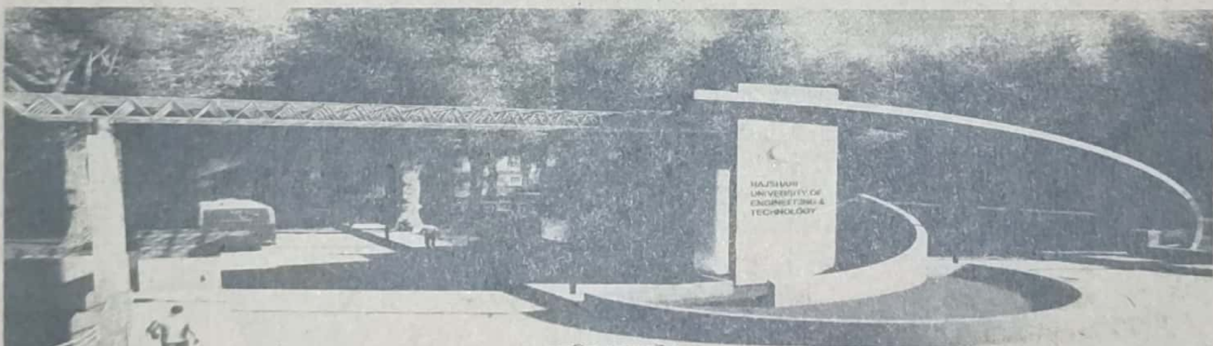
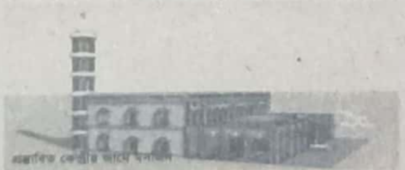
"Heaven's Light is our Guide"

RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

RUET



প্রস্তাবিত প্রধান ফটক



Name: Md. Ismail 1

Subject: Automata (CSE-2205)

Year: ○ 1st ✓ 2nd ○ 3rd ○ 4th

Roll No.: 1503094

Semester: ○ 1st ○ 2nd ○ 3rd ✓ 4th ○ 5th ○ 6th ○ 7th ○ 8th

Department: Computer Science & Engineering

- Design a DFA of bit string ~~othe~~ which has 01 pattern at any position

- Design a DFA of ~~othe~~ bit string which can read even number '0' and even number '1'

- Determine whether the given transition diagram is DFA or NFA

- Design a DFA to read a ~~deg~~ decimal number.

Finite automata:

Automata theory is the study of abstract computing $\boxed{\text{devices}}$ or $\boxed{\text{"machines"}}$

A finite automata (FA) is a simple idealized machine used to recognize patterns within input taken from some character set C.

The job of finite automata is to accept or reject an input whether the pattern defined by the FA occurs in the input.

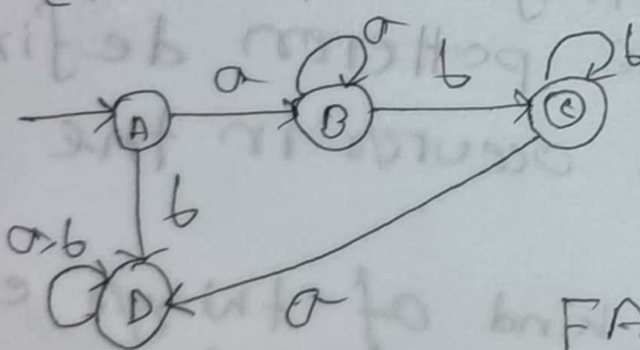
→ It is mainly what kind of things can we really ~~can~~ compute mechanically, how fast and how much space does it take to do so.

$\boxed{FA}$ FA → Finite Automata

- FA with output
  - moore
  - mealy
- FA without output
  - DFA
  - NFA
  - ε-NFA

⊞ $\{a^n b^n / n \geq 1\}$

$L = \{ab, aabb, aaabbb ---\}$

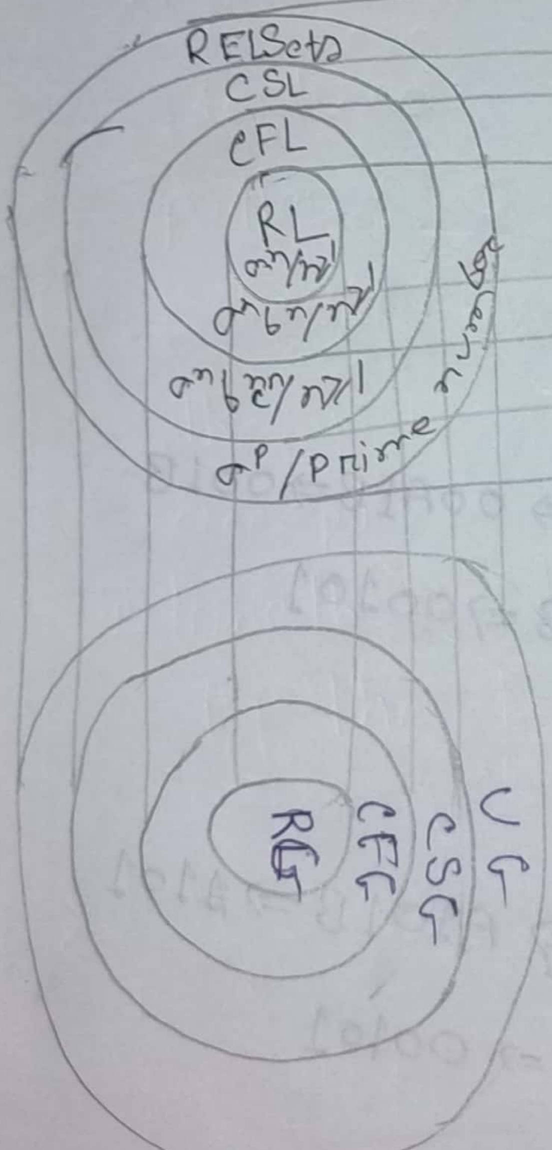


FA + memory = PDA

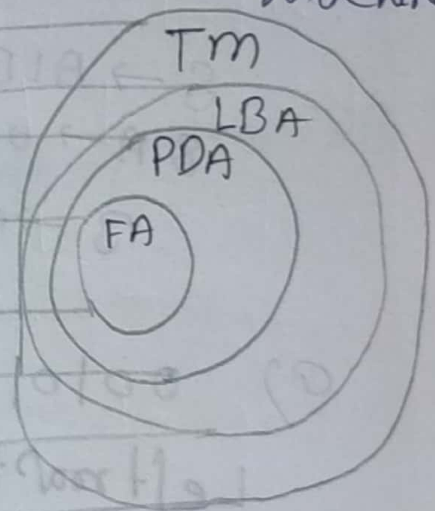
TOC → Theory of computation

Family of language

Family of languages



Family of machine



FA → Finite Automata
PDA → Push down Automata
LBA → Linear Bounded Automata
TM → Turing Machine

Family of grammar

RG → Regular Grammar
CFG → context Free Grammar
CSG → Context Sensitive Grammar
UG → Unrestricted grammar

RL → Regular Language
CFL → context Free Language
CSL → Context Sensitive Language
REL → Recursively Enumerable Language

~~Symbol~~

Chapter-5

Exercise 5.1.2

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

$S \rightarrow A1B$

$A \rightarrow 0A \mid \epsilon$

$B \rightarrow 0B \mid 1B \mid \epsilon$

a) 00101

Leftmost,

$S \Rightarrow A1B \Rightarrow 0A1B \Rightarrow 00A1B \Rightarrow 001B$

$\Rightarrow 0010B \Rightarrow 00101B \Rightarrow 00101$

Rightmost:

$S \Rightarrow A1B \Rightarrow A10B \Rightarrow A101B \Rightarrow A101$

$\Rightarrow 0A101 \Rightarrow 00A101 \Rightarrow 00101$

b 1001

Leftmost :

$S \Rightarrow A1B \Rightarrow 1B \Rightarrow 10B \Rightarrow 100B \Rightarrow 1001B$

$\Rightarrow 1001$

Rightmost :

$S \Rightarrow A1B \Rightarrow A10B \Rightarrow A100B \Rightarrow A1001B$

$\Rightarrow A1001 \Rightarrow 1001$

c 00011

Leftmost

$S \Rightarrow A1B \Rightarrow 0A1B \Rightarrow 00A1B \Rightarrow 000A1B$

$\Rightarrow 0001B \Rightarrow 00011B \Rightarrow 00011$

Rightmost :

$S \Rightarrow A1B \Rightarrow A11B \Rightarrow 1A11 \Rightarrow 0A11$

$\Rightarrow 00A11 \Rightarrow 000A11 \Rightarrow 00011$

Sol 5.1.1 :

a. $L = \{0^n 1^n \mid n \geq 1\}$

$\therefore$ 01 will be accepted

E

0011 will be accepted

CFG =

$A \to C01D$	$S \to 0S1$
	$S \to 01$
$C \to 0C \mid \varepsilon$	
$D \to 1D \mid \varepsilon$	

$\therefore G = (\{A, C, D\}, \{0, 1\}, A, P)$

b. $L = \{a^i b^j c^k \mid i \neq j \text{ or } j \neq k\}$

$S \to AB \mid CD$

$A \to aA \mid \varepsilon$

$B \to bBc \mid E \mid cD$

$C \to aCb \mid E \mid aA$

$D \to cD \mid \varepsilon$

$E \to bE \mid b$

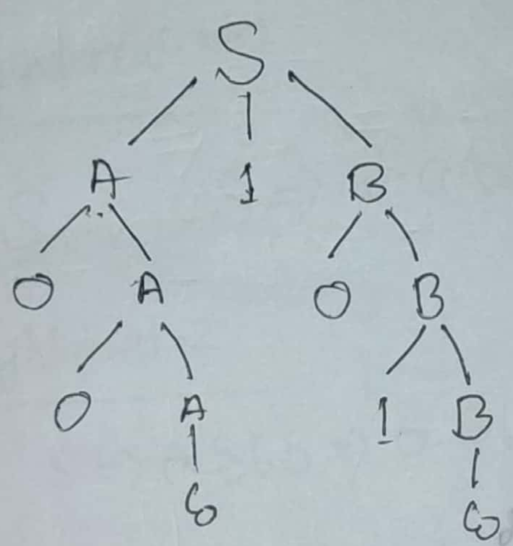
CamScanner

Exercise
5.2.1

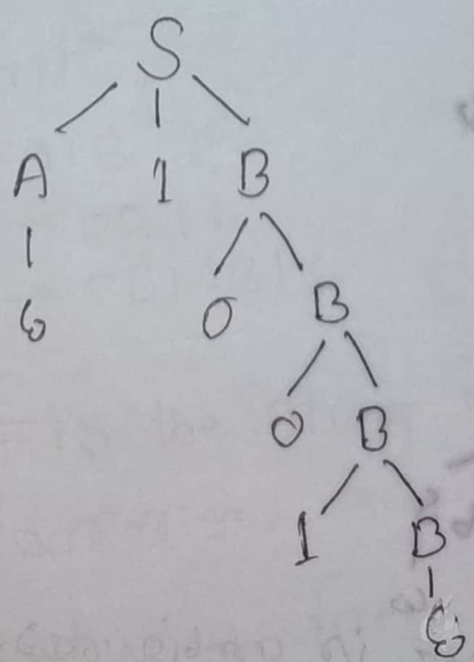
$0^*1(0+1)^*$

$S \rightarrow A1B$
$A \rightarrow 0A \mid \epsilon$
$B \rightarrow 0B \mid 1B \mid \epsilon$

