$\{Q_0, Q_1, Q_2, Q_4\}$	$\{Q_0, Q_1, Q_2, Q_4\}$

Let $\{Q_0, Q_1\} = A$, $\{Q_0, Q_1, Q_2\} = B$,
 $\{Q_0, Q_1, Q_2, Q_3\} = C$ and
 $\{Q_0, Q_1, Q_2, Q_3, Q_4\} = D$

DFA is



Vimp The pumping lemma for regular sets
 (repetition)
 → to prove not regular

Theorem Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton with 'n' state that accept regular language 'L'. Let w be the string such that $w \in L$ and $|w| \geq m$. If $m \leq n$, then there exists x, y, z (substring) such that
 i) $w = xyz$
 ii) $y \neq \epsilon$ (y cannot be null)
 iii) $xy^iz \in L$ for each $i \geq 0$

Properties of regular expression or language

1. Commutative
2. Associativity
3. Identities
4. Annihilator
5. Idempotent law
6. Distributive law
7. Law of closure

Algebraic Rule / law for regular expression.

① Commutativity

This law means we can switch the order of its operand and get the same result. The union of regular expression is commutative but concatenation of regular expression is not commutative i.e. If x and y are regular expression representing like language $L(x)$ and $L(y)$ then $x \cup y = y \cup x$ i.e. $x \cup y = y \cup x$ but $x \cdot y \neq y \cdot x$

② Associativity

The union as well as concatenation of regular expression are associative i.e. If x, y, z are regular expression representing regular language

$L(t)$, $L(r)$ and $L(s)$ then
 $t + (r+s) = (t+r)+s$
 and $t \cdot (r \cdot s) = (t \cdot r) \cdot s$

(3) Distributive law

For any regular expression r, s, t representing regular language $L(r)$, $L(s)$ and $L(t)$ then
 $r(stt) = rs + rt$ — left distribution
 $(s+t)r = sr + tr$ — right distribution

(4) Identity law

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

$$r + \phi = \phi + r = r \text{ i.e. } \phi \cup r = r$$

ϵ is identity for concatenation i.e. $\epsilon \cdot r = r = r \cdot \epsilon$

(5) Annihilator

It is a value such that when the operator is applied to the annihilator and some other value, the result is annihilator ϕ is annihilator for concatenation i.e. $\phi \cdot r = r \cdot \phi = \phi$

(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

$$-(r^*)^* = r^*$$

$$-\text{Closure of } \phi = \phi^* = \epsilon$$

$$-\text{Closure of } \epsilon = \epsilon^* = \epsilon$$

$$-\text{positive closure of } r, r^+ = rr^*$$

Proof: Let $w = a_1 a_2 \dots a_m$, $m \geq 1$

$$S(a_1 a_2 \dots a_m) = a_i \text{ for } i=1, 2, 3, \dots, m$$

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

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

Now input string w can be decomposed into substring as

$$x = a_1 a_2 \dots a_j$$

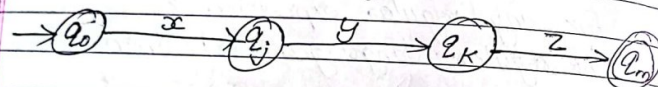
$$y = a_{j+1} \dots a_k$$

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

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.

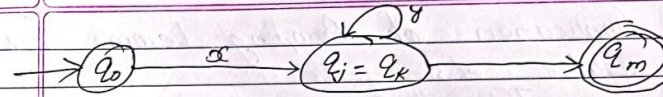
For w , the finite automata can be constructed as shown in figure.



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 unapplying the input string, the state become $n+1$. Thus among various pair of repeated state, 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, $n = 3$
i.e. $m \geq n$ (True)

Since,

$$w = xyz$$

$$= a_1 a_2 \dots a_j a_{j+1} \dots a_k a_{k+1} \dots a_m a_{m+1}$$

So we can write $0 \leq j \leq k \leq n$ and this implies that $|xy| \leq n$

Here $xy^i z \in L$ for each $i \geq 0$

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

and so on.

Hence we can conclude that this condition i.e. $xy^i z \in L$ 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 string. Hence $xy^i z \in L$. As every state in Q is obtained by applying an input symbol, $y \neq \epsilon$.

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:

Step 1: Assume that L is regular. Let n be no. of state in FA.

Step 2: Choose a string w set such that $|w| \geq n$. Use pumping lemma to write, $w = xyz$, $|xy| \leq n$ and $|y| > 0$ or $|y| \geq 1$.

Step 3: 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.

Step 1: Assume L is regular.

Step 2: Let $ww = a^n b a^n b$. 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 part of a

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

Case 2: y has only b 's

$$aaaaaa \underbrace{ab}_{x} \underbrace{aaaaaa}_{y} \underbrace{baaa}_{z} b$$

Case 3: y has both a and b

$$aaaaaa \underbrace{ab}_{x} \underbrace{a}_{y} \underbrace{aaaaaa}_{z} b$$

Step 3: we want to find i such that $xy^iz \notin L$

Take $i = 2$,

In case 1, xy^2z

$$\text{i.e. } xy^2z = xyyz \\ = aaaaaaaaaa b aaaaaa b \\ = a^{12} b a^7 b$$

This is not in pattern

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

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

Take $i = 3$

In case 2, xy^3z

$$= xyyy z$$

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

This is not in pattern

Thus $xy^iz \notin L$

Also

In case 2,

$$|xy| = 8, n = 7, |xy| \leq n \text{ (False)}$$

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

Soln

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 1: y has 0's parts

$$\text{i.e. } \underbrace{00}_x \underbrace{000}_y \underbrace{11111}_z$$

Case 2: y has only 1's

$$\text{i.e. } \underbrace{00000}_x \underbrace{11111}_y \underbrace{1}_z$$

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

$$\text{i.e. } \underbrace{0000}_x \underbrace{01111}_y \underbrace{1}_z$$

Step 3: We want to find i , such that $xy^iz \notin L$
Take $i = 2$

In case 1, xy^2z

$$= xy y z$$

$$= 00000000 11111$$

$$= 0^8 1^5$$

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 $i = 2$

In case 2, xy^2z

$$= xy y z$$

$$= 00000 11111111$$

$$= 0^5 1^7$$

not in pattern

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

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

Take $i = 2$

$$\begin{aligned} \text{In case 3, } xy^2z &= xy^2z \\ &= 00000101111 \\ &= 0^61^6 \end{aligned}$$

not in pattern so, $xy^2z \notin L$ for $i=3$
 $|xy| = 6, n = 5$ (false)

Thus in all cases, we get contradiction
So given language.

$L = \{0^n1^n \mid n \geq 1\}$ is not regular.

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

Solⁿ

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 $i = 2$

$$\begin{aligned} \text{In case 1, } xy^2z &= aaaaaaaaaaaaaa \\ &= a^{14} \end{aligned}$$

This is not prime. So not in pattern

$\therefore xy^2z \notin L$ also $|xy| = 7, n = 7$
 $|xy| \leq n$ (true)

Closure properties of regular set / language

- (2) If L_1 and L_2 are regular, then $L_1 \cdot L_2$ is also regular.

Proof:

Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ be NFA that accept language L_1 , i.e. $L_1 = L(M_1)$ and

$M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ be another NFA that accept language L_2 , i.e. $L_2 = L(M_2)$

Now we construct NFA, $M = (Q, \Sigma, \delta, q, F)$ such that it accept $L_1 \cdot L_2$ i.e.

$$L(M) = L_1(M_1) \cdot L_2(M_2)$$

$$\text{where } Q = Q_1 \cup Q_2$$

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

$$\delta = \delta_1 \cup \delta_2 \cup [(F_1, \lambda \rightarrow q_2)]$$

$$q = q_1$$

$$F = F_2$$

Formally,

$$(q, w) \xrightarrow[M]{*} (p, \epsilon) \text{ for some } p \in F$$

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

Date _____
Page _____

$$w = w_1 \cdot w_2, (q_1, w_1) \xrightarrow{M_1} (p_1, \epsilon)$$

and

$$(q_2, w_2) \xrightarrow{M_2} (p_2, \epsilon)$$

Hence $L(M) = L_1(M_1) \cdot L_2(M_2)$

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

Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ be DFA that accept 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 DFA that accept language L_2 i.e. $L_2 = L_2(M_2)$.

We assume that alphabet of both automata are same and its Σ is union of alphabets of L_1 and L_2 . If they are different

Now we construct FA, $M = \{Q, \Sigma, \delta, q, F\}$ such that it accept $L_1 \cap L_2$, i.e.

$$L(M) = L_1(M_1) \cap L_2(M_2)$$

Date _____
Page _____

where

$$Q = Q_1 \times Q_2 \quad Z = (q_1, q_2)$$

$$\Sigma = \Sigma_1 \cup \Sigma_2 \quad F = F_1 \times F_2$$

$$\delta = (\delta_1, \delta_2)$$

If M_1 goes from state p to state s on reading 'a' 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 $(p, a) \xrightarrow{M_1} (s, \epsilon)$ and $(q, a) \xrightarrow{M_2} (t, \epsilon)$ then

$$(\{p, q\}, a) \xrightarrow{M} (\{s, t\}, \epsilon)$$

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 .

Hence $L_1 \cap L_2$ is regular.

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

~~Q~~

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

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

Let $M_1 = \{Q_1, \Sigma, \delta_1, q_1, F_1\}$ be an NFA that accept regular language $L = L(M_1)$.
Now

We construct NFA, $M = \{Q, \Sigma, \delta, q, F\}$ such that it can accept $L = L(M) = L(M_1)^*$ where

$$Q = Q_1 \cup \{q_1\}$$

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

$$\delta = \delta_1 \cup [(q, \lambda) \rightarrow q_1, (q_1, \lambda) \rightarrow F_1, (F_1, \lambda) \rightarrow q]$$

$$q = \text{start state} = q_1$$

$$F = F_1 \cup \{q_1\}$$

Here M consists of the state of M_1 and all transition of M_1 , 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 λ) is accepted.

If follows the inspection of M that if $w \in L(M)$ then either $w = \epsilon$ or $w = w_1 w_2 \dots w_k$ for $k \geq 1$, $i = 1, 2, 3, \dots, k$

There is $F_i \in F$ such that $(q_1, w) \xrightarrow{*}_{M_2} (F_i, \epsilon)$

6. 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 language goes under complement and intersection

Decision Algorithm for regular sets/language

Algorithm for regular language that tells whether or not following property hold.

Types:

- 1) The membership problem
- 2) The Emptiness problem
- 3) The infiniteness problem

1) The membership problem

- Is string w in regular language L ?
- Assume L is represented by a DFA M
- Processes 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 .

2) The Emptiness problem

- Given a regular language, those the language contain any string at all.
- Assume a DFA for language L
- Construct a transition graph
- Compute the set of state reachable from the start state.

→ If for any string the final state is reached then the string is present in L else no.

g) The infiniteness problem

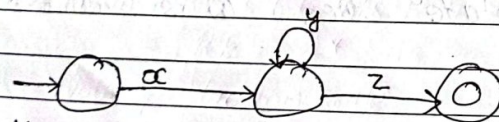
→ Is a given regular language infinite?

→ Start with a DFA for the language L .

→ If the DFA has ' n ' state and language contain any string of length n or more then the string is present in L , else no. language is 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, $w = \alpha y z$

Then, $\alpha y^i z \in L$ 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

* Context free language

↳ Language generated by context free grammar

↳ Language accepted by PDA

* Context-Free Grammar

↳ A context-free grammar G is a ^{quad}tuple (4-tuple) defined as $G = \{V, \Sigma, P, S\}$

where

V = set of variable or non-terminal symbol

Σ = set of terminal symbol

P = production rule

S = starting symbol

A CFG has production rule in the form, $A \rightarrow \alpha$

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

or

non-terminal $\rightarrow$ non-terminal
non-terminal $\rightarrow$ terminal

Capital letter

represented by small letter/
digits / symbol

Why context free grammar

① It provide a simple and mathematically precise mechanism for describing the method by which phrases in some natural language are built from smaller block

② Its simplicity make the formalism amenable to vigorous mathematical study

Eg: Consider $S \rightarrow aaAb$

A is non-terminal and a, b are terminal

[Sentence $\rightarrow$ Noun Verb]

Noun $\rightarrow$ Ram / Sita

Verb $\rightarrow$ goes / writes / sings

Eg: Construct a CFG for the language having any number of a's over $\Sigma = \{a\}$

$\Rightarrow$ we know, R.E for this

$RE = a^*$

Now production rule

$S \rightarrow \epsilon$

$S \rightarrow aS$

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

$S \rightarrow aS$

$\rightarrow aas \quad (S \rightarrow as)$

$\rightarrow aaaS \quad (S \rightarrow as)$

$\rightarrow aaaaS \quad (S \rightarrow as)$

$\rightarrow aaaa \epsilon \quad (S \rightarrow \epsilon)$

$\rightarrow aaaa$

CFG for given problem is

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

where,

$V = \{S\}$

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

$S = \{S\}$

P is, $S \rightarrow \epsilon / as$

2. Construct a CFG for RE $(a+1)^*$

$\Rightarrow$ Here R.E. = $(a+1)^*$

Possible string are $\{\epsilon, a, 1, 01, 10, \dots\}$

Production rule is

$S \rightarrow 0S / 1S / \epsilon$

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

$\Rightarrow$ Here RE = $(a+b)^* a (a+b)^* a (a+b)^*$

Possible string are $\{aa, aba, baaa, \dots\}$

Now,

Production rule

$S \rightarrow AaAaA$

$A \rightarrow \epsilon / aA / bA$

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

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

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

Possible string are $\{01, 10, 001, 110, \dots\}$

Thus production rule,

$S \rightarrow 0A1$

$S \rightarrow 1A0$

$A \rightarrow \epsilon \mid 0A \mid 1A$

5. Construct a CFG for language $L = wCw^T$, $w \in (a,b)^*$, $w^T = \text{reverse}$

$\Rightarrow$

Production rule:

$S \rightarrow C \mid aSa \mid bSb$

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

$\Rightarrow$ Possible string are $L = \{ab, aab, abb, \dots\}$

Now, Production Rules

$A \rightarrow a \mid aA$

$B \rightarrow b \mid bB$

$S \rightarrow AB$

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

$\Rightarrow$ Possible string, $L = \{a, ab, aab, aabb, \dots\}$

Now, Production Rules

$A \rightarrow a \mid aA$

$B \rightarrow \epsilon \mid bB$

$S \rightarrow AB$

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

$\Rightarrow$ As we know,

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

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

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

Also even + even = even

~~odd~~ 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

$A \rightarrow \epsilon \mid aaA$

Production rule for $(bb)^*$ is

$B \rightarrow \epsilon \mid bbB$

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

$S \rightarrow AbB \mid aAB$

$A \rightarrow \epsilon \mid aaA$

$B \rightarrow \epsilon \mid bbB$

9. Construct a CFG for $L = \{a^n b^m \mid n, m \text{ is odd}\}$
 $\Rightarrow$ [Same as 8]

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

Derivation of CFG

$\hookrightarrow$ process of deriving strings from grammar

Eg: Derive a^4 from grammar
 $S \rightarrow as \mid \epsilon$

$\Rightarrow$

$$\begin{aligned} S &\rightarrow as \\ &\rightarrow aas \quad [S \rightarrow as] \\ &\rightarrow aaas \quad [S \rightarrow as] \\ &\rightarrow aaaaS \quad [S \rightarrow as] \\ &\rightarrow aaaa\epsilon \quad [S \rightarrow \epsilon] \\ &\rightarrow aaaa \end{aligned}$$

Hence, a^4 is accepted by grammar

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

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

$S \rightarrow aCa$

$C \rightarrow aCa \mid b$

And $L(G)$ is language accepted by grammar (CFG)

$\Rightarrow$

$$\begin{aligned} S &\rightarrow aCa \\ &\rightarrow aba \quad [C \rightarrow b] \end{aligned}$$

$aba \in L(G)$

Also, $S \rightarrow aCa$
 $\rightarrow aCaCa \quad [C \rightarrow aCa]$

$\rightarrow aabga \ [c \rightarrow b]$
 $\therefore aabaa \in L(G)$

Similarly,

$s \xrightarrow{*} aaabaaa \in L(G)$

$s \xrightarrow{*} aaaabaaaa \in L(G)$

and so on.

