Page No.: _____

Date __/__/____

1. $(P, e, e) \rightarrow (q, S)$ - (as 'S' is starting non terminal in CFG)
2. $(q, e, A) \rightarrow (q, \alpha)$ - for each rule $A \rightarrow \alpha$ in CFG.
3. $(q, a, a) \rightarrow (q, e)$ - for each $a \in \Sigma$.

Example:

Design a PDA for the following CFG

$G = (V, \Sigma, R, S)$ with

$V = \{S\}$

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

R is defined as

$S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow c$

$S \Rightarrow aSa$

$\rightarrow aaSaa$

$\rightarrow abSbaa$

$\rightarrow abcbaa$

Date __/__/____

सबैभन्दा पहिले PUSH गर्ने
मिल्दै पहिले POP गर्ने

SOL

Let PDA be

$M = (K, \Sigma, \Gamma, \Delta, s, F)$

$K = \{P, q\}$

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

$\Gamma = \{S, a, b, c\}$

$s = \{P\}$

$F = \{q\}$

and Δ is defined as

1. $(P, e, e) \rightarrow (q, S)$
2. $(q, e, S) \rightarrow (q, aSa)$
3. $(q, e, S) \rightarrow (q, bSb)$
4. $(q, e, S) \rightarrow (q, c)$
5. $(q, a, a) \rightarrow (q, e)$
6. $(q, b, b) \rightarrow (q, e)$
7. $(q, c, c) \rightarrow (q, e)$ //

State	unread input	Stack	Transition
P	abbcbba	e	-
q	abbcbba	S	1
q	abbcbba	aSa	2
q	abbcbba	Sa	5
q	abbcbba	bSba	3
q	bbcbba	Sba	6
q	bcbba	bSbba	3
q	bcbba	Sbba	6
q	cbba		

q	cbba	cbba	8
q	bba	bba	7
q	ba	ba	6
q	a	a	6
q	e	e	5

properties of CFL

Closure property

The family of CFLs are closed under union, concatenation and kleene star.

proof let L_1 and L_2 be two CFL generated by the following CFGs respectively

$G_1 = (V_1, \Sigma_1, R_1, S_1)$

$G_2 = (V_2, \Sigma_2, R_2, S_2)$

For union: Consider the language $L(G_1)$, generated by the following grammar.

$G = (V, \Sigma, R, S)$

$V = \{V_1 \cup V_2 \cup \{S\}$

$\Sigma = \Sigma_1 \cup \Sigma_2$

S is start symbol and, productions rules R is defined as follows:

$R = \{R_1 \cup R_2 \cup \{S \to S_1 / S_2\}$

Now, let us choose a string $w \in \{\Sigma_1 \cup \Sigma_2\}^*$ then if $S_1 \overset{*}{\Rightarrow} w$ or $S_2 \overset{*}{\Rightarrow} w$ and.

in our grammar

$S \to S_1 | S_2$ hence S will lead.

∴ Hence G is a CFL language so $L(G_1)$ is CFL

i.e $L(G_1) = L_1 \cup L_2$ is a CFG

For concatenation:

Conser the language $L(G_1)$, generated by the following grammar

$G = (V, \Sigma, R, S)$

$V = \{V_1 \cup V_2 \cup \{S\}$

$\Sigma = \Sigma_1 \cup \Sigma_2$

S is start symbol and productions rules R is defined as follows

$R = \{R_1 \cup R_2 \cup \{S \to S_1 S_2\}$.

let w_1 be a string $w_1 \in L_1$ and w_2 is a string $w_2 \in L_2$ we know that $S_1 \Rightarrow^* w_1$ and $S_2 \Rightarrow^* w_2$ but in above grammar G_1 $S \to S_1 S_2$ so, S will lead to $w_1 w_2$ in broder way the language of grammer G will be in $L_1 L_2$, since L_1 is CFL and L_2 is CFL and G is CFG so $L_1 L_2$ also CFL

Date __/__/____ Page No.: _____

Regular expression & language

Regular expression this descrip a language by means of symbol and $\emptyset$ combine with symbol '$\cup$' and '$*$'

→ The regular expression over & alphabet Σ^* are all string over alphabet $\Sigma \cup \{\epsilon, \supset, \emptyset, \cup, *\}$ that can be obtained as

1) $\emptyset$ and each member of Σ is regular
2) If α and β are regular expression then so is $\alpha\beta$
3) If α and β are regular expression then so is $\alpha \cup \beta$
4) If α is a regular expression then so is α^*
5) Nothing is a regular expression unless, it follows from ① to ④

⇒ Every regular expression is associated with same language

Example of regular expression and language

$(ab)^* = \{\epsilon_\perp, ab, abab, ababab \cdots\}$ - zero or more

$ab(ab)^* = \{ab, abab, ababab \cdots\}$ - one or more

$(a+b+c)^*$ = any string with $\Sigma = \{a, b, c\}$

$(a+b+c)^* (a+b+c)$ = any non empty string with $\Sigma = \{a, b, c\}$

Date __/__/____

Regular languages - The regular languages are those languages that can be obtained from the ~~key~~ big three set operation (a) union (b) concatemation (c) kleen star

A regular language can be defined as follows. Let Σ be an alphabet, the class of regular languages over Σ is defined inductively as

a) $\emptyset$ is a regular language.
b) If α and β are regular language the so is $\alpha \cup \beta$
c) If α and β are regular expression ~~then~~ then so is α
d) If α is a regular expresion then so is α^*
e) Nothing is a regular expresion unless it follows from (a) to (d)

Writing Regular expression:

1) Write a regular expression for the language $L = \{w \in \{a,b\}$ ~~has~~ has even number of 'a' followed by odd numb of $\{b\}$.

2) Ending with a '0' and not containg '11' as substr

$0 \Rightarrow (0 \cup 10)$

$10 \quad (0 \cup 10)^* (0 \cup 10)$

"Failure comes only when we forget our ideals and objectives and principles." —Jawaharlal Nehru

$\Rightarrow (0+10)^* \quad 11(0+1)^*$

Application of Regular expression:

Regular Expression are widely used in various fields of computation and fields of applications are overlapped with that of finite automata.

Some of the application are:-

1) Regular expression in UNIX
   - → The symbol . (dot) represents any character.
   - → (A-Z) represents uppercase characters
   - → The sequence $[a_1, a_2, a_3 \ldots\ldots a_k]$ represents the regular expression $a_1 \cup a_2 \cup a_3 \ldots\ldots a_k$

→ [+ - . 0-9] represents digits with plus minus and dot

2) Lexical Analysis

- Lexical analyzer is a software (part of compiler) which seperatep the source program into piece called tokens which represents identifies, keywords and operators

3) Finding patterns in texts - text scanning

Algebraic laws for regular expression:-

1) Associative and commutative:-

$L \cup M = M \cup L$

$(L \cup M) \cup N = L \cup (M \cup N)$

$(LM)N = L(MN)$

2) Identities and Annililators:-

$\emptyset \cup L = L \cup \emptyset = L$

$\epsilon \cdot L = L \cdot \epsilon = L$

$\emptyset L = L \emptyset = \emptyset$

3. Distributive law

$L(M \cup N) = LM \cup LN$

$(M \cup N)L = ML \cup NL$

4. The idempotent law

$L \cup L = L$

5. Laws of closure

$(L^*)^* = L^*$

$\emptyset^* = \epsilon$

$L^+ = L \cdot L^*$

$L^* = L^+ \cup \epsilon$

State Minimization:-

→ State minimization is the process of minimizing the given DFA by removing unreachable states and by merging the similar/equivalent states

Steps

1. Remove unrechable states from given DFA (if any)
2. Construct a transition table for the rest of the states
3. Divide the transition table of step 2 into two parts as-
   2) Rows that start with final states.

