Mod3: Myhill-Nerode Theorem / Table filling method

John Myhill & Anil Nerode Proposed a algorithm called Myhill-Nerode theorm which is used to eliminate useless stage from a dfa.

Steps:

1) Draw a table for all pairs of states (P, Q)

2) Mark all pairs where P is an element of F & Q is not an element of F

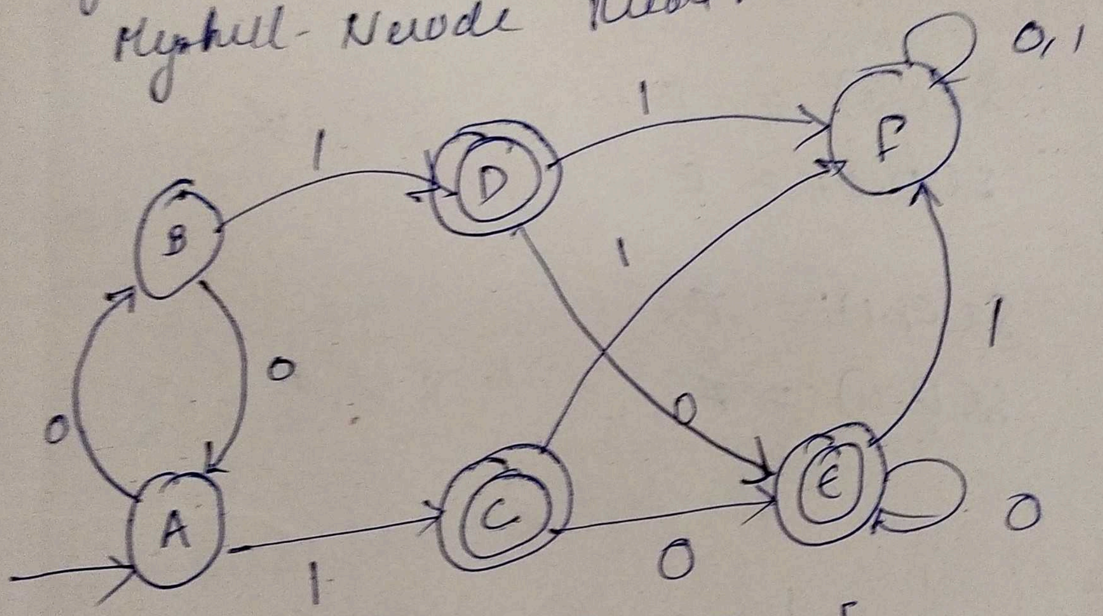
3) Mark all pairs with one of them is final state & other is not a final state. Do not mark if both of them are final state or both of them are non-final state.

3) If there are any unmarked pairs (P, Q) such that $\delta(P, x), \delta(Q, x)$ is marked. Then marked (P, Q)

where x is an input symbol.
Repeat this act until no more marking can be made.

4) Combine all the unmarked pairs & make them a single state in the minimised dfa.

Q: Minimise the following using Myhill-Nerode theorem.



	A	B	C	D	E	F
A						
B						
C	✓	✓				
D	✓	✓				
E	✓	✓				
F	✓	✓	✓	✓	✓	

AB - unwanted pair ~~cannot be marked~~

$$\left.\begin{array}{l}\delta(A,0) = B \\ \delta(B,0) = A\end{array}\right\} (AB)$$

$$\left.\begin{array}{l}\delta(A,1) = C \\ \delta(B,1) = D\end{array}\right\} (C;D)$$

CD

$\delta(C,0) = E$

$\delta(D,0) = E$

$\delta(C,1) = F$

$\delta(D,1) = F$

CE

$\delta(C,0) = E$

$\delta(E,0) = E$

$\delta(C,1) = F$

$\delta(E,1) = F$

DE

$\delta(D,0) = E$

$\delta(E,0) = E$

$\delta(D,1) = F$

$\delta(E,1) = F$

AF

$\delta(A,0) = B$ } (B,F)

$\delta(F,0) = F$

& $\delta(A,1) = C$ } (C,F) marked so we can remove (A,F)

$\delta(F,1) = F$

BF

$\delta(B,0) = A$ } (A,F)

$\delta(F,0) = F$

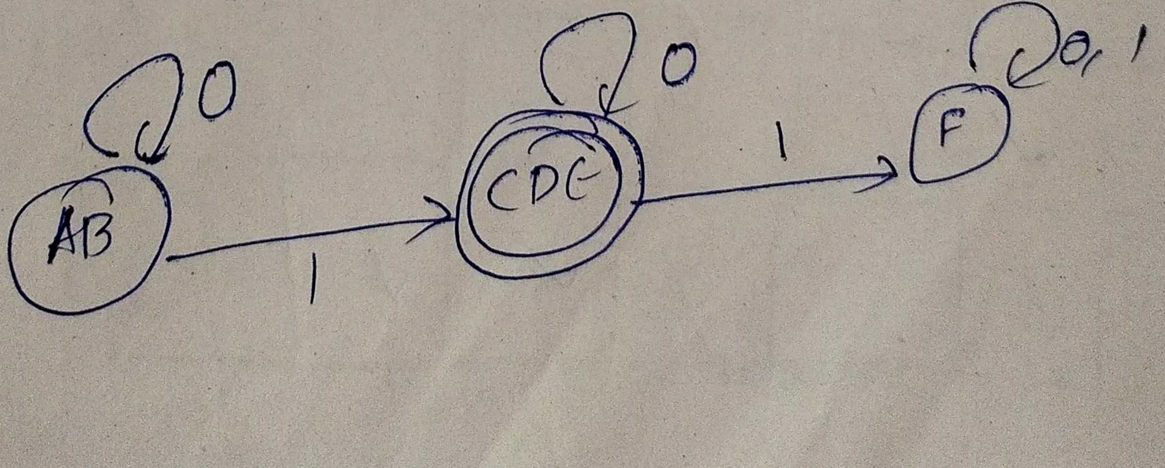
Combine Unmarked pairs :-

here (A,B)