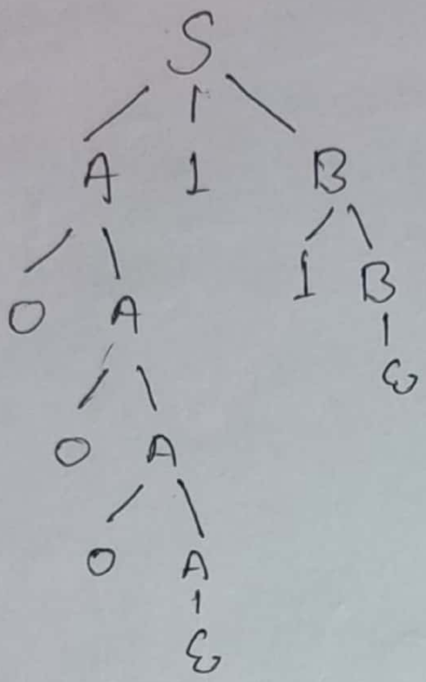
a) 00101



b) 1001



c) 00011

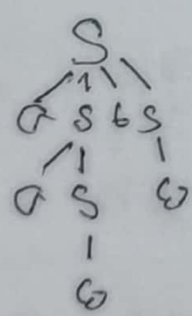


Exercise 5.4.1

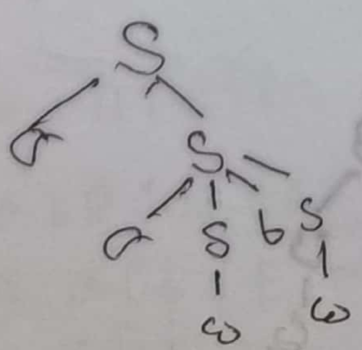
$S \rightarrow aS \mid aSbS \mid \epsilon$

Here String $w = aab$

a)



again



∴ This grammar is ambiguous.

b

Leftmost-1

$S \to aS \Rightarrow aaSbS \Rightarrow aabS \Rightarrow aab$

Leftmost-2

$S \Rightarrow aSbS \Rightarrow aaSbS \Rightarrow aabS \Rightarrow aab$

c

Rightmost-1

$S \Rightarrow aS \Rightarrow aaSbS \Rightarrow aaSb \Rightarrow aab$

Rightmost-2

$S \Rightarrow aSbS \Rightarrow aSb \Rightarrow aaSb \Rightarrow aab$

5.4.5

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

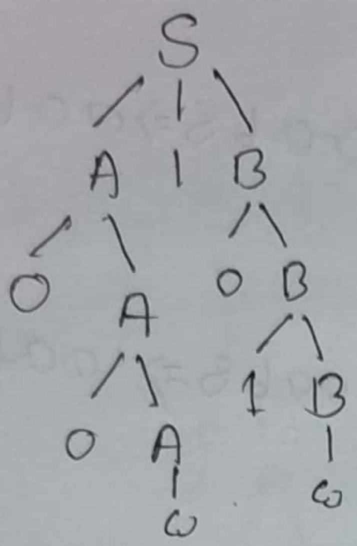
$S \to A1B$

$A \to 0A \mid \varepsilon$

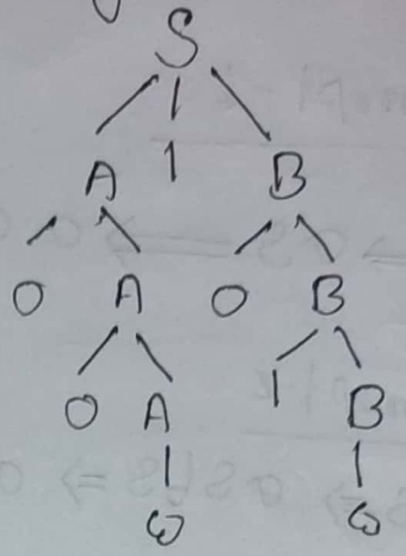
$B \to 0B \mid 1B \mid \varepsilon$

00101 is the string accepted by this grammar. Now find Left and right parse tree of this string
rightmost

Left most tree



Right most tree



As ~~tre~~ two parse tree is same
So this grammar is unambiguous.

5.4.3

$S \rightarrow aS \mid aTbS \mid \varepsilon$

$T \rightarrow aTbT \mid \varepsilon$

Chapter-7

Exercise 7.1.1

$S \to AB \mid CA$
$A \to a$
$B \to BC \mid AB$
$C \to aB \mid b$

Soln:

$T = \{a, b\}$

$W_1 = \{A, C\}$

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

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

$\therefore$ Productions without non generating is.

$S \to CA$
$A \to a$
$C \to b$

Now, $y_1 = \{S\}$

$y_2 = \{C, A, S\}$

$y_3 = \{a, b, C, A, S\}$

∴ production without useless symbols

$S \to CA$

$A \to a$

$C \to b$

Exercise - 7.1.2

$S \to ASB \mid \epsilon$

$A \to aAS \mid a$

$B \to SbS \mid A \mid bb$

Soln :

a Here $S \to \epsilon$

∴ S is nullable

∴ $S \to ASB \mid AB$

$A \to aAS \mid aA \mid a$

$B \to SbS \mid Sb \mid bS \mid b \mid A \mid bb$

b ~~Only Here B → A is~~

Q b Here, $B \to A$ is a unit production

$\therefore B \to SbS \mid Sb \mid bS \mid b \mid bb \mid aAS \mid aA \mid a$

So All productions are,

$S \to ASB \mid AB$
$A \to aAS \mid aA \mid a$
$B \to SbS \mid Sb \mid bS \mid b \mid aAS \mid aA \mid a \mid bb$

c Here A and B each derive terminal strings and therefore so does S.
Thus there is no useless symbols.

d $S \to \cancel{AC}\ ASB \mid AB$
$A \to aAS \mid aA \mid a$
$B \to SbS \mid Sb \mid bS \mid b \mid aAS \mid aA \mid a \mid bb$

Now reduce ~~two~~ three length ~~vari~~ string to two length variable

∴ $S \rightarrow AE \mid AB$

$A \rightarrow aD \mid aA \mid a$

$B \rightarrow Sb \mid Sb \mid bS \mid b \mid$

$S \rightarrow CB \mid AB$

$A \rightarrow aC \mid aA \mid a$

$B \rightarrow DS \mid Sb \mid bS \mid b \mid aC \mid aA \mid a \mid bb$

$C \rightarrow AS$

$D \rightarrow Sb$

again,

$S \rightarrow CB \mid AB$

$A \rightarrow EC \mid EA \mid a$

$B \rightarrow DS \mid SF \mid FS \mid b \mid EC \mid EA \mid a \mid FF$

$C \rightarrow AS$

$D \rightarrow SF$

$E \rightarrow a$

$F \rightarrow b$

Exercise 7.1.3

$S \to 0A0 \mid 1B1 \mid BB$

$A \to C$

$B \to S \mid A$

$C \to S \mid \varepsilon$

a

Here C is nullable
again A derive C so A is also nullable
again B derive A so B is also nullable

Now, ~~For~~ Remove $C \to \varepsilon$

$S \to 0A0 \mid 1B1 \mid BB$

$A \to C$

$B \to S \mid A$

$C \to S$

Now remove $B \to \varepsilon$

$S \to 0A0 \mid 1B1 \mid 11 \mid B \mid BB$

$A \to C$

$B \to S \mid A$

$C \to S$

Now remove, $A \to \epsilon$

$A \to \epsilon$

$C \to \epsilon$

$B \to \epsilon$

$S \to 0A0 \mid 00 \mid 1B1 \mid 11 \mid B \mid BB$

$A \to C$

$B \to S \mid A$

$C \to S$

AnD

Exerc

Chapter-1

The methods and madness

■ **Automata Theory**: Automata theory is the study of abstract computing devices or "machines".

■ **Finite automata**: A finite automata is a simple idealized machine used to recognize patterns (চেনা) within input taken from some character set / alphabet set C.

The job of finite automata is to accept or reject an input depending on whether defined by the FA occurs in the input.

■ **Necessity of finite automata**:

Finite automata are a useful model for important kinds of hardware and software.

Some important kind of use of finite automata is given below:

1. Software for designing and checking the behavior of digital circuit.

2. The "lexical analyzer" of a typical compiler that is the compiler component that breaks the input text into logical units such as identifiers, keywords and punctuation.

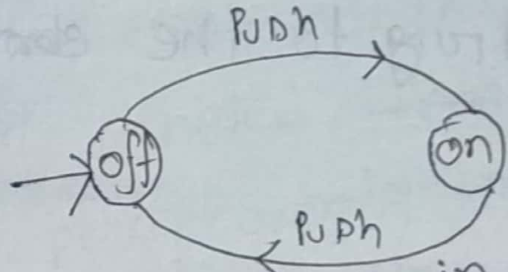
3. Software for scanning large bodies of text.

Such as collections of web pages to find occurrences of words, phrases or other patters.

4. Software for verifying systems of all types that have finite number of distinct states. Such as communications protocols or po protocols for secure exchange of information

⊞ The simplest finite automaton is an on/off switch. The device remembers whether it is in the "On" state or "off" state and it allows the user to press a button whose effect is different, depending on the state of switch.

If the switch is in the off state then pressing the button changes it to on state



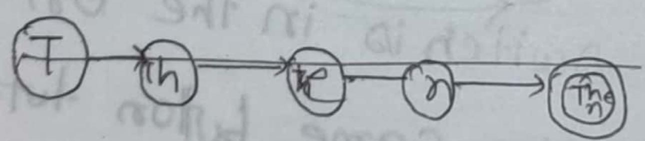
and if the switch is in the on state, then pressing the same button turns it to the off state.

