Context - free language

CFG

push down Automata (machine)

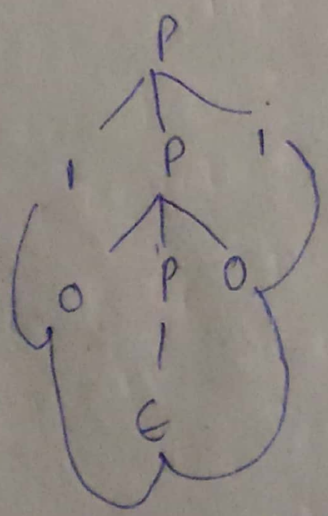
-> Syntax analysis
parser (IInd phase of compiler design)

-> DTD $G(V, T, P, S)$

context free grammer for even pallindrome

$$P \rightarrow \epsilon / 0 / 1 / 0P0 / 1P1.$$

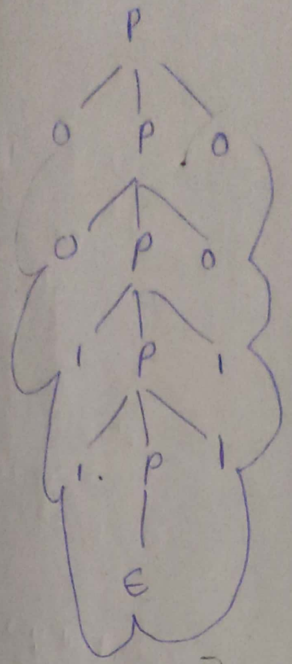
$P \rightarrow 0$
$P \rightarrow 1$
$P \rightarrow \epsilon$
$P \rightarrow 0P0$
$P \rightarrow 1P1$



The context free languages have a natural recursive notations called context free grammars.

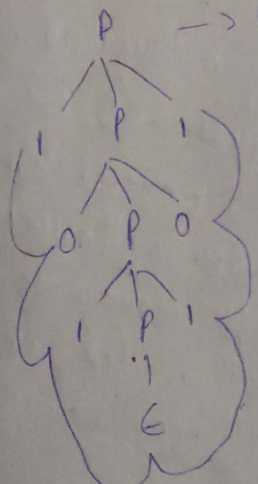
there are 4 components in a grammatical representation of language.

00111100



(00111100)

1011 → not valid



→ parse tree

i) A finite set of symbols that form the strings of the language being defined Eg: {0, 1}

we call this alphabet the terminals (or) terminal symbols.

Scanned by CamScanner

(iii) there is a finite set of variables also called non-terminals (or) syntactic categories each variable represents a language i.e a set of strings

p -> opo

(iii) one of the variables represents the language being defined. it is called start symbol other variables represents auxiliary classes of strings that are used to help define the lang

(iv) there is a finite set of productions or rules that represents the recursive definition of language. Each production consists of

a) The variable that is being (partially) defined by the production. This variable is often called the head of the production.

b) The production symbol ->

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

we leave terminals unchanged and substitute for each variable of the body any string that is known to be in the language of that variable.

$p \to v \,/\, v\, \varphi\, v$

$E \to E + E$

$E \to E * E$

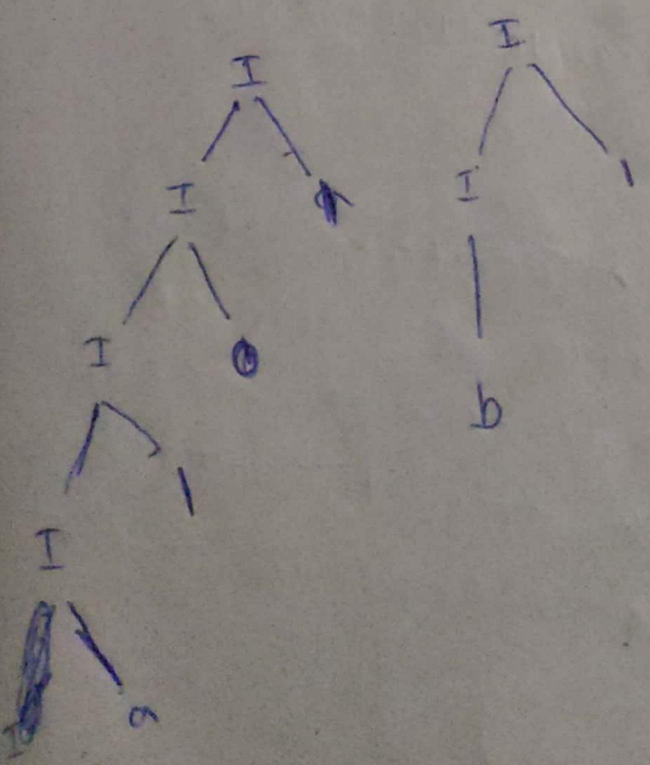
$E \to a \mid d$

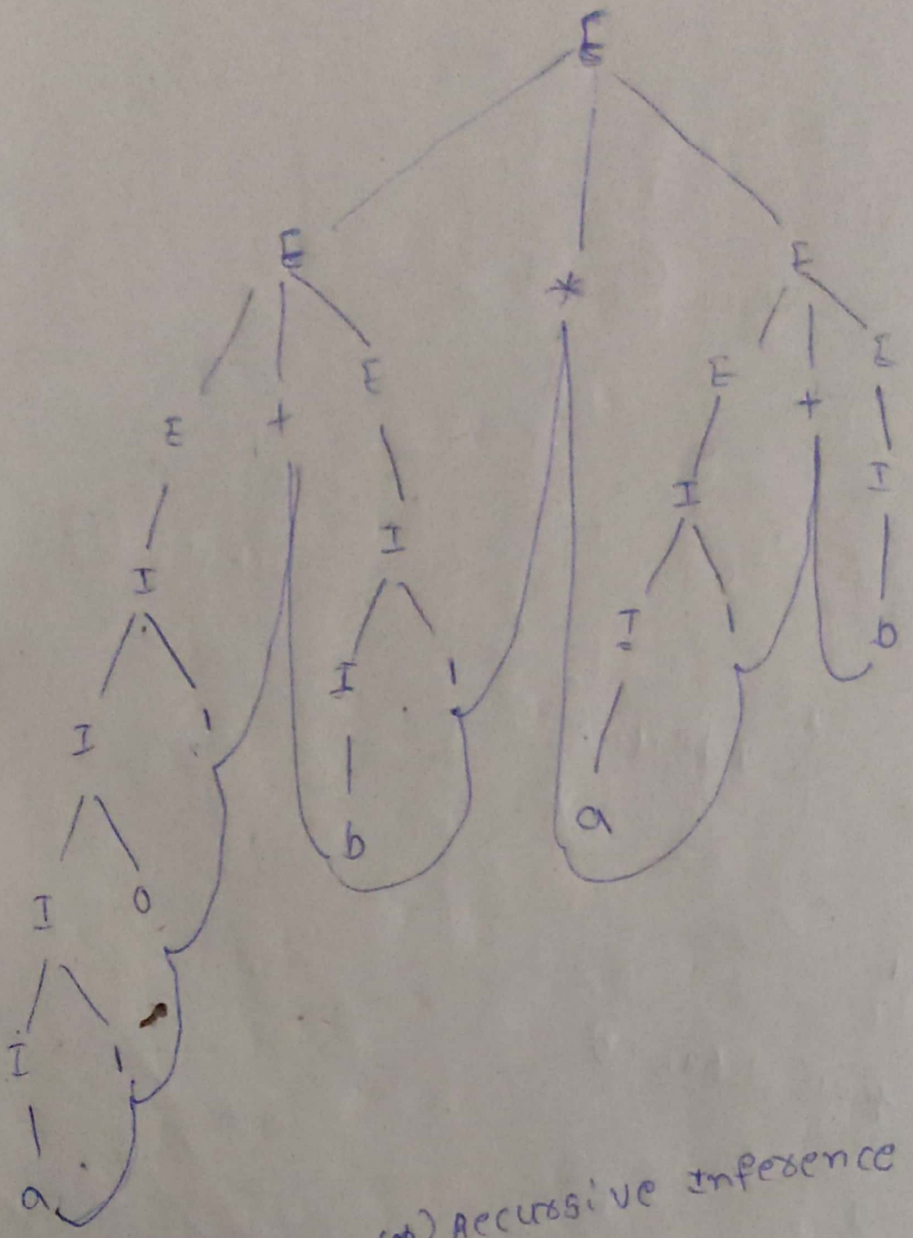
Expression grammar

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

$\left.\begin{matrix} 5.\ I \to a \\ 6.\ I \to b \\ 7.\ I \to Ia \\ 8.\ I \to Ib \\ 9.\ I \to I0 \\ 10.\ I \to I1 \end{matrix}\right\}$ identifiers

$(a101 + b1) * (a1 + b)$





(i) Rule of Inference (or) Recursive inference

(ii) Derivation

(i) Rules from body to head we take strings known to be in the language of each of the variables of the body concartinate them ~~with~~ in the proper order with any terminals appearing in the body and infer that

$(a101 + b1) * (a1 + b)$

String infered	for language of	production used
		5
(i) a	I	10
(ii) a1	I	9
(iii) a10	I	10
(iv) a101	I	5
(v) b	I	10
(vi) b1	E	1
(vii) a101	E	1
(viii) b1	E	2
(ix) a101 + b1	E	4
(x) (a101 + b1)	E	5
(xi) a1	E	1
~~(xii) a1~~	E	2
(xii) b	E	4
(xiii) a1 + b	E	3
(xiv) (a1 + b)	E	
(xv) (a101 + b1) * (a1 + b)		

Strings used	
—	(ii) The process of deriving strings
(i)	by applying productions from head
(ii)	to body requires the definition of
(iii)	new relation symbol (⟹).
—	Suppose $G = (V, T, P, S)$ is a context
(v)	free grammer.
(iv)	Let $\alpha A \beta$ be a string of terminals
(vi)	and variables, with A a variable.
(vii) (viii)	i.e α, β are strings in $(V \cup T)^*$
(ix)	and A is in V. Let $A \to \gamma$ be a
(ii)	production. of G then we say
(v)	$\alpha A \beta$ derives $\alpha \gamma \beta$ by using
xi, xii	grammar G.
xiii	$\alpha A \beta \Rightarrow \alpha \gamma \beta$ in one derivation
x, xiv	step replaces any variable
	any where in the string

Scanned by CamScanner

in the body of any one of its production

$(a1 0 1 + b 1) * (a 1 + b)$

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

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

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

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

$E \underset{lm}{\Rightarrow} (I1 + E) * E \quad (I \to I1)$

$E \underset{lm}{\Rightarrow} (I01 + E) * E \quad (I \to I0)$

$E \underset{lm}{\Rightarrow} (I101 + E) * E \quad (I \to I1)$

$E \underset{lm}{\Rightarrow} (a101 + E) * E \quad (I \to a)$

$E \underset{lm}{\Rightarrow} (a101 + I) * E \quad (E \to I)$

$E \underset{lm}{\Rightarrow} (a101 + I1) * E \quad (I \to I1)$

$E \underset{lm}{\Rightarrow} (a101 + b1) * E \quad (I \to b)$

$E \underset{lm}{\Rightarrow} (a101 + b1) * (E) \quad (E \to (E))$

$E \underset{lm}{\Rightarrow} (a101 + b1) * (E + E) \quad (E \to E + E)$

$E \underset{lm}{\Rightarrow} (a101 + b1) * (I + E) \quad (E \to I)$

$E \underset{lm}{\Rightarrow} (a101 + b1) * (I1 + E) \quad (I \to I1)$

$E \underset{lm}{\Rightarrow} (a101 + b1) * (a1 + E) \quad (I \to a)$

$E \underset{lm}{\Rightarrow} (a101 + b1) * (a1 + I) \quad (E \to I)$

$E \underset{lm}{\Rightarrow} (a101 + b1) * (a1 + b) \quad (I \to b)$

$E \underset{lm}{\Rightarrow} (a101 + b1) * (a1 + b)$

$$\boxed{E \overset{*}{\underset{lm}{\Rightarrow}} (a101 + b1) * (a1 + b)}$$

$S \rightarrow \epsilon / aS$ → context free grammar for any number of a's.

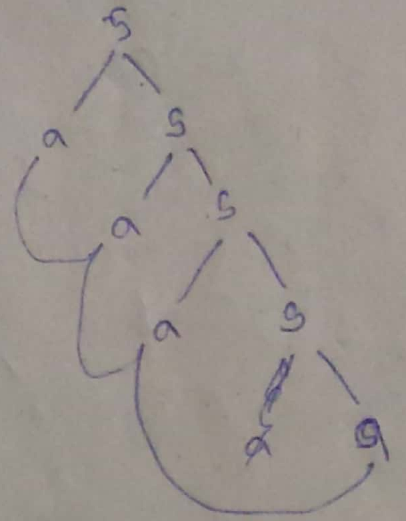
context free grammar for any numbers of a's and b's

$S \rightarrow \epsilon / aS / bS$

Atleast one a

$S \rightarrow a / aS$

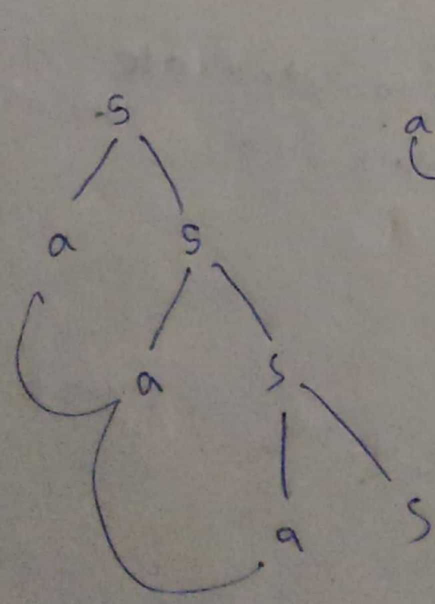
aaaa

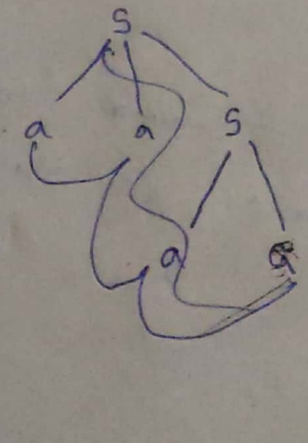


Atleast 2 a's $\Sigma = \{a\}$

$S \rightarrow aa / aaS$

aaaa

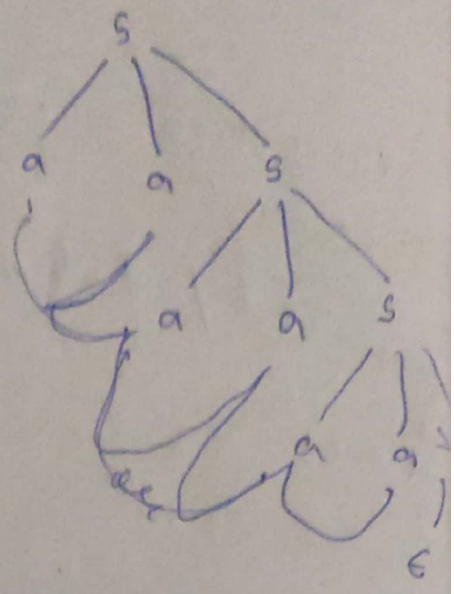




ven numbers of a's $\Sigma = \{a\}$

$S \to \epsilon / aaS$

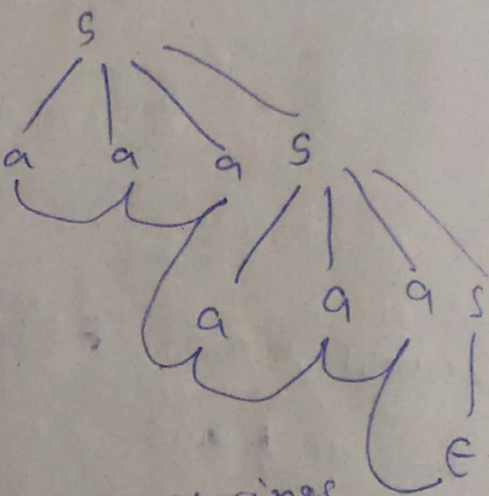
aaaaaa



Multiples of 3 $\Sigma = \{a\}$

aaaaaa

$S \to \epsilon / aaaS$



Obtain a grammar to generate strings of a's and b's such that string length is multiple of 3.

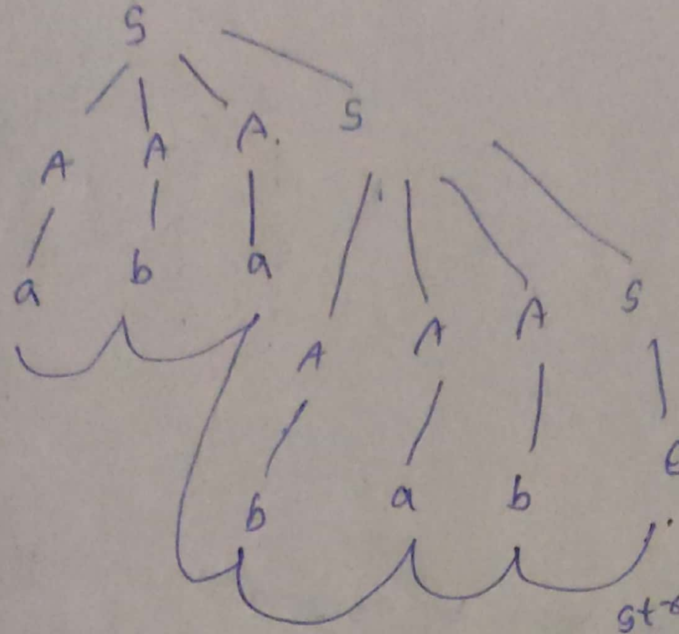
$\Sigma = \{a, b\}$

aaa, bbb, aab, aba, abb, bab, aaabab ...