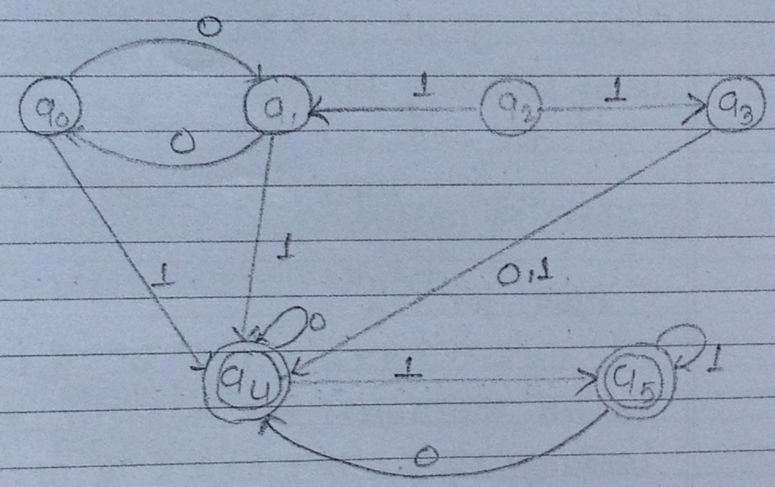
"Whenever you take a step forward, you are bound to disturb something." —Indira Gandhi

"Failure comes only when we forget our ideals and objectives and principles." —Jawaharlal Nehru

b) Rows that start with non-final states.

4. Eliminate equivalent/similar states from table 3(a) and (b) separately.
5. Repeat step 4 until equivalent states are appeared
6. Combine/ merge updated tables
7. Draw the minimize diagram.

Minimize the following DFA



Soln Here, q_2 and q_3 are unreachable states, so we have to eliminate that states and transition table for the rest of the state is

δ/Σ	0	1
$\rightarrow q_0$	q_1	q_4
q_1	q_0	q_4
$*q_4$	q_4	q_5
$*q_5$	q_4	q_5

Dividing the above transition table as: Table 1

δ/Σ	0	1
q_4	q_4	q_5
q_5	q_4	q_5

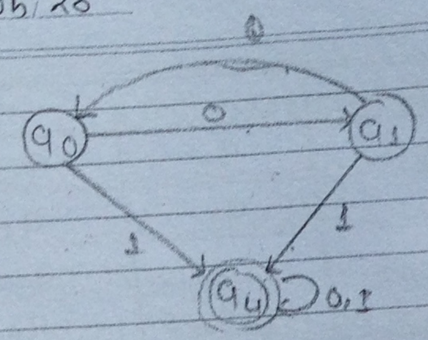
δ/Σ	0	1
$\rightarrow q_0$	q_1	q_4
q_1	q_0	q_4

δ/Σ	0	1
q_4	q_4	q_4

Merging the two tables

δ/Σ	0	1
q_0	q_1	q_4
q_1	q_0	q_4
q_4	q_4	q_4

Date 07/05/28



Pumping lemma for Regular languages (proof by contradiction - used to show that the given language is not regular.

Statement

Let L be regular language and w be any string we L, n be a pumping constant with $n \geq 1$ and $|w| \geq n$. Then w can be rewritten as $w = xyz$ such that

$|xy| \leq n$

$|y| \geq 1$

$xy^i z \in L$ for all $i \geq 0$

Example

Show that $L = \{a^i b^i : i \geq 1\}$ is not regular by using pumping lemma for regular language.

Proof:

Let $L = \{a^i b^i : i \geq 1\}$ is a regular language. w be any string $w \in L$ and n be a pumping constant (a constant integer $n \geq 1$) with $|w| \geq n$ Then w can be re-written as

"Whenever you take a step forward, you are bound to disturb something." —Indira Gandhi

Date

Page No.:

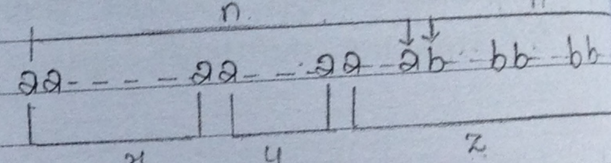
$w = xyz$ such that

$|xy| \leq n$

$|y| \geq 1$

$xy^i z \in L$ for all $i \geq 0$.

For the pumping lemma, we can write $w = xyz$ which can be written as



Here $y = a^k$; $k \geq 1$

using pumping lemma,

for $i = 0$; we get

$xy^i z = xy^0 z = xz = a^{n-k} b^n \notin L$

Also for $i = 2$

$xy^2 z = xyyz = a^{n+k} b^n \notin L$

which contadicts the theorem. Hence the given language $L = \{a^i b^i : i \geq 1\}$ is not regular.

"Failure comes only when we forget our ideals and objectives and principles." —Jawaharlal Nehru

Date __/__/__ Page No.:

Properties of Regular languages

1) Closure properties
2) Decision properties.

1) Closure properties.

The closure language accepted by finite automate is closed under

- Union
- Concatenation
- Kleene star.
- complementation
- Intersection
- Reverse
- Set difference.
- Homomorphism (substitution of string for symbols)
- Inverse homomorphism.

2) Decision properties.

1) membership
   - Given L, is w in L?
2) Empty Test
   - is $L = \emptyset$?
3) finiteness
   - is L finite or infinite.

"Whenever you take a step forward, you are bound to disturb something." —Indira Gandhi

Date __/__/__ Page No.:

→ Union
→ concatenation
→ Kleene star

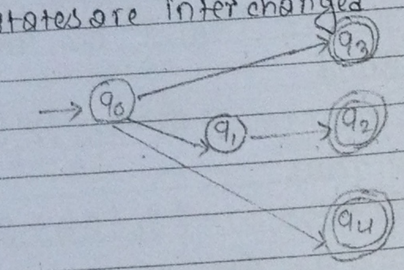
Complementation

→ start with a complete DFA, not with a NFA
→ Make every final state non final and every non final state final

So, let $M = (K, \Sigma, \delta, s, F)$ be a DFA. The complimentary language is $\bar{L} = \Sigma^* - L(M)$ is accepted by the DFA

$$\bar{M} = (K, \Sigma, \delta, s, K - F)$$

i.e $\bar{M}$ is identical to M except that final and non final states are interchanged



DFA for L

"Failure comes only when we forget our ideals and objectives and principles." —Jawaharlal Nehru

Date 07/07/22. Page No.:

Pushdown Automata (PDA')

PDA = FA + stack

PDA are essentially the finite automates with auxiliary storage element called stack. Hence PDA was control to both input ~~tap~~ tape and stack.

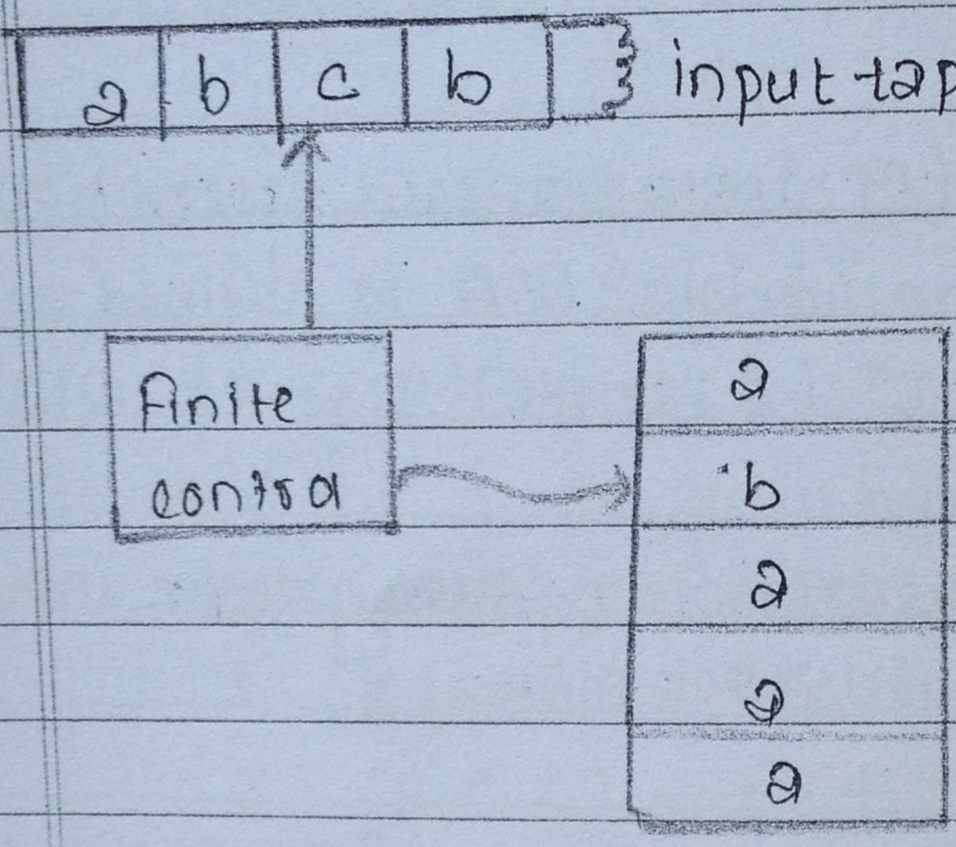


fig: A PDA

Formal Definition

A pushdown automata (PDA) $M = (K, \Sigma, T, \Delta, S, F)$ is a 6-tuple where

K is a finite set of states

Σ is an alphabet (finite set of input symbols)

T is an alphabet (finite set of stack symbols

S ∈ K is an initial state

F is a set of final states with $F \subseteq K$

Δ is a transition relation.