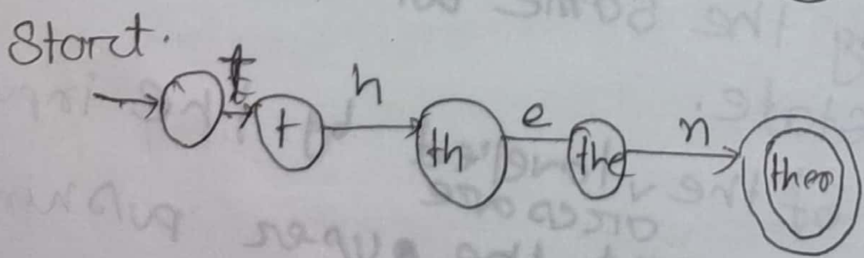
Here both of the arcs are labeled by the input push, which represent the user pushing a the button.

□ Finite automaton for lexical analyzer;

The job of finite automaton is to recognize the keyword then. It thus need five states, each of which represents a different position in the word then that has been reached so far.

These positions correspond to the prefixes of the word, ranging from the empty string to the complete word.





Structural representations:

There are two notation, that play on important role in the study of automata and their applications

1. Grammer; Grammers are useful model when designing software that processes data with a recursive structure. The best example is a "parser" the component of compiler that deals with the recursively nested ~~feathers~~ features of typical programming language, such as expressions- arithmetic, conditional and so on.

2. Regular expression: The regular expression also denote the structure of data especially strings. The style of these expressions differs significantly from that of grammer. The ~~uxi ow~~. UNIX style regular Expression "[A-Z] [a-z] * [ ] [A-Z] [A-Z]"

represent capitalized word followed by a space and two capital letters.

Automata and complexity:

Automata are essential for the study of the limits of computation.

There are two important issues:

1. What can a computer do at all?
 This study is called "decidability". and the problems that ~~a counter~~ can be solved by computer are called 'decidable'

2. What can a computer do efficiently?
 This study is called 'intractability'? and the problems that can be solved by a computer using no more time than some slowly growing function of the size of the input are called "tractable"

Alphatet: An alphabet is a finite, nonempty set of symbols.

, symbol Σ ~~β~~ is used for or alphabet.

1. $\Sigma = \{0, 1\}$, the binary alphabet
2. $\Sigma = \{a, b, \ldots, z\}$ the set of all lower-case letters.

String: A string is a finite sequence of symbols chosen from some alphabet.

~~the string~~ 0100 is a string from the binary ~~ab~~ alphabet $\Sigma = \{0, 1\}$

Empty String: Empty string is the string with zero occurrences of symbols. This string denoted by ϵ.

Length of String: It is the ~~no~~ number of positions for symbols in the string.

It is common to say that the length of string is "the ~~m~~ number of symbols" in the string.

Power of an alphabet: If Σ is an alphabet, we can express the set of all strings of a certain length from that alphabet by using an exponential notation.

Example:

if $\Sigma = \{0,1\}$ the $\Sigma^1 = \{0,1\}$

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

$\Sigma^3 \{010, 011, 100, 101, 000, 001, 111, 110\}$

The set of all string over an alphabet Σ is conventionally denoted Σ^*. $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cdots$

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

$\Sigma^* = \Sigma^+ \cup \{\epsilon\}$

Concatenation of String: Let x and y be string. Then xy denoted the concatenation of x and y. That is the string formed by making a copy of x and following it by a copy of y.

Language: A set of ~~at~~ strings all of which are chosen from some Σ^* where Σ is a particular alphabet is called language. If Σ is an alphabet and $L \subseteq \Sigma^*$ then L is a language over Σ.

Problem: In automata theory, a problem is the question of deciding whether a given string is a member of some particular language.

Chapter- 4

Pumping lemma: Pumping lemma is used to prove that a language in not Regular.

not to prove that a language is Regular

If A is a Regular language than A has a pumping length P such that any String "S" where $|S| \geq P$ may be ~~devi~~ divided into 3 parts xyz such that the following condition must be true:

i) $xy^iz \in A$ for every $i \geq 0$

ii) $|y| > 0$

iii) $|xy| \leq P$

To prove that a language is not regular using Pumping Lemma follow the bellow steps:
(we prove using contradiction)

→ Assume that A is regular

→ It has to have a pumping length (say p)

→ All strings longer than p can be pumped $|S| \geq p$

→ Now find a string 'S' in A such that $|S| \geq p$

→ Divide S into xyz

→ Show that $xy^iz \notin A$ for some i

→ Then consider all ways that S can be divided into xyz

→ Show that none of these can satisfy all the 3 pumping conditions at the same time

→ S cannot be pumped == (Contradiction)

Exercise 4 for 4.2

4.2.1

a) Given

$h(0) = a$

$h(1) = ab$

$h(2) = ba$

$\therefore h(0120) = aabbaa$

b) $h(21120) = baababbaa$

c) $L(01^*2)$

$\therefore h(L) = a(ab)^* ba$ The language of regular expression

d) $L(0+12)$

$\therefore h(L) = a + abba$

e) $L = \{ababa\}$

$\therefore h^{-1}(L) = \{022, 102, 110\}$

f) $L(a(ba)^*)$

$\therefore h^{-1}(L) = 0(2)^*$

Convertion & Complexity

Convertion name	Complexity
NFA to DFA	$O(n^3 2^n)$
DFA to NFA	$O(n)$
Automata (DFA) to RE	$O(n^3 4^n)$
Automata (NFA) to RE	$O(8^n 4^{2^n})$ doubly exponential
RE to ENFA	$O(n)$
ENFA to NFA	$O(n^3)$

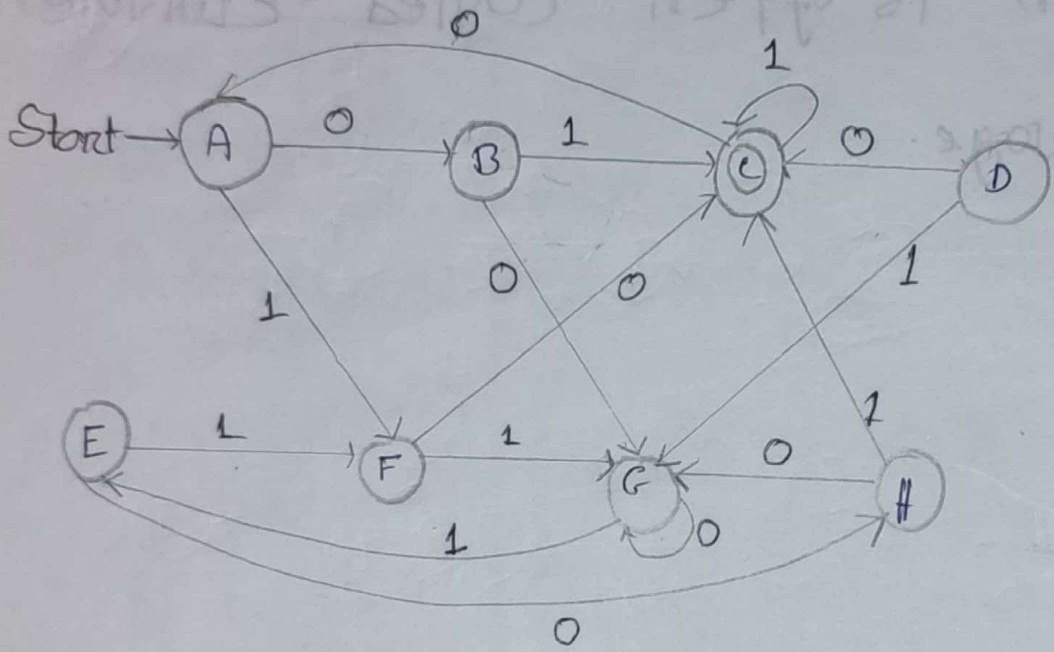
Decision properties of RE

We now consider some of the fundamental questions about language:

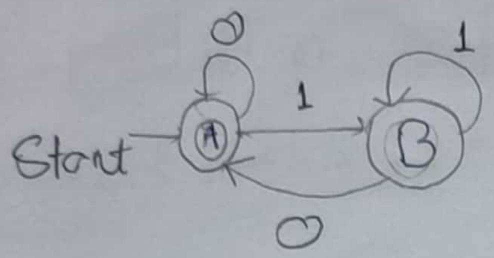
1. Is the language described empty?
2. Is ~~the~~ a particular string ω in the described language?
3. Do two descriptions of a language actually describe the same

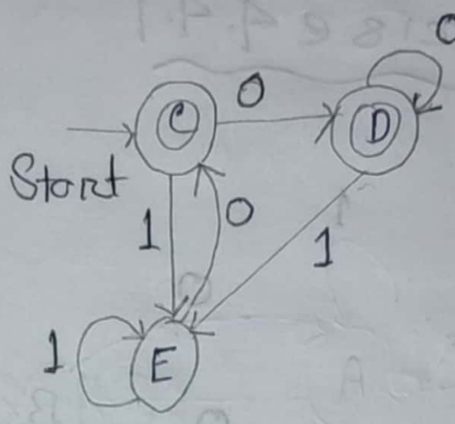
language!

This question is often called "equivalence of language."