$S \to \epsilon / aaaS / bbbS$

$S \to \epsilon / AAAS$ ababab

$A \to a/b$



obtain a grammer to generate string consisting of any number of a's & b's with atleast one a or one b.

$S \to a/b/aS/bS$

$S \to B/AS$

$B \to a/b$

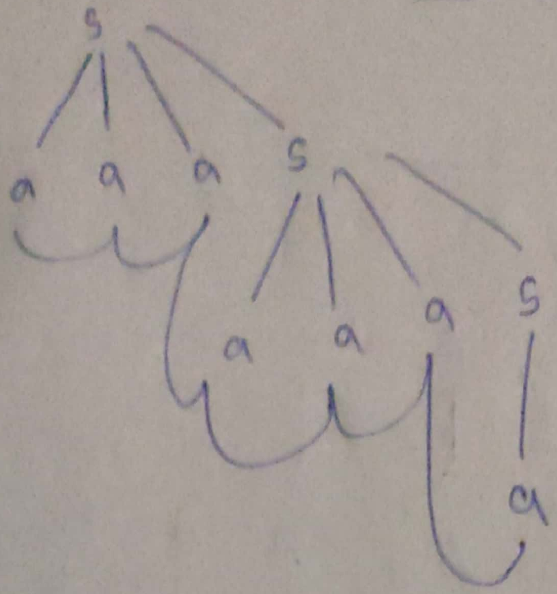
$A \to a/b$

obtain a grammer to accept the following language.

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

$S \to a/aa/aaaS$

aaaaaaa



S -> A / AA / AAAS

A -> a / b.

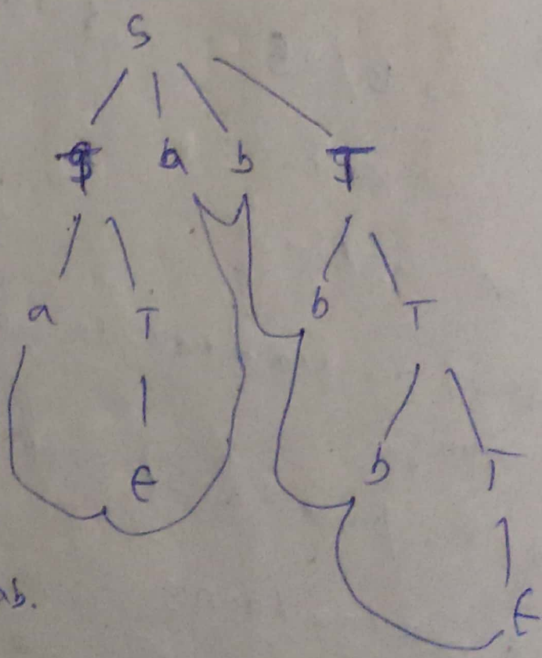
obtain a grammar to generate strings of a's and b's having a substring ab.

~~S -> ab / Sab / abS / SabS~~

ab
a ab
a ababa
a abbb

S -> T ab T

T -> ϵ / aT / bT



S -> abT   starting with ab

T -> ϵ / aT / bT

S -> T ab.   ending with ab.

T -> ϵ / aT / bT

obtain a grammar to generate the following language :

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

aa / aaST

S → aa / aaST

aa

aabaa

aabbbbaaaa

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

Sol

S → ε / bS / TaTaT / TaTaTS

T → ε / bS

(or)

S → ε / bS / SaSaS

Leftmost derivation & Rightmost derivation :-

At each step replace the left most variable by one of its production bodies such derivation is called a leftmost derivation. we indicate leftmost derivation by the using the ~~notation~~ relations $\underset{lm}{\Rightarrow}$ and $\overset{*}{\underset{lm}{\Rightarrow}}$ for one or more steps respectively.

At each step the right most variable is replaced by one of its bodies use the symbols right most derivation by using the relations $\underset{rm}{\Rightarrow}$ and $\overset{*}{\underset{rm}{\Rightarrow}}$ to indicate on or more right-most derivation steps respectively.

Scanned by CamScanner

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

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

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

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

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

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

$E \underset{rm}{\Rightarrow} E * (I1 + b) \quad (I \rightarrow I1)$

$E \underset{rm}{\Rightarrow} E * (a1 + b) \quad (I \rightarrow a)$

$E \underset{rm}{\Rightarrow} (E) * (a1 + b) \quad (E \rightarrow (E))$

$E \underset{rm}{\Rightarrow} (E + E) * (a1 + b) \quad (E \rightarrow E + E)$

$E \underset{rm}{\Rightarrow} (E + I) * (a1 + b) \quad (E \rightarrow I)$

$E \underset{rm}{\Rightarrow} (E + I1) * (a1 + b) \quad (I \rightarrow I1)$

$E \underset{rm}{\Rightarrow} (E + b1) * (a1 + b) \quad (I \rightarrow b)$

$E \underset{rm}{\Rightarrow} (I + b1) * (a1 + b) \quad (E \rightarrow I)$

$E \underset{rm}{\Rightarrow} (I1 + b1) * (a1 + b) \quad (I \rightarrow I1)$

$E \underset{rm}{\Rightarrow} (I01 + b1) * (a1 + b) \quad (I \rightarrow I0)$

$E \underset{rm}{\Rightarrow} (I101 + b1) * (a1 + b) \quad (I \rightarrow I1)$

$E \underset{rm}{\Rightarrow} (a101 + b1) * (a1 + b) \quad (I \rightarrow a)$

$\therefore E \overset{*}{\Rightarrow} (a101$

Language of CFG :-

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

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

If a language L is the language of some CFG, then L is said to be context free language.

$P \rightarrow \epsilon / a / b / apa / bpb$

baa bab

$P \underset{lm}{\Rightarrow} bpb$ $(P \rightarrow bpb)$

$P \underset{lm}{\Rightarrow} bapab$ $(P \rightarrow apa)$

$P \underset{lm}{\Rightarrow} baapaab$ $(P \rightarrow apa)$

~~P ⇒ baapaab~~

$P \underset{lm}{\Rightarrow} baabaab$ $(P \rightarrow b)$

not accepted

sentential forms :-

derivations from the start symbol produce strings if G(V, T, P, S) is a CFG then any string α in $(V \cup T)^*$ such that the $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$, then α is a left sentential form. if $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha$, then α is a right sentential form. The language L(G) is those sentential forms that are in T^* i.e they consists solely of terminals only

parse trees :-

constructing parse tree :-

Grammar G (V, T, P, S). The parse tree for G are trees with the following conditions.

(i) Each interior node is labled by a variable in V.

(ii) Each leaf is labled by either a variable, a terminal or ϵ.
If the leaf is labled ϵ, then it must be the only child of its parent.

(iii) If an interior node is labled A and its children are labled $x_1, x_2 \ldots x_k$ respectively from the left. then $A \rightarrow x_1 x_2 \ldots x_k$ is a production in P

yeild of a parse tree :-

The leaves of any parse tree and concatinate them from the left we get a string called the yeild of the tree which is always a string that is derived from a root variable.
The yeild is the terminal string i.e all leaves are labled either with a terminal or ϵ.

(ii) The root is labled by the sta...
(iii) The set of yeilds of those parse trees having the start symbol at the root and the terminal as the yeild describes the language of a grammar.

$S \to icts / ictses / a$

$c \to b$

ibtibtaea

parse tree

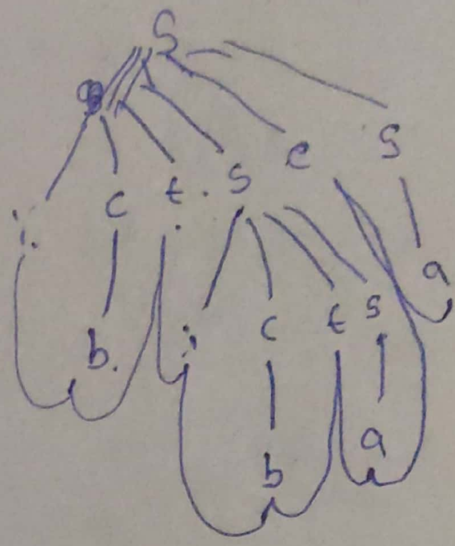
$S \Rightarrow icts$

$\Rightarrow ibts$

$\Rightarrow ibticts$

$\Rightarrow ibtibts$

Left



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

$\underset{lm}{\Rightarrow} ibtses$

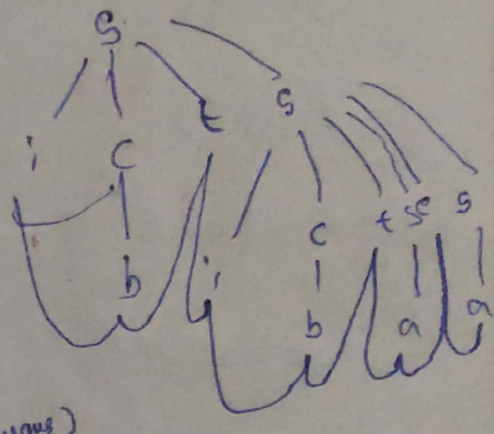
$\underset{lm}{\Rightarrow} ibtictses$

$\underset{lm}{\Rightarrow} ibtibtses$

$\underset{lm}{\Rightarrow} ibtibtaea$

$\boxed{S \overset{*}{\underset{lm}{\Rightarrow}} ibtibtaea}$

Left



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

$\underset{rm}{\Rightarrow} ictsea$

$\underset{rm}{\Rightarrow} icticts ea$

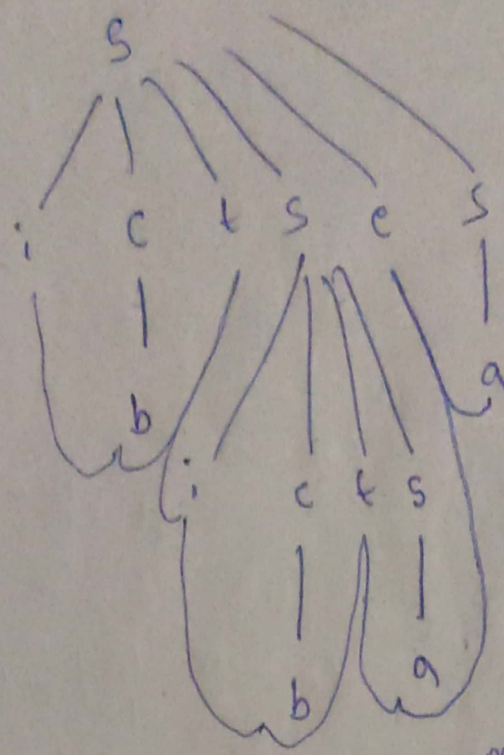
$\underset{rm}{\Rightarrow} icticta ea$

$\underset{rm}{\Rightarrow} ictibtaea$

$\underset{rm}{\Rightarrow} ibtibtaea$

(ambiguous)

$\boxed{S \overset{*}{\underset{rm}{\Rightarrow}} ibtibtaea}$



Right most derivation parse tree

$S \to aB \,/\, bA$

$A \to aS \,/\, bAA \,/\, a$

$B \to bS \,/\, aBB \,/\, b$

aabbab

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

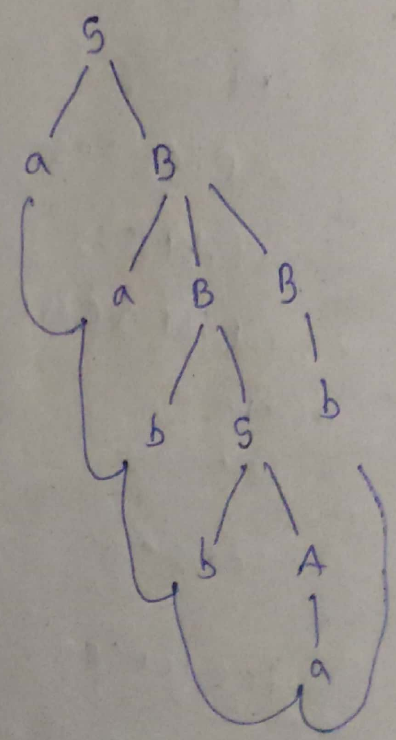
$\underset{lm}{\Rightarrow} aaBB$

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

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

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

$\boxed{S \overset{*}{\underset{lm}{\Rightarrow}} aabbab}$



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

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

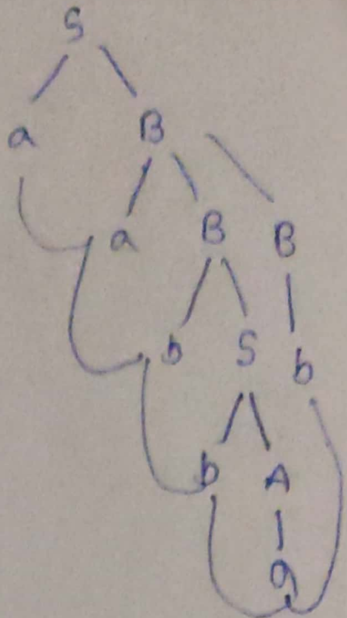
$\underset{rm}{\Rightarrow} aaBb$

$\underset{rm}{\Rightarrow} aabSb$

$\underset{rm}{\Rightarrow} aabbAb$

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

$\boxed{S \overset{*}{\underset{rm}{\Rightarrow}} aabbab}$



(non ambiguous)

$S \rightarrow aB / bA$

$A \rightarrow aS / bAA / a$

$B \rightarrow bS / aBB / b$

aaabbabbba

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

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

$\underset{lm}{\Rightarrow} aaaBBB$

$\underset{lm}{\Rightarrow} aaabBB$

$\underset{lm}{\Rightarrow} aaabbB$

$\underset{lm}{\Rightarrow} aaabbaBB$

$\underset{lm}{\Rightarrow} aaabbabB$

$\underset{lm}{\Rightarrow} aaabbabbS$

$\underset{lm}{\Rightarrow} aaabbabbbA$

$\underset{lm}{\Rightarrow} aaabbabbba$

$\boxed{S \overset{*}{\underset{lm}{\Rightarrow}} aaabbabbba}$

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

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

$\underset{lm}{\Rightarrow} aaaBBB$

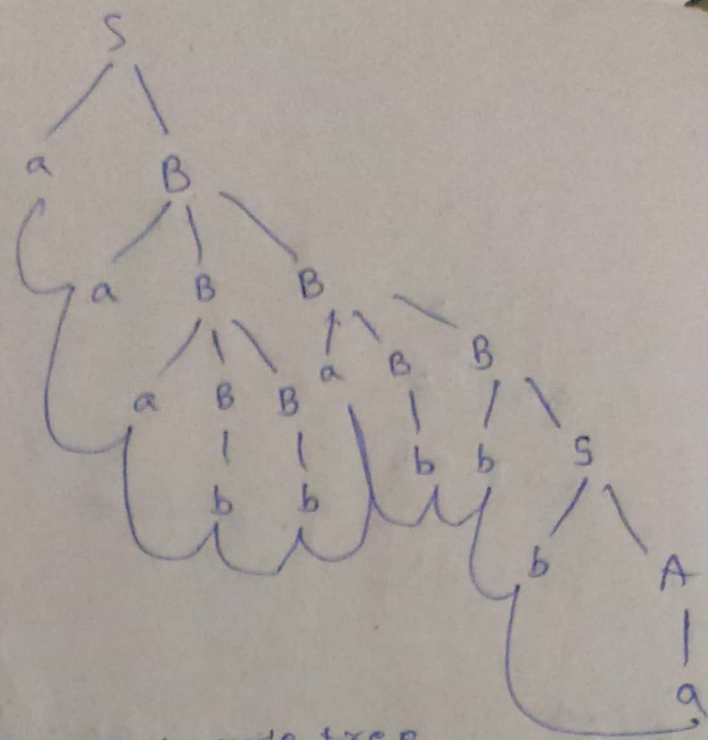
$\underset{lm}{\Rightarrow} aaabBB$

$\underset{lm}{\Rightarrow} aaabbBB$

$\underset{lm}{\Rightarrow} aaabbaBB$

$\underset{lm}{\Rightarrow} aaabba\ bSB$

$\underset{lm}{\Rightarrow} aaabbabbAB$



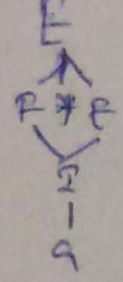
Left most parse tree

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

$\underset{rm}{\Rightarrow} abS$

Ambiguity in grammars:-

A context free grammar $G = (V, T, P, S)$ is ambiguous if there is atleast one string w in T^* for which we can find 2 different parse trees each with root labeled S and yeild w. if each string has atmost one parse tree in the grammar then the grammar is unambiguous.