So the language is

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

Q. Find $L(G)$ for $S \rightarrow as \mid bs \mid a \mid b$

$\Rightarrow S \rightarrow a$

$S \rightarrow b$

$\rightarrow as$

$\rightarrow aa \ [S \rightarrow a]$

$S \rightarrow bs$

$\rightarrow bb \ [S \rightarrow b]$

Similarly, $S \xrightarrow{*} aba$

$S \xrightarrow{*} aaba$

and so on

So, the language is $L = \{a^n b^m \mid (n+m) \geq 1\}$

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 Most Derivation

Combination of left and right most

Derivation Tree

$\rightarrow$ representation of derivation using tree

$\rightarrow$ also called parse tree

Definition:

$\rightarrow$ 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 Λ)
2. The root has label S
3. The label of an internal vertex is a variable.

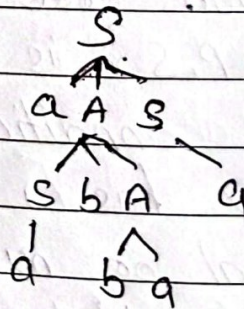
4. If the vertices $n_1, n_2, \dots, n_k$ written with label $x_1, x_2, \dots, x_k$ are the child nodes of vertex n with label 'A', then $A \rightarrow x_1 x_2 \dots x_k$ is a production rule.

Q. Consider a grammar G which has
 $S \rightarrow aAS \mid a$
 $A \rightarrow sbA \mid ss \mid ba$

Construct a parse tree which yield 'aabbbaa' where s is starting symbol

$\Rightarrow$ $S \rightarrow aAS$
 $\rightarrow asbAS \quad [A \rightarrow sbA]$
 $\rightarrow aabAS \quad [s \rightarrow a]$
 $\rightarrow aabbas \quad [A \rightarrow ba]$
 $\rightarrow aabbbaa \quad [s \rightarrow a]$

Parse Tree:



Q. Derive the string 'aabbabba'

$$S \rightarrow aB/bA$$

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

$$B \rightarrow b/bs/aBB$$

Solⁿ ① Left most derivation

$$S \rightarrow aB \quad [S \rightarrow aB]$$

$$\rightarrow aaBB \quad [B \rightarrow aBB]$$

$$\rightarrow aaBSB \quad [B \rightarrow bs]$$

$$\rightarrow aaBSB \quad [S \rightarrow bA]$$

$$\rightarrow aabbabB \quad [A \rightarrow a]$$

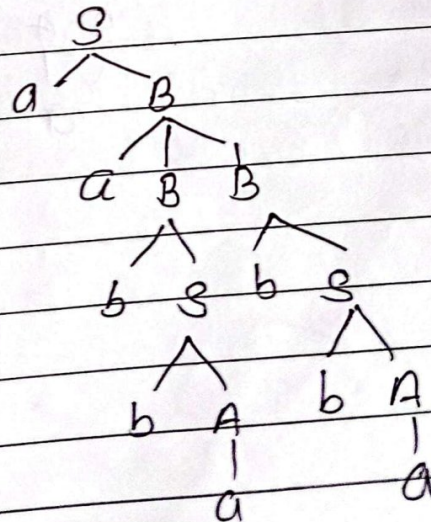
$$\rightarrow aabbabAB \quad [S \rightarrow bA]$$

$$\rightarrow aabbabbs \quad [B \rightarrow bs]$$

$$\rightarrow aabbabba \quad [S \rightarrow bA]$$

$$\rightarrow aabbabba \quad [A \rightarrow a]$$

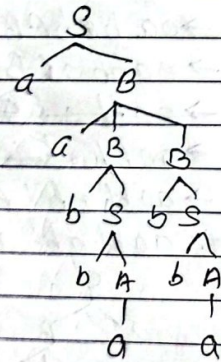
Parse tree:



② Right most derivation

$$\begin{aligned} s &\rightarrow aB & [s \rightarrow aB] \\ &\rightarrow aaBB & [B \rightarrow aBB] \\ &\rightarrow aaBbs & [B \rightarrow bs] \\ &\rightarrow aaBbba & [s \rightarrow ba] \\ &\rightarrow aaBbbba & [A \rightarrow a] \\ &\rightarrow aa bs bba & [B \rightarrow bs] \\ &\rightarrow aa bba bba & [B \rightarrow ba] \\ &\rightarrow aa bba bba & [A \rightarrow a] \end{aligned}$$

Parse tree:



Q. Derive the string "00110101"

$S \rightarrow 00^U / 1A$
 $A \rightarrow 0 / 0S / 1AA$
 $B \rightarrow 1 / 1S / 0BB$

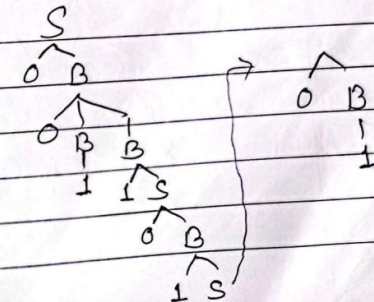
by

- ① Left most derivation
- ② Right most derivation
- ③ Mixed most derivation
- ④ make parse tree of each derivation

(1) Left most derivation

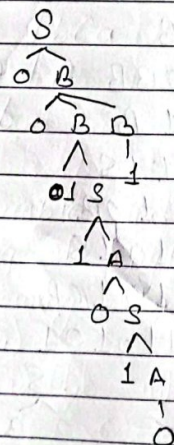
$$\begin{array}{ll}
 S \rightarrow 0B & [S \rightarrow 0B] \\
 \rightarrow 00BB & [B \rightarrow 0BB] \\
 \rightarrow 001B & [B \rightarrow 1] \\
 \rightarrow 0011S & [B \rightarrow 1S] \\
 \rightarrow 00110B & [S \rightarrow 0B] \\
 \rightarrow 001101S & [B \rightarrow 1S] \\
 \rightarrow 0011010B & [0S \rightarrow 0B] \\
 \rightarrow 00110101 & [B \rightarrow 1]
 \end{array}$$

Parse tree:



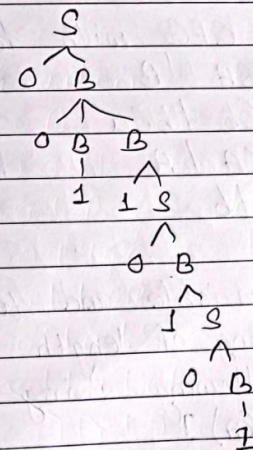
2. Right Most derivation

$S \rightarrow 0B$ [$S \rightarrow 0B$]
 $\rightarrow 00BB$ [$B \rightarrow 0BB$]
 $\rightarrow 00B1$ [$B \rightarrow 1$]
 $\rightarrow 001S1$ [$B \rightarrow 1S$]
 $\rightarrow 0011A1$ [$S \rightarrow 1A$]
 $\rightarrow 00110S1$ [$A \rightarrow 0S$]
 $\rightarrow 001101A1$ [$S \rightarrow 1A$]
 $\rightarrow 00110101$ [$A \rightarrow 0$]



② Mixed Most derivation:

$S \rightarrow 0B$ [$S \rightarrow 0B$]
 $\rightarrow 00BB$ [$B \rightarrow 0BB$]
 $\rightarrow 0011S$ [$B \rightarrow 1, B \rightarrow 1S$]
 $\rightarrow 00110B$ [$S \rightarrow 0B$]
 $\rightarrow 001101S$ [$B \rightarrow 1S$]
 $\rightarrow 0011010B$ [$S \rightarrow 0B$]
 $\rightarrow 00110101$ [$B \rightarrow 1$]



Assignment

Q. Derive the string "aaabbaabba" 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 most 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) abba iv) abbb

- d) True or False, $C \rightarrow bab$
- e) True or False, $C \xrightarrow{*} bab$
- f) True or False, $C \xrightarrow{*} AAA$

Ambiguity in CFG

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

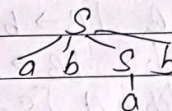
Eg: Show that the grammar

$S \rightarrow a \mid absb \mid aAb$

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

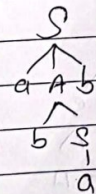
1st method

$S \rightarrow absb$ [$S \rightarrow abSb$]
 $\rightarrow abab$ [$S \rightarrow a$]



2nd method

$S \rightarrow aAb$ [$S \rightarrow aAb$]
 $\rightarrow absb$ [$A \rightarrow bS$]
 $\rightarrow abab$ [$S \rightarrow a$]



Since more than two derivation for "abab"
 So, grammar is ambiguous.

Assignment

Date
Page

Q. Derive the string "aaabbaabba" by

$$S \rightarrow aB \mid bA$$

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

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

i) Left most derivation

$$S \rightarrow aB \quad [S \rightarrow aB]$$

$$\rightarrow aaBB \quad [B \rightarrow aBB]$$

$$\rightarrow aaaBBB \quad [B \rightarrow aBB]$$

$$\rightarrow aaabBB \quad [B \rightarrow b]$$

$$\rightarrow aaabbb \quad [B \rightarrow b]$$

$$\rightarrow aaabbaBB \quad [B \rightarrow aBB]$$

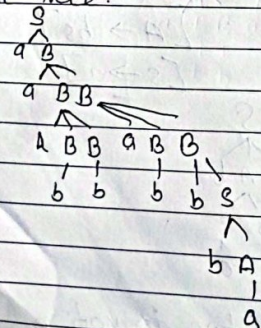
$$\rightarrow aaabbaabB \quad [B \rightarrow b]$$

$$\rightarrow aaabbaabBS \quad [B \rightarrow bS]$$

$$\rightarrow aaabbaabbaA \quad [S \rightarrow bA]$$

$$\rightarrow aaabbaabbaa \quad [A \rightarrow a]$$

Parse tree:



ii) Right most derivation

$$S \rightarrow aB \quad [S \rightarrow aB]$$

$$\rightarrow aaBB \quad [B \rightarrow aBB]$$

$$\rightarrow aaBabBB \quad [B \rightarrow aBB]$$

$$\rightarrow aaBabBbS \quad [B \rightarrow bS]$$

$$\rightarrow aaBabBbbA \quad [S \rightarrow bA]$$

$$\rightarrow aaBabBbbba \quad [A \rightarrow a]$$

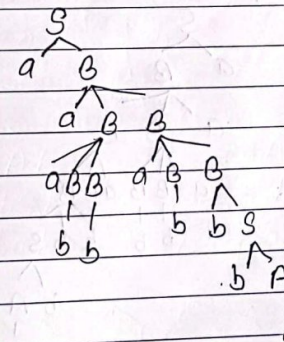
$$\rightarrow aaBabbbba \quad [B \rightarrow b]$$

$$\rightarrow aaabBBabbbba \quad [B \rightarrow aBB]$$

$$\rightarrow aaabbbabbbba \quad [B \rightarrow b]$$

$$\rightarrow aaabbaabbbba \quad [B \rightarrow b]$$

Parse tree:



Date _____
Page _____

$S \rightarrow aB$ [$S \rightarrow aB$]
 $\rightarrow aaBB$ [$B \rightarrow aBB$]
 $\rightarrow aabbs$ [$B \rightarrow b, B \rightarrow bs$]
 $\rightarrow aabbab$ [$S \rightarrow aB$]
 $\rightarrow aabbbaBB$ [$B \rightarrow aBB$]
 $\rightarrow aabbbaa$
 $\rightarrow aaaaBBbaBB$ [$B \rightarrow aBB, B \rightarrow aBB$]
 $\rightarrow aaaaabbbs$ [$B \rightarrow b, B \rightarrow b, B \rightarrow b, B \rightarrow bs$]
 $\rightarrow aaaaabbbaA$ [$S \rightarrow bA$]
 $\rightarrow aaaaabbaabbaa$ [$A \rightarrow a$]

$$\begin{array}{l} S \rightarrow ASA \mid B \\ B \rightarrow aCb \mid bCa \\ C \rightarrow ACA \mid A \\ A \rightarrow a \mid b \end{array}$$

⇒ Variable : A, S, B, C,
Terminal : a, b

三

$\Rightarrow$ Yes, there is a string in language
 i.e. abb

⇒ Yes, there is a string in language.
i.e. abb

2/14

Date
Page

Q. Show that grammar G is ambiguous
 $S \rightarrow aB \mid ab$
 $B \rightarrow ABb \mid b$
 $A \rightarrow aAB \mid a$

1st method:

$S \rightarrow aB$ [$S \rightarrow aB$]
 $\rightarrow aABb$ [$B \rightarrow ABb$]
 $\rightarrow aaABbb$ [$A \rightarrow aAB$]
 $\rightarrow aaaaBBBB$ [$A \rightarrow a$]
 $\rightarrow aab$ [$B \rightarrow b$]

2nd method:

$S \rightarrow ab$ [$S \rightarrow ab$]
 $\begin{matrix} & S \\ & \swarrow \searrow \\ a & & b \end{matrix}$

Since two different derivation tree. So grammar is ambiguous

2/14

Date
Page

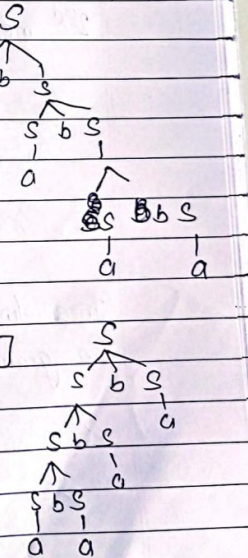
Q. If G is the grammar, $S \rightarrow sbS \mid a$
 Show that grammar is ambiguous for
 abababa string

1st method:

$S \rightarrow sbS$ [$S \rightarrow sbS$]
 $\rightarrow abs$ [$S \rightarrow a$]
 $\rightarrow absbS$ [$S \rightarrow sbS$]
 $\rightarrow ababs$ [$S \rightarrow a$]
 $\rightarrow ababsbS$ [$S \rightarrow sbS$]
 $\rightarrow abababs$ [$S \rightarrow a$]
 $\rightarrow abababa$ [$S \rightarrow a$]

2nd method:

$S \rightarrow sbS$ [$S \rightarrow sbS$]
 $\rightarrow sb a$ [$S \rightarrow a$]
 $\rightarrow sb sba$ [$S \rightarrow sbS$]
 $\rightarrow sb a ba$ [$S \rightarrow a$]
 $\rightarrow sb s b a ba$ [$S \rightarrow sbS$]
 $\rightarrow sb a b a ba$ [$S \rightarrow a$]
 $\rightarrow a b a b a b a$ [$S \rightarrow a$]



Q. Show that grammar G is ambiguous

$$S \rightarrow as | osb | x$$

$$x \rightarrow xa | a$$

1st method

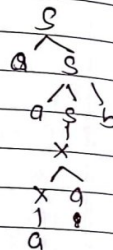
$$S \rightarrow as \quad [S \rightarrow as]$$

$$\rightarrow aasb \quad [S \rightarrow asb]$$

$$\rightarrow aaxb \quad [S \rightarrow x]$$

$$\rightarrow aaxab \quad [x \rightarrow xa]$$

$$\rightarrow aqaab \quad [x \rightarrow a]$$



2nd method

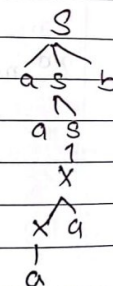
$$S \rightarrow asb$$

$$\rightarrow aasb \quad [S \rightarrow as]$$

$$\rightarrow aaxb \quad [S \rightarrow x]$$

$$\rightarrow aaxab \quad [x \rightarrow xa]$$

$$\rightarrow aqaab \quad [x \rightarrow a]$$



Since two different derivation tree
So grammar is ambiguous

Q. IF G is the grammar for palindrom

$$S \rightarrow asa | bsb | a | b | \epsilon$$

Check for ambiguity

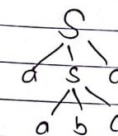
1st method

$$S \rightarrow asa$$

$$[S \rightarrow asa]$$

$$S \rightarrow aba$$

$$[S \rightarrow b]$$



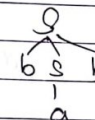
2nd method

$$S \rightarrow bsb$$

$$[S \rightarrow bsb]$$

$$\rightarrow bab$$

$$[S \rightarrow b]$$



Here ~~there~~ no different derivation tree
So, grammar is no ambiguity

Assignment

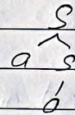
Date _____
Page _____

1. Q) Show that grammar G is ambiguous.
 $S \rightarrow as / sa / a$

1st method:

$$S \rightarrow as$$

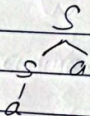
$$\rightarrow aa \quad [S \rightarrow a]$$



2nd method:

$$S \rightarrow sa$$

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



Since more than two derivation form. So,
Grammar is ambiguous.

Assignment

Date _____
Page _____

2. Check for ambiguity.

$$S \rightarrow ices / ictses / o$$

$$| c \rightarrow b$$

$$S = ices$$

$$\Rightarrow ibta \quad [c \rightarrow b, s \rightarrow a]$$

It can't be derived from any other way. So
it is not ambiguous.