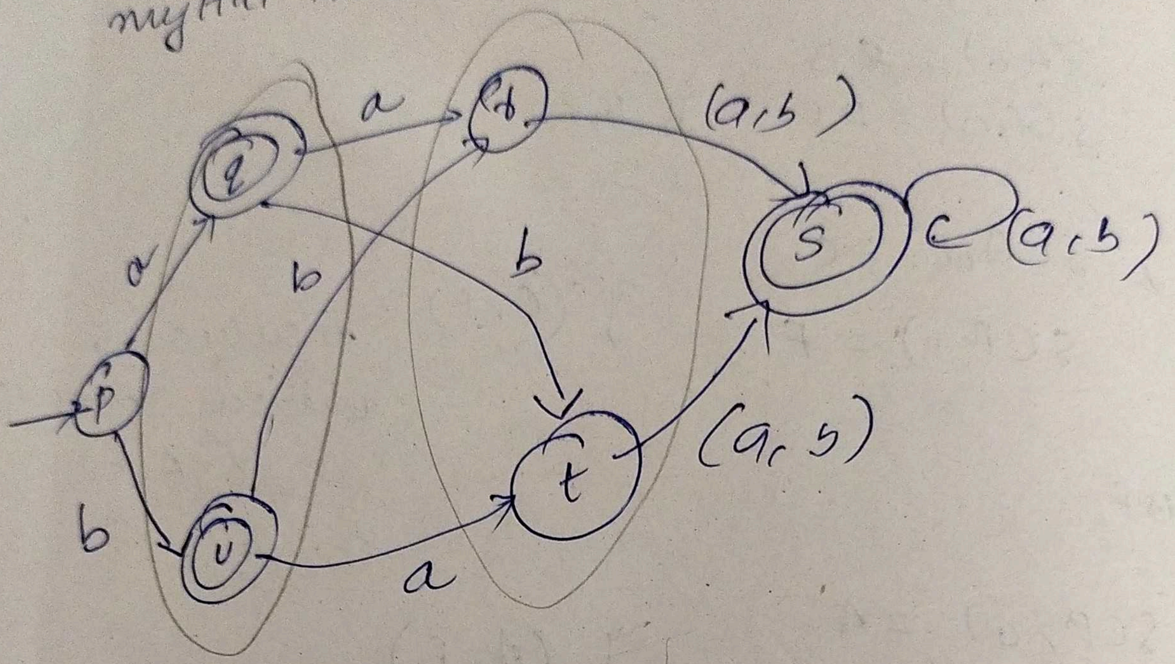
(C,D)

(C,E)

(D,E)



Q Minimise the following DFA using myHill Nerode Theorem



	p	q	r	s	t	u
p						
q	✓					
r	✓	✓				
s	✓	✓	✓			
t	✓	✓		✓		
u	✓		✓	✓	✓	

~~PR~~ Pr :

$\delta(P, a) = q$ (q, s) - nM

$\delta(r, a) = s$

$\delta(P, b) = u$ (u, s) - nM

$\delta(r, b) = s$

(If any of them is marked, we can mark PR)

Sq

$\delta(s, a) = s$ (s, r) - M

$\delta(q, a) = r$

$\delta(s, b) = s$ (s, t) - M

$\delta(q, b) = t$

(so we can mark SQ)

Pt

$\delta(P, a) = q$ (q, s) - M

$\delta(t, a) = s$

$\delta(P, b) = u$ (u, s) - NM.

$\delta(t, b) = s$

we can mark Pt

rt

$\delta(r,a) = s$ (s,s) - NM

$\delta(t,a) = s$

$\delta(r,b) = s$ (s,s) NM)

$\delta(t,b) = s$

t,u

$\delta(t,a) = r$ (r,t) - NM

$\delta(u,a) = t$

$\delta(t,b) = t$ (r,t) - NM

$\delta(u,b)$ r

su

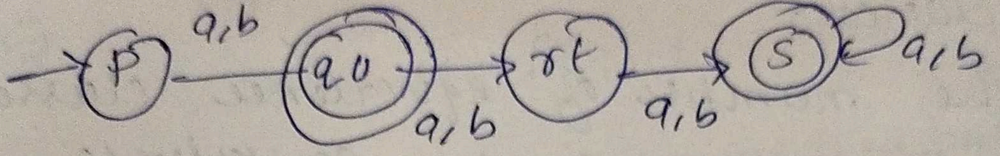
$\delta(s,a) = s$ (s,t) - M

$\delta(u,a) = t$

$\delta(s,b) = s$ (s,r) - M

$\delta(u,b) = r$

(Pr) can be marked



Myhill Nerode Theorem (MNR)

Let L in Σ^* be a regular language & $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA for L with no inaccessible states. The automaton M induces an equivalence relations R on Σ^* defined by, ~~x rela~~

$$xRy \Rightarrow \delta(q_0, x) = \delta(q_0, y)$$

This relation R is an equivalence relation as it is reflexive, symmetric & transitive.

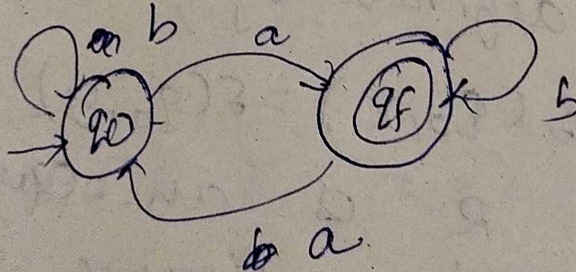
1. xRx
2. $xRy \Rightarrow yRx$
3. xRy & $yRz \Rightarrow xRz$.

The equivalence relation R, partitions the underline Σ^* in classes which are called equivalence classes.

The number of equivalence classes is the index of the equivalence relation.

In the case of myhill Nerode relation, the index is of atmost the number of states in the finite automata

eg:- Consider the given DFA, M over $\Sigma = \{a, b\}$ an equivalence relation $x R y \Rightarrow q_0, x = q_0, y$



The DFA has 2 states q_0 & q_f, so the equivalence R o Σ^* partitions into 2 equivalence classes c_1 & c_2 as follows,

C_1 = set of all strings which satisfies $\delta(q_0, x) = \delta(q_0, y) = q_0$

$C_1 = \{b, aa, aba, baa, baba \ldots\}$

C_2 = set of all strings that satisfies $\delta(q_0, x) = \delta(q_0, y) = q_s$

$C_2 = \{ba, a, ab, abb, abaa \ldots\}$

Here the index of equivalance relation is 2 (no. of equivalance classes)

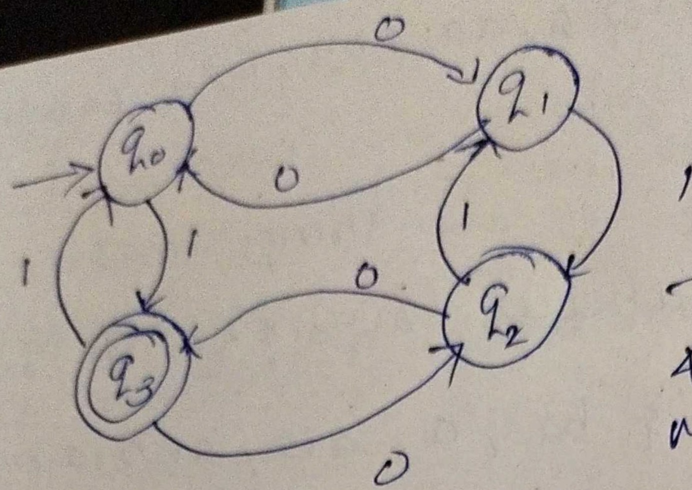
Qn1. Show the equivalence classes of the cannonical Myhill-Nerode Relation for the language of binary strings with odd number of 1's & even no. of 0's

Qn. Write the equivalence classes of the cannonical Myhill Nerode relation for the language.

$L = \{x \in \{a, b\}^* \mid x$ ends with 3 consecutive a

~~L=~~

1.)



This DFA has 4 states, so MNR partitions Σ^* into 4 equivalence classes.