	A	B	C	D	E	F	G
B	X						
C	X	X					
D	X	X	X				
E		X	X	X			
F	X	X	X		X		
G	X	X	X	X	X	X	
H	X		X	X	X	X	X

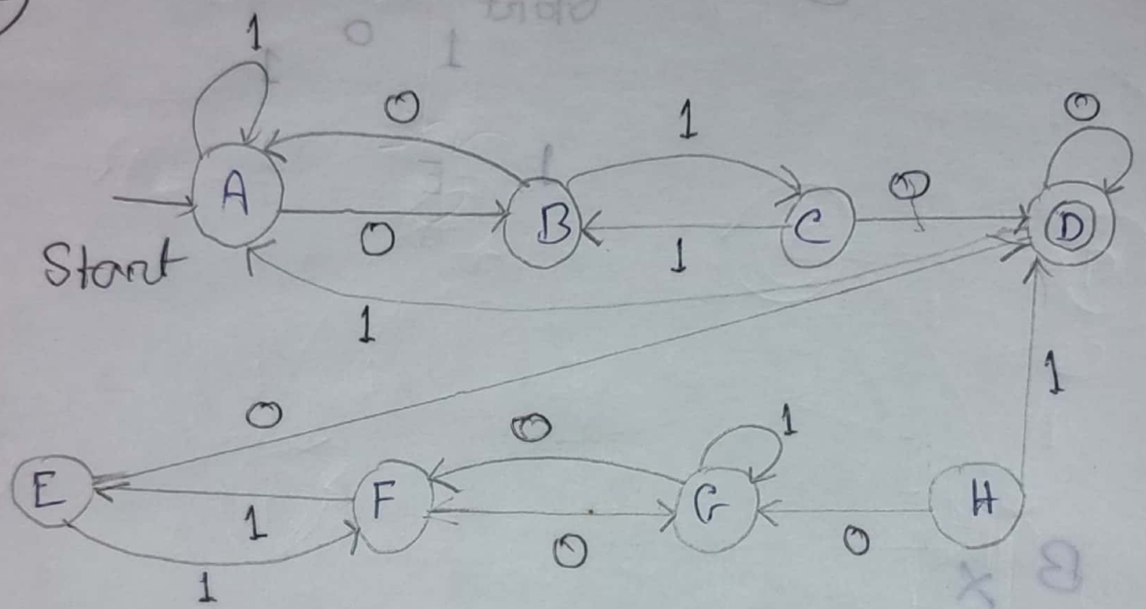




B	X			
C		X		
D		X		
E	X		X	X
	A	B	C	D

Exercise 4.4.1

a)



B	x						
C	x	x					
D	x	x	x				
E	x	x		x			
F	x		x	x	x		
G		x	x	x	x	x	
H	x	x	x	x	x	x	x
	A	B	C	D	E	F	G

b) The partition of the states into equivalent blocks is

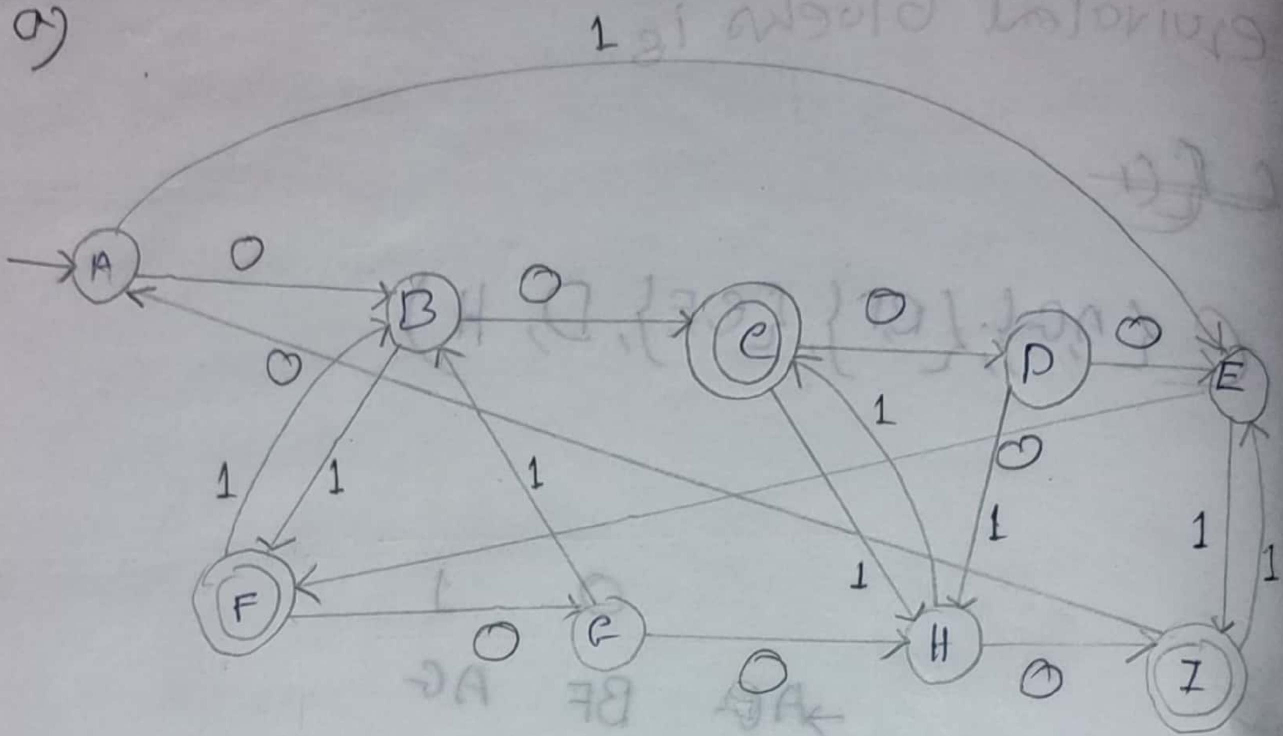
~~C {G~~

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

	0	1
→AG	BF	AG
BF	AG	CE
CE	D	BF
*D	D	AG
H	AG	D

Exercise: 4.4.2

a)



B	x							
C	x	x						
D		x	x					
E	x	x	x	x				
F	x	x		x	x			
G		x	x		x	x		
H	x		x	x		x	x	
I	x	x		x	x		x	x
	A	B	C	D	E	F	G	H

6) The partition of the state into block is

({A, G, D}, {B, H, E}, {C, F, I})

	0	1
→AGD	BEH	BEH
BEH	CFI	CFI
*CFI	AGD	BEF

Chapter-5

Context free grammer

Context free grammer: A CFG is a way of describing language by recursive rules called productions. A ~~CGG~~ CFG consists of a set of variables, a set of symbols and a start variable, as well as the productions.

Each production consists of a head variable and a body consisting of a string of zero or more variables and/or terminals.

- **Palindrome:** A palindrome is a string that reads the same forward and backward such as otto, madamImadam.

Basis: ϵ, 0, 1 are palindromes.

Induction: If w is a palindrome so are $0w0$ and $1w1$.

Example 5.1: Production for context free grammer ~~for~~ of a palindrome number.

$P \rightarrow \epsilon$

$P \rightarrow 0$

$P \rightarrow 1$

$P \rightarrow 0P0$

$P \rightarrow 1P1$.

Definition of Context free grammar:

A Context free grammar or CFG "G" defined by it's four components, that is

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

where V is the set of variables

T is the set of terminals

P is the set of productions

S the start symbol.

Example 5.2 :

A context free grammar for the palindromes is represented by

$G_{pal} = (\{P\}, \{0, 1, \epsilon\}, A, P)$

A represents the set of five productions:

$P \rightarrow \epsilon$

$P \rightarrow 0$

$P \rightarrow 1$

$P \rightarrow 0P0$

~~P → 1P0~~

$P \rightarrow 1P1$

Example 5.3:

Context ~~gr~~ free grammar for $(a+b)(a+b+0+1)^*$

Here, a or b is followed by $\{0,1,a,b\}^*$

Production's ~~set~~ are:

$E \rightarrow I$

$E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow (E)$

$I \rightarrow a$

$I \rightarrow b$

$I \rightarrow Ia$

$I \rightarrow Ib$

$I \rightarrow I0$

$I \rightarrow I1$

$\therefore$ Grammar, $G = (\{E, I\}, \{+, *, (,), 0, 1, a, b\}, P, E)$

Derivation and Languages: Begining with the start symbol, ~~we~~ derive terminal string by repetedly replacing a variable by the body of some production with that variable in the head.

The language of CFG is the set of terminal string.

There are two approaches to infer that certain strings are in the language of a certain variable

→ rules from body to head

→ rules from head to body

Example: ~~5.4~~:

Example 5.4 :

Recurreive inferrence of RE

$(a+b)(a+b+0+1)^*$

Productions are

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

Serial No.	String inferred	For language of	Production used	Strings used
i.	a	I	5	—
ii	b	I	6	—
iii	b0	I	9	ii
iv	b00	I	9	iii
v	a	E	1	i
vi	b00	E	1	iv
vii	a+b00	E	2	(v), (vi)
viii	(a+b00)	E	4	vii
ix	a×(a+b00)	E	3	(v), (viii)

□ Leftmost derivation/Right most derivation:

~~If we always replace~~ the

Beginnig with the start symbol,
we derive terminal strings, by
repededly replacing the leftmost/
rightmost variable in a string ; the
the resulting derivation is a leftmost/
rightmost derivation.

Every string in the language of a CFG
has at least one leftmost and at least
one right most derivation.

Sentential form : Any step in a derivation
is a string of variables and /or terminal
we call such a string a sentential form.
If the derivation is leftmost/ Rightmost
then the string is a ~~res~~ left /right
sentential form.

Example 5.5

~~Rightmost~~ Leftmost derivation of $a*(a+b00)$

$$E \Rightarrow E*E \Rightarrow \cancel{a*E \Rightarrow a*}(E)$$

$$\Rightarrow I*E \Rightarrow a*E \Rightarrow a*(E)$$

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

$$\Rightarrow a*(a+I) \Rightarrow a*(a+I0) \Rightarrow a*(a+I00)$$

$$\Rightarrow a*(a+b00)$$

$\Rightarrow_{rm}$ ~~or~~ and $\overset{*}{\Rightarrow}_{rm}$ is used to indicate one or many rightmost derivation steps respectively.

Example 5.6 :

Rightmost derivation of $a*(a+b00)$

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

$$\underset{rm}{\Rightarrow} E*(E+I) \underset{rm}{\Rightarrow} E*(E+I0) \underset{rm}{\Rightarrow} E*(E+I00)$$

$$\underset{rm}{\Rightarrow} E*(E+b00) \underset{rm}{\Rightarrow} E*(I+b00) \underset{rm}{\Rightarrow} E*(a+b00)$$

$$\underset{rm}{\Rightarrow} I*(a+b00) \underset{rm}{\Rightarrow} a*(a+b00)$$

$\therefore$ Thus : $E \overset{*}{\underset{rm}{\Rightarrow}} a*(a+b00)$

Language of a Grammar:

If $G=\langle V, T, P, S\rangle$ is a CFG the language of G denoted L(G) is the set of terminal strings that have derivations from the start symbol. That is,

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

Parse tree:

A parse tree is a tree that shows the essentials relationship of derivation.

If a grammar $G=\langle V, T, P, S\rangle$ The parse trees for G are ~~tee~~ tree with the following condition:

1. Each interior node is labeled by a variable in V.
2. Each leaf is labeled by either a variable, a terminal or ε. If the leaf is labeled by ε, then it

must be the only child of its parent.

3. If an interior node is labeled A and its children are labeled $X_1, X_2 --- X_k$.

$A \rightarrow X_1, X_2 --- X_k$ is a production of P.