Simplification of Grammar

↳ removing all unnecessary symbol

1. Removal of useless symbol
2. Removal of NULL Symbol
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 derived some terminal and $i=1$

Step 2: Include all symbols W_{i+1} that derives W_i

Step 3: Implement 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 α (derives from start symbol)

Procedure:

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

Step 2: Include all symbol 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 $Y_{i+1} = Y_i$

2. Removal of useless Symbol

3. Find a reduced grammar equivalent to G having production rule P as

$S \rightarrow AC / B$

$A \rightarrow a$

$C \rightarrow c / BC$

$E \rightarrow aA / e$

Q1

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

$P = S \rightarrow AC, A \rightarrow a, C \rightarrow c, E \rightarrow aA/c$

Phase 2:

- $X_1 = \{S\}$
- $X_2 = \{S, A, C\}$
- $X_3 = \{S, A, C, a, c\}$
- $X_4 = \{S, A, C, a, c\}$

$G' = (\{S, A, C\}, \{a, c\}, P, S)$

where

$P = S \rightarrow AC$
 $A \rightarrow a$
 $C \rightarrow c$

G, W_3 is reduced grammar.

Q Find a reduced grammar equivalent to G having production rule P as
 $S \rightarrow ABa/BC, A \rightarrow BCC/CE, C \rightarrow a, B \rightarrow bcc$
 $D \rightarrow E, E \rightarrow d, F \rightarrow c$

Phase 1:

- $T = \{a, b, c, d, e\}$
- $W_1 = \{C, B, A, E, ES\}$
- $W_2 = \{A, B, C, E, F, S, D\}$
- $W_3 = \{A, B, C, D, E, F, S\}$

$G' = (\{A, B, C, D, E, F, S\}, \{a, b, c, d, e\}, P, S)$

where

$P = S \rightarrow ABa/BC$
 $A \rightarrow BCC/CE$
 $C \rightarrow a$
 $B \rightarrow bcc$
 $D \rightarrow E$
 $E \rightarrow d$
 $F \rightarrow c$

Now Phase 2:

- $X_1 = \{S\}$
- $X_2 = \{S, A, B, c\}$
- $X_3 = \{S, A, B, C, a, b, c\}$

$$Y_4 = \{S, A, B, C, \epsilon, a, b, c\}$$

$$G'' = \{S, A, B, C, \epsilon, a, b, c\}, P, S\}$$

where

$$P = S \rightarrow ABa / BC$$

$$A \rightarrow ac / BCC$$

$$C \rightarrow a$$

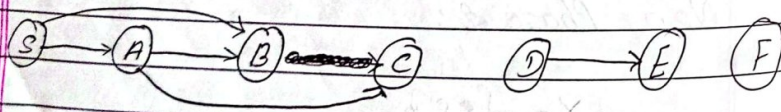
$$B \rightarrow bcc$$

So this is the reduced grammar.

Alternate method:

Step 1: Eliminate non-generating symbol. Here all symbol are generating.

Step 2: Eliminate non-reachable symbol (variable)
Draw dependency graph



Here D, E and F are non-reachable from S
After removing useless symbol, we get

$$P: S \rightarrow ABa / BC, A \rightarrow ac / BCC, C \rightarrow a$$

$$B \rightarrow bcc$$

Q. Find a reduced grammar equivalent to G having production rule P as

$$S \rightarrow AB / CA, A \rightarrow a$$

$$S \rightarrow BC / AB, C \rightarrow ab / b$$

Phase 1:

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

$$W_1 = \{A, C\}$$

$$W_2 = \{A, C, S, \epsilon\}$$

$$W_3 = \{A, C, S\}$$

$$W_4 = \{A, C, S\}$$

where

$$G' = \{A, C, S, \epsilon, a, b\}, P, S\}$$

where

$$P = S \rightarrow AC$$

$$A \rightarrow a$$

$$C \rightarrow ab / b$$

Phase 2:

$$Y_1 = \{S\}$$

$$Y_2 = \{A, C\}$$

$$Y_3 = \{A, C, S, a, b, B\}$$

$$Y_4 = \{A, C, S, a, b, B\}$$

$$G'' = \{A, C, S, a, b, B\}, (a, b), P, S\}$$

where

$$P = S \rightarrow AC$$

$$A \rightarrow a$$

$$C \rightarrow aB/b$$

So this is reduced grammar.

Find a reduced grammar equivalent to G having production rule P as

$$S \rightarrow aAa$$

$$B \rightarrow ab$$

$$A \rightarrow bBB$$

$$C \rightarrow ab$$

$$S \rightarrow aS/A/BC$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$C \rightarrow aCb$$

Here C is non-reachable

from S . So, remove

it. we get,

$$S \rightarrow aAa$$

$$A \rightarrow bBB$$

$$B \rightarrow ab$$

Here C is useless

as it does not

derive any string.

After removing C ,

we get

$$S \rightarrow aS/A$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

Again

B is non-reachable

from S . So remove it.

we get

$$S \rightarrow aS/A$$

$$A \rightarrow a$$

2. Reduction 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 grammar rule whenever $B \rightarrow x$ ($x \in$ terminal or $x \in \text{null}$)

Step 2: Remove $A \rightarrow B$ from grammar

Step 3: Repeat step 1 and 2 until all unit production are removed

Q. Remove unit production from given grammar

P: $S \rightarrow XY$, $X \rightarrow a$, $Y \rightarrow z/b$, $Z \rightarrow M$
 $M \rightarrow N$, $N \rightarrow a$

Solⁿ Here, unit production are
 $Y \rightarrow Z$, $Z \rightarrow M$, $M \rightarrow N$

(1) Since $N \rightarrow a$, add $M \rightarrow a$, we get

P: $S \rightarrow XY$, $X \rightarrow a$, $Y \rightarrow z/b$, $Z \rightarrow M$, $M \rightarrow a$,
 $N \rightarrow a$

(2) Since $M \rightarrow a$, add $Z \rightarrow a$, we get

P: $S \rightarrow XY$, $X \rightarrow a$, $Y \rightarrow z/b$, $Z \rightarrow a$, $M \rightarrow a$,
 $N \rightarrow a$

(3) Since $Z \rightarrow a$, add $Y \rightarrow a$, we get

P: $S \rightarrow XY$, $X \rightarrow a$, $Y \rightarrow a/b$, $Z \rightarrow a$, $M \rightarrow a$,
 $N \rightarrow a$

Also remove non-reachable symbol
Here M, N, Z are non-reachable. So reduced
Grammar is

P: $S \rightarrow XY$, $X \rightarrow a$, $Y \rightarrow a/b$

Q. Remove unit production from given grammar

$$\begin{aligned} RS &\rightarrow A/bb \\ A &\rightarrow B/b \\ B &\rightarrow S/a \end{aligned}$$

⇒ Here unit production are
 $S \rightarrow A$, $B \rightarrow S$, $A \rightarrow B$

① Since $B \rightarrow a$ add $A \rightarrow a$ we get
P: $S \rightarrow A/bb$
 $B \rightarrow S/a$
 $A \rightarrow a/b$

② Since $S \rightarrow bb$ add $B \rightarrow bb$ we get
P: $S \rightarrow A/bb$
 $B \rightarrow bb/a$
 $A \rightarrow a/b$

③ Since $A \rightarrow a/b$ add $S \rightarrow a/b$ we get
 $S \rightarrow a/b/bb$
 $A \rightarrow a/b$
 $B \rightarrow bb/a$

Here B, A are not reachable so
remove it the the grammar is
 $S \rightarrow a/b/bb$

Elimination of NULL production

In CFG or non-terminal symbol 'A' is null production if $A \rightarrow \epsilon$ (or $A \rightarrow \lambda$) or a production starting at A leads to ϵ or λ (i.e. $A \rightarrow \dots \epsilon$ or $A \xrightarrow{*} \epsilon$)

Procedure:

1. To remove $A \rightarrow \epsilon$, look for all production whose RHS contain A.
2. Replace each occurrence of A in each of those production with ϵ .
3. Add the resultant production to the Grammar.

Q. Eliminate null production from grammar
 $S \rightarrow ABAC$, $A \rightarrow aA/\epsilon$, $B \rightarrow bB/\epsilon$
 $C \rightarrow c$

Solⁿ Here null production $A \rightarrow \epsilon$, $B \rightarrow \epsilon$

① Remove $A \rightarrow \epsilon$

$S \rightarrow ABAC$	$S \rightarrow ABAC$	$S \rightarrow ABAC$
$\rightarrow \epsilon BAC$	$\rightarrow AB\epsilon C$	$\rightarrow \epsilon B\epsilon C$
$\rightarrow BAC$	$\rightarrow ABC$	$\rightarrow BC$

$$\textcircled{iv} \begin{aligned} A &\rightarrow aA \\ &\rightarrow aE \\ &\rightarrow a \end{aligned}$$

now, new production rule are
 $S \rightarrow ABAC / BAC / ABC / BC$
 $A \rightarrow aA / a, B \rightarrow bB / E, C \rightarrow C$

① Remove $B \rightarrow E$

$$\begin{array}{lll} \textcircled{1} S \rightarrow ABAC & \textcircled{2} S \rightarrow BAC & \textcircled{3} S \rightarrow ABC \\ \rightarrow AEAC & \rightarrow EAC & \rightarrow AEC \\ \rightarrow AAC & \rightarrow AC & \rightarrow AC \end{array}$$

$$\begin{array}{ll} \textcircled{4} S \rightarrow BC & \textcircled{5} B \rightarrow bB \\ \rightarrow BC & \rightarrow bE \\ \rightarrow C & \rightarrow b \end{array}$$

now, production rule are
 $S \rightarrow ABAC / ABC / BAC / BC / AAC / AC / C$
 $A \rightarrow aA / a$
 $B \rightarrow bB / b$
 $C \rightarrow C$

② Eliminate null production from grammar
 $S \rightarrow ABAC, A \rightarrow BC, B \rightarrow b/E, C \rightarrow D/E$
 $D \rightarrow d$

Here null production.
 $B \rightarrow E$
 $C \rightarrow E$

① Remove $B \rightarrow E$

$$\begin{array}{ll} S \rightarrow ABAC & A \rightarrow BC \\ \rightarrow AAC & \rightarrow C \end{array}$$

New P: $S \rightarrow ABAC / AAC$
 $A \rightarrow BC / C$
 $B \rightarrow b$
 $C \rightarrow D / E$
 $D \rightarrow d$

② Removing $C \rightarrow E$

P: $S \rightarrow ABAC / AA / ABAC / AAC$
 $B \rightarrow b$
 $A \rightarrow B / E / BC / C$
 $C \rightarrow D$
 $D \rightarrow d$

Assignment

Date _____
Page _____

Again Removing $A \rightarrow \epsilon$

P:
 $S \rightarrow ABaC / AaC / ABa / Aa / BaC / ac /$
 Ba / a

$A \rightarrow BC / C / B$

$B \rightarrow b$

$C \rightarrow D$

$D \rightarrow d$

2. Simplify the grammar

$S \rightarrow aA / aBb$

$A \rightarrow aAA / \epsilon$

$B \rightarrow bB / bBc$

$C \rightarrow B$

1) Remove

① NULL

② Unit

③ Useless

2) Eliminating ϵ -production gives result grammar as

$S \rightarrow aA / aBb / a$

$A \rightarrow aAA / aA / a$

$B \rightarrow bB / bBc$

$C \rightarrow B$

3) Removing unit production gives resulting grammar

$S \rightarrow aA / aBb / a$

$A \rightarrow aAA / aA / a$

$B \rightarrow a bB / bBc$

$C \rightarrow bB / bBc$

② 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.

$$S \rightarrow aA|a$$

$$A \rightarrow aAA|aA|a$$

This is simplified grammar.

Normal Form

↳ production rule that has certain restriction

Types:

- ① Chomsky Normal Form (CNF)
- ② Greibach Normal Form (GNF)

Imp

① Chomsky Normal Form (CNF)

A ~~CFG~~ CFG is in CNF if production rule are of the following form:

$$A \rightarrow a$$

$$A \rightarrow BC$$

where A, B, C are variable and 'a' is terminal

Procedure for converting into CNF

Step 1: IF START symbol 'S' occurs on some right side, take 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 no. of variables on RHS, introduced new variable and separate them as follows:

Suppose we have production with 'n' non-terminal as shown below with 5 non-terminal

$$Y \rightarrow X_1 X_2 X_3 X_4 X_5$$

Add n-2 new production rule using n-2 new non-terminals and modify the production as below:

$$Y \rightarrow X_1 R_1$$

$$R_1 \rightarrow X_2 R_2$$

$$R_2 \rightarrow X_3 R_3$$

$$R_3 \rightarrow X_4 X_5$$

where R_1, R_2, R_3 are new non terminal

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

$$P: S \rightarrow ASA / aB$$

$$A \rightarrow B / S$$

$$B \rightarrow b / \epsilon$$

Solⁿ

1. Since S appears in RHS, introduce new start symbol S' and add $S' \rightarrow S$ to production rule.

$$P: S' \rightarrow S, S \rightarrow ASA / aB, A \rightarrow B / S, B \rightarrow b / \epsilon$$

2. Remove null production: $B \rightarrow \epsilon$

After removing: $B \rightarrow \epsilon$

$$P: S' \rightarrow S$$

$$S \rightarrow ASA / aB / a$$

$$A \rightarrow \epsilon / S / B$$

$$B \rightarrow b$$

Again removing $A \rightarrow \epsilon$ the we get

$$P: S' \rightarrow S, S \rightarrow ASA / aB / a / SA / AS / S, A \rightarrow S / B$$

$$B \rightarrow b$$

3. Remove unit production:

$$S' \rightarrow S, S \rightarrow S, A \rightarrow S, A \rightarrow B$$

After removing $S \rightarrow S$

$$P: S' \rightarrow S$$

$$S \rightarrow ASA / aB / a / SA / AS$$

$$A \rightarrow S / B$$

$$B \rightarrow b$$

After removing: $S' \rightarrow S$

$$P: S' \rightarrow aB / a / ASA / SA / AS$$

$$S \rightarrow ASA / aB / a / SA / AS$$

$$A \rightarrow S / B$$

$$B \rightarrow b$$

After removing: $A \rightarrow S$

$$P: S' \rightarrow aB / a / ASA / SA / AS$$

$$S \rightarrow ASA / aB / a / SA / AS$$

$$A \rightarrow aB / a / SA / AS / B$$

$$B \rightarrow b$$

After removing : $A \rightarrow B$

$$P: S' \rightarrow ASA / aB / a / SA / AS$$

$$S \rightarrow ASA / aB / a / SA / AS$$

$$A \rightarrow ASA / aB / a / SA / AS / b$$

$$B \rightarrow b$$

④ Now check for more than two variable on R.H.S.

$$S' \rightarrow ASA, S \rightarrow ASA, A \rightarrow ASA$$

Removing these

$$S' \rightarrow ASA$$

$$\rightarrow AX \quad [x \rightarrow SA]$$

Also

$$S' \rightarrow ASA$$

$$\rightarrow AX \quad [x \rightarrow SA]$$

$$A \rightarrow ASA$$

$$\rightarrow AX \quad [x \rightarrow SA]$$

Now we get

$$P: S' \rightarrow AX / aB / a / SA / AS$$

$$S \rightarrow AX / aB / a / SA / AS$$

$$A \rightarrow AX / aB / a / SA / AS / b$$

$$B \rightarrow b$$

$$X \rightarrow AB$$

⑤ Change production $S' \rightarrow aB, S \rightarrow aB, A \rightarrow aB$ as

$$S' \rightarrow aB$$

$$\rightarrow YB \quad \text{where } Y \rightarrow a$$

also,

$$S \rightarrow YB \quad \text{and} \quad A \rightarrow YB$$

Finally, we get

$$P: S' \rightarrow AX / YB / a / SA / AS$$

$$S \rightarrow AX / YB / a / SA / AS$$

$$A \rightarrow AX / YB / a / SA / AS / b$$

$$B \rightarrow b$$

$$X \rightarrow AS$$

$$Y \rightarrow a$$

Assignment

Q. Convert following CFG into CNF:

$$S \rightarrow bA / aB$$

$$A \rightarrow bAA / aS / a$$

$$B \rightarrow aBB / bS / b$$

1. Here no S is in R.H.S, so no need to introduce start symbol

2. Here no empty and no unit production.

3. Now check for more than two variable on R.H.S

$$A \rightarrow bAA, B \rightarrow aBB$$

Removing this,

$$A \rightarrow bAA$$

$$\rightarrow bX \quad [x \rightarrow AA]$$

also

$$B \rightarrow aBB$$

$$\rightarrow aY \quad [y \rightarrow BB]$$

we get

$$S \rightarrow bA / aB$$

$$A \rightarrow bX / aS / a$$

$$B \rightarrow aY / bS / b$$

$$X \rightarrow AA$$

$$Y \rightarrow BB$$

Q. Change production: $S \rightarrow bA / S \rightarrow aB$, $A \rightarrow bX, A \rightarrow aS$
 $B \rightarrow bS, B \rightarrow aY$ as