~~class 1 = C_1 = {~~

class 1 = C_1 = $\{x \in \Sigma^* \mid \delta(q_0, x) = q_0\}$

$C_1 = \{0110, 00, 0110, \epsilon 01 \ldots\}$

class 2, C_2 = $\{x \in \Sigma^* \mid \delta(q_0, x) = q_1\}$

$C_2 = \{0, 000, 011, \ldots\}$

class 3, C_3 = $\{x \in \Sigma^* \mid \delta(q_0, x) = q_2\}$

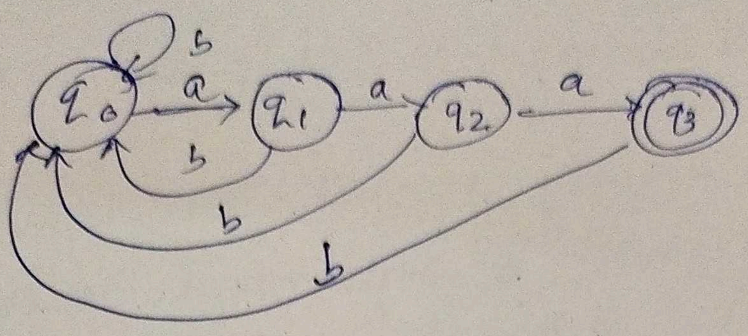
$C_3 = \{01, 0100, \ldots\}$

class 4, C_4 = $\{x \in \Sigma^* \mid \delta(q_0, x) = q_3\}$

$C_4 = \{1, 111, 010 \ldots\}$

Index = 4.

Q-2.



This DFA has 4 states, so MNR partitions Σ^* into 4 equivalence classes,

class 1, $C_1 = \{x \in \Sigma^* \mid \delta(q_0, x) = q_0\}$

$C_1 = \{b, bbb, bb, bab, ab, \epsilon \ldots\}$

class 2 = $C_2 = \{x \in \Sigma^* \mid \delta(q_0, x) = q_1\}$

$C_2 = \{a, ba, bba, aba, \ldots\}$

Class 3, $C_3 = \{x \in \Sigma^* \mid \delta(q_0, x) = q_2\}$

$C_3 = \{baa, aa, bbaa, babaa, \ldots\}$

Class 4, $C_4 = \{x \in \Sigma^* \mid \delta(q_0, x) = q_3\}$

$C_4 = \{aaa, baaa, bbaaa, babaab, aaa, \ldots\}$

Index = 4

Properties of Myhill-Nerode Relations

In addition to the properties, reflexive, symmetric & transitive. The Myhill-Nerode relation satisfies other properties,

1) It is a right congruence
i.e, for any $x, y \in \Sigma^*$ and $a \in \Sigma$

$$xRy \Rightarrow xaRya.$$

To prove this, assume xRy

then $\delta(q_0, xa) = \delta(\delta(q_0, x), a)$

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

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

2) It refines L from Σ^*,

For any $x, y \in \Sigma^*$,

$xRy \Rightarrow$ ~~xR~~ $x \in L \Rightarrow y \in L$

i.e, if x is accepted by M, y is also accepted.

if x is rejected by M y is also rejected.

3). It is of finite index,

The myhill-Nerode relation has ~~only~~ ~~finitely~~ finite equivalence classes because there is exactly one equivalence class

$\{x \in \Sigma^* / \delta(q_0, x) = q\}$

corresponding to each state $q(M)$.

Myhill-Nerode theorem (MNT)

Myhill Nerode theorem states that the following 3 statements are equivalent. ~~The set~~

1. The set L is in Σ^* is accepted by some finite state automata.

2. L is the union of some of the equivalence classes of right invariant equivalence relation of

finite index:

3. Let equivalence relation R_L be defined by xRy if and only if for all z in Σ^* xz is in L exactly when yz is in L. Then R_L is of finite index.

Equivalence of NFA with & without ε-moves:

Theorem:-

If L is accepted by an NFA with ~~ε-tr~~ ε-transitions, then L is accepted by an NFA without ε-transitions

Proof:- Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA with ε-transitions

Then construct $M' = (Q, \Sigma, \delta_N', q_0, F_N)$ be an NFA without ε-transitions

where $F_N = \{ F \cup \{q_0\}$, if ε-closure (q_0) cont where $F_N = \begin{cases} F \cup \{q_0\}, & \text{if } \epsilon\text{-closure}(q_0) \text{ contains a state of } F \\ F \end{cases}$

and $\delta(q, a)$ is $\delta'(q, a)$ for q in Q and a in Σ.

~~We have to prove, $\hat{\delta}(q_0, w) = \hat{\delta}_N(q_0, w)$~~

By mathematical induction, we have to prove that $\hat{\delta}(q_0, w) = \hat{\delta}_N(q_0, w)$.

Base: $|w| = 1$.

Let $w = a$.

$\hat{\delta}_N(q_0, w) = \hat{\delta}(q_0, a)$

So, as we know, $\hat{\delta}(q_0, a) = \delta_N(q_0, a)$
$= \hat{\delta}_N(q_0, a)$

By induction hypothesis

Assume this for all strings of length x, $\hat{\delta}_N(q_0, x) = \hat{\delta}(q_0, x) = \{P_1, P_2, P_3 \ldots P_k\}$

Induction steps:-

$|w| >= x$

Let $w = xa$.

Then, $\hat{\delta}_N(q_0, w) = \hat{\delta}_N(q_0, xa) =$

$\hat{\delta}_N(\hat{\delta}_N(q_0, x), a) =$

$\delta_N(\{P_1, P_2, P_3 \ldots P_K\}, a)$

$= \bigcup_{i=1}^{K} \delta_N(P_i, a)$

$\hat{\delta}(q_0, w) = \hat{\delta}(q_0, xa)$

$= \hat{\delta}(\hat{\delta}(q_0, x), a)$

$= \delta(\{P_1, P_2, P_3 \ldots P_K\}, a)$

$= \bigcup_{i=1}^{K} \delta(P_i, a).$

Thus, $\delta(q_0, wa) = \delta_N(q_0, w)$

MODULE - 3

Context Free Grammers & Languages

Type 2 languages are also called context free languages.
This languages are recognized by push down automata.

Context Free Grammer. (CFG)

A grammer $G = (V, T, P, S)$ is said to be context free if all the productions of P have the form $A \to \alpha$ where A is a non-terminal or variable. i.e. $A \in V$.
Then α is the string of symbols from $(V \cup T)^*$

V & T are finite set of variables or terminals respectively.

Every regular grammer is a context ~~f~~ free grammer.

Eg: $S \rightarrow ASA \mid BSB \mid a \mid b$

$A \rightarrow a$

$B \rightarrow b$.

where S is the start symbol

From the above productions,

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

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

$T = \{a, b\}$

$P = \{S \rightarrow ASA \mid BSB \mid a \mid b,\ A \rightarrow a,\ B \rightarrow b\}$

$S = S$

Context Free Languages (CFL):

The language generated by context free grammar is context free languages.

It is a set of strings that can be derived from a context free grammars.