$a * a$  (lm am)

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

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

E

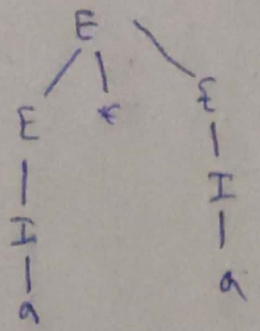
$E \underset{lm}{\Rightarrow} E * E$

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

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

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

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



parse tree for lm

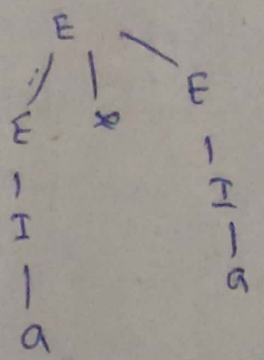
$E \underset{rm}{\Rightarrow} E * E$

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

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

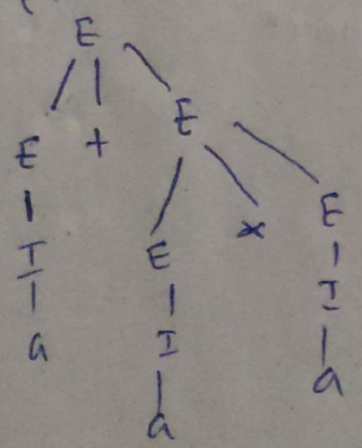
$E \underset{rm}{\Rightarrow} I * a$

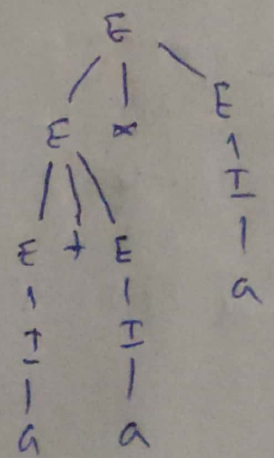
$\underset{rm}{\Rightarrow} a * a$



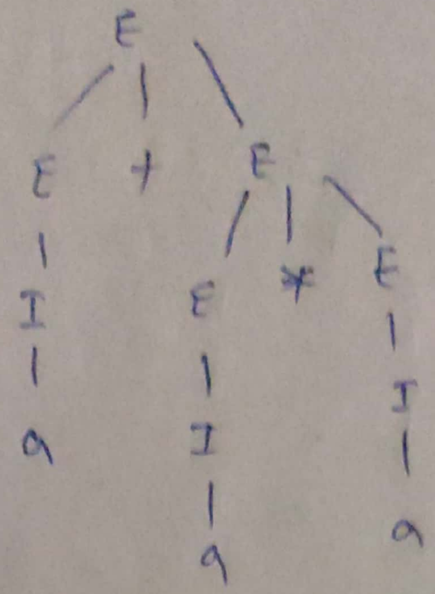
parse tree for rm

$a + a * a$ (Ambiguity)





E+(E*E)



Left most derivation is one way to represent ambiguity. For each grammar G(V,T,P,S) and string w has two distinct parse trees if and only if w has two distinct left most derivations from S.

Inherent ambiguity:-

A CFL L is said to be inherently ambiguoues if all its grammars are ambiguoues. If even one grammar for L is un-ambiguoues then L is an un ambiguoues language.

Applications of CFG:-

1) parses
2) YACC parser generator
3) Mark up Languages
4) XML and document type definitions.

aaabbabbba

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

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

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

$\underset{rm}{\Rightarrow} aaBbbA$

$\underset{rm}{\Rightarrow} aaBbba$

$\underset{rm}{\Rightarrow} aabSbba$

$\underset{rm}{\Rightarrow}$

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

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

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

$\Rightarrow aaBbbA$

aaaBBbba

aaaB bbba

aaabSbbba

aaabbAbbba

aaabbabbba

aaabbabbba

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

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

$\underset{rm}{\Rightarrow} aaBb$

$\underset{rm}{\Rightarrow}$

obtain CFG for $L = \{a^n b^n : n \geq 0\}$

$S \to \cancel{\epsilon b} \mid ab \mid$

ϵ

ab

$aabb$

$aaaabbbb$

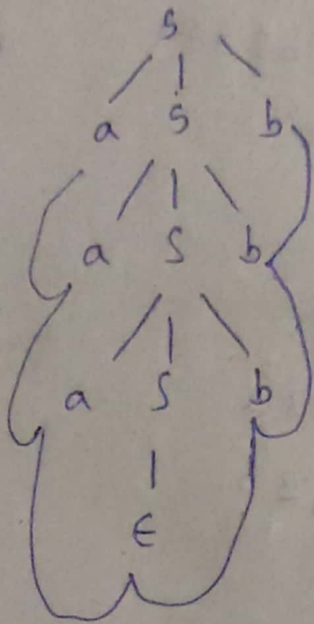
$S \to \epsilon \mid aAbB$

$A \to \epsilon / a / aA$

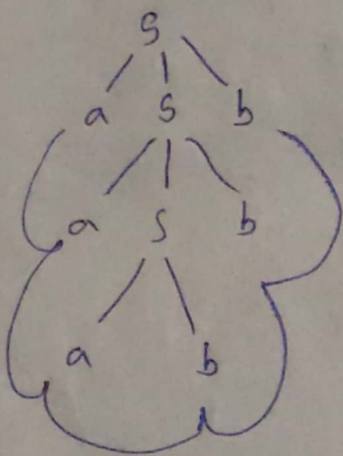
$B \to \epsilon / b / bB$

or

$S \to \epsilon / aSb$



$S \to ab / aSb$ for $n \geq 1$



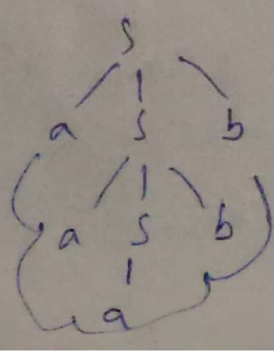
$L = \{a^{n+1} b^n \;\; n \geq 0\}$

a

aab

$aaabb$

$S \to a / aSb$



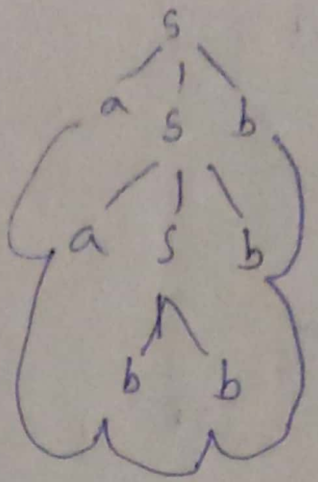
$L = \{a^n b^{n+2} \quad n \geq 0\}$

$S \rightarrow bb / aSb$

bb

abbb

aabbbb



$L = \{a^n b^{2n} \quad n \geq 0\}$

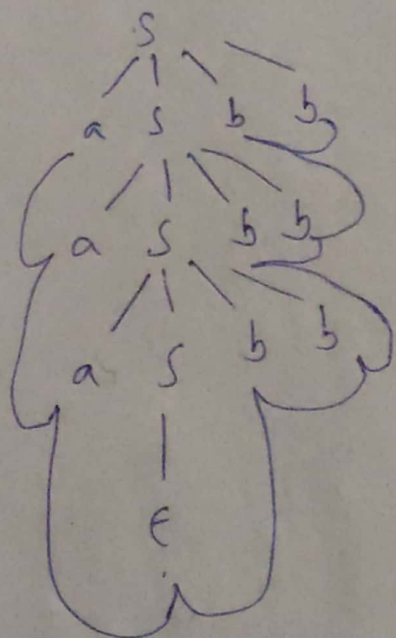
$S \rightarrow \epsilon / aSbb$

ϵ

abb

aabbbb

aaabbbbbb

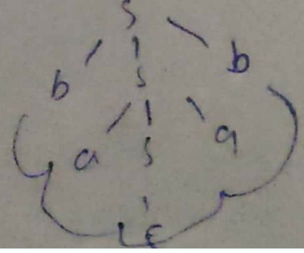


obtain context FG for palindrome $\Sigma = \{a, b\}$

$S \rightarrow \epsilon / aSa / bSb / a / b$

babaabab

baab



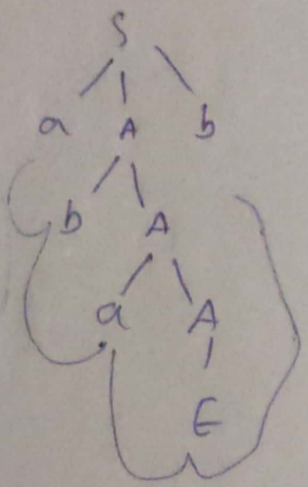
obtain CFG non palindromes over $\Sigma = \{a, b\}$

$S \rightarrow aAb / bAa$

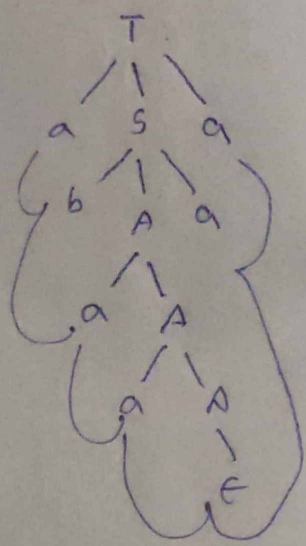
$A \rightarrow \epsilon / aA / abA$

$T \rightarrow aSa / bSb$

abab



abaaaa



$L = \{0^m 1^m 2^n \mid m \geq 1 \text{ and } n \geq 0\}$

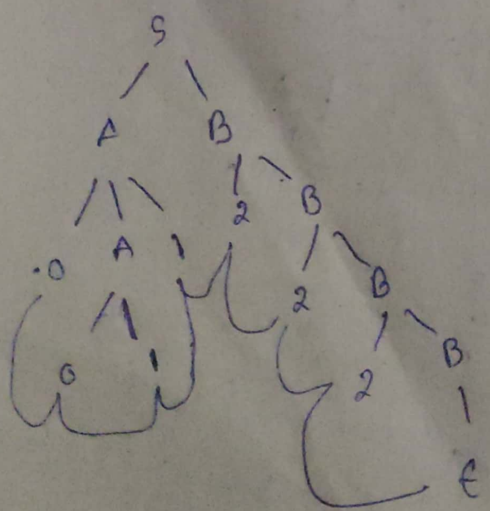
01

$S \rightarrow AB$

$A \rightarrow 01 / 0A1$

$B \rightarrow \epsilon / 2B$

0011222



$L = \{ w \mid n_a(w) = n_b(w) \}$ $\quad \Sigma = \{a, b\}$

$S \to \epsilon / aSb / bSa / SS$

ϵ

ab

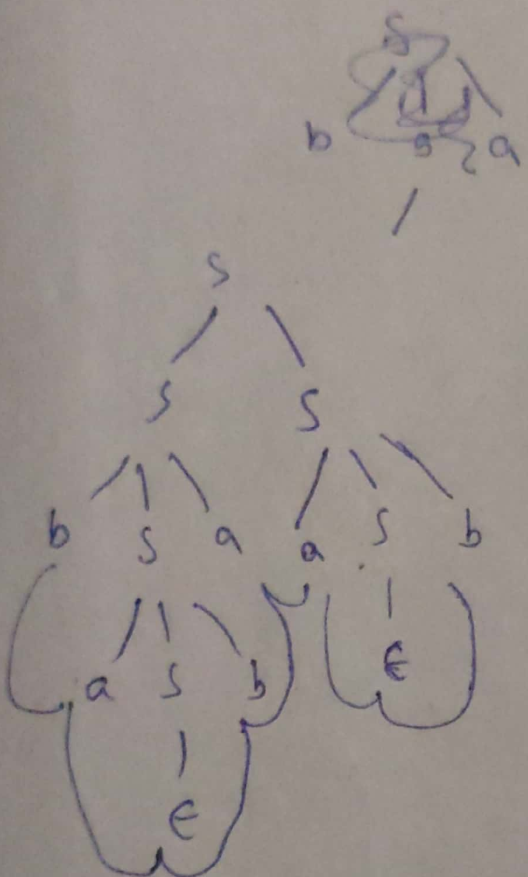
abba

baba

baab

abab

b a b a a b



balanced parenthesis $\Sigma = \{ \text{'('}, \text{'\{'}, \text{'['}, \text{']'}, \text{'\}'}, \text{')'} \}$.

$S \to \epsilon / [S] / (S) / \{S\} / SS$

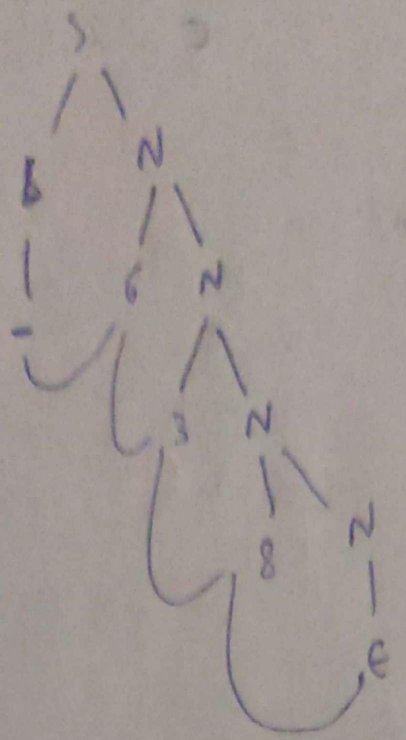
obtain CFG to generate integers

$S \to LN$

$L \to \epsilon / - / +$

$N \to \epsilon / 0N / 1N / 2N / 3N / 4N / 5N / 6N / 7N / 9N$

- 638



S -> SN | UN

SN -> T N

UN -> N

D -> 0|1|2|3|... 9

N -> D/ ND

T -> +/-

identifier CFG

S -> l / lN

N -> ε / lN / dN

l -> ε / al / bl / ... zl

Obtain a CFG to generate all strings with exactly one a $\Sigma = \{a, b\}$

Obtain a CFG to generate a language of strings of 0's and 1's having substring 000.

Obtain CFG for language $w \in \{0,1\}^*$ and of even length.

Language is $w \in \{0,1\}^*$ w contains atleast 3 0's.

$w \in \{0,1\}^*$ length of w is odd.

$S \rightarrow a / ab\cancel{N} / ba\cancel{N}$

$S \rightarrow \cancel{a} aN / Na /$

$N \rightarrow \cancel{\epsilon / bN}$

$S \rightarrow$

a

ab

bab

bbabb

x1 x1x1x

1) $S \to (L) / a$

$L \to L, S / S$

2) $S \to AA$

$A \to aA$

$A \to b$

3) $E \to +EE / *EE / -EE / x / y$

4) $S \to BAAB$

$B \to 0A2 / 2A0 / \epsilon$

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

5) $S \to AB / BC$

$A \to BB / 0$

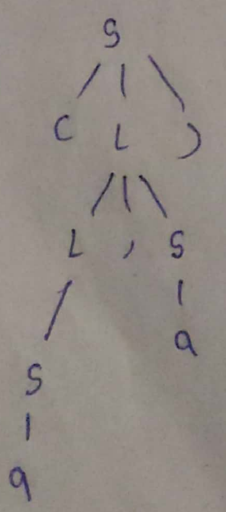
$B \to BA / 1$

$C \to AC / AA / 0$

1) $S \Rightarrow$ a

(L)

(L,S)



a

ab

ba

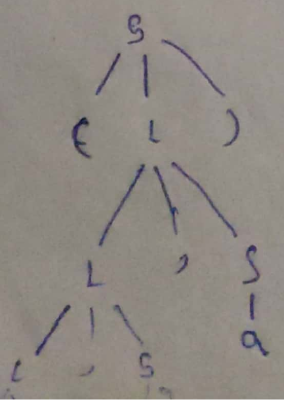
abb

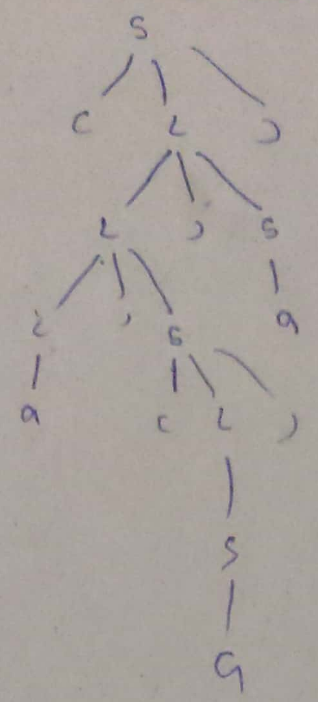
bab

((a,a),a)

(~~S~~)

(a,a)





(a, (a), a)

(a, a, a)

$$S \underset{lm}{\Rightarrow} (L)$$

$$\underset{lm}{\Rightarrow} (L, S)$$

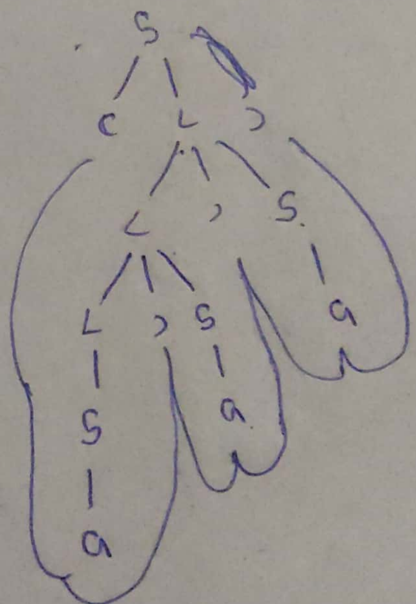
$$\underset{lm}{\Rightarrow} (L, S, S)$$

$$\underset{lm}{\Rightarrow} (S, S, S)$$

$$\underset{lm}{\Rightarrow} (a, S, S)$$

$$\underset{lm}{\Rightarrow} (a, a, S)$$

$$\underset{lm}{\Rightarrow} (a, a, a)$$



$S \underset{rm}{\Rightarrow} (L)$

$\underset{rm}{\Rightarrow} (L, S)$

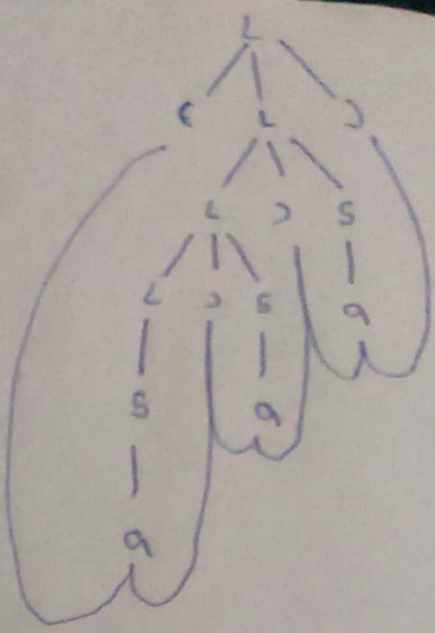
$\underset{rm}{\Rightarrow} (L, a)$

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

$\underset{rm}{\Rightarrow} (L, a, a)$

$\underset{rm}{\Rightarrow} (S, a, a)$

$\underset{rm}{\Rightarrow} (a, a, a)$



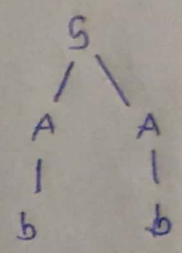
non - ambiguous :