$S \rightarrow bA$
 $\rightarrow PA \quad [P \rightarrow b]$

$S \rightarrow aB$
 $\rightarrow QB \quad [Q \rightarrow a]$

Similarly
 $A \rightarrow PX, A \rightarrow QS, B \rightarrow PS, B \rightarrow QY$

Finally Production:

$S \rightarrow PA / QB$
 $A \rightarrow PX / QS / a$
 $B \rightarrow PS / QY / b$
 $X \rightarrow AA$
 $Y \rightarrow BB$
 $P \rightarrow b$
 $Q \rightarrow a$

Convert into CNF

Q. $S \rightarrow AB / aB$
 $A \rightarrow aab / \epsilon$
 $B \rightarrow bbA$

1. Since no S is in RHS so no need to introduce start symbol

2. Removing null production: $A \rightarrow \epsilon$
 After removing $A \rightarrow \epsilon$

P: $S \rightarrow AB / aB / B$
 $A \rightarrow aab$
 $B \rightarrow bbA / bb$

3. Here no unit production in RHS

4. Now check for more than two variable on RHS
 Here more than two variable is not present in RHS

5. Change production $S \rightarrow aB, A \rightarrow aab, B \rightarrow bbA$
 $B \rightarrow bbAS$

$S \rightarrow aB$
 $\rightarrow XB \quad [X \rightarrow a]$

$A \rightarrow aab$
 $\rightarrow XXbY \quad [X \rightarrow a] [bY \rightarrow b]$

$B \rightarrow bbA$
 $\rightarrow YYA \quad [Y \rightarrow b]$
 $B \rightarrow XY$

Finally we get:

$$S \rightarrow xB \mid AB \mid B$$

$$A \rightarrow xxBy$$

$$B \rightarrow yYA \mid Y$$

$$x \rightarrow a$$

$$y \rightarrow b$$

6. Again check for more than two variable on RHS $A \rightarrow xxy$ $B \rightarrow yYA$

$$A \rightarrow xxy$$

$$\rightarrow x_1 y \quad [x_1 \rightarrow xx]$$

$$B \rightarrow yYA$$

$$\rightarrow y_1 A \quad [y_1 \rightarrow yy]$$

Now, finally we get,

$$P: S \rightarrow xB \mid AB$$

$$A \rightarrow x_1 y$$

$$B \rightarrow y_1 A \mid Y \quad y_1 \rightarrow yy$$

$$x \rightarrow a$$

$$y \rightarrow b$$

$$x_1 \rightarrow xx$$

2/23

$$Q. S \rightarrow AB \mid aB$$

$$A \rightarrow aab \mid \epsilon$$

$$B \rightarrow bbA$$

1. Since no S is in RHS. So no need to introduce start symbol

2. Removing null production: $A \rightarrow \epsilon$

After removing: $A \rightarrow \epsilon$

$$S \rightarrow AB \mid aB \mid B$$

$$A \rightarrow aab$$

$$B \rightarrow bbA \mid bb$$

3. Removing unit production $S \rightarrow B$

$$P: S \rightarrow AB \mid aB \mid bbA \mid bb$$

$$A \rightarrow aab$$

$$B \rightarrow bbA \mid bb$$

4. Now there are more than two variable is not present

5. Change the production $S \rightarrow aB$, $S \rightarrow bbA$, $S \rightarrow bb$
 $A \rightarrow aab$, $B \rightarrow bbA$, $B \rightarrow bb$

$$S \rightarrow aB$$

$$\rightarrow xB \quad [x \rightarrow a]$$

$$S \rightarrow bb$$

$$\rightarrow yy$$

$$A \rightarrow aab$$

$$\rightarrow xxy$$

$$S \rightarrow bbA$$

$$\rightarrow yYA \quad [y \rightarrow b]$$

$$B \rightarrow yYA$$

$$B \rightarrow bb$$

$$\rightarrow yy$$

Pr $S \rightarrow AB / XB / YYA / YY$ $X \rightarrow a$
 $A \rightarrow xxy$ $Y \rightarrow b$
 $B \rightarrow YYA / YY$

(6) Again check more than two variable on RHS

$S \rightarrow YYA$, $A \rightarrow xxy$, $B \rightarrow YYA$ as

$S \rightarrow YYA$
 $\rightarrow Y_1 A$ [$Y_1 \rightarrow YY$]

$A \rightarrow xxy$
 $\rightarrow X_1 Y$ [$X_1 \rightarrow xx$]

$B \rightarrow YYA$
 $\rightarrow Y_1 A$ [$Y_1 \rightarrow YY$]

Now,

Pr $S \rightarrow AB / XB / Y_1 A / YY$
 $A \rightarrow X_1 Y$
 $B \rightarrow Y_1 A / YY$
 $X \rightarrow a$
 $Y \rightarrow b$
 $Y_1 \rightarrow YY$
 $X_1 \rightarrow xx$

Greibach Normal Form (GNF)

A CFG is in GNF if the production rule are of the following form:

$A \rightarrow b$

$A \rightarrow bC_1C_2 \dots C_n$

where $A, C_1, C_2, \dots, C_n$ are non-terminal and b is terminal

Steps to convert into (GNF)

Step 1: Check CFG have any null or unit production if any then, then remove it

Step 2: Check if CFG is in CNF, if not convert it

Step 3: Change the non-terminal symbol into A_i is ascending order of i

Step 4: Alter the rule so that non terminal are in ascending order such that if the production is of the form $A_i \rightarrow A_j Y$, then $i \geq j$ and should never be $i > j$

Step 5: Remove left recursion, if there are any

Eg: $S \rightarrow CA / BB$
 $B \rightarrow b / SB$
 $C \rightarrow a$
 $A \rightarrow a$

- Solⁿ
- (1) No null or unit production
 - (2) Already in CNF

[Note: Left recursion: make problem]

③ Here we replace non-terminals with A_1 ,
 S with A_1
 C with A_2
 A with A_3
 B with A_4

we get

$$A_1 \rightarrow A_2 A_3 / A_4 A_4$$

$$A_4 \rightarrow b / A_1 A_4$$

$$A_2 \rightarrow b$$

$$A_3 \rightarrow a$$

we have

$$A_4 \rightarrow b / A_1 A_4 \quad [i > j]$$

$$A_4 \rightarrow b / A_2 A_3 A_4 / A_4 A_4 A_4$$

$$[A_1 \rightarrow A_2 A_3 / A_4 A_4]$$

$$A_4 \rightarrow b / b A_3 A_4 / A_4 A_4 A_4 \quad [A_2 \rightarrow b]$$

$$A_4 \rightarrow b / b A_3 A_4 / A_4 A_4 A_4$$

Here we get left recursion $[A_2 \rightarrow b]$

To remove left recursion introduce a new non-terminal as

$$A_4 \rightarrow b / b A_3 A_4 / A_4 A_4 A_4$$

$$Z \rightarrow A_4 A_4 Z / A_4 A_4$$

Now

$$A_4 \rightarrow b / b A_3 A_4 / b z / b A_3 A_4 Z$$

The grammar is

$$A_1 \rightarrow A_2 A_3 / A_4 A_4$$

$$A_4 \rightarrow b / b A_3 A_4 / b z / b A_3 A_4 Z$$

$$A_2 \rightarrow b$$

$$A_3 \rightarrow a$$

$$Z \rightarrow A_4 A_4 Z / A_4 A_4$$

Now convert it into GNF as

$$A_1 \rightarrow b A_3 / b A_4 / b A_3 A_4 A_4 / b z A_4 / b A_3 A_4 Z A_4$$

$$A_4 \rightarrow b / b A_3 A_4 / b z / b A_3 A_4 Z$$

$$A_2 \rightarrow b$$

$$A_3 \rightarrow a$$

$$Z \rightarrow b A_4 Z / b A_3 A_4 A_4 Z / b z A_4 Z / b A_3 A_4 Z A_4 Z$$

This is the required GNF

Pushdown Automata (PDA)

- ↳ a PDA is a machine that is designed to implement context-free language
- ↳ CFL is accepted by PDA

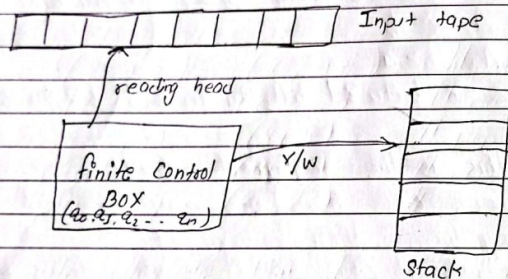


Fig. pushdown Automata

- ↳ Similar to FA, but it has memory called pushdown Automata.
- ↳ element can be read-write on stack

Formally,

A PDA consist 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 read)

(How) Γ = symbol 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 a PDA and instantaneous description is (q, x, y)

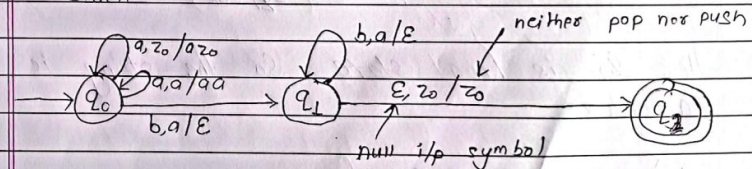
Eg:

$(q_0, a_1 a_2 \dots a_n, z_0 z_1 \dots z_n)$ is ID. This describe when PDA is in current state q_0 and input symbol to be processed is $a_1 a_2 \dots a_n$ and PDA has $z_0 z_1 \dots z_n$ where z_0 is top element, z_1 is the second top element and z_n is the lowest element

Q. Design a PDA for $L = \{a^n b^n \mid n \geq 1\}$ Solⁿ Concept:

→ push every 'a' in the stack

→ Then, reading each 'b', remove each 'a' from the stack.



Thus PDA is

$A = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$

where

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

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

$\Gamma = \{a, z_0\}$

$q_0 = \{q_0\}$

$z_0 = \text{top of stack}$

$F = \{q_2\}$

δ can be written as

$\delta(q_0, a, z_0) = (q_0, a z_0)$ // push 'a'

$\delta(q_0, a, a) = (q_0, a a)$ // push 'a'

$\delta(q_0, b, a) = (q_1, \epsilon)$ // pop 'a'

$\delta(q_1, b, a) = (q_1, \epsilon)$ // pop 'a'

$\delta(q_1, \epsilon, z_0) = (q_2, z_0)$

Now we will simulate PDA for string "aaa bbb" as

$(q_0, aaabbb, z_0) \vdash (q_0, aaabbb, a z_0)$
 $\vdash (q_0, aaabbb, a a z_0)$
 $\vdash (q_0, aaabbb, a a a z_0)$
 $\vdash (q_1, aabbb, a a a z_0)$
 $\vdash (q_1, abb, a a a z_0)$
 $\vdash (q_1, ab, a a z_0)$
 $\vdash (q_1, b, a z_0)$
 $\vdash (q_1, \epsilon, z_0)$
 $\vdash (q_2, z_0)$

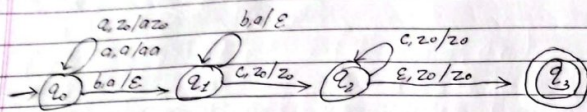
Here q_2 is final state. So the string is accepted.

Assignment: Design PDA for

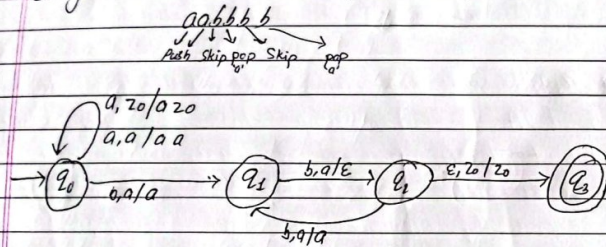
① $L = \{a^n b^{2n} \mid n \geq 1\}$

② $L = \{a^n b^n c^m \mid n, m \geq 1\}$

Q. Design PDA for $L = \{a^n b^m c^n \mid n, m \geq 1\}$

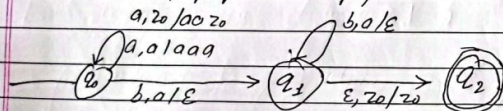


Q. Design PDA for $L = \{a^n b^{2n} \mid n \geq 1\}$



or

For each a push two a
For each b, pop each pop

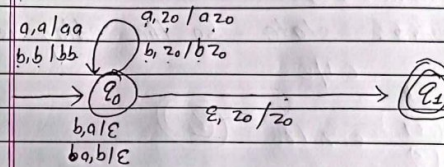


Q. Design PDA to accept $L = \{w \mid w \in (a+b)^*$ and $n_a(w) = n_b(w)\}$ where equal no. of a's and b's

→ Initially when stack is empty, whatever we read push in stack

→ if we read 'a' and top of stack is 'b', pop b

→ if we read 'b' and top of stack is 'a', pop a



Q. Design PDA to accept $L = \{w \mid w \in (a+b)^*$ and $n_a(w) > n_b(w)\}$ where a is greater than b

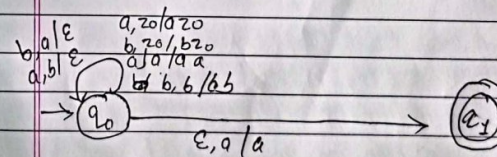
Concept

Initially whatever the symbol push into stack.

If we read a and top of stack b, pop b

If we read b and top of stack a, pop a

Finally if we read ε, then stack should contain 'a' on top of stack.



2/28

Date _____
Page _____

Suppose string 'abbaa'

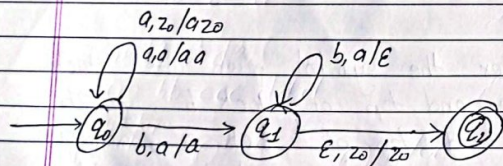
$(q_0, abbaa, z_0) \vdash (q_0, bbaa, az_0)$
 $\vdash (q_0, bbaa, \emptyset z_0)$
 $\vdash (q_0, aa, bz_0)$
 $\vdash (q_0, a, \emptyset z_0)$
 $\vdash (q_1, \emptyset, az_0)$
 $\vdash (q_1, \emptyset, \emptyset)$

Here q_1 is the final state. So the string is accepted.

Q. Design a PDA for $L = \{a^n b b^n \mid n \geq 0\}$

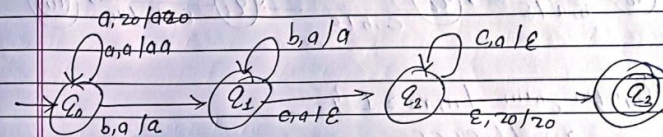
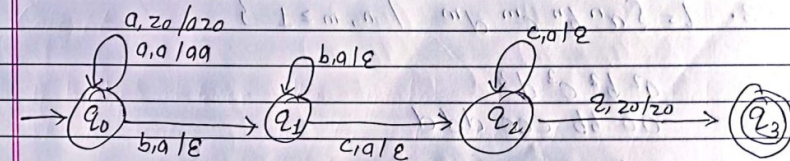
Concept

Skip first 'b'

Q. Design PDA for $L = \{a^n b^m c^n \mid n, m \geq 1\}$

Concept

Skip 'b'

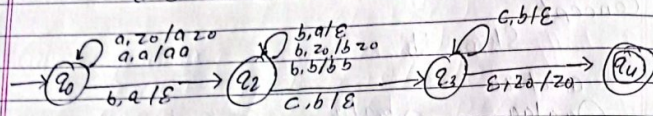
Q. Design PDA for $L = \{a^m b^n c^n \mid n, m \geq 1\}$ Assignment

- ① $L = \{a^n b^{m+n} c^m \mid n, m \geq 1\}$
- ② $L = \{a^n b^m c^{n+m} \mid n, m \geq 1\}$
- ③ $L = \{a^n b^n c^m d^m \mid n, m \geq 1\}$
- ④ $L = \{a^n b^m c^m d^n \mid n, m \geq 1\}$
- ⑤ $L = \{a^i b^j c^k \mid i = j + k\}$

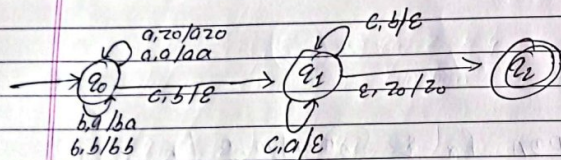
Assignment

Date _____
Page _____

1. $L = \{a^n b^{m+n} c^m \mid n, m \geq 1\}$
 Let $a^n b^m \cdot b^n c^m$
 $a^n b^n b^m c^m$

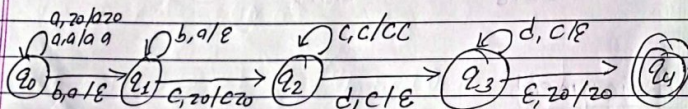


2. $L = \{a^n b^m c^{n+m} \mid n, m \geq 1\}$
 $aa\ bb\ cccc$



3. $L = \{a^n b^n c^m d^m \mid n, m \geq 1\}$

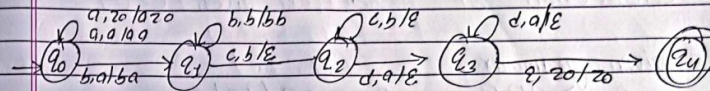
$a^2 b^2 c^3 d^3$
 $aa\ bb\ ccc\ ddd$



Assignment

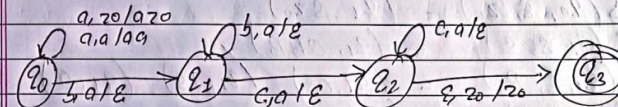
Date _____
Page _____

4. $L = \{a^n b^m c^m d^n \mid n, m \geq 1\}$ $aa\ bbb\ ccc\ dd$



5. $L = \{a^i b^j c^k \mid i = j + k\}$

$aaaa\ bb\ ca$



Equivalence of CFG and PDA

Construction of PDA for given CFG

Eg: Given grammar
 $S \rightarrow 0BB, B \rightarrow 0S/1S/0$
 And PDA.