Formal Notation:

The language generated by 'G' is

$L(G) = \{w \mid w \text{ is in } T^* \ \&\ S \overset{+}{\Rightarrow} w\}$.

i.e w is a string in L(G) if

1) The string consists of only terminals

2) Strings can be derived from S

eg:- Consider the following CFL

$S \rightarrow aA / bB$

$A \rightarrow x$

$B \rightarrow y$

Find $L(G)$?

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

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

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

$P = \{S \rightarrow aA / bB, A \rightarrow x, B \rightarrow y\}$

$S = S$

$S \Rightarrow aA$	$S \Rightarrow bB$
$\Rightarrow ax$	$\Rightarrow by$

$L(G) = \{ax, by\}$

Q. Consider the $G = (\{S\}, \{a, b\}, P, S)$ with productions $S \rightarrow aSa / bSb / \epsilon$

Find $L(G)$?

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

$V = \{S\}$

$T = \{a, b\}$

$P = \{S \to aSa / bSb / \varepsilon\}$

$S = S$

$S \Rightarrow aSa$ | $S \Rightarrow bSb$ | $S \Rightarrow \varepsilon$

$\Rightarrow aaSaa$ | $S \Rightarrow bbSbb$

$\Rightarrow aaaa$ | $\Rightarrow bbbb$

$S \Rightarrow aSa$ | $S \Rightarrow bSb$

$\Rightarrow aa$ | $\Rightarrow bb$

$\cancel{L = \{a^n b^n}$

$L = \{\{a^n, b^n\}, \varepsilon\}$

$S \Rightarrow aSa$

$\Rightarrow aa$

$S \Rightarrow aSa$

$\Rightarrow aaSaa$

$\Rightarrow aaaSaaa$

$\Rightarrow aaabSbaaa$

$\Rightarrow aaabbaaa$

$S \Rightarrow bSb$

$\Rightarrow baS\cancel{b}ab$

$\Rightarrow babSbab$

$\Rightarrow babbab$

$S \Rightarrow bSb$

$\Rightarrow bbSbb$

$\Rightarrow bbaSabb$

$= bbaabb$

$L(G) = \{\epsilon, aa, bb, aaabbaaa \ldots\}$

$\therefore L(G) = \{ww^R \mid w \in (a,b)^*\}$

Q Design a context free grammer for the language.

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

→ $L = \{\epsilon, ab, a^2b^2, a^3b^3 \ldots\}$

$n = 0$ $S \to \epsilon$

$n > 0$ $\cancel{S \to ab}$ $S \to aSb$

$S \Rightarrow aSb$

$\Rightarrow ab$

$S \Rightarrow aSb$

$\Rightarrow aaSbb$

$\Rightarrow aabb$

$S \Rightarrow aSb$

$\Rightarrow aaSbb$

$\Rightarrow aaaSbbb$

$\Rightarrow aaabbb$

$G = (\{S\}, \{a, b\}, \{S \to \epsilon / \cancel{S} \; aSb\},$

$S)$

Q. Design a CFG for the language
$L = \{a^n b^{2n} \mid n \geq 0\}$

$\rightarrow L = \{\epsilon, ab^2, a^2b^4, a^3b^6 \ldots\}$

$n = 0 \qquad S \to \epsilon$

$n > 0 \qquad S \to aSb^2$

$S \Rightarrow aSb^2$
$\Rightarrow aaSb^2b^2$
$\Rightarrow aab^2b^2$

$S \Rightarrow aSb^2$
$\Rightarrow aaSb^2b^2$
$\Rightarrow aaaSb^2b^2b^2$

$S \Rightarrow aSb^2$
$\Rightarrow ab^2$

$G = \{ \{S\}, \{a, b\}, S \to \epsilon / aSb^2,$

$S\}$

Q. Find CFG for the language $a^n b^m$ when $n \neq m$.

$\rightarrow L = \{\epsilon, a, b, ab^2, ba^2b, a^2, b^2,$

→ Case 1

$S \to AC$

$C \to aCb/\lambda$

$A \to aA/a$

Case 2

$S \to CB$

$C \to aCb/\lambda$

$B \to bB/b$

Resulting gramm is

$S \to AC/CB$

$C \to aCb/\lambda$

$A \to aA/a$ & $B \to bB/b$.

$G = \{ \{S, A, B, C\}, \{a, b\},$
$S, \{ \cancel{A} S \to AC/CB, C \to aCb/\lambda,$
$A \to aA/a, B \to bB/b\} \}$

Derivations

It is a Mechanism to check whether ~~str~~ a string w belongs to a content free language.

It is represented by a $\Rightarrow$.

Inorder to restrict the no of choices of replacement of variables, there is

2 types of derivations
* Leftmost derivation
* Rightmost derivation.

A derivation is said to be leftmost if in each step the leftmost variable in the sentential form is replaced it is represented by $\underset{lm}{\Longrightarrow}$ or $\overset{*}{\underset{lm}{\Longrightarrow}}$

A derivation in which rightmost variable is replaced in each step is called rightmost derivation. It is represented by $\underset{rm}{\Longrightarrow}$ or $\overset{*}{\underset{rm}{\Longrightarrow}}$

eg: Consider the grammar G with productions

$S \to aAB$
$A \to bBb$
$B \to A | \epsilon$

* The leftmost derivation of the string abbbb is

$$S \Rightarrow a\underline{A}B \quad (\text{using } S \to aAB)$$
$$\Rightarrow a b\underline{B}bB \quad (A \to bBb)$$
$$\Rightarrow ab\underline{A}bB \quad (B \to A)$$
$$\Rightarrow abb\underline{B}bbB \quad (A \to bBb)$$
$$\Rightarrow abbbb\underline{B} \quad (B \to \epsilon)$$
$$\Rightarrow abbbb \quad (B \to \epsilon)$$

3) The rightmost derivation abbbb

$S \Rightarrow aA\underline{B}$ $(S \to aAB)$

$\Rightarrow aA\underline{A}$ $(B \to A)$

$\Rightarrow aAb\underline{B}b$ $(A \to bBb)$

$\Rightarrow a\underline{A}bb$ $(B \to \epsilon)$

$\Rightarrow ab\underline{B}bbb$ $(A \to bBb)$

$\Rightarrow \underline{abbbb}$ $(B \to \epsilon)$

Same string ~~for~~ by rightmost & leftmost derivation.

Q- ~~S.T.~~ ST $E*(E+E)$ is a right sentential form.

Pdn rules are.

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

→ $E \Rightarrow E * E$

$\Rightarrow E * (E)$

$\Rightarrow E * (E+E)$

Derivation Tree / Parse Tree / Syntax Tree / Generation Tree / Production Tree

The representation for derivation

a derivation of context free Grammar

is called Parse Tree.
It is another way of showing derivations independent of the order in which the productions are used.

A derivation tree is an ordered tree in which nodes are labelled with leftside of the pdns ie in which the children of a node represents its correspon-ding rightside of pdn.

In derivation tree, root is labelled with start symbol ie the leaves are labelled with terminal.