Date __/__/____ Page No.: _____

Δ is $K \times (\Sigma \cup \{e\}) \times \Gamma \to K \times \Gamma$

Languages of PDA

1) language accepted by empty stack
2) language accepted by final state

Configuration (or Instantaneous description (ID) of PDA

Configuration of PDA is defined as member of $K \times \Sigma^* \times \Gamma^+$ i.e triple (q, w, r) where q is the state.

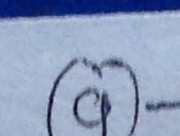
w is the remaining string to be input.

r is the stack content.

Suppose for the PDA M, $\delta(q, a, x)$ contains (p, α) then one of the configuration can be

$(q, aw, xB) \vdash (p, w, \alpha B)$

Chitra



x - input
y - stack top symbol
z - push pop symbol

Date ___/___/___ Page No.:

PDA Design examples

1. Design a PDA M that accepts a language $L = \{wcw^R : w \in \{a,b\}^*\}$.

Soln Let $M = (K, \Sigma, \tau, \Delta, s, F)$ be required PDA where

$K = \{s, f\}$

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

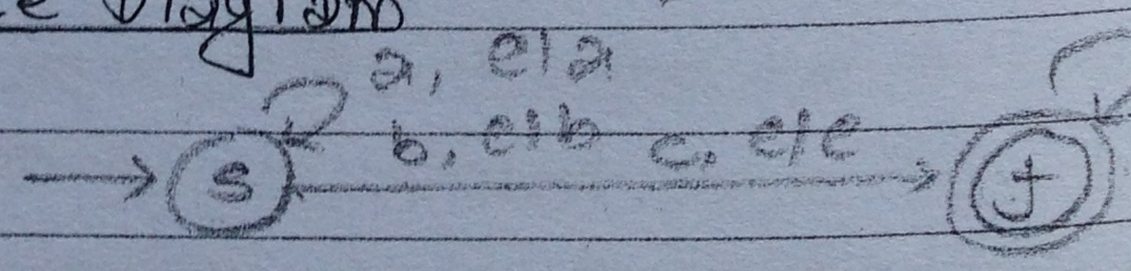
$\tau = \{a, b\}$

$F = \{f\}$

and the Δ is given by

1. $(s, a, e) \rightarrow (s, a)$ // push a
2. $(s, b, e) \rightarrow (s, b)$ // push b
3. $(s, c, e) \rightarrow (f, e)$ // just switch the s
4. $(f, a, a) \rightarrow (f, e)$ // pop a
5. $(f, b, b) \rightarrow (f, e)$ // pop b.

State Diagram



Operation

Let us check for string abcba

Date ___/___/_____ Page No.: _____

Δ is $K \times (\Sigma \cup \{e\}) \times \Gamma \to K \times \Gamma$

Languages of PDA

1) Language accepted by empty stack
2) Language accepted by final state

Configuration (or Instantaneous description (ID) of PDA

Configuration of PDA is defined as member of $K \times \Sigma^* \times \Gamma^*$ i.e triple (q, w, r) where q is the state.

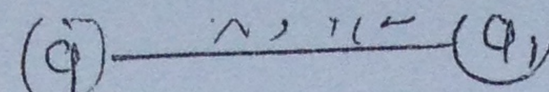
w is the remaining string to be input.

r is the stack content.

Suppose for the PDA M, $\delta(q, a, x)$ contains (p, α) then one of the configuration can be

$(q, aw, xB) \vdash (p, w, \alpha B)$

"All the work you do, is done for your own salvation, is done for your own benefit." —Swami Vivekananda



Date ___/___/___ Page No.: ___

PDA Design examples

1. Design a PDA M that accepts a language $L = \{wcw^R : w \in \{a,b\}^*\}$.

Soln let $M = (K, \Sigma, \tau, \Delta, s, F)$ be required PDA where

$K = \{s, f\}$

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

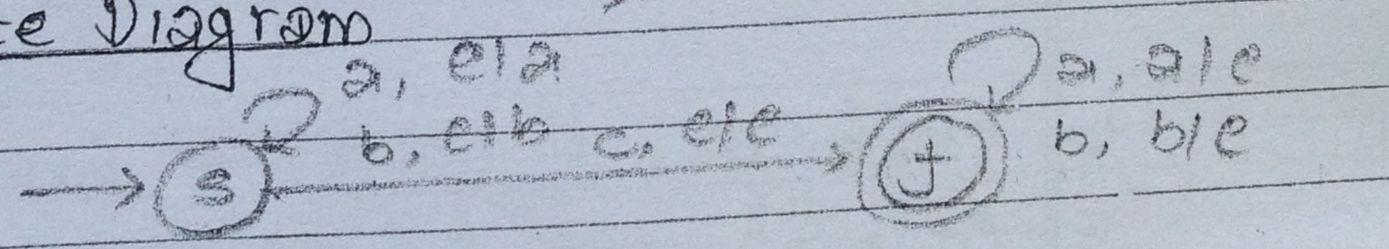
$\tau = \{a, b\}$

$F = \{f\}$

and the Δ is given by

1. $(s, a, e) \rightarrow (s, a)$ // push a
2. $(s, b, e) \rightarrow (s, b)$ // push b
3. $(s, c, e) \rightarrow (f, e)$ // just switch the state.
4. $(f, a, a) \rightarrow (f, e)$ // pop a
5. $(f, b, b) \rightarrow (f, e)$ // pop b.

State Diagram



Operation

Let us check for string abcba

Date __/__/____ Page No.: _____

State	Unread input	Stack	transition used
s	abcba	e	-
s	bcba	a	1
s	cba	ba	2
f	ba	ba	3
f	a	a	5
f	e	e	4

PDA Design a PDA m that accept a language given by

$L = \{a^n b^n : n > 0\}$

Soln.

Δ

1. (s, a, e) → (s, a) // push a
2. (s, a, a) → (s, a) // push a
3. (s, b, a) → (q, e) // pop a when first b is encountered and also switch the state from s to q
4. (q, b, a) → (q, e) // pop a
5. (q, e, e) → (f, e) // go to final state.

Chitra

Date 07/07/25 Page No.:

Normal forms

1) chomsky Normal form (CNF)
2) Greibach Normal form (GNF)

1. chomisky Normal form (CNF)

Any CFL L without any e-production is generated by a grammar in which production are of the form.

$$A \longrightarrow BC$$

or

$A \rightarrow a$, where $A, B \in V$ and i.e if.