We defined PDA as

$$P = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$$

Now define transition function δ as

$$\begin{aligned}\delta(q_0, \epsilon, z_0) &= (q_1, Sz_0) \\ \delta(q_1, \epsilon, S) &= (q_1, 0BB) \\ \delta(q_1, \epsilon, B) &= (q_1, 0S), (q_1, 1S), (q_1, 0) \\ \delta(q_1, 0, 0) &= (q_1, \epsilon) \\ \delta(q_1, 1, 1) &= (q_1, \epsilon) \\ \delta(q_1, \epsilon, z_0) &= (q_f, z_0)\end{aligned}$$

Suppose for "01000"

$$\begin{aligned}(q_0, \epsilon 01000, z_0) &\vdash (q_1, \epsilon 010000, Sz_0) \\ &\vdash (q_1, 010000, 0BBz_0) \\ &\vdash (q_1, 10000, BBz_0) \\ &\vdash (q_1, 1000, 1SBz_0) \\ &\vdash (q_1, \epsilon 0000, SBz_0) \\ &\vdash (q_1, \epsilon 0000, 0BBBz_0)\end{aligned}$$

$$\begin{aligned}&\vdash (q_1, \epsilon 000, BBBz_0) \\ &\vdash (q_1, 000, 0BBz_0) \\ &\vdash (q_1, \epsilon 00, BBz_0) \\ &\vdash (q_1, 00, 0BBz_0) \\ &\vdash (q_1, \epsilon 0, Bz_0) \\ &\vdash (q_1, 0, 0z_0) \\ &\vdash (q_1, \epsilon, z_0) \\ &\vdash (q_f, z_0)\end{aligned}$$

Q. Construct PDA for $S \rightarrow aSa / bSb / c$

Sol: We define PDA as:

$$P = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$$

Now, define transition function as.

$$\begin{aligned}\delta(q_0, \epsilon, z_0) &= (q_1, Sz_0) \\ \delta(q_1, \epsilon, S) &= (q_1, aSa), (q_1, bSb), (q_1, c) \\ \delta(q_1, a, a) &= (q_1, \epsilon) \\ \delta(q_1, b, b) &= (q_1, \epsilon) \\ \delta(q_1, c, c) &= (q_1, \epsilon) \\ \delta(q_1, \epsilon, z_0) &= (q_f, z_0)\end{aligned}$$

Now we simulate PDA for "abbcbbba"

$$\begin{aligned}(q_0, \epsilon abbcbbba, z_0) &\vdash (q_1, \epsilon abbcbbba, Sz_0) \\ &\vdash (q_1, abbcbbba, aSa z_0) \\ &\vdash (q_1, bbcbbba, Sa z_0) \\ &\vdash (q_1, bcbba, bSba z_0) \\ &\vdash (q_1, cbba, Sbba z_0) \\ &\vdash (q_1, cba, Sbba z_0) \\ &\vdash (q_1, ba, Sbba z_0)\end{aligned}$$

$\vdash (q_1, cbba, cbba z_0)$
 $\vdash (q_1, bba, bba z_0)$
 $\vdash (q_1, ba, ba z_0)$
 $\vdash (q_1, a, a z_0)$
 $\vdash (q_1, \epsilon, z_0)$
 $\vdash (q_f, z_0)$

Construct of CFG from PDA

$PDA = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$
 $CFG = \{V, \Sigma, P, S\}$

$q_0, T = \{E\}, S = \{S\}$

Suppose, $Q = \{q_0, q_1\}$, $F = q_1$ of a PDA
 Now, variable 'v' can be defined as
 $[Q \Gamma Q]$

$S_0, v = \{S, [q_0 a q_1], [q_0 a z_0], [q_1 a z_0], [q_1 a q_1]\}$

Suppose we have transition function δ as

1. $\delta(q_0, a, z_0) = (q_0, \epsilon) \Rightarrow POP$
 $[q_0 a z_0] \rightarrow a$

2. $\delta(q_0, a, z_0) = (q_0, a z_0) \Rightarrow push$

$[q_0 z_0] \rightarrow a [q_0 a] [z_0]$

3. $\delta(q_0, b, a) = (q_1, a) \Rightarrow skip$
 $[q_0 a] \rightarrow b [q_1 a]$

4. $\delta(q_0, a, z_0) = (q_1, a x x)$

$[q_0 z_0] \Rightarrow a [q_1 a] [x] [x]$

Q. Construct CFG from PDA

Given PDA $A = \{q_0, q_1\}$ s.t. $\delta(q_0, b, a) = (q_1, z_0)$, $\delta(q_1, z_0, \epsilon) = (q_0, \epsilon)$
 given by

$\delta(q_0, a, z_0) = (q_0, a z_0)$

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

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

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

$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

Solⁿ (1) For $\delta(q_0, a, z_0) = (q_0, a z_0)$

$P_1: [q_0 z_0 z_0] \rightarrow a [q_0 a z_0] [z_0 z_0]$

$P_2: [q_0 z_0 z_0] \rightarrow a [q_0 a q_1] [q_1 z_0 z_0]$

$P_3: [q_0 z_0 q_1] \rightarrow a [q_0 a z_0] [z_0 z_0 q_1]$

$P_4: [q_0 z_0 q_1] \rightarrow a [q_0 a q_1] [q_1 z_0 q_1]$

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

$P_5: [q_0 a z_0] \rightarrow a [q_0 a z_0] [z_0 a z_0]$

$P_6: [q_0 a z_0] \rightarrow a [q_0 a q_1] [q_1 a z_0]$

$P_7: [q_0 a q_1] \rightarrow a [q_0 a q_1] [q_0 a q_1]$

$P_8: [q_0 a q_1] \rightarrow a [q_0 a q_1] [q_1 a q_1]$

$$\textcircled{3} \quad \delta(q_0, b, a) = (q_1, \epsilon)$$

$$p_9: [q_0 a q_1] \rightarrow b$$

$$\textcircled{4} \quad \delta(q_1, b, a) = (q_2, \epsilon)$$

$$p_{10}: [q_0 a q_2] \rightarrow b$$

$$\textcircled{5} \quad \delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$$

$$p_{11}: [q_1 z_0 q_1] \rightarrow \epsilon$$

$$\textcircled{6} \quad S \rightarrow [q_0 z_0 z_0] / [q_0 z_0 q_1]$$

Now we reduce useless symbol (non-reachable and non-generating symbol)

1. Here we remove p_2 and p_6 because $[q_1 z_0 z_0]$ and $[q_1 a q_0]$ are non-generating symbol.
2. Remove p_5 because generating itself
3. Remove p_1, p_3, p_7 because $[q_0 a q_0]$ does not generate any symbol
4. Remove $[q_0 z_0 z_0]$ from S because it is non generating symbol

So, we have CFG,

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

where

$$V = \{S, [q_0 z_0 q_1], [q_0 a q_1], [q_1 z_0 q_1], [q_1 a q_1]\}$$

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

$$S = \{S\}$$

P is given as

$$S \rightarrow [q_0 z_0 q_1]$$

$$p_4: [q_0 z_0 q_1] \rightarrow a [q_0 a q_1] [q_1 z_0 q_1]$$

$$p_8: [q_0 a q_1] \rightarrow a [q_0 a q_1] [q_1 a q_1]$$

$$p_9: [q_0 a q_1] \rightarrow b$$

$$p_{10}: [q_1 a q_1] \rightarrow b$$

$$p_{11}: [q_1 z_0 q_1] \rightarrow \epsilon$$

Closure properties of CFL

(i) Closed under union

Let L_1 and L_2 be two CFL generated by CFG
 $G_1 = \{V_1, \Sigma_1, R_1, S_1\}$ and $G_2 = \{V_2, \Sigma_2, R_2, S_2\}$

Now we construct a new language $L(G)$ using grammar $G = \{V, \Sigma, R, S\}$ such that it can accept $L(G_1) \cup L(G_2)$ where

$$V = V_1 \cup V_2 \cup \{S\}$$

$$\Sigma = \Sigma_1 \cup \Sigma_2$$

$$R = R_1 \cup R_2 \cup \{S \rightarrow S_1 | S_2\}$$

S = start state

Now let us choose a string $w \in \Sigma_1 \cup \Sigma_2$ if $S_1 \xrightarrow{*} w$ or $S_2 \xrightarrow{*} w$ and in our grammar we have $S \rightarrow S_1 | S_2$, so S will lead the string w

$$\text{Hence } L(G) = L(G_1) \cup L(G_2)$$

proved

(ii) Closed under concatenation

Let L_1 and L_2 two context free language ^(CFL) generated by $G_1 = \{V_1, E_1, R_1, S_1\}$ and $G_2 = \{V_2, E_2, R_2, S_2\}$ respectively.

Now, we construct a new language $L(G_1)$ using the grammar $G = \{V, E, R, S\}$ such that it can accept $L(G_1) \cdot L(G_2)$ where

$$V = V_1 \cup V_2 \cup \{S\}$$

$$E = E_1 \cup E_2$$

$$R = R_1 \cup R_2 \cup \{S \rightarrow S_1 S_2\}$$

$$S = \text{start symbol}$$

Now let us choose a string $w_1 \in E_1$ and $w_2 \in E_2$ we know that $S_1 \xrightarrow{*} w_1$ and $S_2 \xrightarrow{*} w_2$ but in the above grammar G , S derives $S \rightarrow S_1 S_2$.

So, S will lead the concatenation of the string w_1 and w_2 i.e. $w_1 \cdot w_2$ and the language will be $L_1 \cdot L_2$. Hence $L_1 \cdot L_2$ is also CFL.

(iii) closed under Kleene star

Let L_1 be the CFL generated by CFG $G_1 = \{V_1, E_1, R_1, S_1\}$. Now we construct a new language $L(G_1)$ using the grammar $G = \{V, E, R, S\}$ such that it can accept Kleene star of the language L_1 i.e. L_1^*

where

$$V = V_1 \cup \{S\}$$

$$E = E_1 \cup \{\Lambda\}$$

$$R = R_1 \cup \{S \rightarrow \Lambda, S \rightarrow S_1 S_1\}$$

$$S = \text{start symbol}$$

Here R follows all the properties of CFG as R_1 is the production of given CFG and $S \rightarrow \Lambda, S \rightarrow S_1 S_1$ also fulfill the requirement so we say that G is a CFG that can generate CFL L_1^*

* Context free language are not closed under Intersection

We know that $L_1 = \{a^n b^n c^n \mid n \geq 1, i \geq 0\}$ and $L_2 = \{a^i b^n c^n \mid n \geq 1, j \geq 0\}$ are CFL.

Now L_1 intersection L_2 i.e. $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$
So, $L = L_1 \cap L_2$

Here, $L = \{a^n b^n c^n\}$ is not CFL

Hence $L_1 \cap L_2$ is not context free

* Context free language are not closed under complement

Suppose L_1 and L_2 are CFL

Assume $\overline{L_1}$ and $\overline{L_2}$ are CFL.

Now using De Morgan's Law

$$L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$$

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.

Hence By contradiction CFL is not closed under complement.

Decision property of CFL (Algorithm)

1. Deciding whether a CFL is finite

Construct a non-redundant CFG, G generating $L \cap \Sigma^n$. We draw a directed graph whose vertices are variable in G . If $A \rightarrow BC$ is a production then there are directed edges from A to B and A to C . L is finite if and only if the directed graph has no cycle.

2. Deciding whether a CFL is empty

Given any CFL, L , there is a CFG, G to generate it. We can determine using the construction describe in the context of elimination of useless symbols, whether the start symbol is useless, if so, then $L(G) = \emptyset$; 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 and past at most length of string w .

If string w is found in any of the derivation trees, then $w \in L$.

Pumping Lemma for CFL

Statement: Let L be a context free language and 'n' be the length of string or pumping length such that

1. Every $z \in L$ and $|z| \geq n$ can be written as $uvwxy$, for some u, v, w, x and y .

2. $|vwx| \geq 1$.

3. $|vwx| \leq n$ (i.e. if u and y are n , then $|vwx| \leq n$).

4. $uv^kwx^ky \in L$ for all $k \geq 0$ (i.e. generate infinite number of string by setting any value for k).

Proof:

To prove the theorem, we consider a CFG whose production are given as

$$S \rightarrow AB \mid a$$

$$A \rightarrow aB \mid a$$

$$B \rightarrow bA \mid b$$

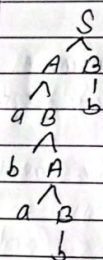
Note: should be derived from grammar

Now let any string $z = ababb$ such that $z \in L$

Thus we decompose z as

$$u = a, v = ba, w = b, x = a, y = b$$

Now, draw a parse tree for given string



$T_0: z = ababb$



$T_1: z_1 = b$

As z and z_1 are derivation of T and a proper subtree T_1 of T , so we can write

$$z = uvwx y$$

$$\therefore z = u z_1 y$$

As z_1 and z_2 are derivation of T_1 and proper subtree T_2 of T_1 we can write $z_1 = v z_2 x$ (i.e. $uvwx$)

Also, we have $|vwx| \geq |w|$

So, we have $|vwx| \geq 1$

Thus we have $z = uvwx y$ with $|vwx| \leq n$ and $|vx| \geq 1$ where n is length of string z .

As T is a S -tree and T_1 and T_2 are B -Tree (i.e. starting from symbol 'B').

We get

$$S \xrightarrow{*} uBy$$

$$B \xrightarrow{*} vBx$$

$$B \rightarrow w$$

Now, $S \rightarrow uBy$

$$\rightarrow uvBxy \quad [B \rightarrow vBx]$$

$$\rightarrow uvWxy \quad [B \rightarrow w]$$

Thus $S \rightarrow uv^k wx^k y$, where $k \geq 1$

Similarly,

$$S \rightarrow uBy$$

$$\rightarrow uvBxy \quad [B \rightarrow vBx]$$

$$\rightarrow uvvBxxy \quad [B \rightarrow vBx]$$

$$\rightarrow uv^2Bx^2y \quad [B \rightarrow w]$$

and so on.

Thus $uv^k wx^k y \in L$ for $k \geq 1$

Hence proved.

* Procedure to prove given language is not CFL.

Step 1: Assume L is context free. let n be pumping length.

Step 2: Choose $z \in L$ so that $|z| \geq n$ write $z = uvwxy$ using pumping lemma.

Step 3: Find a suitable k show that $uv^kwx^ky \notin L$. This is a contradiction so, L is not CFL.

Q. Show that $L = \{a^n b^n c^n \mid n \geq 1\}$ is not CFL.

Solⁿ

Step 1: Assume L is context free language. Let n be pumping length.

Step 2: Let $z = a^n b^n c^n$. Then $|z| = 3n \geq n$. Write $z = uvwxy$, where $|vx| \geq 1$.

Step 3: Here $z = uvwxy$. So, $uvwx = a^n b^n c^n$. As $1 \leq |vx| \leq n$, v or x cannot contain all three symbols.

Suppose $n=4$, so $z = a^4 b^4 c^4$.

Case I:

v and x , each contain only one type of symbol.