<u>Constructing Parse tree</u>

Let $G = (V, T, P, S)$ be a CFG, An ordered tree is a derivation tree of G. If it has the following properties,

(1) The root is labelled by a start symbol (S)

(2) Every leaf has a label which is a Terminal or $\lambda(\varepsilon)$

(3) Every interior vertex has a label from V.

(4) If a vertex has a label $A \in V$ and its children are labelled, $A \in V$ ie its children are labelled $a_1, a_2 \ldots a_n$ then P must contain a

Production of the form:

$A \rightarrow a_1, a_2 \ldots a_m$

(5) A ~~x~~. leaf labelled λ has no siblings.

A tree that has properties (3), (4) & (5) but in which (1) doesn't necessarily hold & in which property (2) is replaced by.

2a) Every leaf has a label from $V \cup T \cup \{\lambda\}$ is said to be partial derivation tree.

Yield of a Derivation Tree

Yield of a derivation tree is a string of symbols obtained by the concatenation of labels of leaves from left to right omitting any λ's encountered.

Yield is the string of terminals in the order they are encountered when the tree is traversed in a depth first manner.

Yield of a derivation tree is a sentential form in a.

Subtree of a derivation tree ?

A subtree of a derivation tree T is a tree:

1) whose root is some vertex v of T

2) whose vertices are the descendants of v together with their labels i.e

3) whose edges are those connecting the descendants of v

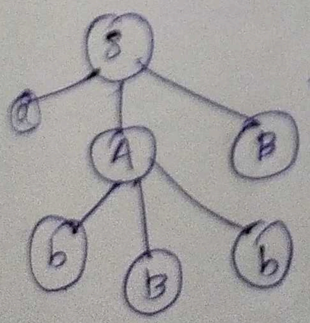
A subtree of a derivation tree looks just like a derivation tree, except that the label of the root may not be the start symbol of the grammar.

eg. Consider the grammar G with production:

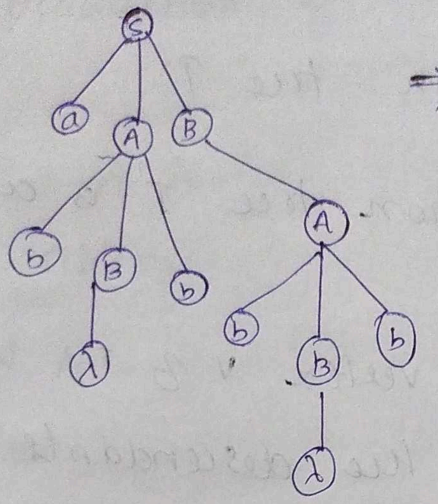
$S \rightarrow aAB$

$A \rightarrow bBb$

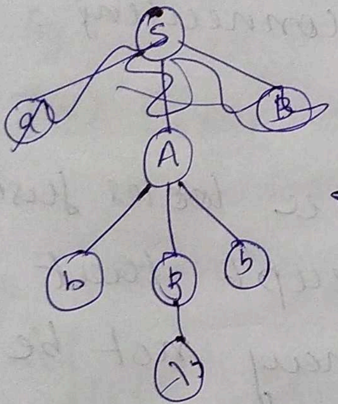
$B \rightarrow A / \lambda$



$\Rightarrow$ This is the partial derivation tree for the above grammar. Here abBbB is the yield of this tree. This in the sentential form of G.



⇒ This is the ~~partial~~ derivation tree for the above grammer. Here abbbb is the yield of this tree. This is the sentential form of ~~G~~ L(G)



⇒ This is the subtree of the above derivation Tree.

(2) Let G be the grammer with

$S \to 0B \mid 1A$, $A \to 0 \mid 0S \mid 1AA$; $B \to 1 \mid 1S \mid 0BB$

For the string 00110101 Find:

(a) Leftmost derivation

(b) Right most derivation

(c) Derivation tree.

→ (a) $S \Rightarrow 0B$

$\Rightarrow 00BB$

$\Rightarrow 001B$

$\Rightarrow 0011S$

$\Rightarrow 00110B$

$\Rightarrow 001101S$

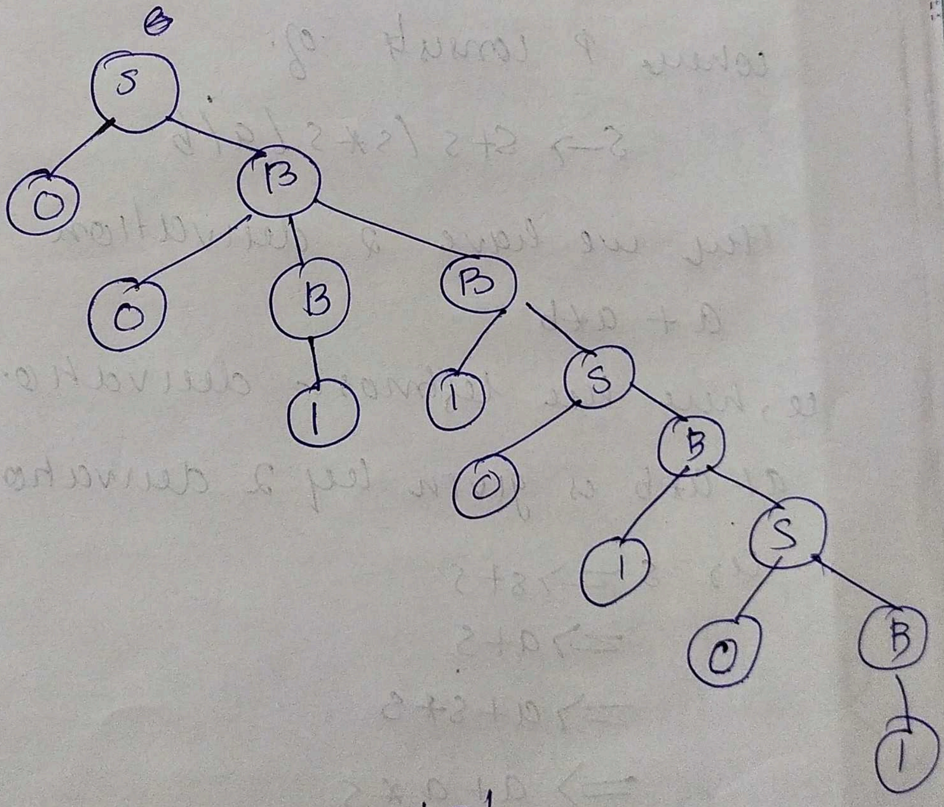
$\Rightarrow 0011010B$

$\Rightarrow 00110101$

(b) ~~$S \Rightarrow 0B$~~
~~$\Rightarrow 00BB$~~
~~$\Rightarrow 001B$~~
~~$\Rightarrow 0011S$~~
~~$\Rightarrow 00110B$~~
~~$\Rightarrow 001101S$~~
~~$\Rightarrow$~~

(b) $S \Rightarrow 0B$
$\Rightarrow 00BB$
$\Rightarrow 00B1S$
$\Rightarrow 00B10B$
$\Rightarrow 00B101S$
$\Rightarrow 00B1010B$
$\Rightarrow 00B10101$
$\Rightarrow 00110101$

(c)



yield = 00110101

Ambiguity in CFG.