Simplified CFG has only production of the form

Non terminal → string of exactly two non-terminal

(NT) → (NT). (NT)

Non terminal → one terminal is said to in chomsky normal form

(NT) → T

procedure to find CNF

1. Eliminate the unit production and e-production if any
2. Eliminate the terminals on the right hand side of length two or more
3. Restrict the number of Variables on the right hand side of production to two

"All the work you do, is done for your own salvation, is done for your own benefit." —Swami Vivekananda

Date __/__/____ Page No.: ____

2. Greibach Normal form:

GNF is of the form $A \to a\alpha$ where A is a non-terminal and 'a' is a exactly one terminal and 'α' is the string of zero or more non-terminal is

Eg Convert the following CFG into GNF

$S \to AB / BC$

$A \to aB / bA / a$

$B \to bc / cC / b$

$C \to c$

Soln

Here the production

$S \to AB / BC$ is not in GNF

On applying substitution rule we get

$S \to aBB / bAB / aB / bcc / cCC / bc$

$B \to bC / cC / b$

$A \to aB / bA / a$

$C \to c$ //

Date 07/07/29 Page No.:

Equivalance Of PDA and CFG

Theorem: The class of languages accepted by PDA is exactly the class of CFL.

lemma 1: Each CFL is accepted by same PDA.
lemma 2: If a language is accepted by a PDA, it is a CFL

Proof of lemma 1: (Constructions Of PDA equivalance of CFG)

Let $G = (V, \Sigma, R, S)$ be a CFG, we must construct a PDA M such that $L(M) = L(G)$. The Machine we construct has only two states p and q and remains permanently in state q after its first move. Also M uses Σ: set of terminals and V; non terminals as its stack alphabet

let $M = (K, \Sigma, \Gamma, \Delta, s, F)$

$K = \{p, q\}$

$\Gamma = V \cup \Sigma$

$s = p$

and transition relation Δ is defined.

Date __/__/____ Page No.: ____

1. $\delta(p, e, e) \rightarrow (q, s)$ - (as 's' is starting non terminal in CFG)
2. $\delta(q, e, A) \rightarrow (q, x)$ - for each rule $A \rightarrow x$ in CFG.
3. $(q, a, a) \rightarrow (q, e)$ - for each $a \in \Sigma$.

Example:

Design a PDA for the following CFG

$G = (V, \Sigma, R, S)$ with

$V = \{S\}$

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

R is defined as

$S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow c$

$S \Rightarrow aSa$

$\rightarrow aaSaa$

$\rightarrow aabSbaa$

$\rightarrow aabcbaa$

.

Date 071/08/07 Page No.

Chapter-3 Turing Machine

- A Turing machine is a very simple machine but logically speaking has all power of any digital computer.
- TM is an abstract model created by Alan Turing.
- A function is computable if it can be computed by a Turing machine
- A Turing machine consist of a finite control, a tape and a head that can be used for reading or writing on that tape.

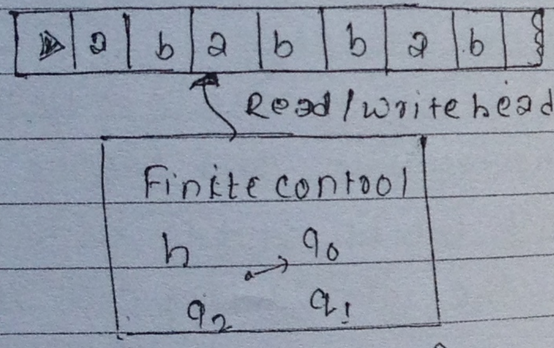


fig: model of Turing machine

Roles of TM
- TM for computing function
- TM for language acceptors
- TM as Enumerators
  → language generators

Date __/__/__ Page No.:

= blank

Formal Definition Of TM

A Turing machine (TM) 'M' can be formally defined as

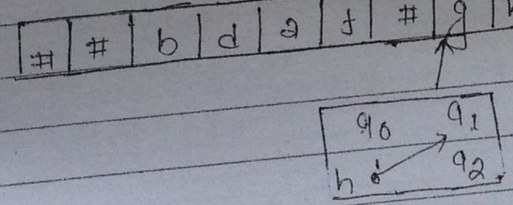
$M = (K, \Sigma, T, \delta, s, H)$ where

K set of states
Σ input alphabet
T tape symbols
s start state
H Halting state
δ is a transition function

configuration of TM (Instantaneous Description)

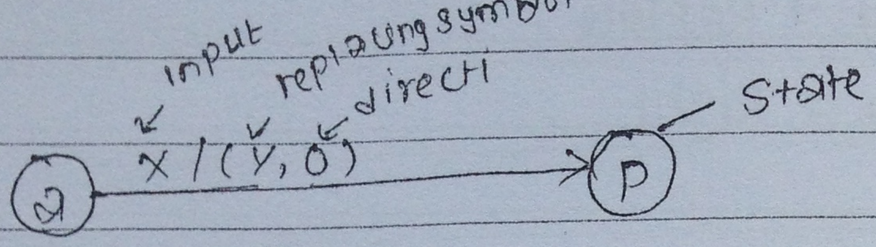
The configuration of TM is the information in respect of
- contents of all cells of the tape
- The cell currently being scanned by the machine.

Eg:



Date __/__/____ Page No.: _____

T.M Transition Diagram



X / (Y, D), where

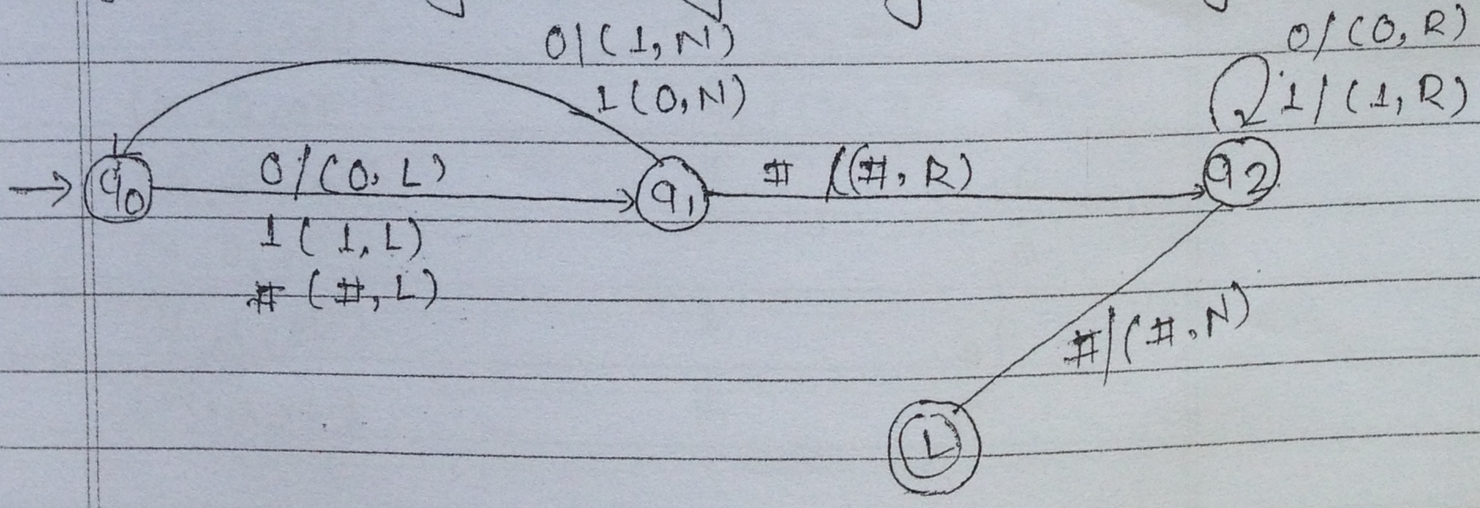
X and Y are tape symbols and D is a direction either R, L or N (neutral) ($\longrightarrow$ $\longleftarrow$)

Right more ↑ Left more

$$\delta(q, x) = (p, y, D)$$

Design Examples

i) let $\Sigma_0 = \Sigma = \{0, 1\}$ and $f: \Sigma_0 \rightarrow \Sigma_0^A$ be defined as: for any $w \in f(w) = w'$, where w' is the result of replacing each occurence of '0' in w by '1' and vice versa: Design a TM which computer f and f and check your design using the string 110



#)

"No religion has mandated killing others as a requirement for its sustenance or promotion." —Dr.A.P.J.Abdul Kalam

q_0 → left move.
q_1 → Replace
q_2 → Right

Soln Let Require TM $M = (K, \Sigma, \delta, s, h)$

Where

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

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

$\Gamma = \{0, 1, \#\}$

$s = q_0$

$h \in K$

and Transition function δ is

q	6	$\delta(q,6)$
q_0	0	$(q_1, 0, 1)$
q_0	1	$(q_1, 1, L)$
q_0	#	$(q_1, \#, L)$
q_1	0	$(q_0, 1, N)$
q_1	1	$(q_0, 0, N)$
q_1	#	$(q_2, \#, R)$
q_2	0	$(q_2, 0, R)$
q_2	1	$(q_2, 1, R)$
q_2	#	$(h, \#, N)$