$\begin{array}{ccccccc} a & aa & a & bbbb & c & c & c \\ \boxed{u} & \boxed{v} & \boxed{w} & \boxed{x} & \boxed{y} & & \end{array}$

We have,

$$uv^kwx^ky$$

For $k=2$

$$\begin{aligned} & uv^2wx^2y \\ &= aaaaaabbbbcccc \\ &= a^6b^4c^6 \end{aligned}$$

Here no. of a 's, b 's and c 's are not equal. So, $uv^kwx^ky \notin L$ for $k=2$.

Case II:

Either v or x can have more than one symbol.

$\begin{array}{ccccccc} aa & aabb & bb & c & cc \\ \boxed{u} & \boxed{v} & \boxed{w} & \boxed{x} & \boxed{y} \end{array}$

We have uv^kwx^ky

So for $k=2$

$$\begin{aligned} & uv^2wx^2y = uvvwxxy \\ &= aaaaaabbbaabbcccc \end{aligned}$$

Since, it is not in pattern. So,

$uv^kwx^ky \notin L$ for $k=2$.

Here in both cases we get contradiction. So given language $L = \{a^n b^n c^n \mid n \geq 1\}$ is not CFL.

Q. Show that $L = \{a^n b^n \mid n \in \mathbb{N}\}$ is not CFL

→

1. Assume L is CFL, n is pumping length

2. Let $z = a^n b^n a^n b^n$ then $|z| = 4n \geq n$
where $z = uvwxy$ where $|ux| \geq 1$

3. Here $z = uvwxy$

so, $uvwxy = a^n b^n a^n b^n$

as, $1 \leq |ux| \leq n$, v or x cannot contain all 4 symbol

Suppose $n = 4$

Case 1: v and x contain only one type of symbol

So $z = \underbrace{00}_u \underbrace{0011}_v \underbrace{1100}_w \underbrace{001111}_x \underbrace{1111}_y$

we have $uv^k wx^k y$

For $k=2$, $uv^2 wx^2 y$

$= 00 0000 11 11 00 00 01 11 11$

$= 0^6 1^4 0^6 1^4$

Here no. of 1's and 0's are not equal so,

$uv^k wx^k y \notin L$

Case 2: Either v or x can have 2 symbol

so $\underbrace{00}_u \underbrace{0011}_v \underbrace{1100}_w \underbrace{001111}_x \underbrace{1111}_y$

for $k=2$

i.e $z = 00 0011 0011 1100 0000 1111$

$= 0^4 1^2 0^2 1^4 0^6 1^4$

Here not in pattern so, it $\notin L$

The both two condition get contradiction. So given L is not in CFL.

Q. Prove that $L = \{a^n b^n c^n \mid n \geq 0\}$ is not CFL

1. Consider L is CFL also Let $z = a^n b^n c^n$, $z \in L$

According to pumping lemma of CFL

z decompose into U, V, W, X and Y as follows:

$U = a^r$

$V = a^s$ ($s \geq 0$)

$W = a^{p-(r+s)}$

$X = b^t$ ($t > 0$)

$Y = b^{(p-t)} c^p$

Now using pumping lemma, $z = UV^k WX^k Y$, $k \geq 0$
We have.

let $k=2$

so, $z = UV^2 WX^2 Y$

$= a^r (a^s)^2 a^{p-(r+s)} (b^t)^2 b^{(p-t)} c^p$

$= a^r a^{2s} a^{p-r-s} b^{2t} b^{(p-t)} c^p$

$= a^{(p+s)} b^{(p+t)} c^p$

$\neq a^p b^p c^p$

Hence our assumption $z \in L$, contradicts with our result

so L is not CFL.

Turing Machine

Turing Machine

- mathematical method of general purpose computer
- TM can solve any problem that can be solved by any computer machine
- developed / designed by 'Alan Turing'

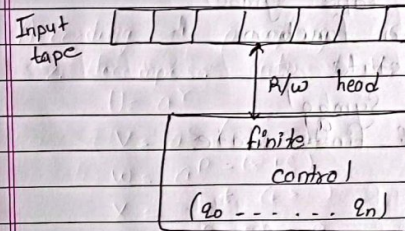


fig: Turing Machine (TM)

→ TM has 1x tape which is divided into number of cell. Each cell can store only one number of system.

→ In one move, the TM machine read the present symbol under the R/w head and on the tape and the present state of finite control to determine

- 1) A new symbol to be written on the tape under the R/w head
- 2) A motion of R/w head along, the head either the head move once at the left ^{or} and once at right
- 3) The next step of automata
- 4) Whether to halt or not.

Mathematically, TM has 7 tuples:

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

where,

Q = set of finite states

Σ = i/p symbol

Γ = set of tape symbol

δ = transition function

q_0 = initial state

b = blank symbol

F = set of final state

Instantaneous Description (ID)

An ID of Turing Machine M is a string $\alpha\beta\gamma$, here β is present state of M , the entire input string is split as $\alpha\beta\gamma$. The first of symbol of γ is the current symbol under the R/w head and α as all the symbol of input string and the input string α is the substring of input string formed by all the symbol to the left of present state.

Example:

Present state	Tape symbol		
	b	0	1
$\rightarrow q_1$	$1Lq_2$	$0Rq_1$	$-$
q_2	bRq_3	$0Lq_2$	$1Lq_2$
q_3	$-$	bRq_4	bRq_5
q_4	$0Rq_5$	$0Rq_4$	$1Rq_4$
(q_5)	$0Lq_2$	$-$	$-$

Consider above TM, draw computation of input string "001"

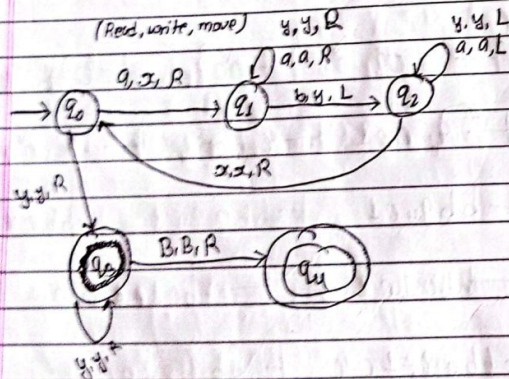
$\Rightarrow (q_1, 00b) \vdash 0q_10b \vdash 00q_1b \vdash 00q_201$
 $\vdash q_2001 \vdash q_2b001 \vdash bq_3001$
 $\vdash bbq_401 \vdash bb0q_41 \vdash bb0q_1q_4b$
 $\vdash bb010q_5b \vdash bb010q_200$
 $\vdash bb0q_2100 \vdash bbq_20100$
 $\vdash bq_2b0100 \vdash bbq_30100$
 $\vdash bbbq_4100 \vdash bbb1q_400 \vdash bbb10q_40$
 $\vdash bbb100q_4b \vdash bbb1000q_5b$
 $\vdash bbb1000q_200 \vdash bbb10q_2000 \vdash bbb10000$

$\vdash bbb a_2 10000 \vdash bb a_2 b 10000 \vdash bbb a_2 10000$
 $\vdash bbb b 2 50000$

Since q_5 is final step so accepted.

Design a TM

Q. Design a TM that accepts $L = \{a^n b^n \mid n \geq 1\}$



Hence, Required TM, M is

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

where

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

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

$$\Gamma = \{a, b, B\}$$

$$q_0 = \{a_0\}$$

$(q_0, a) \rightarrow (q_1, x, R)$
 change state write skip move

$$b = \{B\}$$

$$F = \{q_4\}$$

and δ is

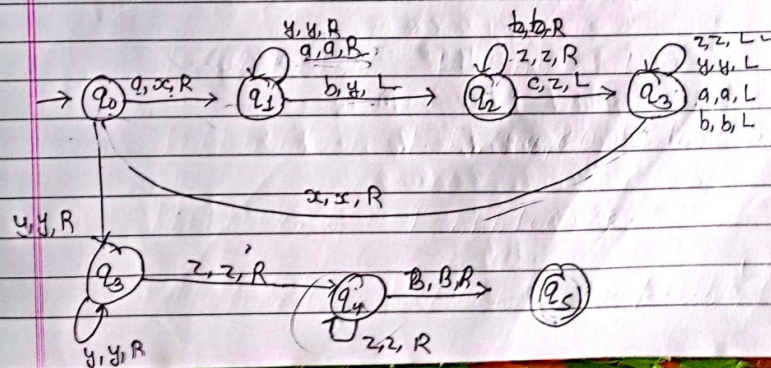
state (symbol)	Tape symbol				
	a	b	B	x	y
$\rightarrow q_0$	xR, q_1	-	-	-	yB, q_3
q_1	aR, q_2	yL, q_2	-	-	yR, q_1
q_2	aB, q_2	-	-	x, q_0	yL, q_2
q_3	-	-	BR, q_3	-	yR, q_3
q_4^*	-	-	-	-	-

Now, we simulate TM for string 'aabb':

$(q_0, aabb) \vdash xq_1abb \vdash xaq_2bb \vdash xq_2ayb$
 $\vdash q_2xayb \vdash xq_0ayb \vdash xxq_1yb$
 $\vdash xxq_1yb \vdash xxq_2yy \vdash xq_2xyy$
 $\vdash xxq_0yy \vdash xxq_3yy \vdash xxq_3yB$
 $\vdash xxq_4Bq_4B$

Q. Design a TM that accept $L = \{a^n b^n c^n \mid n \geq 1\}$

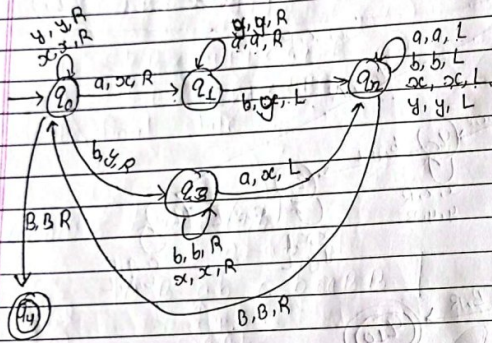
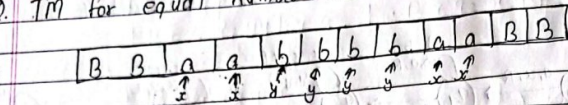
$\overbrace{B B B}^x \overbrace{a a a}^y \overbrace{c c c}^z \overbrace{B B B}^x$
 $\uparrow \quad \uparrow \quad \uparrow$



Date _____
Page _____

Q. TM for equal number of a's & b's

String



B ~~a~~ ~~g~~ ~~h~~ bbb a a B

↑ → ⊕ → ⊗

x ↑

↑ ← ↑ ←

B x a y b b b a a n
 $\uparrow \rightarrow \uparrow \rightarrow x \rightarrow y \rightarrow y$
 $\uparrow \quad \quad \quad \uparrow$

$b x x y y . b b a a$
 $\rightarrow \uparrow \uparrow \uparrow \uparrow, y \uparrow x$

$\uparrow$
 $b x x y y y b x a$ 1
 $\uparrow \rightarrow \dots \rightarrow y \rightarrow \rightarrow \rightarrow$ 5

Here Required Tm, m is

$$m = \{a, e, f, g, q_0, b, F\}$$
$$Q = \{q_0, q_1, q_2, q_3, q_4\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \{a, b, 0\}$$
$$q_0 = q_1$$
$$b = \{a\}$$
$$F = (q_1)$$

8 is

Symbol	a	b	B	x	y
q_0	xRq_1	yRq_2	BRq_4	xRq_0	yRq_0
q_1	aRq_1	yLq_2	—	—	yRq_1
q_2	aLq_2	bLq_2	BRq_0	xLq_2	yLq_2
q_3	xLq_2	bRq_3	—	xRq_3	—
q_4	—	—	BRq_4	—	—

String 'aabbbaa'

$(a_0, a_0 b b b b a a)$
 $\vdash x a_1 a b b b b a a$
 $\vdash x a a_1 b b b b a a$
 $\vdash x a a_1 a y b b b a a$
 $\vdash x a a_1 y b b b a a$
 $\vdash x a x y a_1 b b b a a$
 $\vdash x a x a_2 y y b b a a$
 $\vdash x a_2 x y y b b a a$
 $\vdash a_2 x a x y y b b a a$
 $\vdash a_2 b x x y y b b a a$
 $\vdash b x a_0 x y y b b a a$
 $\vdash b x x a_0 y y b b a a$
 $\vdash b x x y a_0 y b b a a$
 $\vdash b x x y y a_0 b b a a$
 $\vdash b x x y y y a_3 b a a$
 $\vdash b x x y y b a_3 a a$

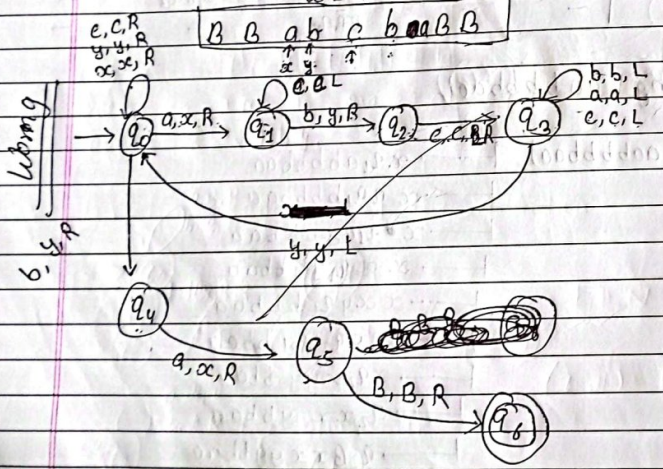
$\vdash Bxxyyzbxz \vdash Bxxyyzbxz$
 $\vdash Bxxzxyybxz \vdash Bxxzxyybxz$

 $\vdash BxxxyyyxxB$
 $\vdash BxxxyyyxxB$

 $\vdash Bxxxyyyxxq_0B$
 $\vdash BxxxyyyxxBq_4$

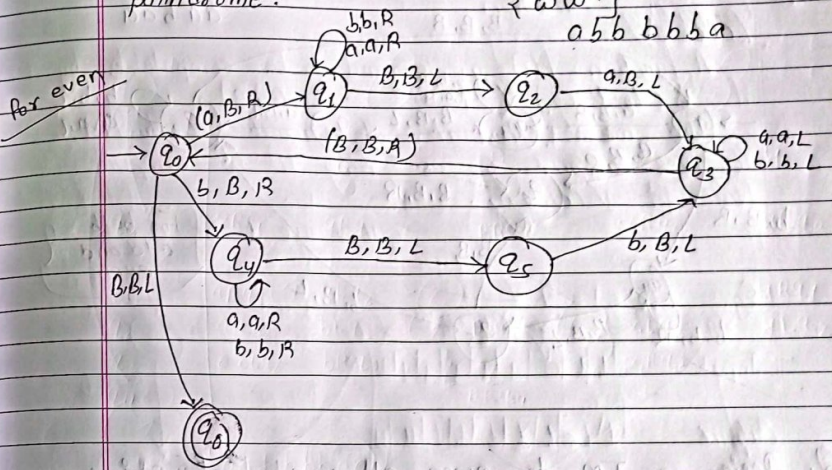
is the final state...

b. $w \in (a,b)^*$
 $w = ab$

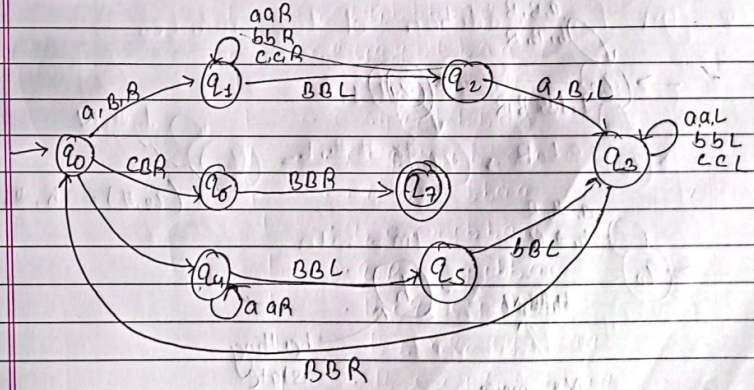


Q. Design TM to check string $w \in (a,b)^*$ is palindrome.

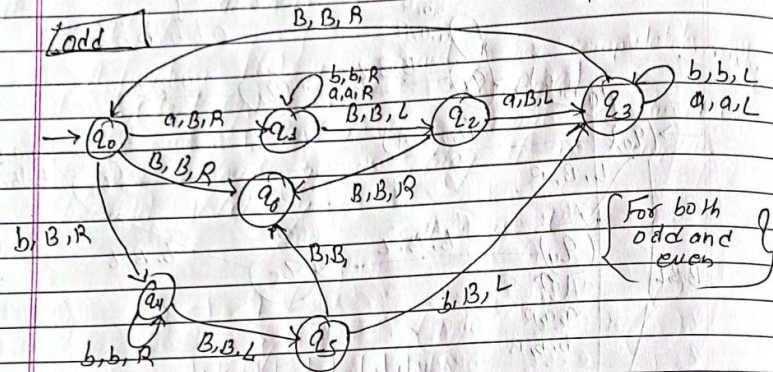
$\{w w^R\}$
 $a b b b b a$



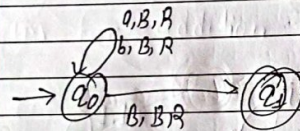
(b) $w \in (a,b)^*$



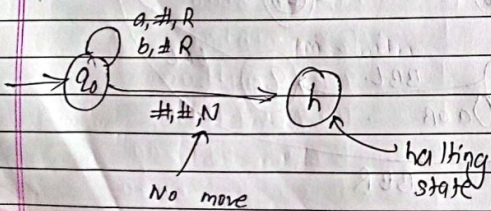
for checking string $w \in (a,b)^*$ is palindrome
a b a b a



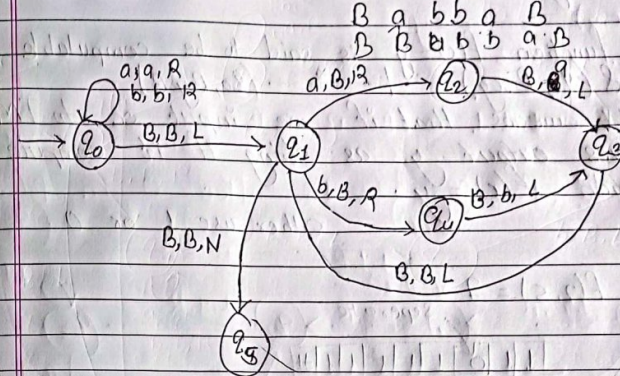
Q. Design a TM to erase all symbol of string over $\Sigma = \{a, b\}$.