(2) $S \rightarrow AA$

$A \rightarrow aA$

$A \rightarrow b$

$\boxed{\text{atmost 2 b's}}$



a A A

aaaaA

aaaaaaa

aaabaab

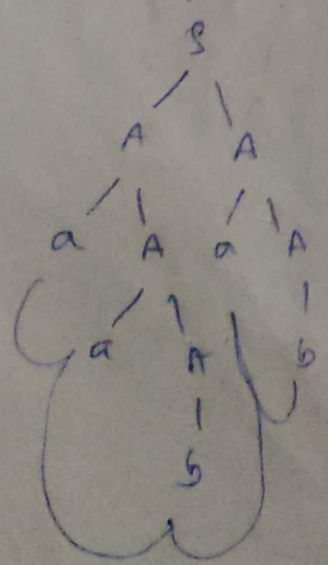
bb

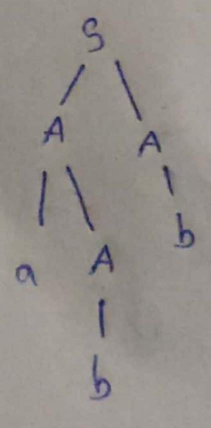
ab

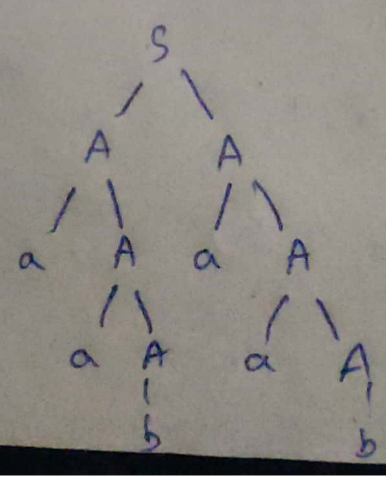
abb

aabaab

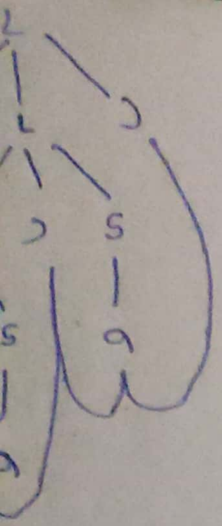
'aabab







aabaab



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

$\underset{lm}{\Rightarrow} aAA$

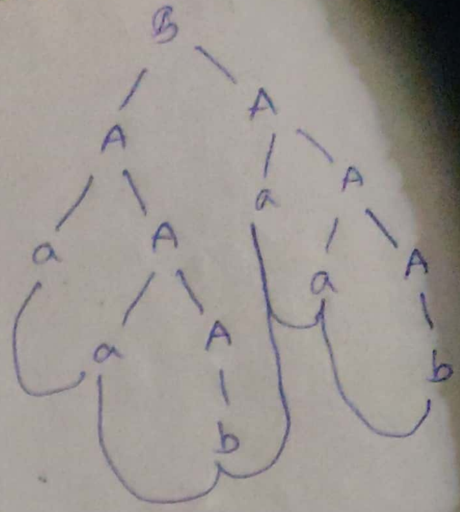
$\underset{lm}{\Rightarrow} aaAA$

$\underset{lm}{\Rightarrow} aabA$

$\underset{lm}{\Rightarrow} aabaA$

$\underset{lm}{\Rightarrow} aabaaA$

$\underset{lm}{\Rightarrow} aabaab$



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

$\underset{rm}{\Rightarrow} AaA$

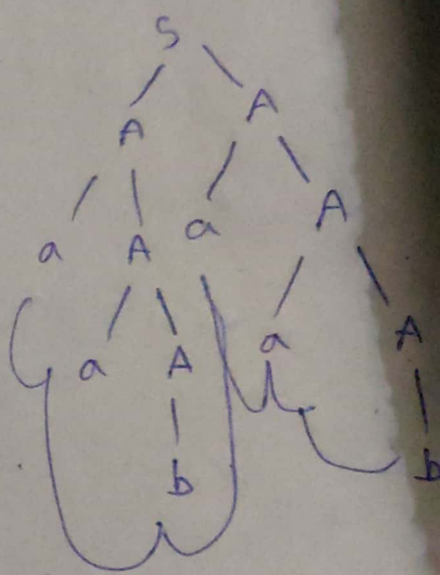
$\underset{rm}{\Rightarrow} AaaA$

$\underset{rm}{\Rightarrow} Aaab$

$\underset{rm}{\Rightarrow} aAaab$

$\underset{rm}{\Rightarrow} aaAaab$

$\underset{rm}{\Rightarrow} aabaab$



a A A

a a A a A

a a a A a a A

a a a b a a b

b b

a b

a b b

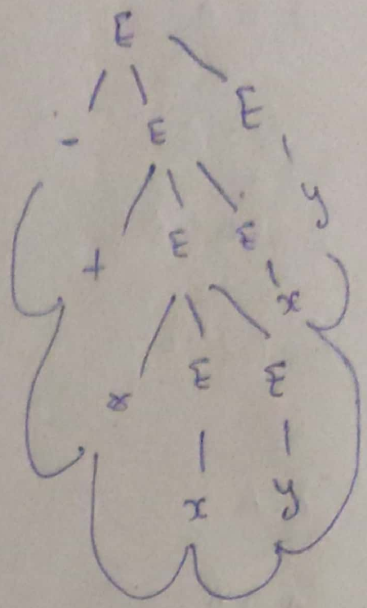
non ambiguous

ab

③ E -> + E E / * E E / - E E / x / y (post-fix expression)

~~- xy + xy * xy~~

- + * x y x y



$E \underset{lm}{\Rightarrow} - E E$

$\underset{lm}{\Rightarrow} - + E E E$

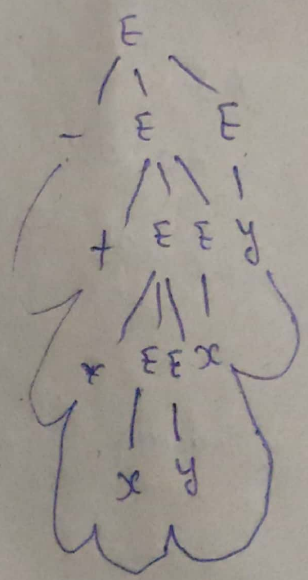
$\underset{lm}{\Rightarrow} - + * E E E E$

$\underset{lm}{\Rightarrow} - + * x E E E$

$\underset{lm}{\Rightarrow} + - * x y E E$

$\underset{lm}{\Rightarrow} + - * x y x E$

$\underset{lm}{\Rightarrow} + - * x y x y$



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

$\underset{rm}{\Rightarrow} - E y$

$\underset{rm}{\Rightarrow} - + E E y$

$\underset{rm}{\Rightarrow} - + E x y$

$\underset{rm}{\Rightarrow} - + * E E x y$

$\underset{rm}{\Rightarrow} - + * E y x y$

$\underset{rm}{\Rightarrow} - + * x y x y$

non ambiguous

(4) S → BAAB ε

B → 0A2 / 2A0 / ε

A → AB / 1B / ε

B A A

ε

A A

AB AB

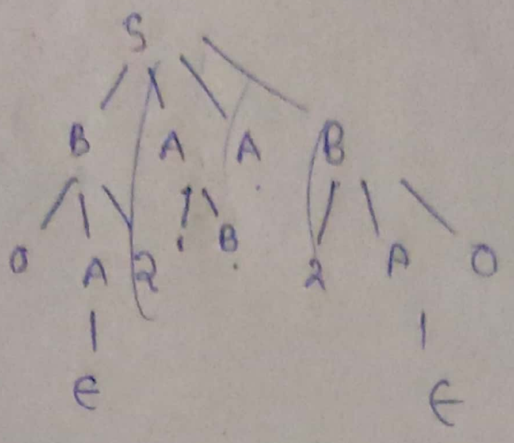
1B 1B

1 0 A 2 0 A 2

0 2

0 2 1 2

0 2 1

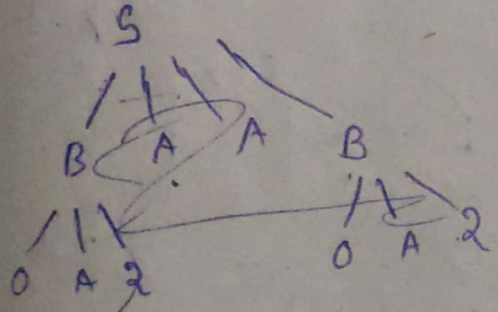


0 A2 AB AB 2A0

0 2 20

$\boxed{0220}$

0 2 1 1 2 0



S ⇒ lm

B A A B

0 A 2 A B 1B 2 A 0

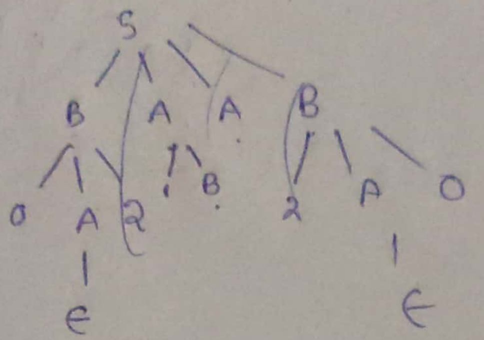
ε ε ε ε ε

$\boxed{0\ 2\ 1\ 2\ 0}$

(4) $S \rightarrow BAAB$

$B \rightarrow 0A2 \;/\; 2A0 \;/\; \epsilon$

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



B A A B

ε ε

A A

AB AB

1B 1B B

1 0A2 0A2

0 2 1 2 1 2 0

0 2 1 2 1 2 0

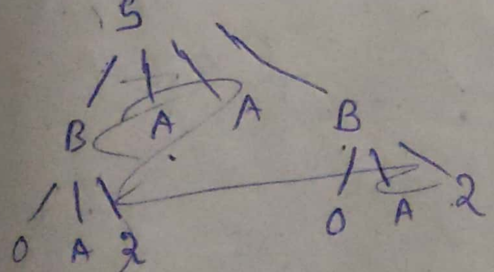
0 A2 AB AB 2A0

20

02

$\boxed{0220}$

021 120



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

B A A B

0A2 AB 1B 2A0

ε ε ε ε ε

$\boxed{02120}$

02120

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

$\underset{lm}{\Rightarrow} 0A2AAB$

$\underset{lm}{\Rightarrow} 02AAB$

$\underset{lm}{\Rightarrow} 02ABAB$

$\underset{lm}{\Rightarrow} 02BAB$

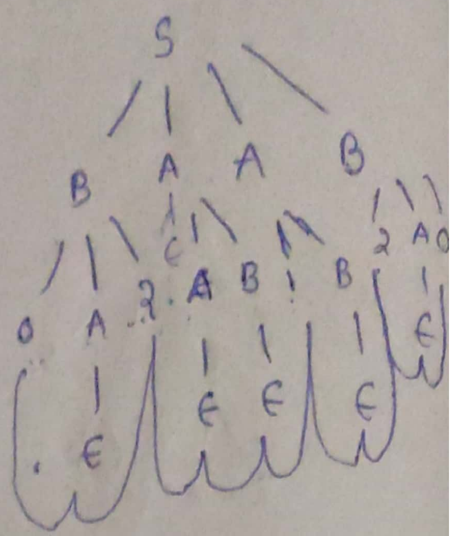
$\underset{lm}{\Rightarrow} 02AB$

$\underset{lm}{\Rightarrow} 021BB$

$\underset{lm}{\Rightarrow} 021B$

$\underset{lm}{\Rightarrow} 0212A0$

$\underset{lm}{\Rightarrow} 02120$



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

$\underset{rm}{\Rightarrow} BAA2A0$

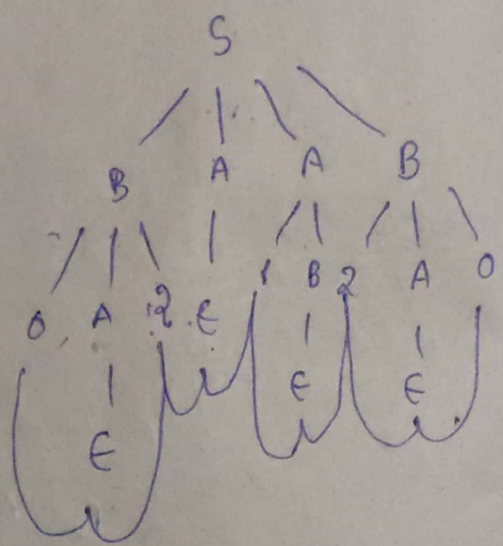
$\underset{rm}{\Rightarrow} BAA20$

$\underset{rm}{\Rightarrow} BA1B20$

$\underset{rm}{\Rightarrow} BA120$

$\underset{rm}{\Rightarrow} B120$

$\underset{rm}{\Rightarrow} 0A2120 \underset{rm}{\Rightarrow} 02120$



Ambiguous

A → BB / 0

B → BA / 1

C → AC / AA / 0

AB

BBBA

1110

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

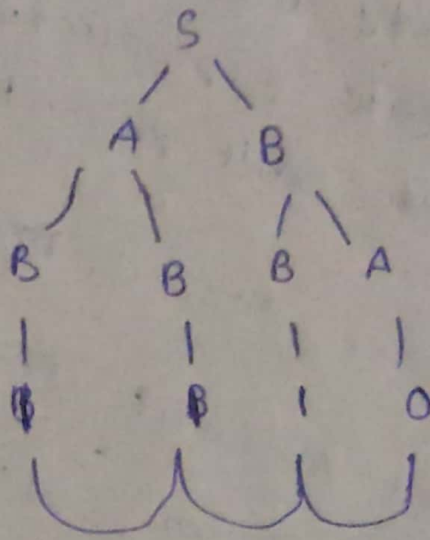
$\underset{lm}{\Rightarrow} BBB$

$\underset{lm}{\Rightarrow} 1BB$

$\underset{lm}{\Rightarrow} 11B$

$\underset{lm}{\Rightarrow} 11BA$

$\underset{lm}{\Rightarrow} 1110$



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

$\underset{rm}{\Rightarrow} ABA$

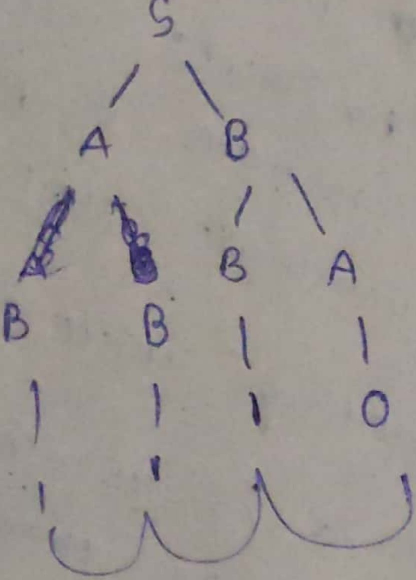
$\underset{rm}{\Rightarrow} AB0$

$\underset{rm}{\Rightarrow} A10$

$\underset{rm}{\Rightarrow} BB10$

$\underset{rm}{\Rightarrow} B110$

rm



non - ambiguous

- CNF Normal Form (chomsky Normal Form)
- pumping Lema
- clousure properties
- Decision properties

(i) eliminate useless symbols < non generating symbols / non reachable symbols
eliminate ε productions
eliminate unit production

Normal forms for CFG :-

(1) chomsky Normal Form :- All productions are of the form $A \to BC$ (or) $A \to a$.

the prilimenary simplifications for CNF are
eliminate useless symbols
eliminate ε productions
eliminate unit productions.

(i) eliminating useless symbols :-

A symbol x is useful for a grammar G = (V, T, P, S) if there is some derivation of the form $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$, where w is in T^*. x may be in either V or T and the sentential form $\alpha X \beta$ might be the first or last in the derivation.

if x is not useful then it is called as useless symbol.

of eliminating useless symbols from a grammar ~~will no the lang~~ will not change the language generated.

Eliminating useless symbols begins by identifying 2 things.

(i) we say x is generating if $x \overset{*}{\Rightarrow} w$ for some terminal string w.

Note:- Every terminal string is generating which since w can be that terminal itself - which is derived by zero step.

(ii) we say x is reachable if there is a derivation $S \overset{*}{\Rightarrow} \alpha x \beta$ for some α and β.

Generating:-

Let $G = (V, T, P, S)$ be a context free grammar and assume that $L(G) \neq \phi$. we

~~Let $G_1 = (V_1, T_1, P_1, S)$ be a grammar obtained by following steps.~~

~~(i) Let (V_2, T_2, P_2, S)~~

To computing the generating symbol of G we perform the following induction.

Basis:- Every symbol of T is generating.