A terminal string $w \in L(G)$ is ambiguous if there exist 2 or derivation tree for w i.e word w has more than one leftmost derivation or right-most derivation.

A CFG for which every CFG is ambiguous is said to be an inherently ambiguous CFG.

eg:- $G = (\{S\}, \{a, b, +, *\}, P, S)$

where P consists of.

$S \to S+S \mid S*S \mid a \mid b$

Here we have 2 derivation trees for

$a + a*b$

i.e, here the leftmost derivation of $a+a*b$ is given by 2 derivation trees.

i.e, $S \Rightarrow S+S$

$\Rightarrow a+S$

$\Rightarrow a+S*S$

$\Rightarrow a+a*S$

$\Rightarrow a+a*b$

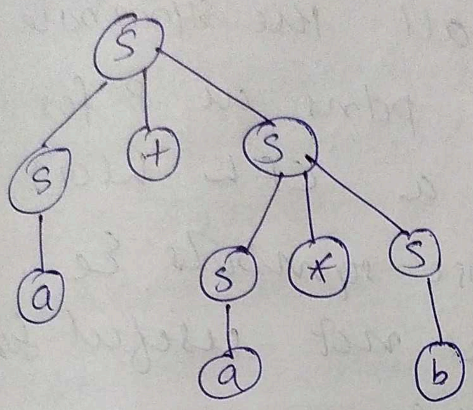
$S \Rightarrow S*S$

$\Rightarrow S+S*S$

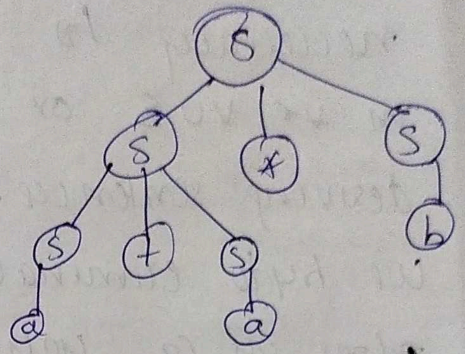
$\Rightarrow a+S*S$

$\Rightarrow a + a * b$

i.e., derivation trees.

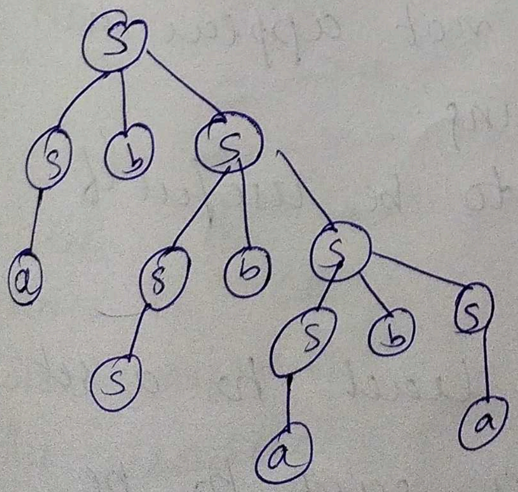


&

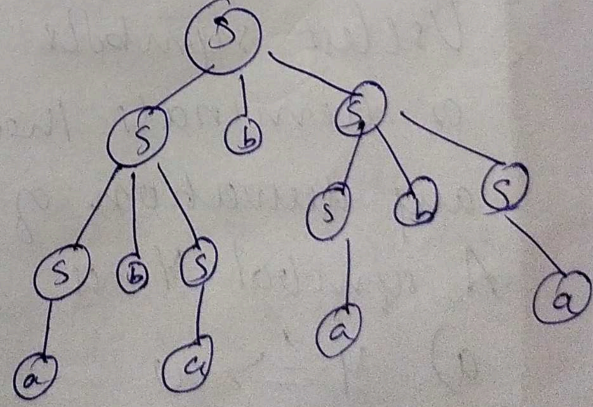


2.) If the CFG is $S \rightarrow SbS \mid a$, show that G is ambiguous.

→ To prove that G is ambiguous, we have to find a string $w \in L(G)$ which is ambiguous. Here we consider $w = abababa \in L(G)$. Here we get 2 derivation trees for w. Thus G is ambiguous.



$\Rightarrow$



Simplification of CFG

In a CFG G, it may not be necessary to use all the symbols in V ∪ Σ or all the prdns in P for deriving sentences. So in a CFL L(G), we try to eliminate those symbols ie prdns in G which are not useful for the derivation of sentences.

We can simplify a CFG ie produce an equivalent reduced CFG. This is done by:

a.) Eliminating useless symbols
b) Eliminating ε productions
c) Eliminating unit productions.

Eliminating useless symbols

Useless symbols are those non-terminals or terminals that do not appear in any derivation of a string.

A symbol Y is said to be useful if

a) $Y \xRightarrow{*} w$

that is, Y should lead to a set of terminals. Here Y is said to be 'Generating'.

b) If there is a derivation.

$$S \overset{*}{\Rightarrow} \alpha Y \beta \overset{*}{\Rightarrow} w$$

then Y is said to be reachable.

Thus a symbol is useful, if it is generating & useful.
Thus a symbol is useless, if it is non-generating & non-reachable.

eg:- Consider the CFG,

$S \longrightarrow AB/a$

$A \longrightarrow b$

where S is the start symbol.

→ $S \longrightarrow AB/a$

$A \longrightarrow b.$

~~In S ⟹~~ Here, B is non generating
since A derives b, S derived a but
B does not derive any string w.
So, we can eliminate $S \rightarrow AB$ from CFG.

Now ~~S~~ CFG becomes,

$S \longrightarrow a$

$A \longrightarrow b.$

Here A is non reachable symbol,
since it cannot be reached from the
start symbol S. So we can eliminate
the production, $A \rightarrow b$ from the CFG
So the reduced gramar is /

$S \longrightarrow a$

This gramar does not contain any useless symbols

Q.) Consider the CFG,

$S \rightarrow aB \mid bX$

$A \rightarrow BAd \mid bSX \mid a$

$B \rightarrow aSB \mid bBX$

$X \rightarrow SBD \mid aBx \mid ad.$

Eliminate usless symbols from this grammer.

→ Here B is ~~unreachable~~ non generating, Eliminating pdns containing B.

$S \rightarrow bx$

$A \rightarrow bsx \mid a$

$x \rightarrow ad.$

Now, A is non-reachable symbol

$S \rightarrow bx$

$x \rightarrow ad.$ ∴ This grammer does not contain any useless symbols.

Q. Consider the CFG

$A \rightarrow xy3 \mid xy33$

$x \rightarrow x3 \mid xy3$

$y \rightarrow xyy \mid xx3$

$z \rightarrow zy \mid 3.$

where A is start symbol.

Eliminate useless symbols from this grammer.

A X is non generating symbol.

$A \to xYz$

$Y \to xYy$

$z \to zy/z$.

Y is non generating.