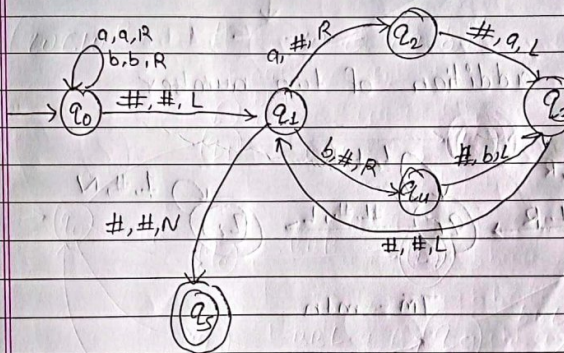
or



Q. Design a TM as a right shift machine that transforms $\#w\#$ to $\#\#w\#$ with $\Sigma = \{a, b\}$



$\# a b a b b a \#$



Computing Function

A function $f(x) = y$ is said to be computable by TM if $\exists (Q_0, B, q_f)$ such that if we input x to TM, it gives output y as string if it compute function $f(x) = y$.
either use 1 or 0.

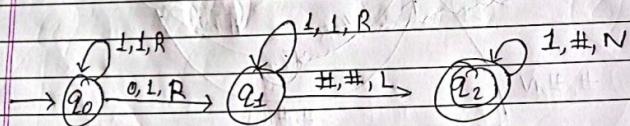
Eg: $f(2, 5) = 7$

Input: 11011111# (separate)

Output: 11101111##

TM

* TM for addition of two number



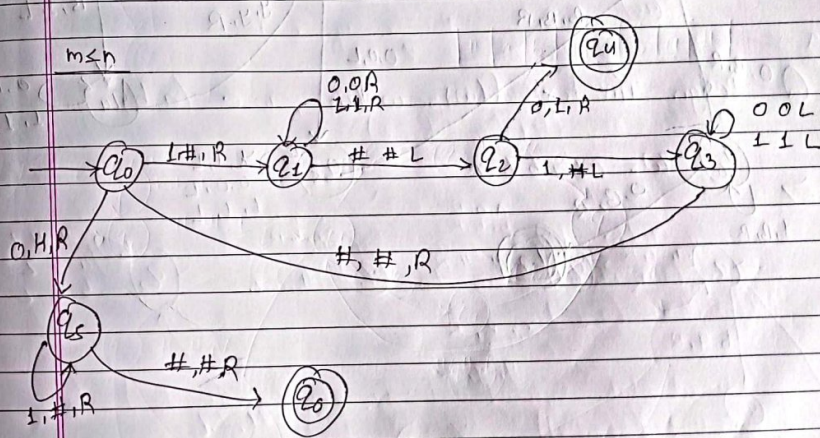
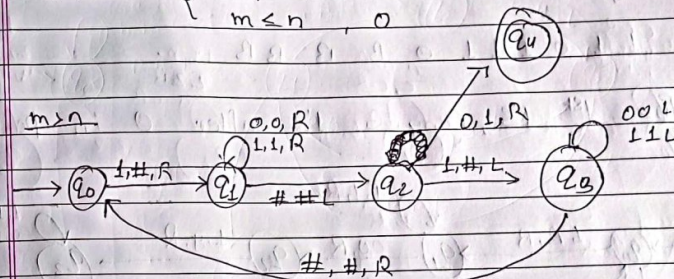
$$TM = m+n$$

Similarly, $f(n) = n+2$

$f(n) = n+1$

⑧ TM for subtraction of two number

$$f(m,n) = \begin{cases} m-n & m \geq n \\ 0 & m < n \end{cases}$$



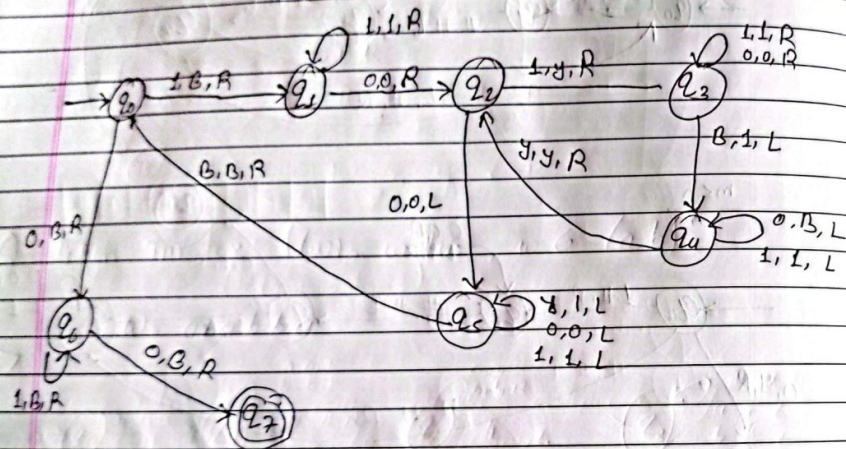
x. Turing Machine for multiplication

$$f(x,y) = x \times y$$

Eg: 2×3

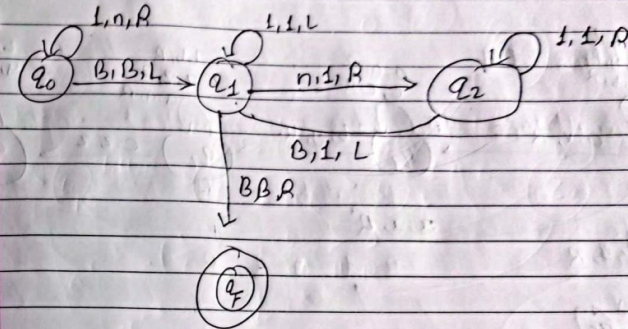
1 1 0 1 1 1 0 B B B B ...

For 1st 1's, Copy 3 1's after 0 in position



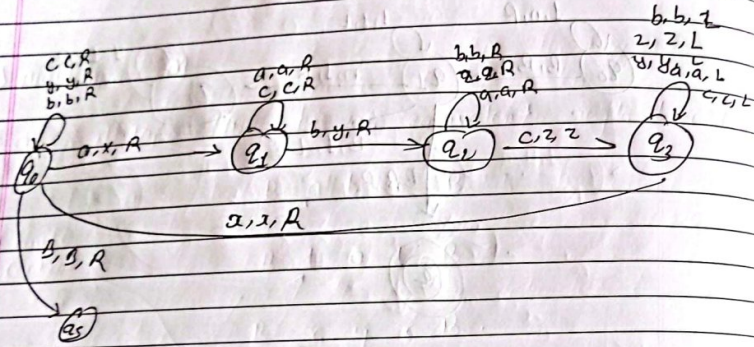
x. T.M for $f(n) = 2^{2n}$

1 1 0 1 1 0 B B n=2
zero mark at end



Date _____
Page _____

Q Design a TM for equal no. of a's, b's and c's.

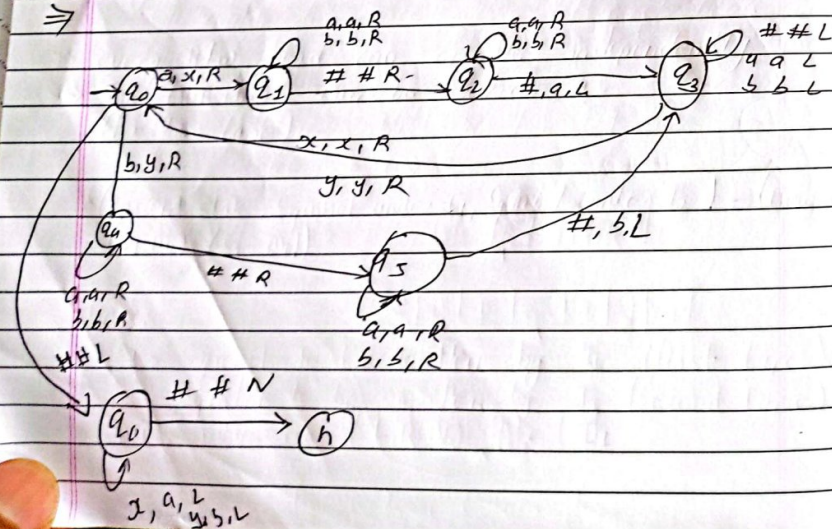


Imp

Q. Design TM for $L(w) = w \neq w, w \in (a,b)^*$

$w = ababba$

7



Date _____
Page _____

Extension (Variation) of Tuning Machine

- 1) Infinite on two side of input tape
- 2) Multi head on single tape
- 3) Multi tape TM
- 4) Non-deterministic TM

1) Infinite on two side of input tape

--- B B a b a B B ---

- Left and Right side have infinite blank symbol.
- In standard TM, blank symbol is define in Γ
- In this TM, we do not include blank symbol in Γ
- Eg: $a^n b^n$

2) Multi head on Single tape.

Suppose two heads are defined in a single tape, H_1, H_2 . These two heads can read/write/move independently.

$\therefore$

B	a	b	b	a	b	a	B
---	---	---	---	---	---	---	---

The transition can be defined as $\delta(\text{current state}, \text{Symbol Under } H_1, \text{Symbol under } H_2) = (\text{new state}, (s_1, m_1), (s_2, m_2))$

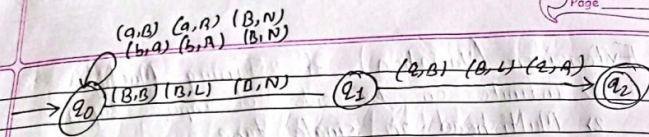
Here, $s_1 \Rightarrow$ symbol to be written under H_1

$S_2 \nrightarrow$ Symbol to be written under H_2

$m_1 = \text{move}$ i.e. (L, R, N) for H_1

$$m_2 = m_{\text{H}_2} \text{ i.e. (LRN) for } H_2$$

Eg: $\begin{array}{ccccccc} & 3 & 3 & 3 & 3 & 3 & 3 \\ & B & B & B & B & B & B \\ B & 1 & 1 & 1 & 1 & 1 & 1 \\ & \uparrow & & & & & \uparrow \\ & H_1 & & & & & H_2 \end{array} \quad (q_L, q, B) = (q_L(B, L), (q,$



3) Multitape TM

→ every multitape TM has an equivalent single tape TM

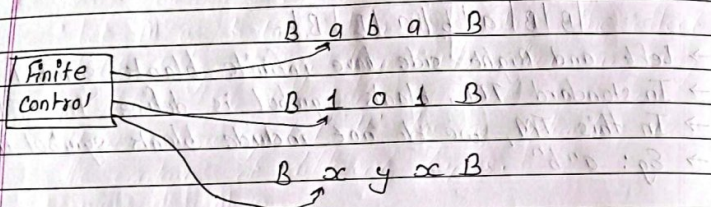


Fig: Multitape TM
(k tape, k=3)

→ every R/w head can operate independently

→ The transition can be define as

(For two tape)

$\delta(\text{current state, Symbol under } I_1, \text{Symbol under } I_2) = (\text{new state, } (S_1, M_1) (S_2, M_2))$

Here,

S_1 = Symbol to be written by I_1 (first tape)

S_2 = Symbol to be written by I_2 (second tape)

M_1 = move i.e. (L, R, N) for I_1

* Multitape TM

Theorem: For any k -tape (multitape) TM, there is an equivalent single tape TM.

Let $k \geq 1$ be any k -tape TM. The key idea to convert multitape TM is to show the simulation of multitape TM 'M' in single tape TM's.

• Let M as k -tape, then S simulate the effect of k -tape by storing their information on single tape.

• It uses new symbol '#' as a delimiter to separate the contents of different tapes.

• In Addition, to the contain of those tapes, S must be ~~keep~~ track of the location of the head by ~~for~~ writing a tape symbol with a dot (.). To make the place where the ~~track~~ head on that tape would be.

Following figure illustrate how 1 tape can be used to represent 3-tape.

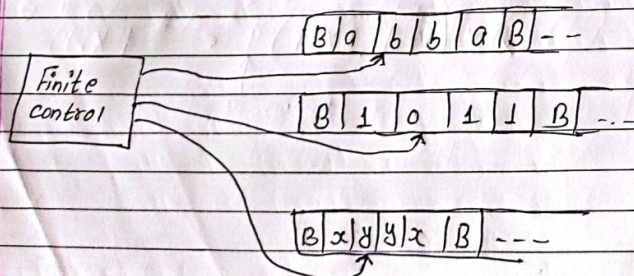


fig: k -tape (where $k=3$)

Finite control

a b b a # 1 0 1 1 # x y y x # B - -

(fig: representing 3 tapes with one tapes)

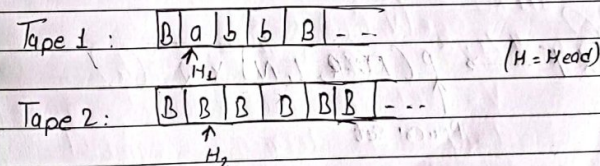
To simulate a transition from state Q , we must scan our tape to see which symbol are under k tape head i.e. scan the dots (.).

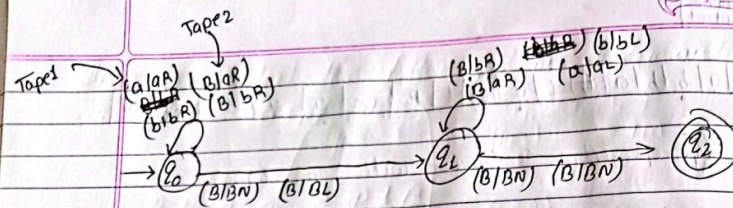
Once we determine this and are ready to make the transition, we must scan across the tape again to update the cell and move the dots.

Whenever one tape, whenever one head moves of the right end; we must shift our tape so we can insert a blank symbol.

Construct a 2 tape TM to convert an input w into ww^R , $w \in (a,b)^*$.

Initially tape 1 contains w and tape 2 is blank. Let w be 'abb'.





* Non-Deterministic TM

A TM is non-deterministic if there are multiple possibilities for the current input and present state.

Formal definition

The non-deterministic is combined of 6 Tuples: $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

$Q \rightarrow$ set of state

$\Sigma \rightarrow$ set of i/p state

$\Gamma \rightarrow$ set of tape symbol

$q_0 \rightarrow$ initial state

$F \rightarrow$ set of final state

δ is

$$Q \times \Gamma \rightarrow P(\Gamma \times R, L, N) \times Q$$

Power set

In this transition function the left side of transition is enclosed in power set, which means that a state can move to more than one state on getting a particular input.

All possible transition sequence of non-deterministic TM on a given input string can be represented by computation tree.