- Yield of a parse tree: The leaves of any parse tree and concatenate them from left, we get a string called the yield of the tree.

- Special importance of a parse tree:

1. The yield is a terminal string. That is all leaves are labeled with terminal symbol or with ϵ.

2. The root is labeled by the start symbol.

Ambiguous grammar: A CFG, G = (V, T, P, S) is ambiguous if there is at least one string w in T* for which we can find two ~~parse~~ different parse trees, each with root labeled S and yield w.

~~If If each stri~~

Reason of ambiguity:

1. The precedence of operators is not respected
2. A sequence of identical operators can group either from the left or from the right.

Way of removing ambiguity:

1. A Factor is an expression that cannot be broken apart by any adjacent operator, either a * or a +. The only factors in our expression language are:

   a) Identifiers. It is not possible to separate the letters of an identifier by attaching an ~~facto~~ operator.

   b) Any parenthesized expression no matter what appears inside the parentheses.

2. A term is an expression that can not be broken by the + operator.

3. An expression will refer to any possible expression including those that can be broken by either an adjacent * or an

adjacent +

NB: (1) কম precendence এর operator starting এর কাছে থাকবে, বেশি precedence এর operator starting এর থেকে দূরে থাকবে,

(2) কোন ~~এ~~ এক্সপ্রেশনে একের অধিক ~~অপ~~ operator রাখা যাবে না।

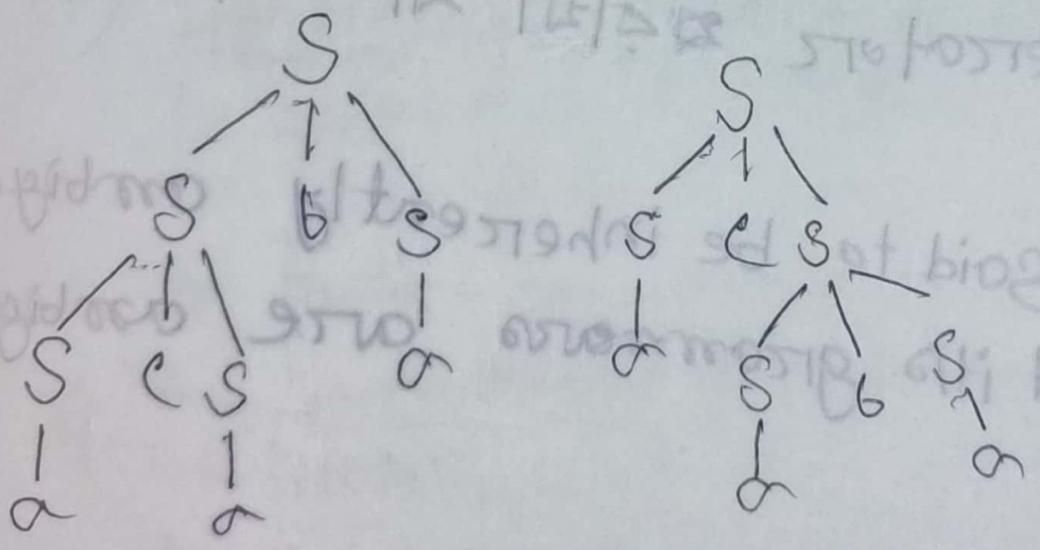
A CFG is said to be inherently ambiguous if all its grammars are ambiguous.

Show that the CFG

$S \to SbS \mid ScS \mid a$ is ambiguous

Soln:

~~we~~ ~~fo~~ we find a parse tree for acaba



As we find two leftmost ~~pa~~ parse tree for a String acaba so the CFG is ambiguous.

Let us now consider the following grammar:

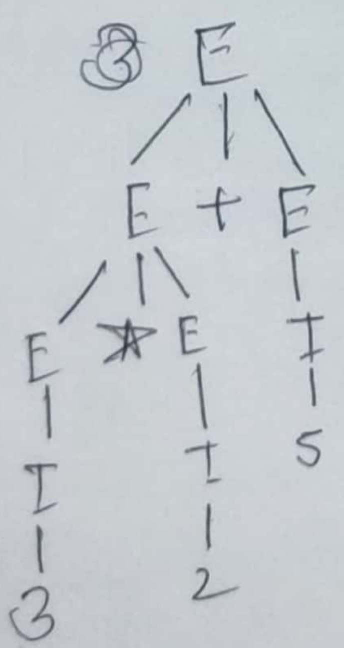
Set of alphabets? = {0 --- 9 +, *, (,)}

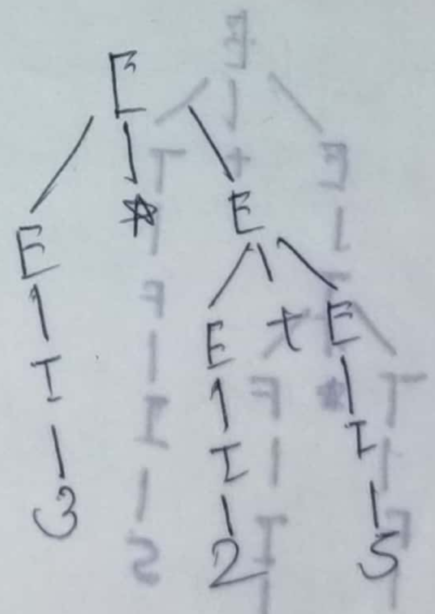
$E \rightarrow I \mid E+E \mid E*E \mid (E)$

$I \rightarrow \epsilon \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

does the string 3*2+5 is ambiguous?

if ambiguous then make it unambiguous.





As we find two leftmost tree for string 3*2+5

∴ String 3*2+5 create ambiguity

$I \rightarrow \epsilon|0|1|2|3|4|5|6|7|8|9$

$E \rightarrow \cancel{E+F \mid F}$

$\cancel{F \rightarrow F+T \mid T}$

$\cancel{T \rightarrow}$

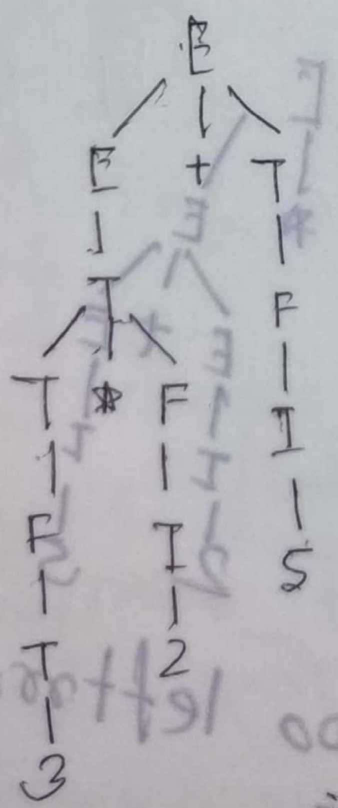
$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow I \mid (E)$

Now

~~Fow~~ Fow $3*2+5$



∴ As there is only one leftmost ~~tre~~ parse tree ambiguity is removed

Exerceise 5.4.3

S → aS | aSbS | ε

Find an unambiguous grammar for given CFG

Soln: S → aS | aTbS | ε

T → aTbF | ε

F → bF | ε

As S can be generated both left or toward right side by ~~ele~~ using a term variable T we remove this. Now S will be generated toward right direction.

Thus ambiguity is removed.

exerceise 5.4.6

T = {0, 1, +, *, (,), φ, ε} a CFG for regular expression.

design it and remove it ambiguity

$E \rightarrow E+E \mid E^* \mid EE \mid I$

$I \rightarrow 0 \mid 1 \mid \epsilon$

Removing ambiguity the CFG will become followings.

$E \rightarrow E+T \mid T$

~~$T \rightarrow T*F \mid F$~~

~~$F \rightarrow (E)$~~

$T \rightarrow TF \mid F$

$F \rightarrow F^* \mid (E) \mid 0 \mid 1 \mid \epsilon$

Chapter ~~X~~

~~Normal~~
Properties of CFG

CNF
~~CFL~~ : Every nonempty CFL $\boxed{\text{without } \epsilon}$ has a grammar G (CFG) in which all productions are in one of two simple forms, either:

1. $\boxed{A \rightarrow BC}$ where A, B and C are each variables.

2. $\boxed{A \rightarrow a}$ where A is a variable and a is a terminal.

Further (আরও) G has no useless symbols. Such a grammar is said to be in Chomsky Normal form or CNF

A CNF has no ϵ production, unit production or useless symbol.

□ **Unit production:** A unit production is a production of the form $\boxed{A \rightarrow B}$ where both A and B are variables.

These production can be useful.

□ **A CFG without ϵ production/ transition:** If language L has a CFG, then $L-\{\epsilon\}$ has a CFG without ϵ productions. If ϵ is not in L then L has a CFG without ϵ-production.

□ **nullable:** A variable is A is nullable if $A \overset{*}{\Rightarrow} \epsilon$. If A is nullable then whenever A appears in a production body $A \rightarrow BC\ CAD$, A might (or might not) derive ϵ. We make two versions of production, one without A in the body ($B \rightarrow CD$) other with A still present ($B \rightarrow CAD$).

Useful symbol: A symbol that is usefull ~~will~~ if ~~both~~ it is both generating and reachable. Otherwise symbol is ~~that~~ useless symbol.

A symbol X is useful for a grammar $G = (V, T, P, S)$ is there is some derivation of the form $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} \omega$ where ω is in T^*.

and ① X is generating

① if $X \overset{*}{\Rightarrow} \omega$ for some terminal String ω.

2) X is reachable if there is a derivation $S \overset{*}{\Rightarrow} \alpha X \beta$ for some α and β.

Eliminating Useless symbol (Technique)

① → Elimination of $\boxed{\text{non generating}}$ symbol

- Process:

① include terminal set T

② Include ω_1 that derives some terminals and initialize $i=1$

③ Include ω_{i+1} ~~from~~ that derives ω_i

④ repeat step 2 until $\omega_{i+1} = \omega_i$

⑤ Include all production ~~for the~~ rules that have ω_i in H.

Terminal/Variable

→ Elimination of $\boxed{\text{non reachable}}$ symbol

Start → Symbol (Variable + terminal)

① Include the start symbol in y_1 and initialize $i=1$

② Include all the symbols y_{i+1} that can be derive from y_i and include all production rules that have been applied.