$A \to xyz$

$z \to zy/z$.

z is non reachable.

$\therefore$ $A \to \underline{\underline{xyz}}$ This grammar does not contain any useless symbols.

Q) Consider the CFG,

$S \to aC/bSB$

$A \to bSCa$

$B \to aSB/bBC$

$C \to aBC/ad$. Eliminate useless symbols

$\to$ B is non generating.

$S \to aC$

$A \to bSCa$

$C \to ad$

A is non reachable

$S \to aC$

$C \to \underline{\underline{ad}}$ This grammar does not contain useless symbols.

Removal of Unit Productions

A unit production is defined as

$A \rightarrow B$

where A is a non-terminal and B is non-terminal.

Thus both LHS & RHS contain single non-terminals.

eg. Consider the CFG.

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow C / b$
$C \rightarrow D$
$D \rightarrow E$
$E \rightarrow a$

where S is the start symbol.

Eliminate unit productions from this grammar.

→ Unit productions here are:-

$B \rightarrow C$
$C \rightarrow D$
$D \rightarrow E$

are unit productions

To remove the production $B \rightarrow C$, check whether there exists a pdn whose LHS is C & RHS is a terminal.

No such pdn. exists

To remove the pdn, $C \to D$, check whether there exists a pdn. whose LHS is D & the RHS is a terminal.

No such pdn. exists.

To remove the pdn. $D \to E$, check whether there exists a pdn. whose LHS is E & RHS is a terminal.

There is a pdn, $E \to a$. So, remove $D \to E$ and add the pdn. $D \to a$, CFG become

$S \to AB$

$A \to a$

$B \to C/b$

$C \to D$

$D \to a$

$E \to a$.

Now remove pdn, $C \to D$ and add the pdn, $C \to a$, we get

$S \to AB$

$A \to a$

$B \to C/b$

$C \to a$

$D \to a$

$E \to a$

Now remove the pdn., $B \to C$ & add the pdn., $B \to a$, we get,

$S \to AB$

$A \to a$

$B \to a/b$

$C \to a$

$D \to a$

$E \to a$

Now the grammer contains no unit pdns.

From the above CFG, it is seen that the pdns. $C \to a$, $D \to a$, $E \to a$ are useless because the symbols C, D & E cannot be reached from start symbol. S

By eliminating these pdns., we get the CFG as

$S \to AB$

$A \to a$

$B \to a/b$

Above is a

Mod-2 Balance

Q.) Eliminate unit productions from this grammar.

$S \rightarrow a / b / Sa / Sb / S0 / S1$

$F \rightarrow S / (E)$

$T \rightarrow F / T * F$

$E \rightarrow T / E + T$

where E is the start symbol

→ Unit pdns are:

$F \rightarrow S$

$T \rightarrow F$

$E \rightarrow T$

The unit pdn $F \rightarrow S$ can be ~~replaced by~~ removed by rewriting it as $F \rightarrow a/b/Sa/Sb/S0/S1$

Now the CFG is

$S \rightarrow a / b / Sa / Sb / S0 / S1$

$F \rightarrow a / b / Sa / Sb / S0 / S1 / (E)$

$T \rightarrow F / T * F$

$E \rightarrow T / E + T$

The unit pdn. T→F can be removed by ~~a~~ rewriting it as ~~T~~

T→a|b| Sa|Sb| S0|S1 |(E)

Now the CFG is

S→a|b|Sa|Sb|S0|S1

F→a|b|Sa|Sb|S0|S1|(E)

T→a|b|Sa|Sb|S0|S1|(E)|T*F

E→T|E+T

The unit pdn E→T can be removed by rewriting it as ~~E→a~~

E→a|b|Sa|Sb|S0|S1|(E)|T*F

Now the CFG is:-

S→a|b|Sa|Sb|S0|S1

F→a|b|Sa|Sb|S0|S1|(E)

T→a|b|Sa|Sb|S0|S1|(E)|T*F|~~E+T~~

E→a|b|Sa|Sb|S0|S1|(E)|T*F|E+T

This is the CFG that does not contain ~~any~~ any unit productions

Q. Consider the CFG

S→A|bb

A→B|b

B→S|a

Where S is the start symbol.
Eliminate unit pdns from this gramar.

→ Unit pdns are

$S \to A$

$A \to B$

$B \to S$

→ ~~$S \to A$ generates $S \to b$~~

~~$S \to b / bb$~~

~~$A \to B / b$~~

~~$B \to S / a$~~

~~⇒ $S \to b / b / b$~~
~~a~~
~~$A \to B / b$~~

~~$B \to S / a$~~

~~$S \to b / bb$~~

~~$A \to B / b$~~

~~$B \to b / bb / a$ (for $S \to b/bb$)~~

~~S.~~
⇒ ~~$S \to b$~~ $/ bb$

$S \to A$ generates $S \to b$

$S \to A \to B$ generates $S \to a$

$A \to B$ generates $A \to a$

$A \to B \to S$ generates $A \to bb$

$B \to S$ generates $B \to bb$

$B \to S \to A$ generates $B \to b$.

The new CFG is

$S \to b \,|\, a \,|\, bb$

$A \to a \,|\, bb \,|\, b$

$B \to bb \,|\, b \,|\, a$

which contains no unit pdns.

Removal of ϵ pdns

Productions of the form $A \to \epsilon$ are called ϵ-productions.

We can eliminate ϵ-production from a grammar in the following way.

If $A \to \epsilon$ is a pdn. to be eliminated, then we look for all pdns whose RHS contains A & replace every occurrence of A at each these pdns. to be obtain non-ε-pdns. The result of non-εe pdns are added to grammer.

eg:- Consider the CFG

$S \to aA$

$A \to b/ε$.

where S is the start symbol. Eliminate epsilon pdns from the grammar.

→ Here $A \to εe$ is an epsilon pdn.

By the following, the abv procedure, put εe at place of A at RH of pdns, we get.

The pdn $S \to aA$ becomes $S \to a$

Then the CFG is

$S \to aA$

$S \to a$ OR $S \to aA/a$

$A \to b$ $A \to b$

OR This CFG does not contain any εe-pdns

2. Consider the CFG,

$S \rightarrow ABAC$

$A \rightarrow aA / \varepsilon$

$B \rightarrow bB / \varepsilon$

$C \rightarrow c$

where S is the start symbol. Eliminate epsilon pdns. from this grammer.

→ Epsilon productions in this CFG are $A \rightarrow \varepsilon$

$B \rightarrow \varepsilon$

To eliminate $A \rightarrow \varepsilon$, ~~repl~~ replace A with epsilon in the RHS of the pdns. , $S \rightarrow ABAC$, $A \rightarrow aA$.

For the pdns, $S \rightarrow ABAC$, replace A with epsilon one by one as, we get

$S \rightarrow BAC$

$S \rightarrow ABC$

$S \rightarrow BC$

for pdn $A \rightarrow aA$, we get

$A \rightarrow a$.