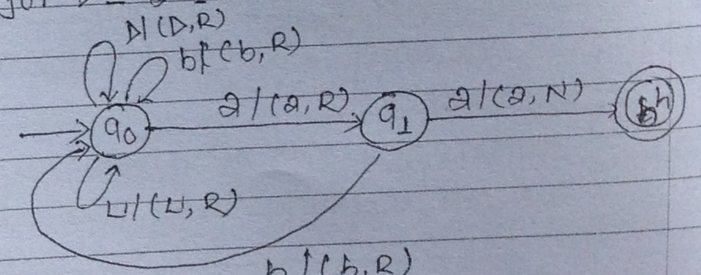
Chitra

"All the work you do, is done for your own salvation, is done for your own benefit." --Swami Vivekananda

Now testing this regular design for #110#

$(q_0, \#110\#) \vdash_M (q_1, \#0110) \vdash_M q_0\#11$
$\vdash_M (q_1, \#111) \vdash_M (q_0, \#101) \vdash_M (q_1, \#10$
$\vdash_M (q_0, \#001) \vdash_M (q_1, \#001) \vdash_M (q_2, \#0$
$\vdash_M (q_2, \#01) \vdash_M (q_2, \#001) \vdash_M (q_2, \#0$

Design a TM that scans to right until to right until it finds two consecutive a's and then halts. The alphabet of the TM should be $\Sigma = \{a, b, \triangleright, \sqcup\}$ Test your Design for $\triangleright\sqcup aa\sqcup abaa$



Chitra

"No religion has mandated killing others as a requirement for its sustenance or promotion." --Dr.

Let $M = (K, \Sigma, \Gamma, \delta, S, H)$ be required TM where

$K = \{q_0, q_1, h\}$

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

$\Gamma = \{a, b, \triangleright, \sqcup\}$

$S = q_0$

$H = \{h\}$

and transition function δ is given by

q	σ	$\delta(q, \sigma)$
q_0	a	(q_1, a, R)
q_0	b	(q_0, b, R)
q_0	$\sqcup$	$(q_0, \sqcup, R)$
q_0	$\triangleright$	$(q_0, \triangleright, R)$
q_1	a	(h, a, N)
q_1	b	(q_0, b, R)
q_1	$\sqcup$	$(q_0, \sqcup, R)$
q_1	$\triangleright$	$(q_0, \triangleright, R)$

for Input $\triangleright\sqcup aba$

$(q_0, \triangleright\sqcup aba) \vdash (q_0, \triangleright\sqcup aba) \vdash$

$\vdash (q_0, \triangleright\sqcup aba) \vdash$

$\vdash (q_0, \triangleright\sqcup a\sqcup ba) \vdash$

$\vdash (q_1, \triangleright\sqcup a\sqcup aba)$

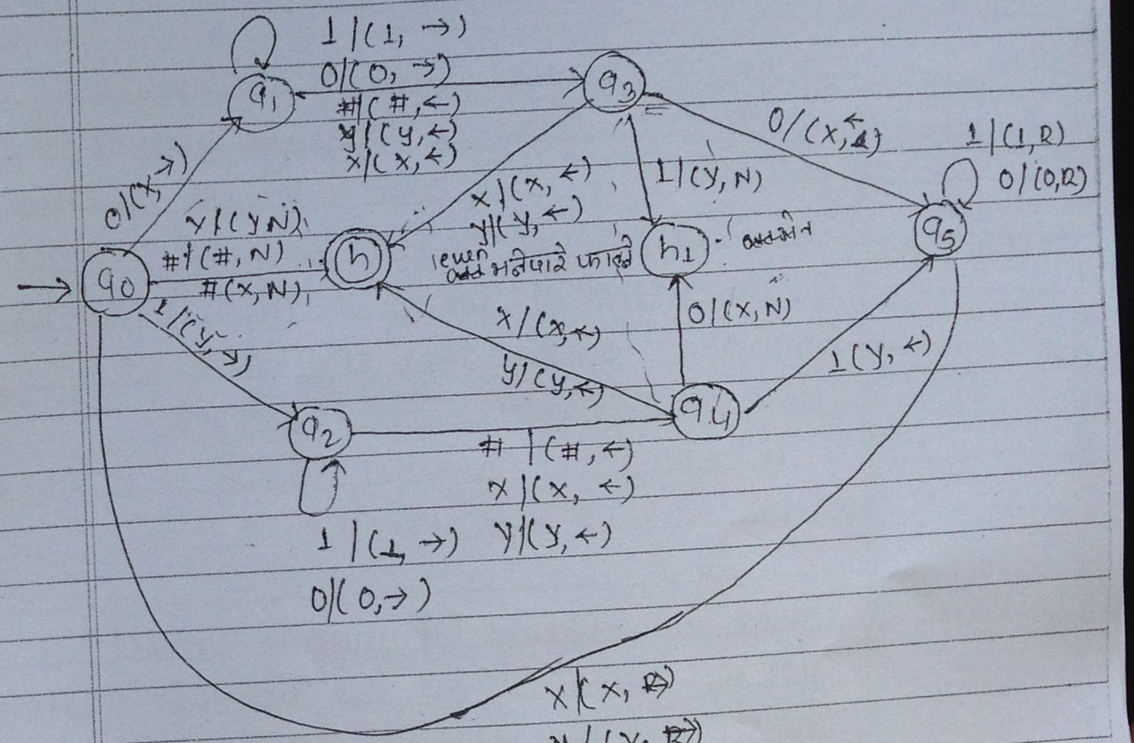
$\vdash (h, \triangleright\sqcup a\sqcup aba)$ //

$(q_1, \triangleright\sqcup\sqcup aba)$

$(q_1, \triangleright\sqcup a\sqcup aba)$

"All the work you do, is done for your own salvation, is done for your own benefit." –Swami Vivekananda

Design a TM which accepts the set of all palindromes over $(0, 1)$



"No religion has mandated killing others as a requirement for its sustenance or promotion." –Dr.A.P.J.Abdul Kalam

Date 07/08/14

TM for Computing function:

A function $f(x) = y$ is said to be computable by a TM $(K, \Sigma, T, \delta, S, H)$ if $(S, \# x \#) \vdash^* (h, \# y \#)$, where x may be in some alphabet Σ_1^* and y may be some alphabet Σ_2^* and $\Sigma_1, \Sigma_2 \subseteq \Sigma$

Turing machine as language Acceptors

A TM works as language acceptor for a language L if it is able to tell us whether a string w belongs to the language L or not. If TM accepts a string w then it will leave. Symbol 'y' on the tape and halts i.e

$(s, \# w \#) \vdash^* (h, \# y \#)$

It will leave symbol 'n' if w is not accepted

$(s, \# w \#) \vdash^* (h, \# n \#)$

TM as Enumerators (generating Devices)

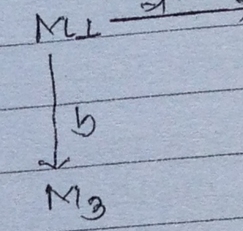
We can define L(TM), the language generated by TM, to be set of w in Σ^* such that w is eventually printed between a pair of #'s on the output tape.

Chitra

"All the work you do, is done for your own salvation, is done for your own benefit." –Swami Vivekananda

Combining Turing Machine

Turing machine can be combined in a way suggestive of the structure of finite automata. Following diagram, shown the simple combining turing machine.