③ Increment i and repeat step 2 until $y_{i+1} = y_i$

Eliminating null production : (Technique)

1. To remove $A \to \epsilon$ look for all productions whose right side contain A.

2. Replace $\boxed{\text{each occurrence}}$ (all possible pattern for each production of 'A') in each of these productions with ϵ

3. Add the resultant production of the grammar.

Eleminitating unit $(A \to B)$ production: (Technique)

1. First find all unit pair by using basis & induction rule.

2. ~~Find elem~~ Keep the production for every pair by removing $(A \to B)$ or unit production

Finding Unit ~~Symbol~~ pair:

Basis: (A, A) is a unit pair for any variable A. That is $A \overset{*}{\Rightarrow} A$ by zero steps.

Induction: Suppose we have determined that (A, B) is a unit pair and $B \to C$ is a production, where C is a variable. Then (A, C) is a unit production.

Eliminating ϵ - production

Basis: If $A \to \epsilon$ is a production of G then A is nullable.

Induction: If there is a production $B \to C_1 C_2 \cdots C_i$ where each C_i is nullable, then B is nullable.

Example 7.1

Eliminate useless symbol from following production,

$S \to AB \mid a$

$A \to b$

Soln:

Eliminating nongenerating symbol,

$T = \{a, b\}$

$\omega_1 = \{A, S\}$

$\omega_2 = \{A, S\}$

$\therefore S \to a$

$A \to b$

Now, Eliminate non reechable symbol,

$\therefore y_1 = \{S\}$

$y_2 = \{S\}$

$\therefore S \to a$ is the production after eliminating useless

Symbol.

Example 7.8:

Consider the grammar

$S \rightarrow AB$

$A \rightarrow aAA \mid \epsilon$

$B \rightarrow bBB \mid \epsilon$

Find nullable symbol and eliminate ϵ production

Soln :

Here,

$A \rightarrow \epsilon$

$B \rightarrow \epsilon$

$\therefore$ A and B is nullable.

again $S \rightarrow AB$

$\therefore$ S is also nullable.

Eliminate $A \rightarrow \epsilon$

$S \rightarrow AB \mid B$

$A \rightarrow aAA \mid aA \mid a$

$B \rightarrow bBB \mid \epsilon$

Eliminate $B \rightarrow \epsilon$

$S \rightarrow AB \mid B \mid A$

$A \rightarrow aAA \mid aA \mid a$

$B \rightarrow bB \mid b \mid bBB$ ~~when is out~~

$\therefore$ by eliminating ϵ production we obtain

$S \to AB \mid A \mid B$

$A \to aAA \mid aA \mid a$

$B \to bBB \mid bB \mid b$

Example 7.10

Consider the production

$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$F \to I \mid (E)$

$T \to F \mid T * F$

$E \to T \mid E + T$

Find the unit pair

Soln: According to basis rule

$(E,E); (T,T); (F,F); (I,I)$ are unit pair

Now

1. (E,E) unit pair and $E \to T \therefore (E,T)$ is unit pair
2. (E,T) " " " $T \to F \therefore (E,F)$ " "
3. (E,F) " " " $F \to I \therefore (E,I)$ " "

~~4. (F, I) is unit pair~~ and

4. (T, T) is unit pair and T → F ∴ (T, F) is unit pair

and F → I ∴ (F, I) " " "

5. (T, F) " " " and F → I ∴ (F, I) " " "

6. (F, F) " " "

∴ Unit pairs are,

(E, E) ; (E, T) ; (E, F) ; (E, I) ; (T, T) ; (T, F) ; (T, I) ;
(F, F) ; (F, I) , (I, I)

Example 7.12 :

Eliminate unit production for example

7.10

Soln :

Productions are,

I → a | b | Ia | Ib | I0 | I1

F → I | (E)

T → F | T * F

E → T | E + T

Pair	Production
(E, E)	E+T
	T*F
	(E)
(E, T)	
(E, F)	a|b| Ia|Ib|I0|I1
(E, I)	
(T, T)	T*F
(T, F)	(E)
(T, I)	a|b|Ia| Ib|I0 |I1
(F, F)	(E)
(F, I)	a|b| Ia| Ib| I0|I1
(I, I)	a|b|Ia|Ib|I0|I1

$\therefore E \rightarrow E+T \mid T*F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$T \rightarrow T*F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$F \rightarrow (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

which is the production without unit production.

Example: 7.15:

Convert the grammar of 7.12 into a CNF.

Sol^n: Productions are

E → E+T | T*F | (E) | a | b | Ia | Ib | I0 | I1
T → T*F | (E) | Ia | Ib | I0 | I1 | a | b
F → (E) | a | b | Ia | Ib | I0 | I1
I → a | b | Ia | Ib | I0 | I1

Now, create productions for all terminals.

O → 1
Z → 0
A → a
B → b
R →)
L → (
P → +
M → *

~~E → EMT~~ | T
E → EPT | TMF | LER | a | b | IA | IB | IZ | IO
T → TMF | LER | IA | IB | IZ | IO
F → LER | a | b | IA | IB | IZ | IO
I → a | b | IA | IB | IO | IZ

Now create production for RHS of length's 3 or more.

$E \to EC_1 \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$

$T \to TC_2 \mid LC_3 \mid IA \mid IB \mid IZ \mid IO$

$F \to LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$

$I \to a \mid b \mid IA \mid IB \mid IO \mid IZ$

$A \to a$

$B \to b$

$O \to 1$

$Z \to 0$

$R \to)$

$L \to ($

$P \to +$

$M \to *$

$C_1 \to PT$

$C_2 \to MF$

$C_3 \to ER$

Which is our required CNF (Chomsky normal form).

Fresh

Exercise Book

Name Md Ismail (2)

Institute RUET

Roll/ID 1503094 Year 2nd

Section/Dept. CSE(B) Subject Automata/CSE-2205

Chapter 7

Exercise 7.1.1

Soln:

$S \to AB \mid CA$

$A \to a$

$B \to BC \mid AB$

$C \to aB \mid b$

First eliminate nongenerating symbol.

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

$\omega_1 = \{C, A\}$

$\omega_2 = \{C, A, S\}$

$\omega_3 = \{C, A, S\}$

$\therefore$ B is our nongenerating symbol.

$S \to CA$

$A \to a$

$C \to b$

Now eliminate non reachable symbol

$y_1 = \{S\}$

$y_2 = \{C, A, S\}$

$y_3 = \{C, A, S, a, b\}$

$\therefore$ ~~$S \rightarrow$~~

$y_3 = \{C, A, S, a, b\}$

$\therefore$ $S \rightarrow CA$

$A \rightarrow a$

$C \rightarrow b$

which are the production without useless symbol.

Exercise 7.1.2

Given,

$S \rightarrow ASB \mid \epsilon$

$A \rightarrow aAS \mid a$

$B \rightarrow SbS \mid A \mid bb$

a) $S \rightarrow \epsilon$

$\therefore$ S is nullable

Now remove $S \rightarrow \epsilon$

$S \rightarrow ASB \mid AB$

$A \rightarrow aAS \mid aA \mid a$

$B \rightarrow SbS \mid bS \mid Sb \mid b \mid A \mid bb$

which are the production without ϵ production.

6

Now,

$S \rightarrow ASB | AB$
$A \rightarrow aAS | aA | a$
$B \rightarrow SbS | bS | Sb | b | A | bb$

Here B → A is a unit production.

∴ ~~and~~ (S,S), (A,A) (B,B) (B,A) is unit pair

Pair	Production
(S,S)	~~A~~ $S \rightarrow ASB \| AB$
(A,A)	~~A→~~ $aAS \| aA \| a$
(B,B)	$B \rightarrow SbS \| bS \| Sb \| b \| bb$
(B,A)	$B \rightarrow aAS \| aA \| a$

∴ $S \rightarrow ASB | AB$
$A \rightarrow aAS | aA | a$
$B \rightarrow SbS | bS | Sb | b | bb | aAS | aA | a$

which are the production without unit ~~pa~~ production,

Ans

c Here,

$S \to ASB \mid AB$

$A \to aAS \mid aA \mid a$

$B \to SbS \mid bS \mid Sb \mid b \mid aAS \mid aA \mid a \mid bb$

First remove non generating symbol.

$T = \{a, b\}$

$W_1 = \{A, B\}$

$W_2 = \{A, B, S\}$

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

$\therefore$ There is no non generating symbol.

$\therefore S \to ASB \mid AB$

$A \to aAS \mid aA \mid a$

$B \to SbS \mid bS \mid Sb \mid b \mid aAS \mid aA \mid a \mid bb$

Now eliminate non reachable symbol.

$y_1 = \{S\}$

$y_2 = \{S, A, B\}$

$y_3 = \{a, A, B, S\}$

$y_4 = \{b, A, B, S, a\}$

$y_5 = \{b, A, B, S, a\}$

$\therefore$ There is no non reachable symbol

$\therefore$ $S \to ASB | AB$

$A \to aAS | aA | a$

$B \to SbS | bS | Sb | b | aAS | aA | a | bb$

Ans

d) Here,

$S \to ASB | AB$

$A \to aAS | aA | a$

$B \to SbS | bS | Sb | b | aAS | aA | a | bb$

Now create production for terminal symbol,

$S \to ASB | AB$

$A \to XAS | XA | a$

$B \to SYS | YS | SY | b | AY | XA | a | YY$

$X \to a$

$Y \to b$

Now create production for 3 or more length's RHS)

$S \to AE | AB$

$A \to XF | XA | a$

$B \to SG | YS | SY | b | AY | XA | a | YY$

$X \to a$

$Y - b$

$E \to SB,$

$F \to AS$

$G \to YS$

which is the required chomsky Normal form

Exercise 7.1.5

a) Here, $S \to aAa \mid bBb \mid \epsilon$

$A \to C \mid a$

$B \to C \mid b$

$C \to CDE \mid \epsilon$

$D \to A \mid B \mid ab$

Here, $S \to \epsilon$, so S is nullable

$C \to \epsilon$ so C is also nullable.

again $A \to C$

$B \to C$

$D \to A \mid B$

$\therefore$ A, B and D is also nullable.