Induction:- suppose there is a production $A \to \alpha$, and every symbol of α is already known to be generating. Then A is generating.

Note:- This rule includes the case where $\alpha = \epsilon$, all variables that have ϵ as production body are surely generating.

Reachable symbols:-

Basis: S is surely reachable.

Induction: suppose we have discovered that some variable A is reachable, then for all productions with A in the head, all the symbols of the bodies of those productions are also reachable.

Eliminate useless symbols from the grammar

S -> aA/ bB

A -> aA/a

B -> bB

D -> ab/EA

E -> aC/d

Terminals: T: {a, b, d}

A, D, E

A -> a
D -> ab
E -> d

S -> aA
A -> aA
D -> EA

A, D, E, S

S -> aA
A -> aA/a
D -> ab/EA
E -> d

Scanned by CamScanner

where
as
rating.

that some
productions
of
is a

ar

S is reachable = {S}

S -> aA - {S, a, A}

A -> aA
A -> a } {S, a, A}

S -> aa
A -> aA / a

Eliminate useless symbols from the grammar

S -> aA / a / Bb / cC

A -> aB

B -> a / Aa

C -> cCD

D -> ddd

Terminals : T = {a, b, c, d}

S -> a
B -> a } S, B, D
D -> ddd

S -> Bb
A -> aB } S, B, D, A

S -> aA } S, B, D, A
B -> Aa

S -> a / aA / Bb
A -> aB
B -> a / Aa
D -> ddd

S is rechable = {S}

S -> a/aA/Bb {S, a, A, B, b}

A -> aB
B -> a/Aa } {S, a, A, B, b}

S -> a/aA/Bb
A -> aB
B -> a/Aa

S -> aAa/aBC
A -> aS/bD
B -> aBa/b
C -> abb/DD
D -> aDa

Terminals: T: {a, b}

B -> b
C -> abb } B, C

~~B -> aBa~~

S -> aBC
B -> aBa } S, B, C

A -> aS } S, A, B, C

S -> aAa } S, A, B, C

S -> aAa / aBC
A -> aS
B -> aBa / b
C -> abb

Scanned by CamScanner

S is reachable = $\{S\}$

$S \to aAa / aBC$ $\{S, a, B, A, C\}$

$A \to aS$
$B \to aBa$
$C \to abb$ $\Big\}$ $\{S, a, B, A, C\}$

$$\boxed{\begin{array}{l} S \to aAa / aBC \\ A \to aS \\ B \to aBa \\ C \to abb \end{array}}$$

Eliminating ϵ - productions :-

Let $G = (V, T, P, S)$ be a CFG we can find all the nullable symbols of G by the following iterative algorithm.

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

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

So, we only have to consider productions with all variable bodies.

Construction of Grammar with out ϵ - productions;

$G = (V, T, P, S)$ be a context F G. Determine all the nullable symbols of G.

we construct a new grammar $G_1 = (V, T, P_1, S)$ whose set of productions P_1 is determined as follows.

For each production $A \to x_1, x_2 \cdots x_k$ of P where $k \geq 1$. suppose that m of the k x_i's are nullable symbols the new grammar G_1 will have 2^m versions of this production where the nullable x_i's in all possible combinations are present or absent. there is one exception.

If $m = k$ i.e all symbols are nullable then we do not include the case where all x_i's are absent.

Note:- If a production of the form $A \to \epsilon$ is in P. we do not place this production in P_1.

Eliminate ϵ-productions from the following grammar.

$S \to ABCa / bD$

$A \to BC / b$

$B \to b / \epsilon$

$C \to c / \epsilon$

$D \to d$

B, C are nullable ~~symbols~~ variables.

$2^2 = 4$

$S \to ABCa / ACa / ABa / Aa / bD$

$A \to BC / b$

$B \to b$

$C \to c$

$B \to \epsilon$
$C \to \epsilon$ } B, C are nullable variables

$A \to BC$ } A is nullable

A, B, C are nullable variables.

$S \to \underline{ABCa}$

$S \to ABCa / BCa / ACa / ABa / Ca / Aa / Ba / a$

$S \to bD$

$\cancel{AB} \to BC$

$A \to b$

$B \to b$

$C \to c$

$D \to d$

$S \to bD$

$A \to BC/B/C$

$A \to b$

$B \to b$

$C \to c$

$D \to d$

$S \to BAAB$
$A \to 0A2/2A0/\epsilon$
$B \to AB/1B/\epsilon$

$S \to ABC$
$A \to BC/a$
$B \to bAC/\epsilon$
$C \to cAB/\epsilon$

$B \to \epsilon$
$C \to \epsilon$ } B, C are nullable variables

$A \to BC$ } A, B, C are nullable

Unit productions:-

To eliminate unit productions we proceed as follows.

Given a $G(V,T,P,S)$, construct CFG $G_1 = (V_1, T, P_1, S)$

1. Find all the unit pairs of G.
2. For each unit pair (A, B) add to P_1 all the productions $A \to \alpha$ where $B \to \alpha$ is a non unit production in P.

Note:- A = B is possible in that way P_1 contains all non unit productions in P.

	non unit productions	unit productions
S -> AB	S -> AB	B -> C
A -> a	A -> a	C -> D
B -> C/b	B -> b	D -> E
C -> D	D -> bC	
D -> E/bC	E -> d/Ab	
E -> d/Ab		

E/Bd

B -> C -> D -> E

D -> E

⇓

D -> d/Ab

⇓

D -> d/Ab/bC

C -> D

C -> d/Ab/bC

B -> C

B -> d/Ab/bC

⇓

B -> d/Ab/bC/b

S -> AB

A -> a

B -> d / Ab / bc / b

C -> d / Ab / bc

D -> d / Ab / bc

S -> Aa / B / Ca

B -> aB / b

C -> Db / D

D -> E / d

E -> ab

non unit

S -> Aa / B / Ca

B -> aB / b

C -> Db

D -> d

E -> ab

unit

S -> D

C -> D

D -> E

C -> D -> E

S -> B

S -> aB / b

D -> E

⇓

D -> ab / d

C -> D

⇓

C -> Db / ab / d

$$\boxed{\begin{array}{l} S \to Aa / Ca / aB / b \\ B \to aB / b \\ C \to Db / ab / d \\ D \to ab / d \\ E \to ab \end{array}}$$

Eliminate unit-productions :

S -> A0/B

B -> A/11

A -> 0/12/B

~~Non~~ unit	non unit
S -> B	S -> A0
B -> A	B -> 11
A -> B	A -> 0/12

S -> B -> A

A -> 0/12/11

B -> 11/0/12

S -> A0/11/0/12

CNF :-

E productions

unit productions

useless symbols

~~S -> A0/BC~~

In CNF productions are A -> BC where A, B and C are variables

A -> a A is variable & a is terminal symbol

To convert into CNF first eliminate
ϵ - productions
unit productions and
useless symbols.
From the resulting grammar do the following task.
(a) Arrange that all bodies of length 2 or more consist only of variables.
(b) Break bodies of length 3 or more each into a cascade of productions each with a body consists of 2 variables.

construction of '(a)' case is as follows :-

For every terminal a that appear in a body of length 2 (or) more, create a new variable, say A, This variable has only one production of $A \to a$, Now we use A in place of terminal a every where a appears in a body of length 2 or more at this point every production has a body that is either a single terminal or at least two variables and no terminals

construction of '(b)' case is as follows :-

i.e we must break those productions $A \to B_1 B_2 \ldots B_k$ for $k \geq 3$ into a group of productions with two variables in each body.

we introduce k-2 new variables i.e $C_1, C_2 \ldots C_{k-2}$. The original production is replaced by the k-1 productions.

$A \to B_1 C_1$

$C_1 \to B_2 C_2$

$C_2 \to B_3 C_3$

$C_{k-3} \to B_{k-2} C_{k-2}$

$C_{k-2} \to B_{k-1} B_k$

convert the following grammar into CNF

E -> E + T

E -> T * F

E -> (E)

E -> a | b | Ia | Ib | I0 | I1

T -> T * F

T -> (E)

T -> a | b | Ia | Ib | I0 | I1

F -> (E)

F -> a | b | Ia | Ib | I0 | I1

I -> a | b | Ia | Ib | I0 | I1

Terminals: { +, *, (,), a, b, 0, 1 }

(I)

A -> a　　M -> *

B -> b　　L -> (

Z -> 0　　R ->)

O -> 1

P -> +

E -> EPT / T...

T -> TMF / LE...

F -> LER / a...

I -> a/b/IA...

E -> EP₁

P₁ -> PT

E -> TM₁

M₁ -> MF

E -> LE₁

E₁ -> ER

T -> TM...

T -> LE...

F -> L...

E -> EP₁ / T...

T -> TM₁

F -> LE₁

I -> a/b...

P₁ -> PT

M₁ -> MF

E₁ -> ER

A -> a

B -> b

Z -> 0

i.e $C_1, C_2 ...$ replaced

$S \to Bb$
$A \to aB$ }
$S \to aA$
$B \to Aa$ }
$\mid Bb$

$E \to EPT / TMF / LER / a / b / IA / IB / IZ / IO$

$T \to TMF / LER / a / b / IA / IB / IZ / IO$ (II)

$F \to LER / a / b / IA / IB / IZ / IO$

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

* CNF

$E \to EP_1$

$P_1 \to PT$

$E \to TM_1$ (III)

$M_1 \to MF$

$E \to LE_1$

$E_1 \to ER$

$T \to TM_1$

$T \to LE_1$

$F \to LE_1$

$E \to EP_1 / TM_1 / LE_1 / a / b / IA / IB / IZ / IO$

$T \to TM_1 / LE_1 / a / b / IA / IB / IZ / IO$

$F \to LE_1 / a / b / IA / IB / IZ / IO$

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

Final Answer

$\{ (,), a, b, 0, 1 \}$

$P_1 \to PT$

$M_1 \to MF$

$E_1 \to ER$

$A \to a$

$B \to b$

$Z \to 0$

$O \to 1$

$P \to +$

$M \to *$

$L \to ($

$R \to)$

Scanned by CamScanner

$\to 0A/1B$

$\to 0AA/1S/1$

$\to 1BB/0S/0$

$X \to 0$

$Y \to 1$

$S \to XA/YB$

$A \to XAA/YS/1$

$B \to YBB/XS/0$

$A \to XA_1/YS/1$

$A_1 \to AA$

$B \to YB_1/XS/0$

$B_1 \to BB$

$S \to XA/YB$
$A \to XA_1/YS/1$
$B \to YB_1/XS/0$
$A_1 \to AA$
$B_1 \to BB$
$X \to 0$
$Y \to 1$

$\to$ CNF

Scanned by CamScanner

$S \to ABC \mid BaB$

$A \to aA \mid BaC \mid aaa$

$B \to bBb \mid a \mid D$

$C \to CA \mid AC$

$D \to \epsilon$

ϵ - eliminating:

$D \to$ nullable variables

$B \to D$ } B D nullable variables.

$S \to ABC \mid AC \mid aB \mid Ba \mid BaB \mid a$

$A \to aA \mid Bac \mid ac \mid aaa$

$B \to bBb \mid bb \mid a \mid D$

$C \to CA \mid AC$

unit-productions:-

no

useless - symbols:-

unit-production:-

$B \to D$

non-unit

for D no non-unit

$S \to ABC \mid AC \mid aB \mid Ba \mid BaB \mid a$

$A \to aA \mid BaC \mid aC \mid aaa$

$B \to bBb \mid bb \mid a$

$C \to CA \mid AC$

Terminal: $\{a, b\}$

$S \to a$

$A \to aaa$

$B \to a$

S, A, B

$S \to aB$

$S \to Ba$

$S \to BaB$

$A \to$

unit production.

useless symbols:-

generating

$T = \{a, b\}$

$S \to a$
$A \to aaa$
$B \to a$
$B \to bb$

$\left.\begin{matrix} S \to a \\ A \to aaa \\ B \to a \end{matrix}\right\}$ S, A, B

$S \to aB$
$S \to Ba$
$S \to BaB$
$A \to aA$
$B \to bBb$

$\left.\begin{matrix} S \to aB \\ S \to Ba \\ S \to BaB \\ A \to aA \\ B \to bBb \end{matrix}\right\}$ S, A, B

$S \to a / aB / Ba / BaB$

$B \to a / bb / bBb$

$A \to aaa / aA$

Reachability:-

$\{S\}$

$S \to a / aB / Ba / BaB$ $\{S, a, B\}$

$\{S, a, B, b\}$

~~S~~

$\left.\begin{matrix} B \to a \\ A \to aaa \end{matrix}\right\}$ $\{S, a, B, A\}$

$A \to aA$ $\}$ $\{S, a, B, A\}$

$S \to a / aB / Ba / BaB$

$B \to a$

$A \to aaa / aA$

CNF:-

Scanned by CamScanner

abacaba

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

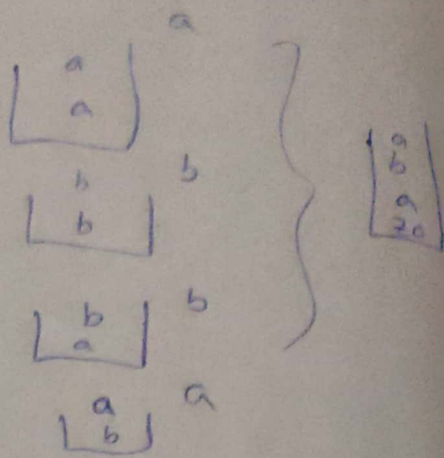
$\delta(q_0, b, Z_0) = (q_0, bZ_0)$

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

$\delta(q_0, b, b) = (q_0, bb)$

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

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



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

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

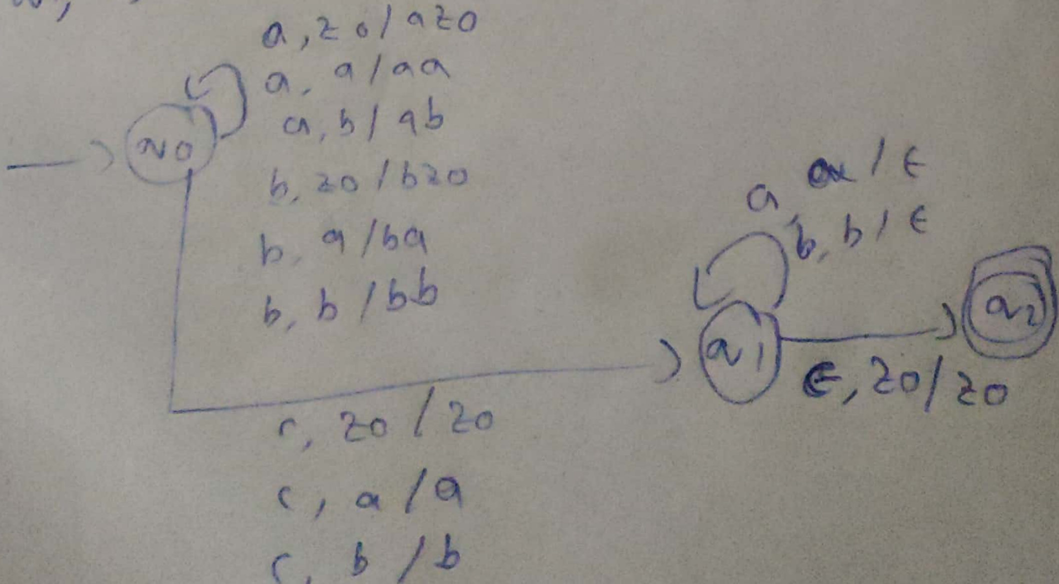
$\delta(q_0, c, Z_0) = (q_1, Z_0)$

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

$\delta(q_1, b, b) = (q_1, \epsilon)$

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

$\delta(q_1, \epsilon, Z_0) = (q_2, Z_0)$



$P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

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

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

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

q_0 = start state

z_0 = stack initial state

$F = \{q_2\}$

a
b
a
z_0

Instantaneous discription:-

$(q_0, abacaba, z_0) \vdash (q_0, bacaba, az_0)$

$\vdash (q_0, acaba, baz_0)$

$\vdash (q_0, caba, abaz_0)$

$\vdash (q_1, aba, abaz_0)$

$\vdash (q_1, ba, baz_0)$

$\vdash (q_1, a, az_0)$

$\vdash (q_1, \epsilon, z_0)$

$\vdash (q_2, \epsilon, z_0)$

The current configuration of PDA at any instance can be described by an ID (instantaneous description) ID gives the current state of PDA the remaining string to be processed the entire string to be stacked.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA an

ID is defined as 3-tuple ~~or~~ triple (q, w, α)

Let the current configuration of PDA be

$(q, aw, z\alpha)$

$\downarrow$ top of stack

$\delta(q, a, z) = (p, \beta)$ then new configuration

will be. $(p, w, \beta\alpha)$

$(q, aw, z\alpha) \vdash (p, w, \beta\alpha)$

$(q, aw, z\alpha)$ derives

the configuration $(q, aw, z\alpha)$ If an arbitrary

$(p, w, \beta\alpha)$ in one move. If an arbitrary number of moves are used to move from one configuration to another configuration then the moves are denoted by the symbol

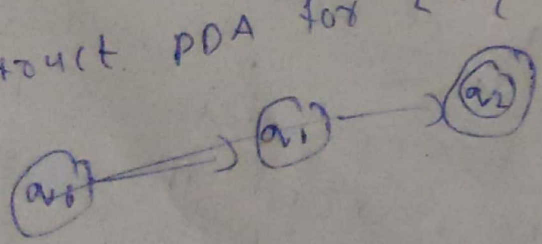
$\vdash^{*}$ (or) $\vdash^{+}$

$(q_0, abacaba, z_0) \vdash^{*} (q_2, \epsilon, z_0)$

$(q_0, abacaba, z_0) \vdash^{+} (q_1, caba, abaz_0)$

$q_0, abacaba, z_0)$

Construct PDA for $L = \{a^n b^n \; n > 0\}$



aaabbb

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

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

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

~~$\delta(q_0, a, a) = (q_0, aa)$~~

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

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

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

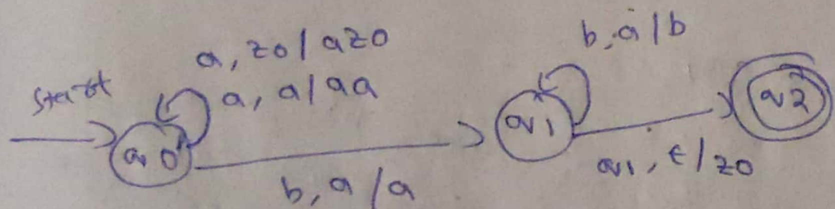
$\delta(q_1, b, a) = (q_1, b)$

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

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

$\delta(q_1, \epsilon, Z_0) = (q_2, Z_0)$

a
a
a
Z_0



$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

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

$\delta(q_0, b, Z_0) = (q_1, bZ_0)$

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

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

$\delta(q_1, \epsilon, Z_0) = (q_2, Z_0)$ → by final state

$\delta(q_1, \epsilon, Z_0) = (q_2, \epsilon)$ → on empty stack.