It operates start in the initial state of M_1 as M_1 would operate M_1 would halts then currently scanned symbol is an 'a' initiate and operate as M2 would operate, otherwise if the currently scanned symbol is 'b' then M_3 and operate as M_3.

let $M_1 = (K_1, \Sigma, T_1, \delta_1, S_1, H_1)$

$M_2 = (K_2, \Sigma, T_2, \delta_2, S_2, H_2)$

$M_3 = (K_3, \Sigma, T_3, \delta_3, S_3, H_3)$

Combining M_1, M_2, M_3 (assuming set of M_1, M_2 and M_3 are disjoint)

$M = (K, \Sigma, T, \delta, S, H)$ where

$K = K_1 \cup K_2 \cup K_3$

$S = S_1$

$H = H_2 \cup H_3$

for each $\sigma \in \Sigma$ and $q \in (K - H)$

"No religion has mandated killing others as a requirement for its sustenance or promotion." –Dalai Lama

$\delta(q,\sigma)$ is defined as

a) If $q \in (K_1 - H_1)$, then $\delta(q,\sigma) = \delta_1(q,\sigma)$

b) If $q \in (K_2 - H_2)$, then $\delta(q,\sigma) = \delta_2(q,\sigma)$

c) If $q \in (K_3 - H_3)$, then $\delta(q,\sigma) = \delta_3(q,\sigma)$

d) Finally if $q \in H$, then only case remain $\delta(q,\sigma) = S_2$ if $\sigma = a$, $\delta(q,\sigma) = S_3$ if $\sigma = b$ and $\delta(q,\sigma) \in H$ otherwise.

Symbolic representation of TM.

1) $q \xrightarrow{\sigma(\sigma, L)}$ - Moves left by one cell

2) $q \xrightarrow{\sigma(\sigma, R)}$ - Moves right by one cell

3) $q \circlearrowleft \overline{\#}/(\#, L)$, $q \xrightarrow{\#/(\#, N)}$ Left moving machine)
- Moves towards left till first blank symbol is encountered.

4) $q \circlearrowleft \overline{\#}/(\#, R)$, $q \xrightarrow{\#/(\#, N)}$ (Right moving machine)
- Moves towards right till first blank symbol is encountered.

5) $q \circlearrowleft \#/(\#, L)$, $q \xrightarrow{\overline{\#}/(\#, N)}$ - Moves towards left till first non blank symbol is encountered.

6) $q \circlearrowleft \#/(\#, L)$, $q \xrightarrow{\overline{\#}/(\#, N)}$ - Moves towards right till first non blank symbol is encountered.

Example: Design a TM which works as eraser

$q_0 \xrightarrow{\overline{\#}(\#, L)} q_1 \circlearrowleft \overline{\#}/(\#, L)$, $q_1 \xrightarrow{\#(\#, R)} h$

"All the work you do, is done for your own salvation, is done for your own benefit." —Swami Vivekananda

"No religion has mandated killing others as a requirement for its sustenance or promotion." —Dr. A.P.J. Abdul Kalam

Date 07/08/21

Extension of Turing Machine / Variation of TM

1) Multiple Tape TM
2) Two-way infinite tape TM
3) Multiple head TM
4) K-dimentional TM
5) Non-dimentional deterministic TM
6) Random access TM.

Recursively enumerable and recursive language.

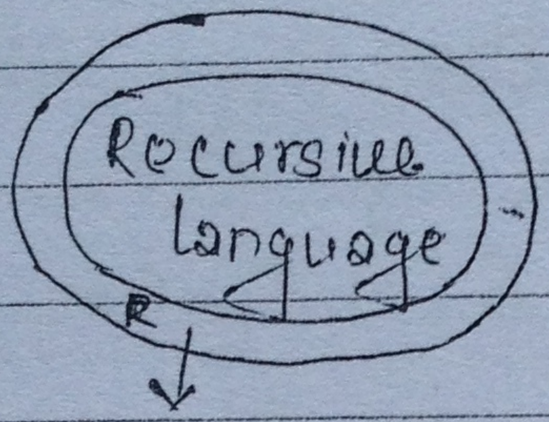
When a TM executes an input, there are four possible outcomes for executes, ~~an input~~ there are ~~four possible~~ Then TM.

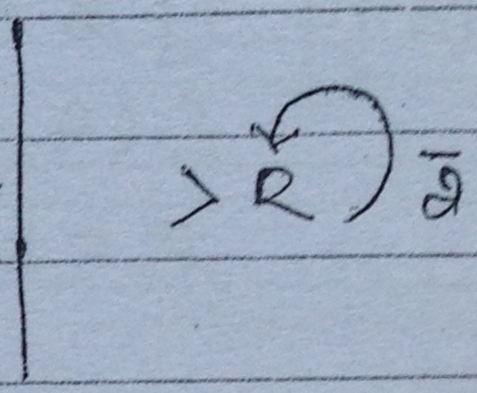
i) Halts and accepts the input
ii) Halts and reject the input
iii) Never halts (fall into loop) or
iv) crash.

The set of recursively enumerable language are precisely those languages that can be listed (ehumelated) by a TM. Basically TM can generate all the strings in a language, another TM can accept all the strings that are wet in the given language (strings that are not in the language may be rejected or may cause the TM to go into an infinite loop).

Date __/__/____ Page No.: ____

A language is recursive if there exits a TM that accep
every string of the language and rejects every strings that
on language. so we are sure about the rejection of the stri
which are not on the given recursive language.



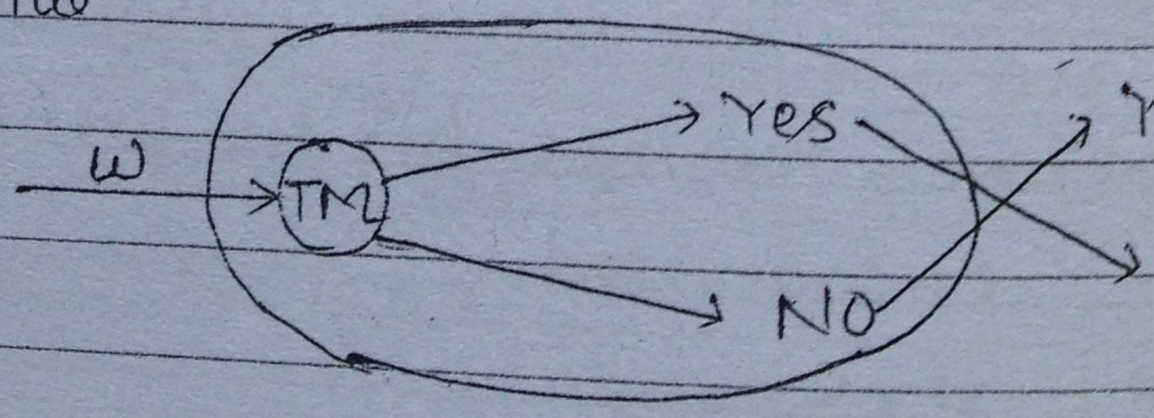


proof: A language is recursively enumerable if there exist
a TM that accepts every strings of the language and a
language is recursive if there is a TM that accepts every
string of the language and doesnot accept string that are
on language

So we can say that the statement is true.

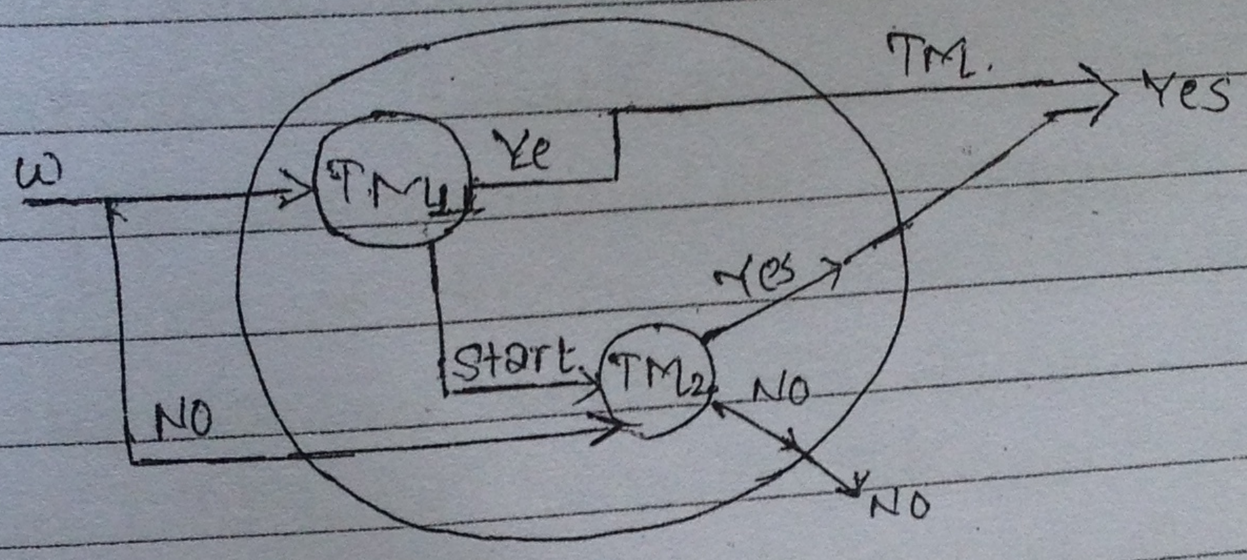
properties of Recursive and Recursively enumerable languages

Theorem 1: The complement of recursive language is recursive.