$\therefore A \to \epsilon$

$B \to \epsilon$

$D \to \epsilon$

First remove $S \to \epsilon$

$S \to aAa \mid bBb$
$A \to C \mid a$
$B \to C \mid b$
$C \to CDE \mid \epsilon$
$D \to A \mid B \mid ab$

remove $C \to \epsilon$

$S \to aAa \mid bBb$
$A \to C \mid a$
$B \to C \mid b$
$C \to CDE \mid DE$
$D \to A \mid B \mid ab$

remove $D \to \epsilon$

$S \to aAa \mid bBb$
$A \to C \mid a$
$B \to C \mid b$
$C \to CDE \mid CE \mid DE \mid E$
$D \to A \mid B \mid ab$

remove $A \to \epsilon$ and $B \to \epsilon$

$S \to aAa \mid aa \mid bBb \mid bb$
$A \to C \mid a$
$B \to C \mid b$
$C \to CDE \mid CE \mid DE \mid E$
$D \to A \mid B \mid ab$ which is required introduction without

b

Here,

$S \to aAa \mid bBb \mid aa \mid bb$

$A \to C \mid a$

$B \to C \mid b$

$C \to CDE \mid CE \mid DE \mid E$

$D \to A \mid B \mid ab$

(S,S) (A,A) (B,B), (C,C) (D,D) is unit pair.

again,

(D, D) A is a unit pair. $D \to A$ ∴ (D,A) is a unit pair

(D, A) " " " $A \to C$ ∴ (D,C) " " " "

(D, C) " " " $C \to E$ ∴ (D,E) " " " "

" $D \to B$ ∴ (D,B) " " "

(D, D) " " $B \to C$ (D,C) " " "

(D, B) " " $C \to E$ (C,E) " " "

(C, C) " $B \to C$ (B,C) " "

(B, B) " " $C \to E$ (B,E) " " "

(B, C) " " $A \to C$ (A,C) " "

(A, A) " " $C \to E$ (A,B) " "

(A, C) "

Pair	Production
(S,S)	$S \to aAa \mid bBb \mid aa \mid bb$
(D,D)	$D \to ab$
(D,A)	$D \to a$
(D,C)	$D \to cDE \mid cE \mid DE$
(D,E)	ϕ
(C,C)	$C \to cDE \mid cE \mid DE$
(C,E)	ϕ
(B,B)	$B \to b$
(B,C)	$B \to cDE \mid cE \mid DE$
(B,E)	ϕ
(A,A)	$A \to a$
(A,C)	$A \to cDE \mid cE \mid DE$
(A,E)	ϕ

$\therefore$ $S \to aAa \mid bBb \mid aa \mid bb$

$A \to a \mid cDE \mid cE \mid DE$

$B \to b \mid cDE \mid cE \mid DE$

$C \to cDE \mid cE \mid DE$

$D \to ab \mid a \mid cDE \mid cE \mid DE$

which ~~is~~ are the production without unit ~~sym~~ production.

Here,

$S \to aAa \mid bBb \mid aa \mid bb$

$A \to a \mid CDE \mid CE \mid DE$

$B \to b \mid CDE \mid CE \mid DE$

$C \to CDE \mid CE \mid DE$

$D \to ab \mid a \mid CDE \mid CE \mid DE$

Now eliminate non generating symbol.

$T = \{a, b\}$

$W_1 = \{S, A, B, D\}$

$W_2 = \{A, B, D, S\}$

$\therefore$ C and E is our non generating symbol. now remove them from production.

$S \to aAa \mid bBb \mid aa \mid bb$

$A \to a$

$B \to b$

$D \to ab \mid a$

Now, eliminate non reachable symbol.

$Y_1 = \{S\}$

$Y_2 = \{S, a, A, b\}$

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

~~$Y_4 = \{S, a, A, b, D\}$~~

~~$S \to aA$~~

∴ ~~There is no non reachable~~

∴ D is our non reachable symbol.

Now eliminate it from production.

∴ $S \to aAa \mid bBb \mid aa \mid bb$

$A \to a$

$B \to b$

which is our required production without useless symbol.

d Now, $S \to aAa \mid bBb \mid aa \mid bb$

$A \to a$

$B \to b$

∴ Now create production by replacing a and b by A and B respectively when they are not single.

$S \to AAA \mid BBB \mid AA \mid BB$

$A \to a \quad B \to b$

Now create new production for 3 or more lengths RHS.

$S \to AC \mid BD \mid AA \mid BB$
$A \to a$
$B \to b$
$C \to AA$
$D \to BB$

Which is the required Chomsky ~~no~~ normal form

Pumping lemma for CFL:

Let L be ~~a~~ a CFL. Then there exists a ~~conston~~ constant n such that if z is any string in L such that $|z|$ is at least n ($|z| \geq n$) then we can write $z = uvwxy$ subject to the following:

1. ~~tvwxy~~ $|vwx| \leq n$
2. $vx \neq \epsilon$ Since v and x are the pieces to be "pumped" this

condition says that at least one of the strings we pump must not empty

③ For all $i \geq 0$ $\quad$ uv^iwx^iy is in L.

④ Application of CFL → Text 301

Chapter-6
Pushdown Automata

CFG to PDA

Rule-1:

For each variable A

$\delta(q, \epsilon, A) = \{(q, B) \mid A \to B$ is a production of $G\}$

Rule-2:

For each terminal a,

$\delta(q, a, a) = \{q, \epsilon\}$

Example-6.12:

Consider the following CFG

$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

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

The set of input symbols for the PDA is $\{a, b, +, *, (,), 0, 1\}$ the stack alphabet symbols I and E are also the terminals.

The transition function for the PDA is,

a) $\delta(q, \epsilon, I) = \{(q, a), (q, b), (q, I0), (q, I1), (q, Ia), (q, Ib)\}$

b) $\delta(q, \epsilon, E) = \{(q, I), (q, E*E), (q, E+E), (q, (E))\}$

c) $\delta(q, a, a) = \{(q, \epsilon)\}$; $\delta(q, b, b) = \{(q, \epsilon)\}$
$\delta(q, 0, 0) = \{(q, \epsilon)\}$; $\delta(q, 1, 1) = \{(q, \epsilon)\}$
$\delta(q, +, +) = \{(q, \epsilon)\}$; $\delta(q, *, *) = \{(q, \epsilon)\}$
$\delta(q, (, () = \{(q, \epsilon)\}$; $\delta(q,),)) = \{(q, \epsilon)\}$

$\therefore$ PDA $P = (\{q\}, \{0, 1, a, b, +, *, (,)\}, \{0, 1, a, b, +, *, (,), I, A\}, \delta, q, E)$

Am

Exercise 6.3.1

Given CFG,

$$S \rightarrow 0S1 \mid A$$

$$A \rightarrow 1A0 \mid S \mid \epsilon$$

The $\{0, 1, \epsilon\}$ is the input symbol's set,

$\{0, 1, \epsilon, S, A\}$ is the stack element set,

a) $\delta(q, \epsilon, S) = \{(q, 0S1), (q, A)\}$

b) $\delta(q, \epsilon, A) = \{(q, 1A0), (q, S), (q, \epsilon)\}$

c) $\delta(q, 0, 0) = \{(q, \epsilon)\}$; $\delta(q, 1, 1) = \{(q, \epsilon)\}$;

$\therefore$ PDA, $P = (\{q\}, \{0, 1, \epsilon\}, \{0, 1, \epsilon, S, A\}, \delta, q, S)$

Ans

Exercise 6.3.2

Given,

$$S \rightarrow aAA$$

$$A \rightarrow aS \mid bS \mid a$$

$\therefore$ $\{a, b\}$ is the input symbol's set,

$\{a, b, S, A\}$ is the stack element set,

a) $\delta(q, \epsilon, S) = \{(q, aAA)\}$

b) $\delta(q, \epsilon, A) = \{(q, aS), (q, bS), (q, a)\}$

c) $\delta(q, a, a) = \{(q, \epsilon)\}$

d) $\delta(q, b, b) = \{(q, \epsilon)\}$

$\therefore$ PDA $P = (\{q\}, \{a, b\}, \{a, b, S, A\}, \delta, q, S)$

Chapter-8

Turing machine

□ Turing machine: A turing machine TM is described by 7-tuple

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

where:

Q = The finite set of states of the finite control.

Σ = The finite set of input symbols.

Γ = The complete set of tape symbols; Σ is always a subset of Γ.

δ = The transition function. The arguments of $\delta(q, X)$ are a state q and a tape symbol X. The value of $\delta(q, X)$ is it is defined is a triple (P, Y, D). where:

1. P is the next state in Q

2. Y is the symbol in Γ.

3. D is the direction either L or R respectively and telling us the direction in which the head moves.

q_0 = The start state, a member of Q

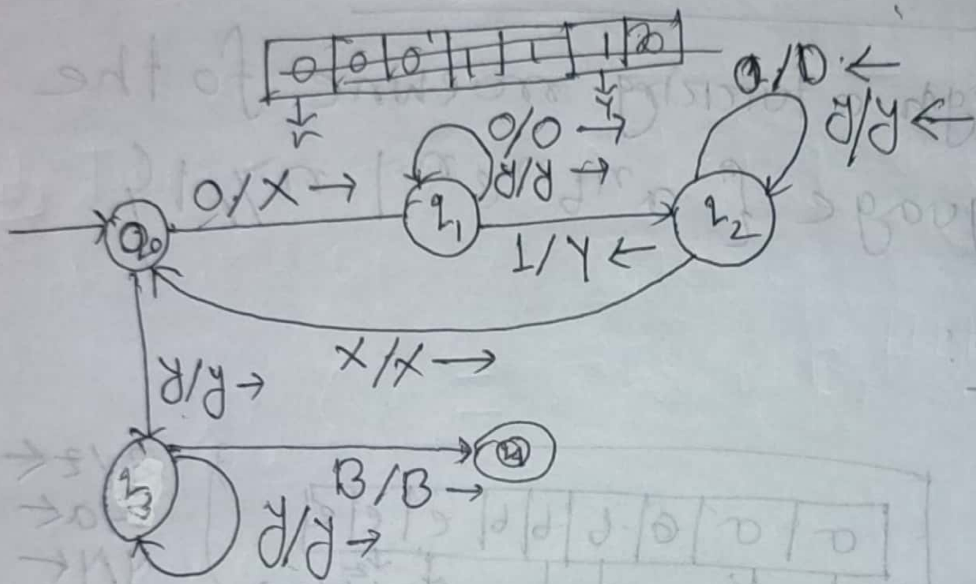
B = The blank symbol.

This symbol is in Γ but not in Σ. i.e, it is not an input symbol.