Design PDA for balanced [illegible]

$(q_0, abbaaabb, z_0) \vdash (q_0, bbaaabb, az_0)$

$\vdash (q_0, baaabb, z_0)$

$\vdash (q_0, aaabb, bz_0)$

$\vdash (q_0, aabb, z_0)$

$\vdash (q_0, abb, az_0)$

$\vdash (q_0, bb, aaz_0)$

$\vdash (q_0, b, az_0)$

$\vdash (q_0, \epsilon, z_0)$

$\vdash (q_1, \epsilon, z_0)$

([ ])

$\delta(q_0, (, z_0) = (q_0, (z_0)$

$\delta(q_0, [, z_0) = (q_0, [z_0)$

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

$\delta(q_0, [, [) = (q_0, [[)$

$\delta(q_0, (, [) = (q_0, ([)$

$\delta(q_0, [, () = (q_0, [()$

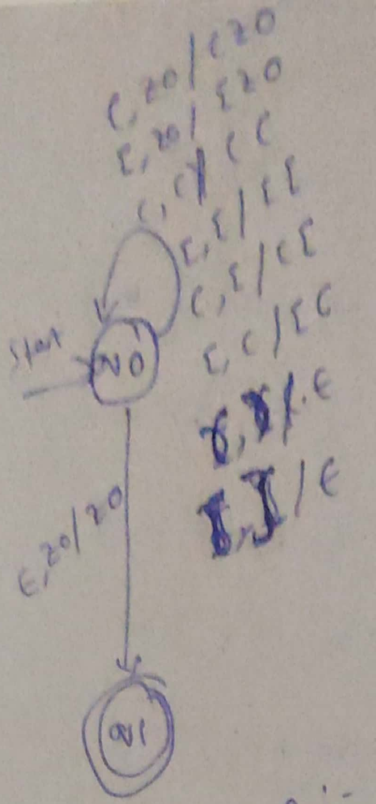
$\delta(q_0,), () = (q_0, \epsilon)$

$\delta(q_0,], [) = (q_0, \epsilon)$

$\cancel{\delta(q_0,), () = (q_0, \epsilon)}$

$\cancel{\delta(q_0,], [) = (q_0, \epsilon)}$

$\delta(q_0, \epsilon, z_0) = (q_1, \epsilon, z_0)$



Languages of PDA :-

Acceptance by Final state :-

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA then $L(P)$, then language accepted by P by Final state is $\{ w / (q_0, w, z_0) \vdash_P^* (q, \epsilon, \alpha) \}$

For some state q in F and any stack string α i.e starting in the initial ID with w waiting on the i/p P consumes w from the i/p and enters an accepting state. at that time the contents of the stack is irrelevent.

Acceptance by empty stack :-

For each PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ we also define $N(P) = \{ w / (q_0, w, z_0) \vdash^* (q, \epsilon, \epsilon) \}$ for any state q that is $N(P)$ is the set of inputs w that P can consume and at the same time empty its stack.

Design PDA for the language $L = \{ a^n b^{2n}$

$\delta(q_0, a, z_0) = (q_0, aaz_0)$ aabbbb

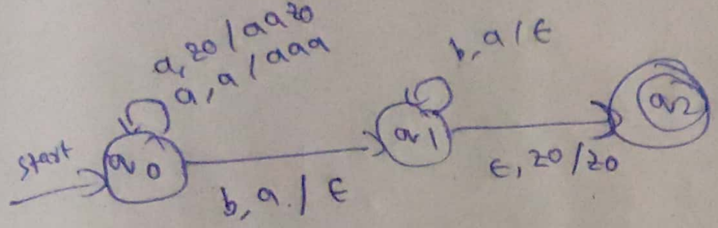
$\delta(q_0, a, a) = (q_0, aaa)$

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

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

$\delta(q_1, \epsilon, z_0) = (q_2, \epsilon, z_0)$

$= (q_2, \epsilon, \epsilon)$ by stack empty.



(i) $n_a > n_b$

(ii) $n_a < n_b$

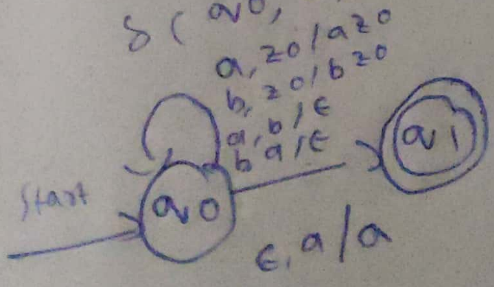
i) $\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, b, z_0) = (q_0, bz_0)$

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

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

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



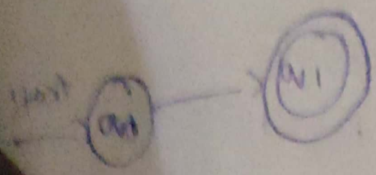
$\delta(q_0, a, z_0) = (q_0, a z_0)$

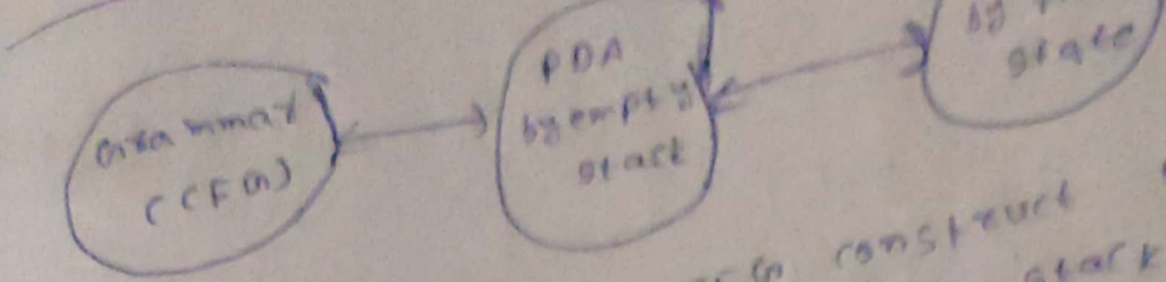
$\delta(q_0, b, z_0) = (q_0, b z_0)$

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

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

$\delta(q_0, b, z_0) = (q_1, \epsilon z_0)$





Let $G = (V, T, P, S)$ be a CFG construct the PDA P that accepts $L(G)$ by empty stack as follows $P = (\{q\}, T, V \cup T, \delta, q, S)$ where transition function δ is defined by

1. For each variable A

$$\delta(q, \epsilon, A) = \{ (q, \beta) \mid A \to \beta \text{ is a production of } P \}$$

2. For each terminal a $\delta(q, a, a) = \{ (q, \epsilon) \}$

(I) CFG is $I \to a / b / Ia / Ib / I0 / I1$

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

term: $\{0, 1, a, b, c, (,), +, *\}$

stack symbols = terminals + I + E

$\{0, 1, a, b, (,), +, *, I, E\}$

$\delta(q, \epsilon, I) = (q, a)$

$\delta(q, \epsilon, I) = (q, b)$

$\delta(q, \epsilon, I) = (q, Ia)$

$\delta(q, \epsilon, I) = (q, Ib)$

$\delta(q, \epsilon, I) = (q, I0)$

$\delta(q, \epsilon, I) =$

$\delta(q,$

$\delta(q, \epsilon, I) = (q, I1)$

$\delta(q, \epsilon, E) = (q, I)$

$\delta(q, \epsilon, E) = (q, E * E)$

$\delta(q, \epsilon, E) = (q, E + E)$

$\delta(q, \epsilon, E) = (q, (E))$

$\delta(q, 0, 0) = (q, \epsilon)$

$\delta(q, 1, 1) = (q, \epsilon)$

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

$\delta(q, b, b) = (q, \epsilon)$

$\delta(q, (, () = (q, \epsilon)$

$\delta(q,),)) = (q, \epsilon)$

$\delta(q, +, +) = (q, \epsilon)$

$\delta(q, *, *) = (q, \epsilon)$

II) $S \to aABC$

$A \to aB/a$

$B \to bA/b$

$C \to a$

Terminals : $\{a, b\}$

Stack symbols : $\{a, b, S, A, B, C\}$

$\delta(q, \epsilon, S) = (q, aABC)$

$\delta(q, \epsilon, A) = (q, aB)$

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

$\delta(q, \epsilon, B) = (q, bA)$

$\delta(q, \epsilon, B) = (q, b)$

$\delta(q, \epsilon, C) = (q, a)$

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

$\delta(q, b, b) = (q, \epsilon)$

$a \to S$

$S \epsilon$

PDA to CFG :-

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ be a PDA then there is a CFG G such that $L(G) = N(P)$
$G = (V, \Sigma, R, S)$

where the set of variables V consists of

1. The special symbol S, which is the start symbol &
2. All symbols of the form $[pXq]$ where p & q are states in Q and X is a stack symbol in T

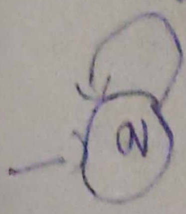
The productions of G are as follows.

a. For all states p, G has the production
$S \rightarrow [q_0 Z_0 p]$

b. Let $\delta(q, a, X)$ contains the pair $(r, Y_1, Y_2 \cdot Y_3)$

where
1. a is either a symbol in Σ or $a = \epsilon$
2. K can be any number including 0 in which case the pair is (r, ϵ) then for all lists of states $r_1 r_2 \dots r_k$ G has the production

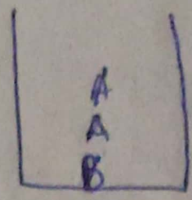
$[qXr_k] \rightarrow a [rY_1 r_1] [r_1 Y_2 r_2] \dots [r_{k-1} Y_k r_k]$



0, B/ε
1, A/ε
ε, Z/ε
0, Z/AZ
1, Z/BZ
0, A/AA
1, B/BB

101001

10110100



$\delta(q, 0, B) = (q, \epsilon)$

$\delta(q, 1, A) = (q, \epsilon)$

$\delta(q, \epsilon, Z) = (q, \epsilon)$

$\delta(q, 0, Z) = (q, AZ)$

$\delta(q, 1, Z) = (q, BZ)$

$\delta(q, 0, A) = (q, AA)$

$\delta(q, 1, B) = (q, BB)$.

For pop:-

$qBq \to 0$

$qAq \to 1$

$qZq \to \epsilon$

$S \to qZq$

For push:-

$qZq \to 1\,(qBq)\,(qZq)$

$qAq \to 0\,(qAq)\,(qAq)$

$qBq \to 1\,(qBq)\,(qBq)$

(ii) $\delta(q_0, a, Z) = (q_0, AZ)$

$\delta(q_0, a, A) = (q_3, \epsilon)$

$\delta(q_3, \epsilon, Z) = (q_0, AZ)$

$\delta(q_0, b, A) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, Z) = (q_2, \epsilon)$

101001

10110100

A
A
B

$S \to q_0 Z q_0 \mid q_0 Z q_1 \mid q_0 Z q_2 \mid q_0 Z q_3$

For pop:-

$q_0 A q_3 \to a$

$q_0 A q_1 \to b$

$q_1 Z q_2 \to \epsilon$

For push:-

$q_0 Z q_0 \to a (q_0 A q_0)(q_0 Z q_0)$

$\to \epsilon$ ~~(q0~~

~~q3 Z q0~~

$\to a (q_0 A q_1)(q_1 Z q_0)$

$\to a (q_0 A q_2)(q_2 Z q_0)$

$\to a (q_0 A q_3)(q_3 Z q_0)$

$\to a (q_0 A q_0)(q_0 Z q_1)$

$q_0 Z q_1 \to a (q_0 A q_1)(q_1 Z q_1)$

$\to a (q_0 A q_2)(q_2 Z q_1)$

$\to a (q_0 A$

If $\delta(q, a, x)$ contains more than one pair, then surely the PDA is non-deterministic.

Let PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ to be deterministic if and only if the following conditions are met.

1). $\delta(q, a, x)$ has atmost one member for any q in Q, a in Σ or $a = \epsilon$ and x in Γ.

2). If $\delta(q, \epsilon, x)$ is non empty, for some a in Σ, then $\delta(q, a, x)$ must be empty.

wcw^R

$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, b, z_0) = (q_0, bz_0)$

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

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

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

$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, \epsilon, z_0) = (q_1, z_0)$

$L: wcw^R$ is a Deterministic pushdown automata.

$L : \{a^n b^n \mid n \geq 0\} \rightarrow$ DPDA.

$L(M) = \{ w \mid w \in (a+b)^+ \text{ and } n_a(w) = n_b(w) \}$

deterministic ?

$\delta(q_0, a, z_0) = (q_0, a z_0)$ $\rightarrow$ 2nd condition

$\delta(q_0, b, z_0) = (q_0, b z_0)$

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

$\delta(q_0, b, b) = (q_0, bb)$

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

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

$\delta(q_0, \epsilon, z_0) = (q_1, z_0)$

not deterministic.

$\boxed{\text{NDPA}}$

Balanced paranthesis is NDPA

$L = \{a^n b^{2n} \mid n \geq 0\}$ is DPA

DPDA $\rightarrow$ unambigua

but unambiguous $\not\rightarrow$ DPDA

R.L < CFL

CFL → DPDA → Some CFL

CFL → NDPDA → CFL

DPDA $\subset$ NDPDA

$L(\text{DPDA}) \subset L(\text{NPDA})$

CYK Algorithm :- → membership problem.

Testing membership in CFL :-

Given a CFG G and string w

G = (V, Σ, P, S) where

V → finite set of variables

Σ → input symbols

P → productions

S → start symbol.

G is used to generate the string of a language

Q : is w in L(G).

Basics : structure of rules in a CNF grammars. Table filling is depending on dynamic programming

construct a triangular table :-

1) Each row corresponds to one length of substrings.

Bottom row is string of length 1

second from Bottom row is string of length 2.

⋮

Top row is string w

$x_{i,i}$ – is the set of variables A such that A → w_i is a production of G compare at most n pairs of previously computed sets

$(x_{i,i}, x_{i+1,j})$ $(x_{i,i+1}, x_{i+2,j})$ $(x_{i,i+2}, x_{i+3,j})$

...... $(x_{i,j-1}, x_{j,j})$

show that $S \to AB \mid BC$

$A \to BA \mid a$

$B \to CC \mid b$

$C \to AB \mid a$

w is baaba whether w is a member of L(G) using CYK algorithm.