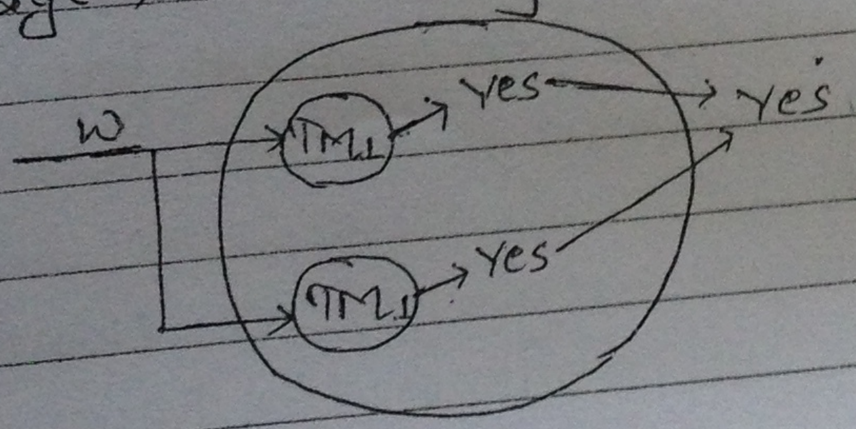


Date ___/___/_____ Page No.: _____

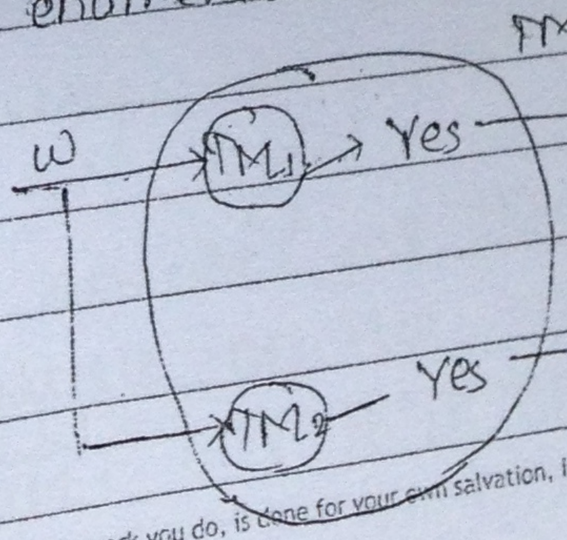
Theorem 2 :- The union of two recursive language is recursive.



Theorem 3 : The union of two recursively enumerable language is recursively enumerable.



Theorem 4 : If a language L and its complement $\bar{L}$ are both recursively enumerable then L (hence $\bar{L}$) is also recursive.

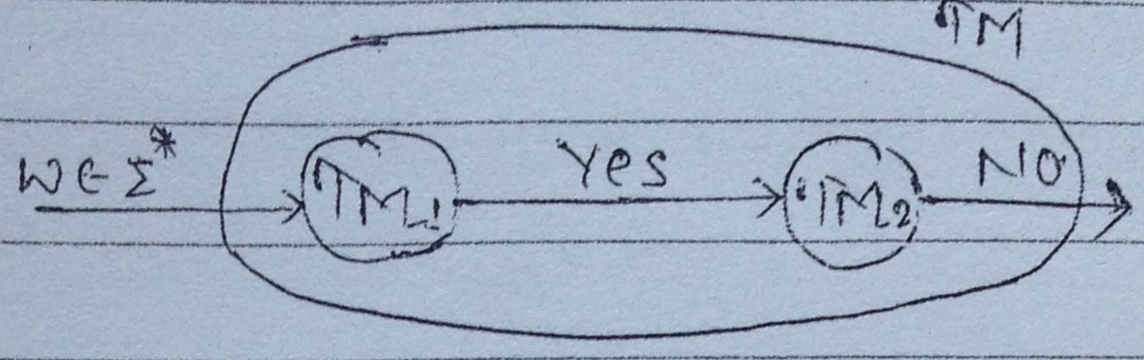


"All the work you do, is done for your own salvation, is done for your own benefit." —Swami Vivekananda

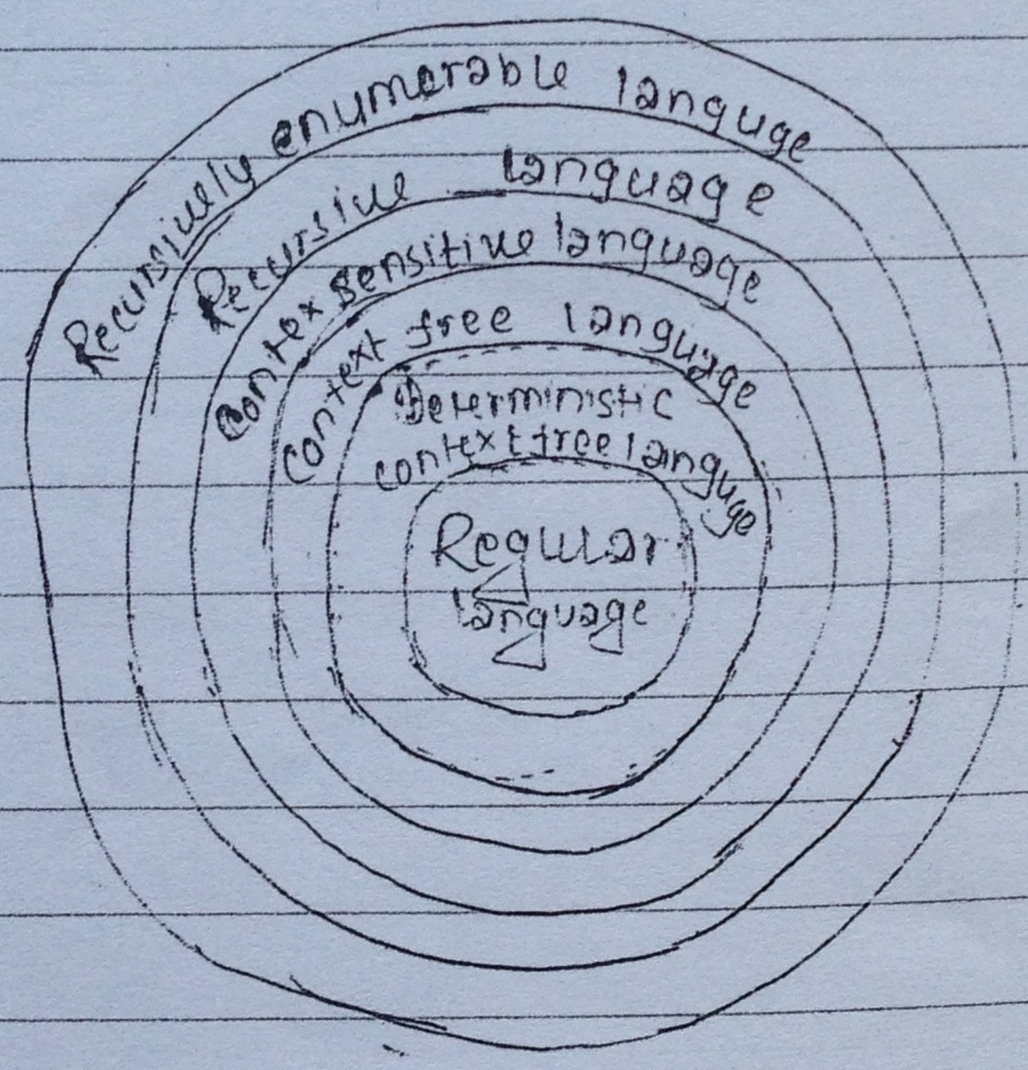
Chitra

Date ___/___/_____ Page No.: _____

Theorem 5:- If L is a recursive language the $\Sigma^* - L$ is recursive.