For eg:

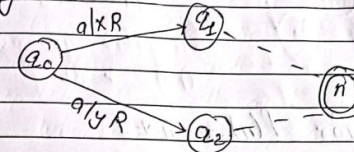
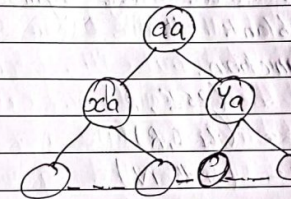


Fig: non deterministic TM

We draw computation tree for string 'aa'



The outcome of NTM is

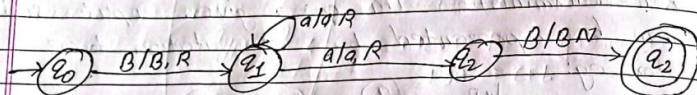
- 1) If any branch accept the input string, then NTM is also accept
- 2) If all branches halt or reject the input, then NTM also rejected

9/18

Date
Page

Eg: Consider the following non-deterministic TM that accept a^+

B B a a a B B



Design a TM for $f(n) = n+1$

$\# / I / I / I / \# / \#$
 $\uparrow$
 Halt

$w \rightarrow i/p$

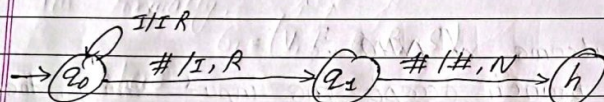
To represent n we use symbol I
 for blank $\#$

Transition function

$\delta(q_0, I) = (q_0, I, R)$

$\delta(q_0, \#) = (q_1, I, R)$

$\delta(q_1, \#) = (h, \#, N)$



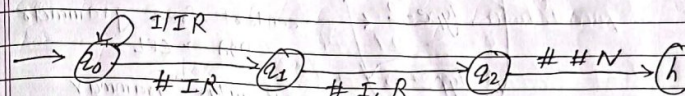
Design a TM for $f(n) = n+2$

$\delta(q_0, I) = (q_0, I, R)$

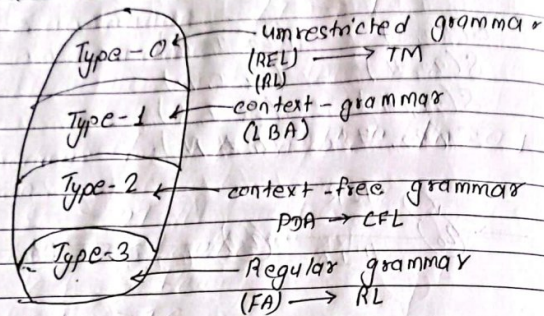
$\delta(q_0, \#) = (q_1, I, R)$

$\delta(q_1, \#) = (q_2, I, R)$

$\delta(q_2, \#) = (h, \#, N)$



Unrestricted Grammar



↳ phase structure grammar

Unrestricted Grammar is the most general in the Chomsky Hierarchy. It is called unrestricted because no other restriction is made on this except each of their LHS being non-empty. The left hand side of their rule can contain terminal and non-terminal but the condition is at least one of them must be non-terminal.

A TM can simulate an unrestricted grammar and unrestricted grammar can simulate TM. Formally, $G = (V, T, S, P, S)$

where

V = set of non-terminal symbol

T = set of terminal symbol

S = starting symbol, $S \in V$

P = set of rules or production

Each production is of the form, $\alpha \rightarrow \beta$

where α is $(V \cup T)^+ V / (V \cup T)^+$
 β is $(V \cup T)^+$

The language generated by an unrestricted grammar is called Recursively Enumerable Language (REL).
 The REL is accepted by TM.

Eg: Given grammar for $L = \{a^{2^n} \mid n \geq 0\}$
 Production s:

$S \rightarrow TaU$

$U \rightarrow E / AU$

$aA \rightarrow Aaa$

$TA \rightarrow T$

$T \rightarrow E$

Derive a^8 .

$S \rightarrow TaU$

$\rightarrow TaAU \quad [U \rightarrow AU]$

$\rightarrow TaAAU \quad [U \rightarrow AU]$

$\rightarrow TaAAA U \quad [U \rightarrow AU]$

$\rightarrow TaAAAC \quad [U \rightarrow E]$

$\rightarrow T Aaa AA \quad [aA \rightarrow Aaa]$

$\rightarrow T Aa Aaa A \quad [aA \rightarrow Aaa]$

$\rightarrow T AAaaaa A$

$\rightarrow T Aaaa Aaa \quad [aA \rightarrow Aaa]$

$\rightarrow T aa Aaaaa \quad [aA \rightarrow Aaa, TA \rightarrow T]$

$\rightarrow TaAaaaaa$

$\rightarrow TAaaaaaa$

$\rightarrow T a a a a a a a a$ $[T \rightarrow T]$
 $\rightarrow \varepsilon a a a a a a a a$ $[T \rightarrow \varepsilon]$
 $\rightarrow a a a a a a a a$

Q. Given grammar for $L = \{a^n b^n c^n \mid n \geq 0\}$
production:

$S \rightarrow UT$

$U \rightarrow \varepsilon / aubc$

$Cb \rightarrow bc$

$CT \rightarrow TC$

$T \rightarrow \varepsilon$

Derive $a^3 b^3 c^3$

$S \rightarrow UT$

$\rightarrow aubcT$ $[U \rightarrow aubc]$

$\rightarrow aaubcbCT$ $[U \rightarrow aubc]$

$\rightarrow aaubcbcbCT$ $[U \rightarrow aubc]$

$\rightarrow aaabcbcbcbCT$ $[U \rightarrow \varepsilon]$ $[Cb \rightarrow bc]$

$\rightarrow aaabbbcbcbCT$ $[Cb \rightarrow bc]$

$\rightarrow aaabbbcbcbCT$ $[T \rightarrow \varepsilon]$

$\rightarrow aaabbbcbcb$

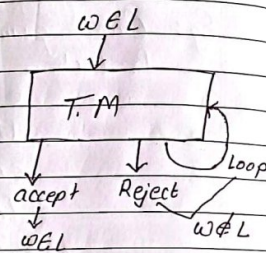
Recursively Enumerable Language (REL)

A language 'L' is said to be recursively enumerable language (REL) if there exists a TM which will accept and halt for all the input string which are in language 'L'. But may or may not halt for input string which are not in L.

It means TM can loop forever for the string which are not in 'L'.

It is also known as Turing
Recognizable language or
Partially decidable

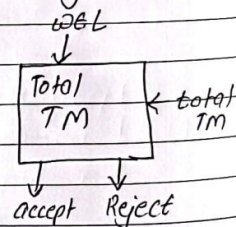
Eg: All regular language,
context-free languages,
Recursive language are REL

# Recursive language

A language 'L' is said to be recursive if there exists a Turing machine which will accept all the string in 'L' and reject all the string not in 'L'.

The Turing machine will halt every time and give an answer (accepted or rejected) for each and every input.

Eg: $a^n b^n$, $a^n b^n c^n$ language
are recursive



language $L = \{a^n b^n c^n \mid n \geq 1\}$ is recursive because we can construct TM which will move to final state if string is of the form $a^n b^n c^n$ else move to nonfinal state. So Turing machine will always halt in this case.

5

Undecidability

Date
Page

Decidable language

3/24
Date
Page
(Yes/no or answer 3/15/24)

↳ if it is recursive language (TM & GE)

Partially decidable language

↳ if it is recursive enumerable language

Undecidable language

↳ no design or made Turing Machine

↳ (Answer is not decidable by yes or no)

↳ It can be partially decidable but not decidable

Decidable problem

↳ solution

Yes No

↳ it has an algorithm for given i/p

Decidable language or Problem:

Eg: Equivalence of two regular language,

① Finiteness of regular language,

② Emptiness of regular language

Example of undecidable problem / language:

- (*) Ambiguity of CFL (No TM)
- (*) Everything or Completeness of CFG (No TM)

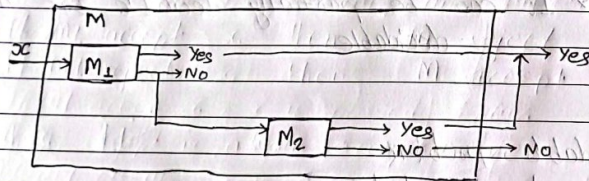
infinite combination

Imp

Properties of Recursive language

(1) Union:

If L_1 and L_2 are two recursive language, then their Union i.e. $L_1 \cup L_2$ will also be recursive.



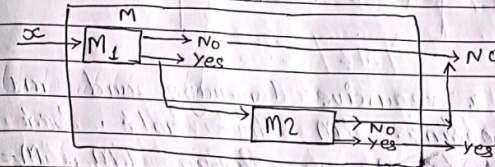
Let L_1 and L_2 be two recursive language and M_1, M_2 be their corresponding TM that halt. Let x belongs to $L_1 \cup L_2$ ($x \in L_1 \cup L_2$) be a string to M . So we design a TM, $M = M_1 \cup M_2$ such that:

- 1) If M_1 accepts x then M accept x .
- 2) If M_1 reject x then x is passed to M_2 .
- 3) If M_2 accept x then M accept x .
- 4) If M_2 reject x then M reject x .

Thus $M = L_1 \cup L_2$ is a recursive language.

(2) Intersection

If L_1 and L_2 are two recursive language then their intersection i.e. $L_1 \cap L_2$ will also be recursive.



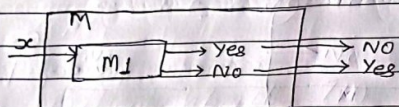
Let L_1 and L_2 be two recursive language and M_1 and M_2 be their corresponding TM that halt. Let $x \in L_1 \cap L_2$ be a string to M . So, we design a TM, $M = M_1 \cap M_2$ such that

- 1) If M_1 reject x then M also reject x .
- 2) If M_1 accept x then x is passed to M_2 .
- 3) If M_2 accept x then M also accept x .
- 4) If M_2 reject x then M also reject x .

Thus $L_1 \cap L_2$ is a recursive language.

③ Complement : $\bar{L}_1$

If L_1 is a recursive language then its complement i.e. $\bar{L}_1$ will also be recursive.



Let L_1 be a recursive language and M_1 be its corresponding TM that halt. Let α be a string to M , we design TM $M = \bar{M}_1$ such that:

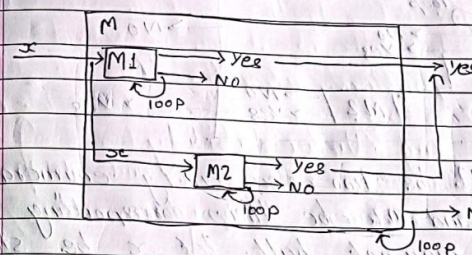
- 1) IF M_1 accept α then M reject α
- 2) IF M_1 reject α then M accept α

Thus $\bar{L}_1$ is also a recursive.

Properties of Recursively Enumerable Language

④ Union

If L_1 and L_2 are recursively enumerable then $L_1 \cup L_2$ will also be recursively enumerable.



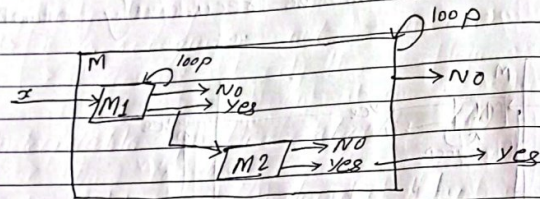
Let L_1 and L_2 be two recursively enumerable language M_1 and M_2 are their corresponding TM that may or may not halt. Let α be string to M . We design TM, $M = M_1 \cup M_2$ such that:

- 1) IF M_1 accept α , then M also accept α .
- 2) IF M_2 accept α , then M also accept α
- 3) IF M_1 and M_2 rejects α or goes infinite loop, then M may reject or goes on infinite loop.

Thus $L_1 \cup L_2$ is also a recursively enumerable language.

② Intersection

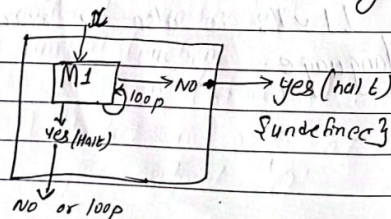
If L_1 and L_2 are two recursively enumerable then their intersection $L_1 \cap L_2$ is also recursively enumerable



Let L_1 and L_2 be two recursively enumerable language. M_1 and M_2 be their corresponding TM that may or may not halt. Let x be string to M . we design TM, $M = M_1 \cap M_2$ such that:

- 1) If M_1 accept x , x is passed to M_2
- 2) If M_2 accept x , M accept x .
- 3) If M_1 or M_2 reject or goes on infinite loop, then M also ^{may} reject or goes on infinite loop

③ Complement: If L_1 is a recursively enumerable language, $\overline{L_1}$ is a not recursively enumerable language.



If M_1 accept x , then and halt then M may goes on loop or reject.

If x does not belong to L_1 then M_1 may goes on loop or reject.

So for this we cannot define accept state in TM M . So L_1 is not recursively enumerable language.

Halting Problem

→ undecidable problem (no TM)

let M be

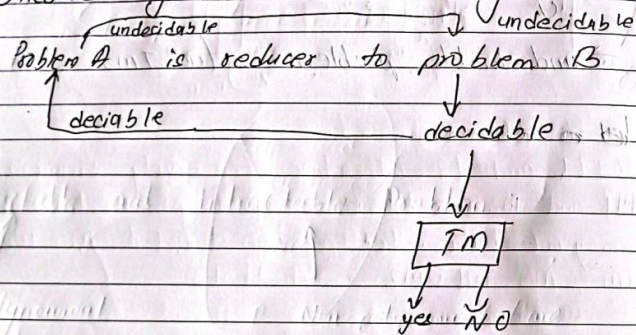
Universal Turing Machine

- ↳ A TM is said to be universal if it accept
 - i) i/p data
 - ii) description (algorithm) for computing

↳ is a recognizer not a decider

Undecidable problem about TM

④ Reducibility and Undecidability



Recursive Function Theory

↳ developed by Alonzo Church,

↳ design for natural number 'N'

There are initial function:

1) 0-place function (z function): $z()$

→ it takes no argument and give output '0' i.e. $z() = 0$

2) 1-place function: successor function

Eg: $f(n) = n + 1$

$f(4) = 5$, $f(3) = 4$

$f(n) = n + 2$

$f(n) = n(n-1)$

$f(n) = n + 3$

$f(x, y) = x \cdot y$

④ Primitive recursive function:

→ A function is primitive recursive function if

1) it is one of the basic (initial) three function.

2) it is obtained by performing operation like composition, recursion or minimization on these basic three function

Three function:

① zero function, $z() = 0$

② Successor function, Eg $f(n) = n + 1$

③ Projector function, $p_k(x_1, x_2, x_3, \dots, x_n) = x_k$

Eg: $p_3(2, 4, 6, 8) = 6$

$p_4(1, 3, 5, 7, 9, 10) = 7$

What do mean by computational or computable?

- ↳ anything that can be accepted by TM
- ↳ if given enough time and space

Complexity Theory

- it consider not only the problem are solvable but also how efficient it can be solved.
- problem are classified on the basis of difficulty
- Time complexity and Space complexity
- ↑ are major two factor of its
- ↑ no. of steps
- ↑ memory used

Tractable and Intractable Problem:

Polynomial time

Non-polynomial time

i.e. exponential time

Factorial time

(*) problem:

recursion language
adding
subtractable

(*) problem:

undecidable problem

Imp

Complexity classes:

- (1) P Classes problem (Easy)
- (2) NP Class problem (Hard)

(1) P Class problem

- polynomial time decidable problem
- having algorithm or TM
- $O(n^k) = n \log^n$
- Deterministic TM
- Tractable problem

(2) NP class problem

→ Non-deterministic polynomial time decidable problem

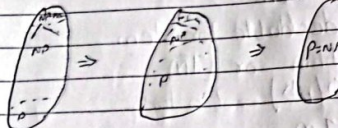
- { non-polynomial
- exponential: $O(k^n)$
- Factorial: $O(n-1)!$
- $\log k^n$

- ~~Decidable~~ Intractable problem
- not practical applied
- there DTM that can verified the ~~sety~~ solution in polynomial time.
- Machine of NP class can be created but not in polynomial time
- also know as intractable problem.
- Eg: (1) Traveling sales man
- (2) Hamilton problem graph

	Decidable	Verifies
P	✓	✓
NP	X	✓

iii) NP-hard

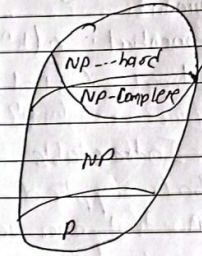
$\Rightarrow A = NP$



iv) NP-complete

if problem is

- NP-hard
- NP



Q. Design DFA that accept three loop keyword used in C.

Keyword: while, for do