$x_{1,5}$				
$x_{1,4}$	$x_{2,5}$			
$x_{1,3}$	$x_{2,4}$	$x_{3,5}$		
$x_{1,2}$	$x_{2,3}$	$x_{3,4}$	$x_{4,5}$	
$x_{1,1}$	$x_{2,2}$	$x_{3,3}$	$x_{4,4}$	$x_{5,5}$

{S, A, C}				
∅	{S, A, C}			
∅	{B}	{B}		
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

$x_{1,1} = b$

$B \to b$

$x_{2,2} = a$

$C \to a$

$A \to a$

~~$x_{1,2} = ba$~~

blem.

f a

F

ing

gth of

ng of

A such

G

$x_{1,2} = (x_{i,i},\ x_{i+1,j})$

$= (x_{1,1},\ x_{2,2})$

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

$= \{BA,\ BC\}$

$S \to BC$

$A \to BA$

$= \{S, A\}$

$x_{2,3} = (x_{i,i},\ x_{i+1,j})$

$= (x_{2,2},\ x_{3,3})$

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

$= \{AA,\ AC,\ CA,\ CC\}$

$= \{B\}$

$x_{3,4} = (x_{i,i},\ x_{i+1,j})$

$= (x_{3,3},\ x_{4,4})$

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

$= \{AB,\ CB\}$

$= \{S\}$

$x_{4,5} = (x_{4,4}, x_{5,5})$

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

$= \{BA, BC\}$

$= \{S, A\}$

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

$= (\{B\}\ \{B\}) \cup (\{S, A\}\ \{A, C\})$

$= \{\{BB\}, \{SA\}, \{SC\}, \{AA\}, \{AC\}\}$

$x_{2,4} = (x_{2,2}, x_{3,4})\ (x_{2,3}, x_{4,4})$

$= (\{A, C\}\ \{S, C\}) \cup (\{B\}, \{B\})$

$= (\{AS\}, \{CS\}, \{BB\}, \{CS\}\ \{CC\}\}$

$= \{\{AS\}, \{CS\}, \{BB\},$

$= \{B\}$

$x_{3,5} = (x_{3,3}, x_{4,5}) \cup (x_{3,4}, x_{5,5})$

$= (\{A, C\}, \{S, A\}) \cup (\{S, C\}, \{A, C\})$

$= \{\{AS\}, \{AA\}, \{CS\}, \{CA\}, \{SA\}, \{SC\}\ \{CA\}\ \{CC\}\}$

$= \{B\}$

$X_{1,4} = (X_{1,1}, X_{2,4})\ (X_{1,2}, X_{3,4})\ (X_{1,3}, X_{4,4})$

$= (\{B\}, \{B\}) \cup (\{S,A\}, \{S,C\}) \cup (\{\emptyset\}, \{B\})$

$= \{\{BB\}, \{SS\}, \{SC\}, \{AS\}, \{AC\}\}$

$X_{2,5} = (X_{2,2}, X_{3,5})\ (X_{2,3}, X_{4,5})\ (X_{2,4}, X_{5,5})$

$= (\{A,C\}\ \{B\}) \cup (\{B\}, \{S,A\}) \cup (\{B\}, \{A,C\})$

$= \{\{AB\}, \{CB\}, \{BS\}, \{BA\}, \{BA\}, \{BC\}\}$

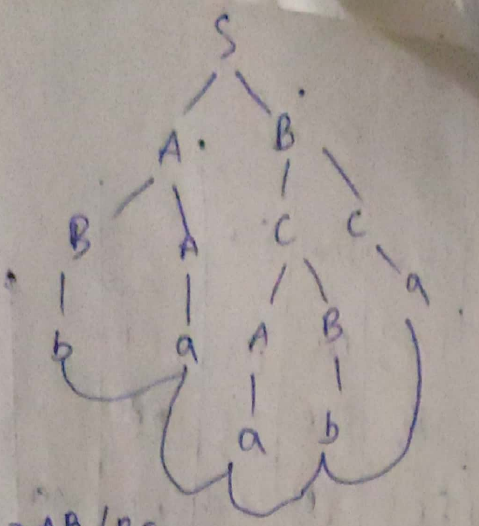
$\{S, A, C\}$ $(S, A, C$

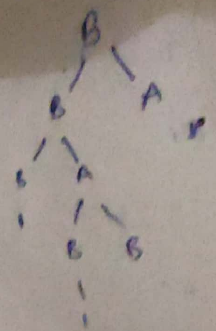
$X_{1,5} = (X_{1,1}, X_{2,5})\ (X_{1,2}, X_{3,5})\ (X_{1,3}, X_{4,5})$
$(X_{1,4}, X_{5,5})$

$= (\{B\}, \{S,A\}) \cup (\{S,A\}, \{B\}) \cup (\{\emptyset\}, \{S,A\})$ $\emptyset$
$\cup\ (\{\emptyset\}, \{A,C\})$ $\emptyset$

$= \{\{BS\}, \{BA\}, \{SB\}, \{AB\}, \{BC\}\}$

$\{A, S, C\}$





$S \to AB \,/\, BC$

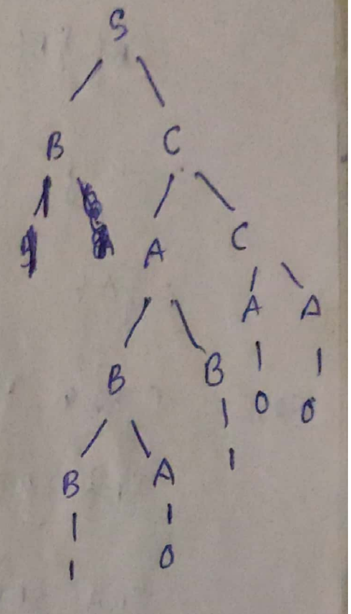
$A \to BB \,/\, 0$

$B \to BA \,/\, 1$

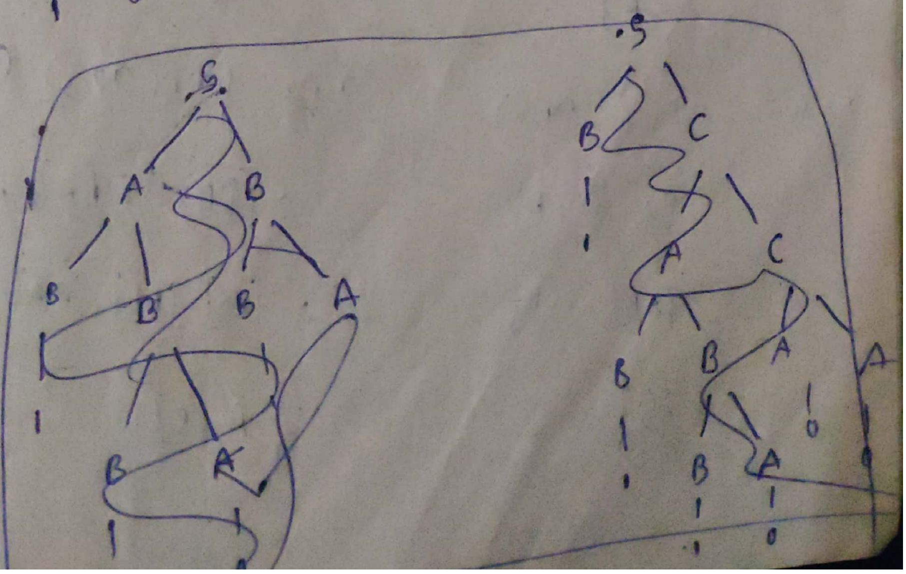
$C \to AC \,/\, AA \,/\, 0$

CYK Algo

110100



$x_{1,6}$					
$x_{1,5}$	$x_{2,6}$				
$x_{1,4}$	$x_{2,5}$	$x_{3,6}$			
$x_{1,3}$	$x_{2,4}$	$x_{3,5}$	$x_{4,6}$		
$x_{1,2}$	$x_{2,3}$	$x_{3,4}$	$x_{4,5}$	$x_{5,6}$	
$x_{1,1}$	$x_{2,2}$	$x_{3,3}$	$x_{4,4}$	$x_{5,5}$	$x_{6,6}$
1	1	0	1	0	0



$x_{1,1} = 1$

$B \to 1$

$x_{3,3} = 0$

$A \to 0$

$C \to 0$

{S,B}					
{S,B}	{A,C}				
{B,S}	{A,C}	{S}			
{A,C}	{A}	{S}	{S,B}		
{A}	{S,B}	{S}	{S,B}	{C}	
{B}	{B}	{A,C}	{B}	{A,C}	{A,C}

$x_{1,2} = (x_{1,1} \cdot x_{2,2})$

$= \{B\}\{B\}$

$= \{BB\}$

$= \{A\}$

$x_{2,3} = (x_{2,2}\ x_{3,3})$

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

$= \{BA\}\{BC\}$

$= \{S, B\}$

$x_{3,4} = (x_{3,3}\ x_{4,4})$

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

$= \{AB\}\{CB\}$

$= \{S\}$

$x_{4,5} = (x_{4,4}\ x_{5,5})$

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

$= \{BA\}\{BC\}$

$= \{B, S\}$

$x_{5,6} = (x_{5,5}\ x_{6,6})$

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

$= \{AA\}\{AC\}\{CA\}\{CC\}$

$= \{C\}$

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

$= (\{B\}\{S, B\}) \cup (\{A\}\{A$

$= \{BS\}\{BB\}\{AA\}\{AC\}$

$= \{A, C\}$

$x_{2,4} = (x_{2,2} \cdot x_{3,4})\ (x_{2,3}\ x_{4,4})$

$= (\{B\}\ \{S\})\ (\{S, B\}\ \{B\})$

$= \{BS\}\ \{SB\}\ \{BB\}$

$= \{A\}$

$x_{3,5} = (x_{3,3}\ x_{4,5})\ (x_{3,4}\ x_{5,5})$

$= (\{A, C\}\ \{S, B\})\ (\{S\}\ \{A, C\})$

$= \{AS\}\ \{AB\}\ \{CS\}\ \{CB\}\ \{SA\}\ \{SC\}$

$= \{S\}$

$x_{4,6} = (x_{4,4}\ x_{5,6})\ (x_{4,5}\ x_{6,6})$

$= (\{B\}\ \{C\}) \cup (\{S, B\}\ \{A, C\})$

$= \{BC\}\ \{SA\}\ \{SC\}\ \{BA\}\ \{BC\}$

$= \{S, B\}$

$x_{1,4} = (x_{1,1}\ x_{2,4})\ (x_{1,2}\ x_{3,4})\ (x_{1,3}\ x_{4,4})$

$= (\{B\}\ \{A\})\ (\{A\}\ \{S\})\ (\{A, C\}\ \{B\})$

$= \{BA\}\ \{AS\}\ \{AB\}\ \{CB\}$

$= \{B, S\}$

$x_{2,5} = (x_{2,2}\ x_{3,5})\ (x_{2,3}\ x_{4,5})\ (x_{2,4}\ x_{5,5})$

$= (\{B\}\ \{S\})\ (\{S,B\}\ \{S,B\})\ (\{A\}\ \{A,C\})$

$= \{BS\}\ \{SS\}\ \{SB\}\ \{BS\}\ \{BB\}\ \{AA\}\ \{AC\}$

$= \{A, C\}$

$x_{3,6} = (x_{3,3}\ x_{4,6})\ (x_{3,4}\ x_{5,6})\ (x_{3,5}\ x_{6,6})$

$= (\{A,C\}\ \{S,B\})\ (\{S\}\ \{S,B\})\ (\{S\}\ \{A,C\})$

$= \{AS\}\ \{AB\}\ \{CS\}\ \{CB\}\ \{SS\}\ \{SB\}\ \{SA\}\ \{SC\}$

$= \{S\}$

$x_{1,5} = (x_{1,1}\ x_{2,5})\ (x_{1,2}\ x_{3,5})\ (x_{1,3}\ x_{4,5})\ (x_{1,4}\ x_{5,5})$

$= (\{B\}\ \{A,C\})\ (\{A\}\ \{S\})\ (\{A,C\}\ \{S,B\})\ (\{B,S\}\ \{A,C\})$

$= \{BA\}\ \{BC\}\ \{AS\}\ \{AS\}\ \{AB\}\ \{CS\}\ \{CB\}\ \{BA\}\ \{BC\}\ \{SA\}\ \{SC\}$

$= \{B, S\}$

$x_{2,6} = (x_{2,2}\ x_{3,6})\ (x_{2,3}\ x_{4,6})\ (x_{2,4}\ x_{5,6})$
$(x_{2,5}\ x_{6,6})$

$= (\{B\}\{S\})\ (\{S,B\}\ \{S,B\})\ (\{A\}\ \{C\})$
$(\{A,C\}\ \{A,C\})$

$= \{BS\}\ \{SS\}\ \{SB\}\ \{BS\}\ \{BB\}\ \{AC\}\ \{AA\}\ \{AA\}$
$\{CA\}\ \{CC\}$

$= \{A, C\}$

$x_{1,6} = (x_{1,1}\ x_{2,6})\ (x_{1,2}\ x_{3,6})\ (x_{1,3}\ x_{4,6})\ (x_{1,4}\ x_{5,6})$
$(x_{1,5}\ x_{6,6})$

$= (\{B\}\ \{A,C\})\ (\{A\}\ \{S\})\ (\{A,C\}\ \{S,B\})\ (\{B,S\}\{C\})$
$(\{S,B\}\ \{A,C\})$

$= \{BA\}\ \{BC\}\ \{AS\}\ \{AS\}\ \{AB\}\ \{CS\}\ \{CB\}\ \{BC\}\ \{SC\}$
$\{SA\}\ \{SC\}\ \{BA\}\ \{BC\}$

$= \{B, S\}$

convert CFG

$S \to S0S1S0S\ /\ S0S0S1S\ /\ S1S0S0S\ /\ \epsilon$ to PDA that accepts the same language by empty stack.

terminals : $\{0, 1\}$

stack symbols : $\{0, 1, S\}$

$\delta(q, \epsilon, S) = (q, S0S1S0S)$

$\delta(q, \epsilon, S) = (q, S0S0S1S)$

$\delta(q, \epsilon, S) = (q, S1S0S0S)$

$\delta(q, \epsilon, S) = (q, \epsilon)$

$\delta(q, 0, 0) = (q, \epsilon)$

$\delta(q, 1, 1) = (q, \epsilon)$

Closure properties of CFL's

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

(i) union

(ii) concatenation

(iii) closure

(iv) Homomorphism

(v) Inverse Homomorphism

$L_1 : \{0^n 1^n 2^i \; n \geq 1 \; i \geq 1\}$

$S \to AB$

$A \to 0A1 / 01$

$B \to 2B / 2$

$L_2 : \{0^i 1^n 2^n \;\; n \geq 1, i \geq 1\}$

$S \to AB$

$A \to 0A/0$

$B \to 1B2/12$

If L is CFL

R is a R.L

$L \cap R$ - CFL

L is CFL

L_1, L_2 is R.L

$L - R$ is CFL

$\bar{L}$ is not necessarily a CFL

$L_1 - L_2$ is not necessarily a CFL.

complexity of converting among CFG's and PDA's

1. CFG - PDA conversion
2. PDA by final state - PDA by empty stack - PDA by final state,
3. by empty stack - PDA by grammer - n^3

conversion from PDA grammer - n

conversion to chomsky

Running time of conversion to chomsky normal form:

→ Reachable and generating symbols of a grammer - $O(n)$ time.

→ Eliminating the useless symbols - $O(n)$ time

→ constructing unit pairs and eliminating unit productions - $O(n^2)$ time .

-> Replacement of terminals by variable.
in product body = $O(n)$ time

-> Breaking of production bodies of length 3 or more into bodies of length 2 - $O(n)$

Given a grammer G of length n we find an equivalent CNF Grammer for G in time $O(n^2)$ the resulting grammer has length $O(n^2)$

-> Testing emptiness of CFL's - $O(n)$
-> Testing Membership is a CFL - $O(n^3)$ by CYK

* preview of Undecidable CFL problems

we cannot solve by any algorithm

1. IS a given CFG G ambiguoues.
2. IS a given CFL inherently ambiguoues.
3. is the intersection of two CFL's is empty
4. Are two CFL's the same
5. IS a given CFL equal to Σ^* where Σ is the alphabet of this language.

R.L < DPDA < NDPDA

Turing Machines

Regular languages - Type 3 -> FA
context free Languages - Type 2 -> PDA
context sensitive Languages - Type 1 -> LBA
Recursively enumerable Languages - Type 0 -> TM

Formal notation for turing machine:-

we will describe turing machine by 7-tuples i.e

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

Q -> Finite set of states of the finite controll

Σ -> set of input symbols.

Γ -> complete set of tape symbols.

$$\Sigma \subset \Gamma$$

δ -> Transition function.

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

(i) P -> next state in Q.

(ii) Y is the symbol in Γ, written in the cell being scanned, replacing whatever symbol was there.

(iii) D is a direction either L or R standing for Left or Right respectively and telling us the direction in which the head moves.