CHOMSKY HIERARCHY:



"No religion has mandated killing others as a requirement for its sustenance or promotion." —Dr.A.P.J.Abdul Kalam

Date __/__/____ Page No.: _____

Type 0 - unrestricted grammar

Type 1 - context sensitive grammar.

Type 2 - context free "

Type 3 - Regular grammar.

$\partial A \rightarrow \partial y$

$\partial AB \rightarrow Ab$

$A \rightarrow B$

A = any string with non-terminal

B = any string.

Type 1 A = any string with non-terminals

B = any string as long as a longer that.

i.e $|A| \leq |B|$

Type 2 A = One non-terminal

B = any string

Type 3 A = One non-terminal

B = ax or B = a where

a is a terminal and 'x' is a non terminal

Date 07/08/21

Church's Thesis

"No computational procedure will be considerd as an algorithm unless ~~as~~ rep it can represents as a Turing machine". This statment is called church Thesis. church actualy said that any machine that can do certain ~~least~~ List of operation will be able to perform all conceivable algorithm

What function can't be computed by turing machine?

It is belived that there are no function that can be defined by humans whose calculation can be described by any well define mathematical algorithm that people can taught to perform, that can't be computed by turing machine. So turing machine is called ultimate calculating machine.

Unfortunatly church's thesis can't be a theorem in mathematics because idea's such as "can never be ever be define by human" and "algorithm that people can taught to perform" are not part of any known ~~matrix~~ mathematics.

Date __/__/____ Page No.: ____

The Halting problem :-

Halting problem is to determine for an aribtirary given turing machine (TM) and input 'W', wheather TM ~~is~~ will enventually halt on input 'W' Let us assume that we have given a discription of turing machine (TM) and input 'W', we aske wheather TM applied to W or simply (TM, W) halts or does not halts. The domain of this problem is to be taken as the set of all turing machine and all W i.e we are looking for singe Turing machine that given the description arbitrary TM and W, will predict wheather or not the computation of TM applied to W will halt

We can't fine the answer by simulating the action of TM on 'W' because there is no limit on the length of computation. If TM enters an infinel loop, loup, then no mater how long wait we can never be sure that TM is in fact in loup. It may be simple case of very long computation. What we need is an algorithm that can determine the correct answer for TM & W by performing some analysis on the machine discription and the input. but it is clear that know no.

"All the work you do, is done for your own salvation, is done for your own benefit." —Swami Vivekananda

Chitra

Date ___/___/_____ Page No.: _____

Search algorithm exit.

Universal Turing Machine (UTM)

We can consider turing machine in both ways

1) Turing machine is an programbi piece pice of Hardware specilized at surving or particular problem with instruction that are hard coild. at the factory.
2) As a software that is there is a certaine generet turing machine that can be programed (general purpose) show the same way that a g. computer can, to solue any problem that can be solued by turing machine. The program that makes this generic machine behave like a specific machine TM will have to be description of TM. Program written in this language can then be enter preated. Universal machine. Such a machine is called Universal turing machine (UTM)

Universal turing machine 'U' takes two arguments, of description of machine (TM), and description of an input string W

U halts input TM, W if and only if TM halts input W

Chitra

"No religion has mandated killing others as a requirement for its sustenance or promotion." —Dr.A.P.J.Abdul Kalam

Date __/__/____ Page No.: _____

U("TM", W) = TM (W) - functional notation of Universal TM.

Let TM = (K, Σ, δ, τ, S, H) be a turing machine and i and j be smallest integers such that $2^i \geq |K|$ and $2^j \geq |\Sigma| + 2$ Each state in K be represented by q. followed by binary strings of length i each symbol in Σ will be represented as letter a followed by a string of j bits.

Consider a example given below.

State	Symbol	δ
S	₹a	(q, #)
S	#	(h, #)
S	b	(S, R)
q	₹a	(S. a)
q	#	(S, R)
q	b	(q, R)

Since there are three states, three symbols in Σ we have i=2, j=3

$2^i \geq 3 \quad 2^j \geq 3+2$

Chitra

State / symbol	Representation.
S	900
q	901
h	911
#	2000
b	2001
L	2010
R	2011
2	2100

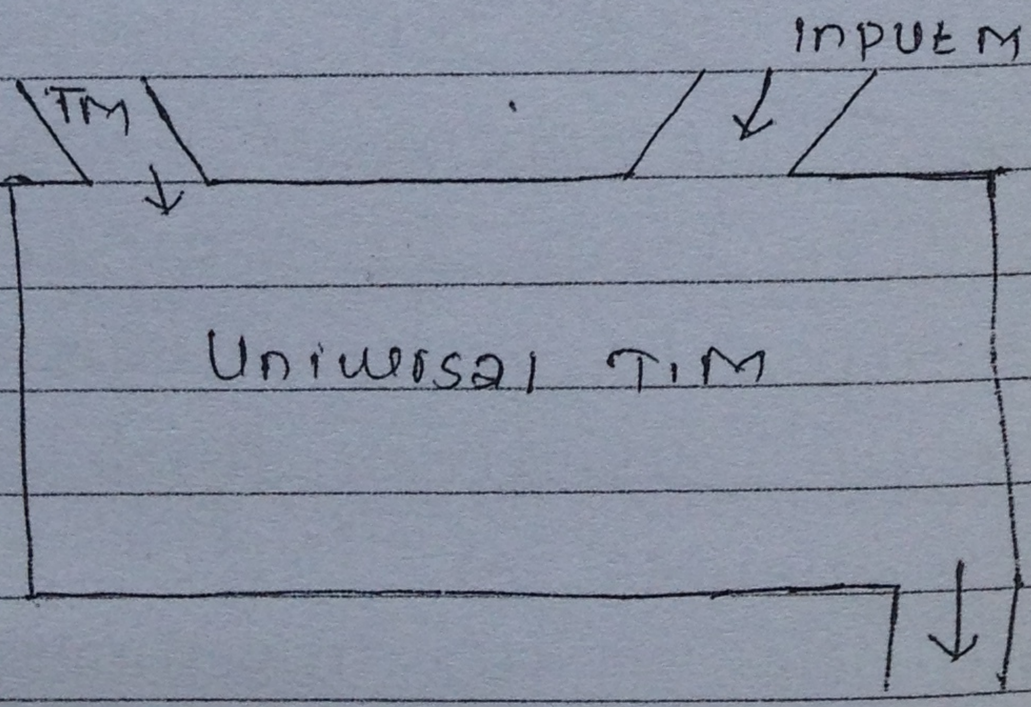
There This representation of string baa#2 is

2001 2100 2100 2000 2100

The representation "TM" of turing machine TM is the following strings

"TM" = (900, 92100, 901, 2000),

(900; 2000, 911, 2000)



Date __/__/____ Page No.: _____

Conversion of Regular grammar into finite automata:

Given a regular grammar G a fine automat accepting L(G) can be obtained as follows

a) the no of ~~starts~~ states in the automata will be equal to the number of(nonterminal +1).
Each state in automata represents each non terminal in the reguar grammer. the additional state will be the final state automat.
The state corrosponding to the start symbol of grammer will be the initial state automata
If L(G) contents $\overset{\text{empty}}{(\epsilon)}$ that is start symbol in grammer derives to empty e then make start stat also final state.

b) The transition for automat are obtained as follows
i) For every production $A \to aB$ make $\delta(A,a) = B$ i.e. make an labelled 'a' from A to B
ii) For every production $A \to a$, make $\delta(A,a)$ = finalstate
iii) For every production $A \to e$, make $\delta(A,e) = A$ and A will be final state.