F = The set of final or ~~accep~~ accepting state a subset of Q

Exercise 8.2.2 (a) / Example-82/83

Design a turring machine for $\{0^n 1^n \mid n \geq 1\}$



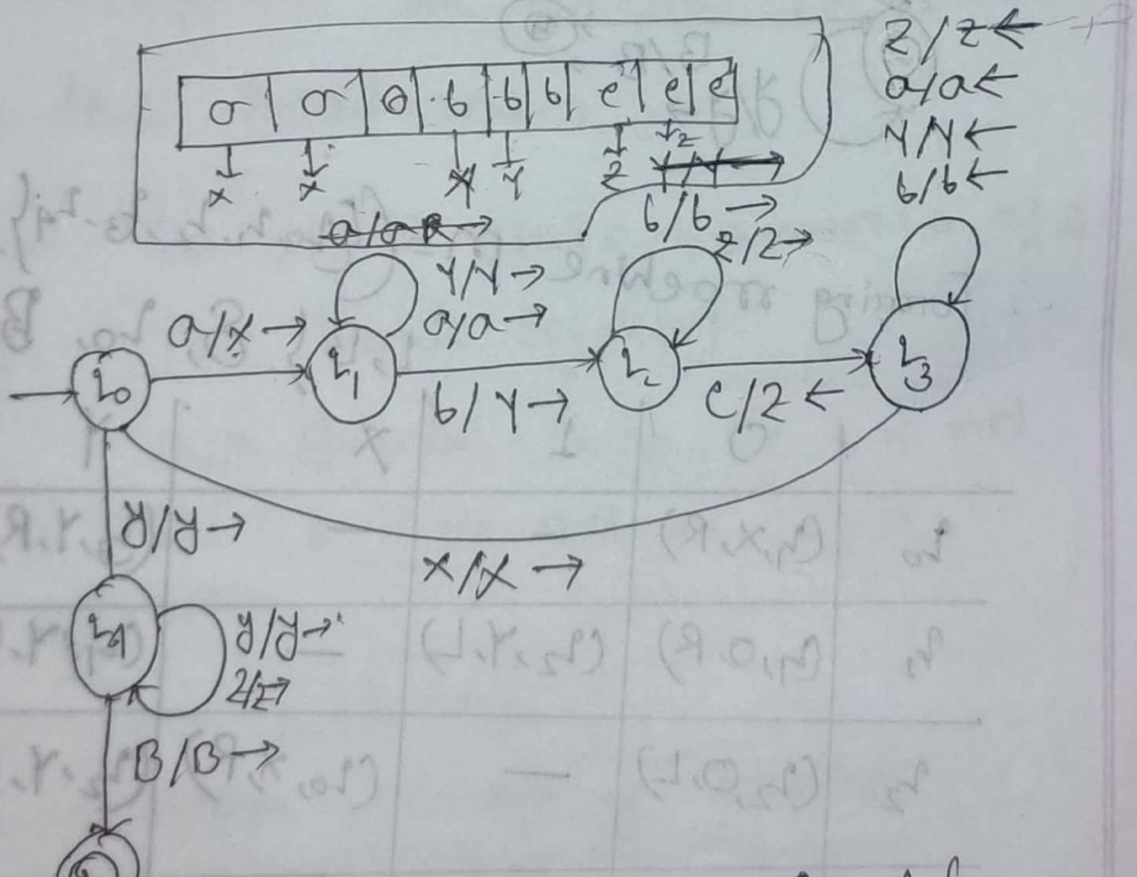
∴ Turring machine $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_4\})$

	0	1	X	Y	B
q_0	(q_1, X, R)	—	—	(q_3, Y, R)	—
q_1	$(q_1, 0, R)$	(q_2, Y, L)	—	(q_1, Y, R)	—
q_2	$(q_2, 0, L)$	—	(q_0, X, R)	(q_2, Y, L)	—
q_3	—	—	—	(q_3, Y, R)	(q_4, B, R)
q_4	—	—	—	—	—

Exercise 8.2.2 (b)

Design a turing machine for the language $\{a^n b^n c^n \mid n \geq 1\}$

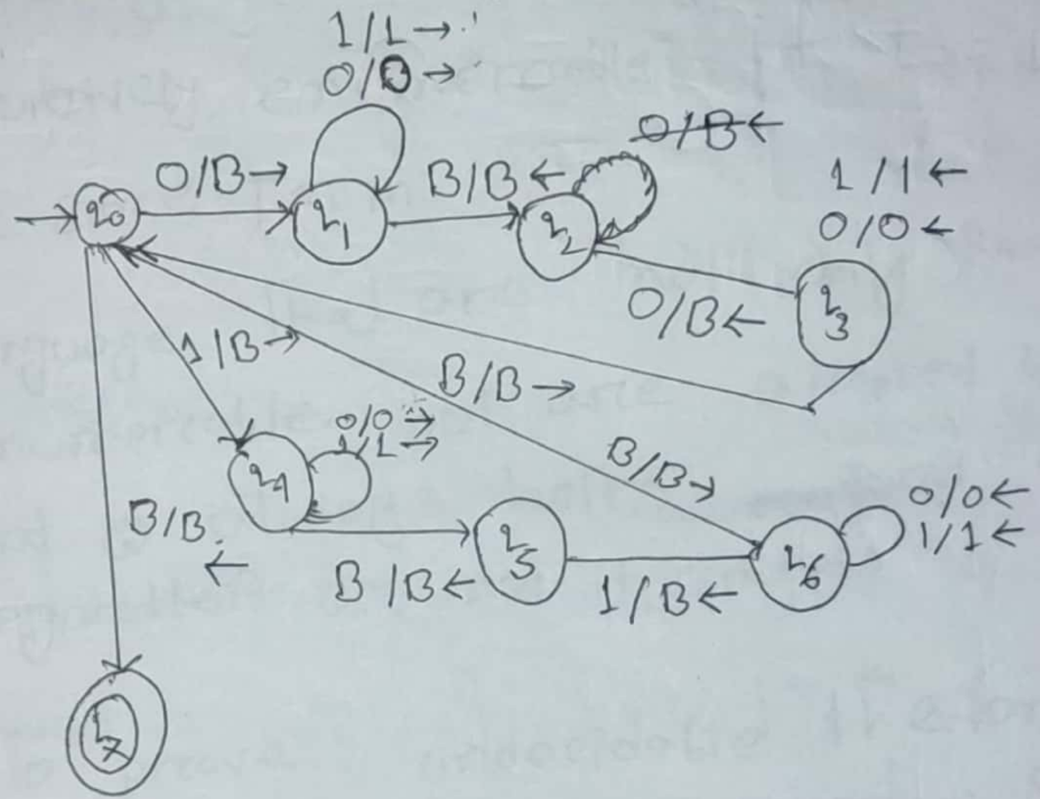
Soln:



$$Tm:, M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b\}, \{a, b, X, Y, Z, B\}, \delta, q_0, B, \{q_5\})$$

Exercise - 8.2.2(e)

Design a Turing machine $\{ww^R \mid w$ in any string of 0's and 1's$\}$



Language of turing machine:

The set of languages we can accept using turing machine is often called recursively enumerable language or RE languages.

Halt with AC
Halt with RE
~~Halt~~ Infinite loop

Chapter-9

Recursively Enumerable: A language L is recursively enumerable if $L = L(M)$ for some TM M.

Language that are not only Recursively enumerable, but are accepted by a TM that always halts ~~regard~~. regardless or not it accepts.

□ To prove undecidable the language consisting of the pairs (M, w) such that:

1. M is turing machine with input alphabet {0, 1}
2. w is a string of 0's and 1's and
3. M accepts input w.

Code of a rule

$\delta(q_i, x_j) = (q_k, x_l, D_m)$

is $0^i 1 0^j 1 0^k 1 0^l 1 0^m$

code for TM.

$\therefore C = C_1 11 C_2 11 C_3 \ldots C_{n-1} 11 C_n$

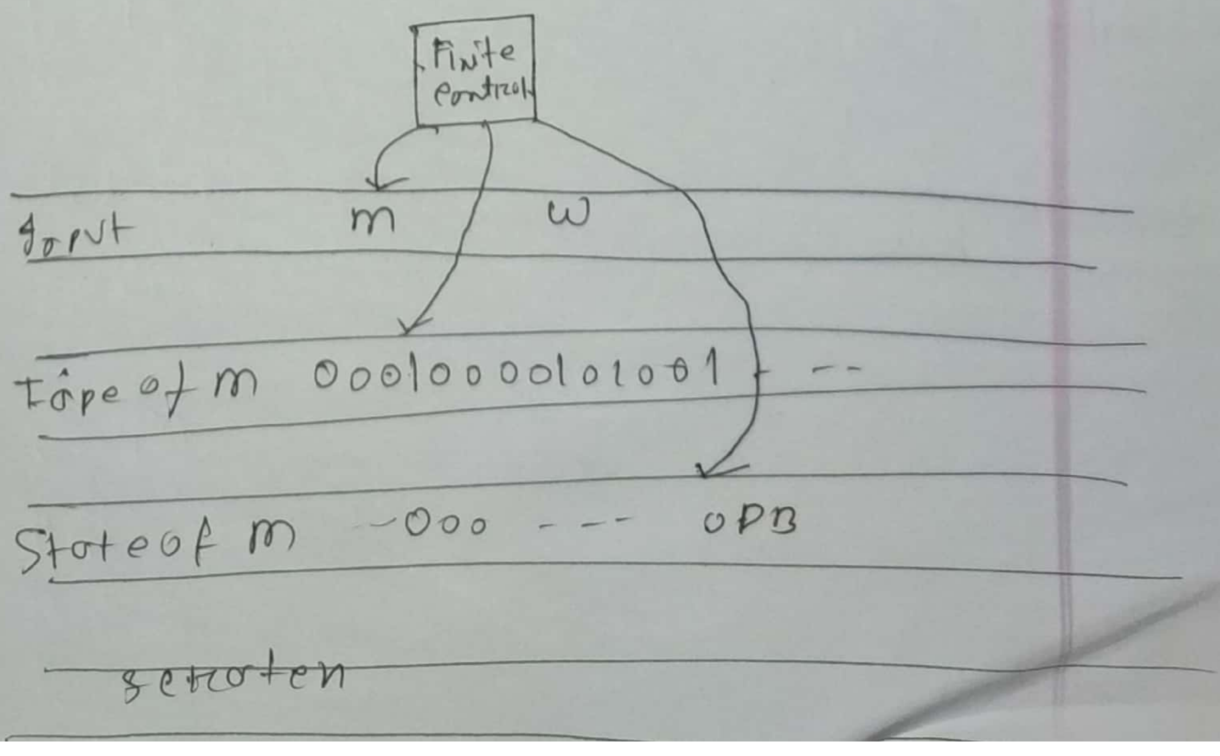
Recursive language:

A language L is recursive if $L = L(M)$ for some ~~turr~~ turing machine M such that

1. If w is in L then M accepts (and therefore halts)
2. If w is not in L, then M eventually halts, although it never enters an accepting state.

A problem L is called decidable if it is a recursive language and it is called undecidable if it is not a recursive language.

language



Design of $\overline{L} = T(\overline{m})$