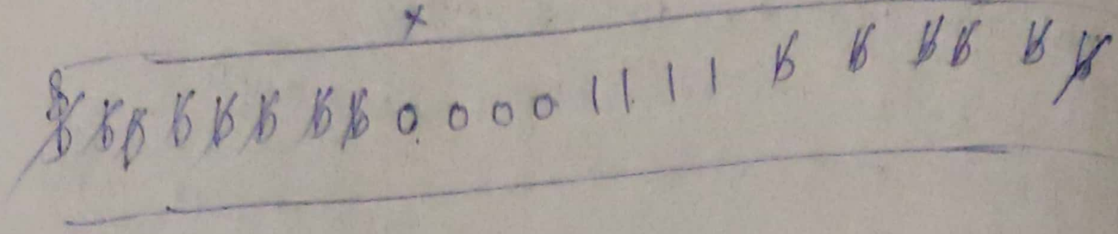
(iv) q_0 is the start state, a member of Q in which a finite controll is found initially

(v) B is the blank symbol which is in Γ but not Σ

The blank appears initially in all but the finite number of initial

(vi) F is the set of final or accepting state

RL < CFL < CSL < REL



(q0, 0) = (q1, x, ->)

(q1, 0) = (q1, 0, ->)

(q1, 1) = (q2, y, <-)

(q2, 0) = (q2, 0, <-)

(q2, x) = (q0, x, ->)

(q2, y) = (q2, y, <-)

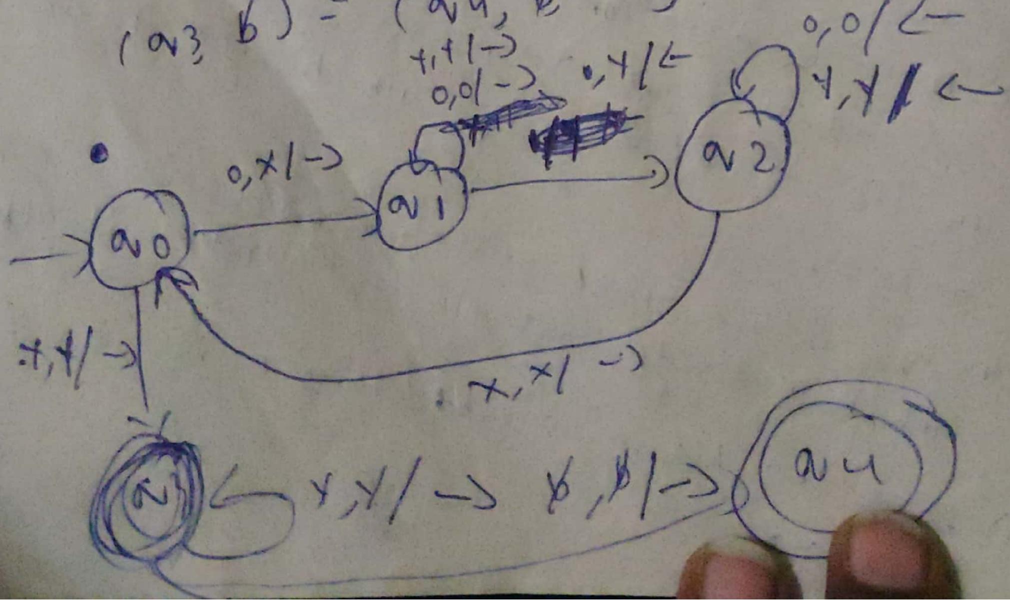
~~(q1, x) = (q1, y, ->)~~

(q1, y) = (q1, y, ->)

(q0, y) = (q3, y, ->)

(q3, y) = (q3, y, ->)

(q3, b) = (q4, b, ->)



	0	1	x	Y	$\not b$
q_0	$(q_1, x, \rightarrow)$	-	-	-	-
q_1	$(q_1, 0, \rightarrow)$	$(q_2, Y, \leftarrow)$	~~$(q_1, Y, \rightarrow)$~~	$(q_3, Y, \rightarrow)$ $(q_1, Y, \rightarrow)$	-
q_2	$(q_2, 0, \leftarrow)$	-	$(q_0, x, \rightarrow)$	$(q_2, Y, \leftarrow)$	-
q_3	-	-	-	$(q_3, Y, \rightarrow)$	$(q_4, \not b, \rightarrow)$
q_4	-	-	-	-	-

Instantaneous description:-

$\not b \not b \not b \not b\ x_1\ x_2\ x_3 \ldots \ldots q\, x_i\ x_{i+1}\ x_{i+2} \ldots x_n \not b \not b \not b$

$(q, x_i) = (p, Y, R)$

$i = 1$

$\not b \not b \not b$

$x_1 \ldots Y\, p\, x_{i+1} \ldots x_n$

$i = n$

$\not b \not b \not b\ x_1 \ldots q\, x_n \ldots \not b \not b \not b \not b$

$\not b\ x_1 \ldots x_{n-1}\, Y\, p\, \not b \not b \not b \not b$

$i = 1$

$q\, x_1\, x_2\, x_3 \ldots x_n \ldots x_n \not b \not b \not b$

$\not b \not b\, Y\, p\, x_2\, x_3$

$q_0 0011 \vdash x q_1 011$

$\vdash x 0 q_1 11$

$\vdash x q_2 0 y 1$

$\vdash q_2 x 0 y 1$

$\vdash x q_0 0 y 1$

$\vdash x x q_1 y 1$

$\vdash x x y q_1 1$

$\vdash x x q_2 y y$

$\vdash x q_2 x y y$

$\vdash x x q_0 y y$

$\vdash x x y q_3 y$

$\vdash x x y y q_3 B$

$\vdash x x y y B q_4 B$

$q_0 00011 \vdash x q_1 0011$

$\vdash x 0 q_1 011$

$\vdash x 0 0 q_1 11$

$\vdash x 0 q_2 0 y 1$

$\vdash x q_2 0 0 y 1$

$\vdash q_2 x 0 0 y 1$

$\vdash x q_0 0 0 y 1$

$\vdash x x q_1 0 y 1$

$\vdash x x 0 q_1 y 1$

$\vdash x x 0 y q_1 1$

$\vdash x x 0 q_2 y y$

$\vdash x x q_2 0 y y$

$\vdash x q_2 x 0 y y$

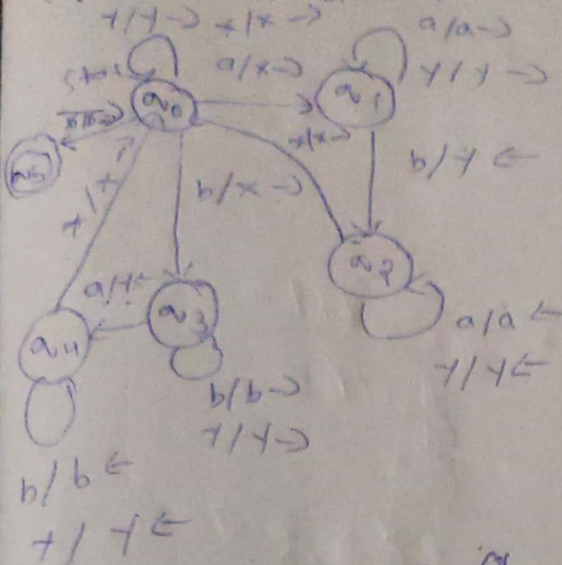
$\vdash x x q_0 0 y y$

$\vdash x x x q_1 y y$

$\vdash x x x y q_1 y$

$\vdash x x x y y q_1 B$

Design a turing machine where no. of a's equal to no. of b's.



b b b a a b a a

	a	b	x	Y	b̸
q_0	$(q_1, x, \rightarrow)$	$(q_3, x, \rightarrow)$	$(q_0, x, \rightarrow)$	$(q_0, Y, \rightarrow)$	$(q_5, b̸, \rightarrow)$
q_1	$(q_1, a, \rightarrow)$	$(q_2, Y, \leftarrow)$	–	$(q_1, Y, \rightarrow)$	–
q_2	$(q_2, a, \leftarrow)$	–	$(q_0, x, \rightarrow)$	$(q_2, Y, \leftarrow)$	–
q_3	$(q_4, Y, \leftarrow)$	$(q_3, b, \rightarrow)$	–	$(q_3, Y, \rightarrow)$	–
q_4	–	$(q_4, b, \leftarrow)$	$(q_0, x, \rightarrow)$	$(q_4, Y, \leftarrow)$	–
q_5	–	–	–	–	–

$q_0abba \vdash x q_1 bba$

$\vdash q_2 xYba$

$\vdash x q_0 Yba$

$\vdash xY q_0 ba$

$\vdash xYx q_3 a$

$\vdash \cancel{xY} q_4 xY$

$\vdash x q_4 YxY$

$\vdash q_4 xYxY$

$\vdash \cancel{x} q_0 YxY$

$\vdash xY q_0 xY$

$\vdash xYx q_0 Y$

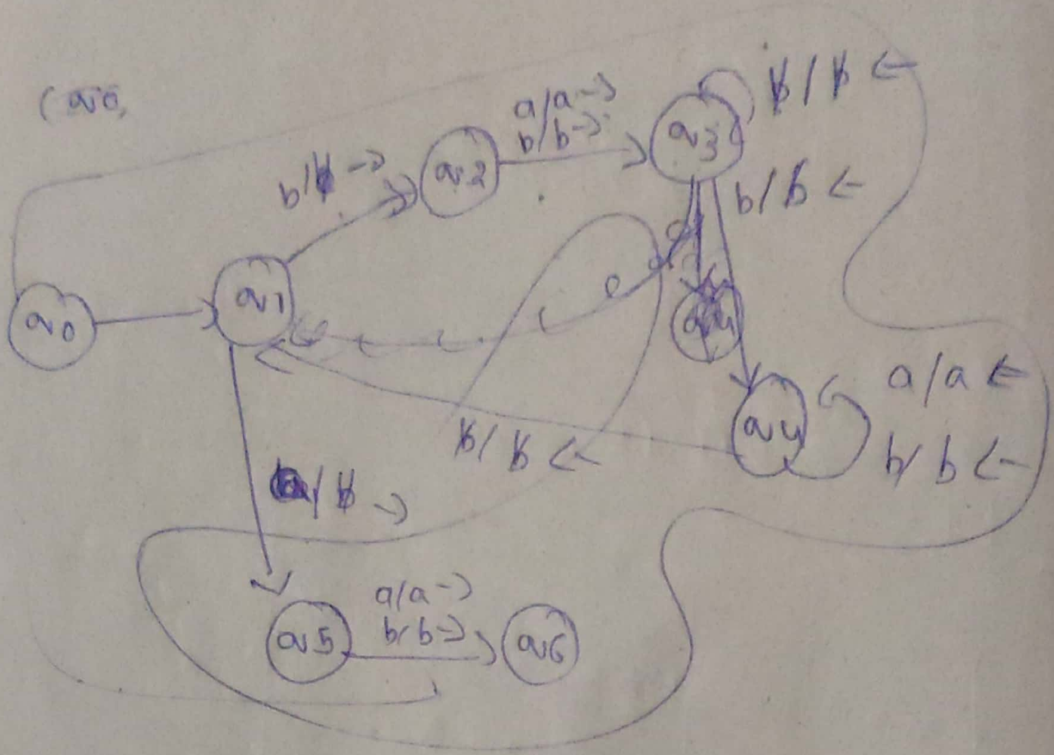
$\vdash xYxY q_0 b̸$

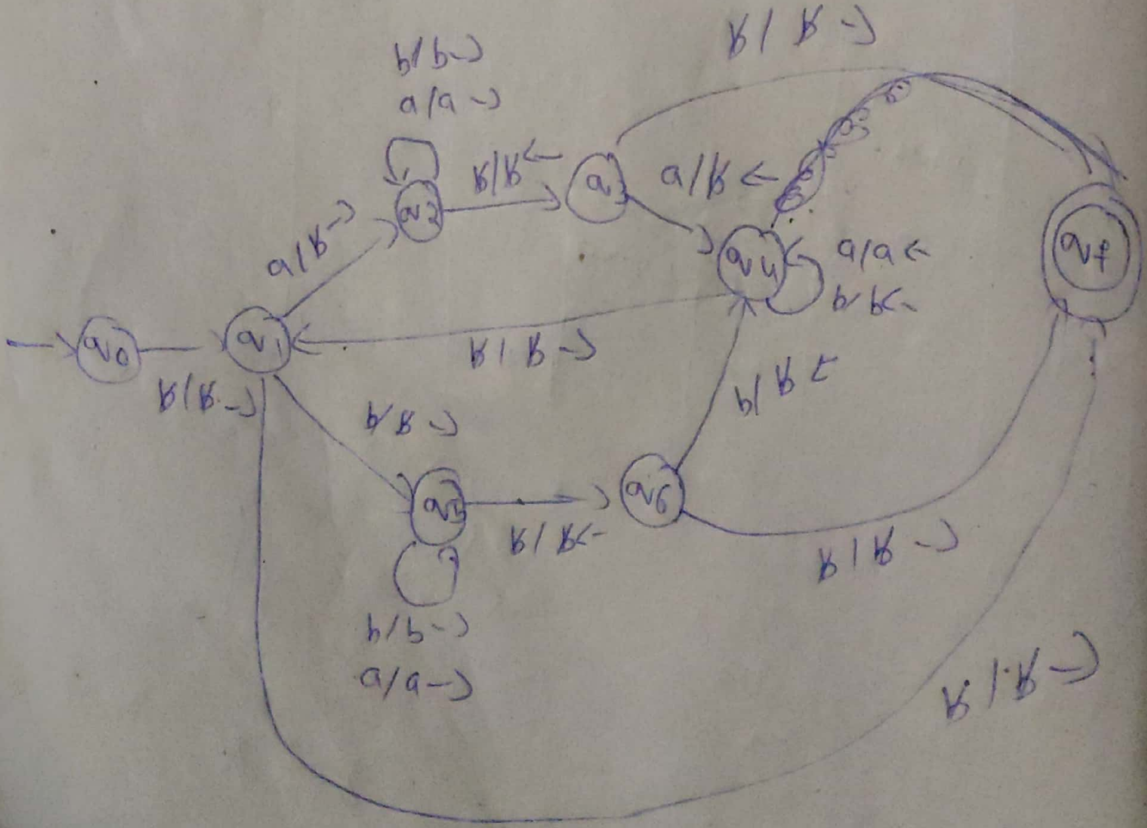
$\vdash xYxY q_5 b̸$

Design a turing machine over input alphabet {a, b} for pallindrome.

a b a b a

b̸ b̸ b̸ · b b a a b b b̸ b̸ b̸ b̸





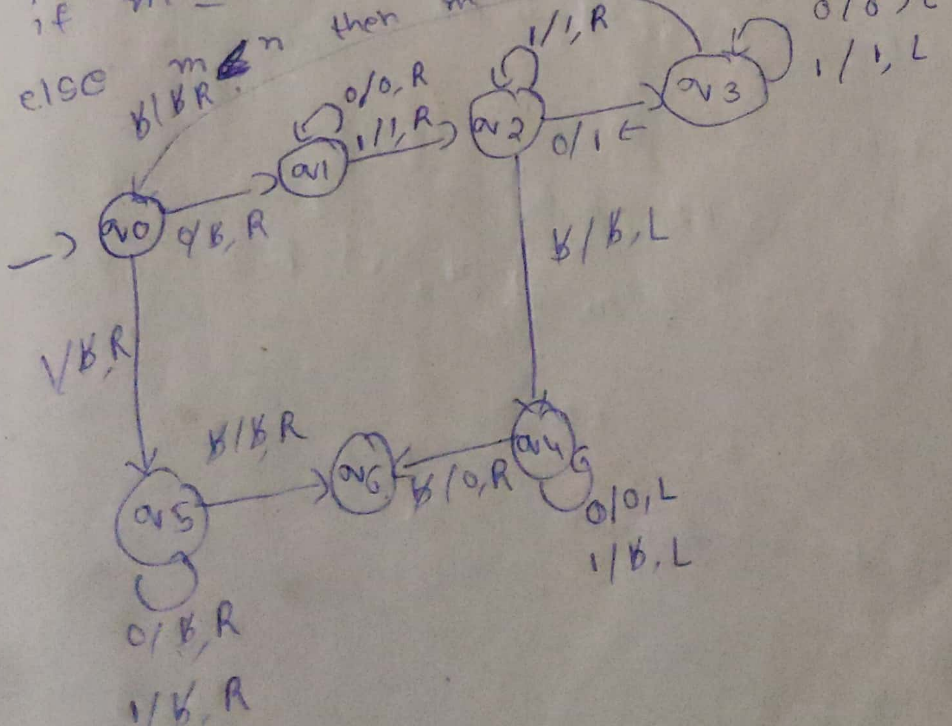
δ	a	b	B
q_0	–	–	$(q_1, B, \rightarrow)$
q_1	$(q_2, B, \rightarrow)$	$(q_5, B, \rightarrow)$	$(q_f, B, \rightarrow)$
q_2	$(q_2, a, \rightarrow)$		
q_3			
q_4			
q_5			
q_6			
q_f			

obtain a turing machine to compute the function minus which is called monus (or) proper substraction and is defined by m-n.

and is $m - n = \max(m-n, 0)$

if $m \geq n$ then $m - n = m - n$

else $m < n$ then $m - n = 0$.



b b b b b 00000 1000 b b b b

Haulting Turing machine

$q_0$000100 ⊢

programming Techniques for Turing Machines

1. storage in the state
2. Multiple tracks
3. subroutines.

① $M = (Q, \{0,1\}, \{0,1,B\}, \delta, [q_0, B], B, [q_0, B])$

$\delta([q_0, B], a) = ([q_1, a], a, R)$

$\delta([q_1, a], b) = ([q_1, a], b, R)$

$\delta([q_1, a], B) = ([q_2, B], a, R)$

② Multiple tracks:-

$M = (Q, \Sigma, \Gamma, \delta, [q_0, B], [B, B],$

$\delta([q_0, B], [B, a]) = ([q_1, a], [*, a], R)$

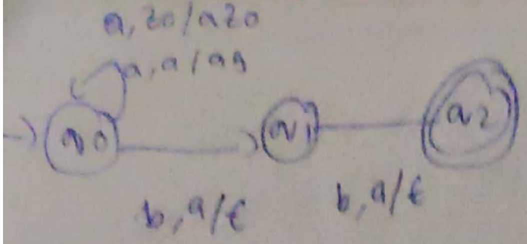
$\delta([q_0, B], [B, 0]) = ([q_1, 0], [*, x], R)$

Extensions to turing machines:-

1. Multi tape TM
2. Non deterministic TM

Language of the Turing Machine:-

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a TM then $L(M)$ is the set of strings w in Σ^* such that $q_0 w \vdash^* \alpha p \beta$ for some state p in F and any tape strings α and β



Design a PDA for no. of a's should be equal to number of b's.

a b

abbabbaabb

$\delta(q_0, a, z_0) = (q_0, a z_0)$

$\delta(q_0, b, z_0) = (q_0, b z_0)$

$\delta(q_0, a, a) : (q_0, aa)$

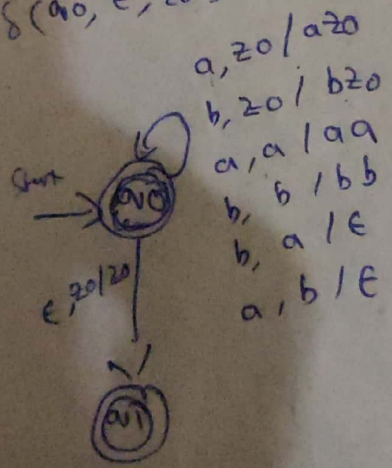
$\delta(q_0, b, b) : (q_0, bb)$

$\delta(q_0, b, a) : (q_0, \epsilon)$

$\delta(q_0, a, b) : (q_0, \epsilon)$

$\delta(q_0, \epsilon, z_0) = (q_1, z_0)$ → by final state

$\delta(q_0, \epsilon, z_0) = (q_0, \epsilon)$
by empty stack.



$Q : \{q_0, q_1\}$

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

δ

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

q_0

z_0

$F = \{q_1\}$