Now the grammar become,

$S \rightarrow ABAC \mid ABC \mid BAC \mid BC$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid \varepsilon$

$C \rightarrow aC$

To eliminate $B \rightarrow \varepsilon$, replace A with epsilon in the RHS of the pdn,

$S \rightarrow ABAC$, $B \rightarrow bB$,

$S \rightarrow ABAC \mid ABC \mid BAC \mid BC$,

for pdn $S \rightarrow ABAC$ ~~eplas~~ epsilon as, replace B with epsilon as,

we get

$S \rightarrow AAC \mid AC \mid C$

For the pdn, $B \rightarrow bB$, we get

$B \rightarrow b$

Now the grammar becoms:

$S \rightarrow ABAC \mid ABC \mid BAC \mid BC \mid AAC \mid AC \mid C$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

$C \rightarrow c$

The grammar does not contain any ε-pdn //

Q.) Consider the CFG

$S \rightarrow aSa$

$S \rightarrow bSb \mid e$.

Where S is the start symbol
Eliminate epsilon prdn from this grammar.

→ ~~S~~ ε-prdns are :-

$S \rightarrow e$.

So $S \rightarrow aSa$ becomes

$S \rightarrow aa$

So $S \rightarrow bSb$ becomes

$S \rightarrow bb$.

So the prdns are

$S \rightarrow aSa \mid bSb \mid aa \mid bb$

This grammar contains no ε - prdns

Q.) Consider the CFG

$S \rightarrow a / Xb / aYa$

$X \rightarrow Y / \varepsilon$

$Y \rightarrow b / X$

where S is the start symbol

Eleminate epsilon prdns. from this grammar.

→ ε-prdns are.

~~S → a+b~~

$X \rightarrow \varepsilon$

So, $Y \rightarrow bX$ become $Y \rightarrow b$.

$X \rightarrow Y / \varepsilon$ " $X \rightarrow Y$

$S \rightarrow aXb$ " $X \rightarrow b$

$S \rightarrow aYa$ " $S \rightarrow aa$

$(X \rightarrow Y \rightarrow \varepsilon)$

So, prdns are.

$S \rightarrow a / Xb / aYa / b / aa$

$X \rightarrow Y$

$Y \rightarrow b / X$

Now this ~~gramer~~ grammar does not contain epsilon pdns.

Exercises

) Simplify the context free gramm

) $S \to AB$

Normal forms for CFG

When the pdns in G satisfy certain restrictions then G is said to be in a normal form.

Two most common normal form of CFG are

(1) Chomsky Normal Form (CNF)

(2) Greibach Normal Form (GNF)

Chomsky Normal Form (CNF)

Any CFL without ε is generated by a grammar in which all pdns are of the form

$A \rightarrow BC$ or.

$A \rightarrow a$.

Here, A, B, C are the non-terminals & 'a' is a terminal symbol.

This is said to be CNF.

Basic Procedure for reduction to Chomsky Normal Form

~~Form~~

Step 1 :-

Elimination of null pdns & unit pdns.

Let the grammar thus obtained be $G = (V, T, P, S)$

Step 2 :-

Elimination of terminals on RHS :-

We define $G_1 = (V_1, T, P_1, S)$ where P_1 & V_1 are construction as follows.

a) * All pdns in P of the form $A \rightarrow BC$ & $A \rightarrow a$ are included in P_1

* All variables in V are included in V_1

b) Consider the pdn $A \rightarrow x_1 x_2 \ldots x_n$ with some terminals on R.H.S.

If ~~*~~ x_i is a terminal say a_i, add a new variable C_{a_i} to V_1 i.e $C_{a_i} \rightarrow a_i$ to P_1

In the pdn $A \rightarrow x_1, x_2, \ldots x_n$ every terminal on RHS is replaced by the corresponding new variable & the other variables on RHS are retained. The resulting pdn is added to P_1: ~~These~~ Thus, we get $G_1 = (V_1, T, P_1, S)$

<u>step3</u>:

Restricting the no. of variables on the RHS.

For any pdn in P_1, the RHS consists of either of 2 or more variables. We define.

$G_2 = (V_2, T, P_2, S)$ as follows.

a) All pdns in P_1 are added to P_2. If they are in the required form

b) Consider $A \rightarrow A_1A_2 \dots A_m$ where $m \geq 3$.

We introduce new pdns $A \rightarrow A_1C_1$, $C_1 \rightarrow A_2C_2$ .. $C_{m-2} \rightarrow A_{m-1}A_m$ & add new variables $C_1 C_2 \dots C_{m-2}$.

These are added to P_2 & V_2 respectively. Thus we get G_2 in CNF.

1. Reduce the following grammar G into CNF.

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

where P is given as follows.

$S \rightarrow aAD$

$A \rightarrow aB / bAB$

$B \rightarrow b$

$D \rightarrow d$.

→ Step 1 : No null pdns & unit pdns.

Step 2 : Let $G_1 = (V_1, T, P_1, S)$ where P_1 & V_1 are constructed as follows.

a) $P_1 = \{B \rightarrow b, D \rightarrow d\}$

$V_1 = \{S, A, B, D\}$

b) $S \rightarrow aAdD$ give rise to

$S \rightarrow C_a AD$

$C_a \rightarrow a$.

$A \rightarrow aB \Rightarrow A \rightarrow C_a B$

$A \rightarrow bAB \Rightarrow A \rightarrow C_b AB$

$C_b \rightarrow b$

Now $V_1 = \{S, A, B, D, C_a, C_b\}$

$P_1 = \{S \rightarrow C_a AD,\ A \rightarrow C_a B,$
$A \rightarrow C_b AB,\ B \rightarrow b,\ D \rightarrow d,$
$C_a \rightarrow a,\ C_b \rightarrow b\}$

Step 3:-

Let $G_2 = (V_2, T, P_2, S)$

a) $P_2 = \{A \rightarrow C_a B$
$B \rightarrow b$
$D \rightarrow d$
$C_a \rightarrow a$
$C_b \rightarrow b\}$

b) $S \rightarrow C_a AD$ is replaced by

$S \rightarrow C_a C_1$

$C_1 \rightarrow AD$

$A \to C_b AB$ is replaced by

$A \to C_b C_2$

$C_2 \to AB$

Let $G_2 = (\{S, A, B, D, C_a, C_b, C_1, C_2\},$
$T, P_2, S)$

where P_2 consists of

$S \to C_a C_1$

$A \to C_a B \mid C_b C_2$

$B \to b$

$D \to d$

$C_a \to a$

$C_b \to b$

$C_1 \to AD$

$C_2 \to AB$

2. Reduce the following grammar into CNF. $G = (\{S, A\}, \{a, b\}, P, S)$ where P is the given as follows,

$S \to a \mid absb \mid aAb$

$A \to bS \mid aAAb$

→ Step 1 :-

No unit pdns & null pdn.

Step 2 :-

Let $G_1 = (V, T, P_1, S)$ where V & P_1 are constructed as follows,

a) $P_1 = \{S \rightarrow a\}$, $V_1 = \{S, A\}$

b) $S \rightarrow absb \Rightarrow S \rightarrow C_a C_b S C_b$

$C_a \rightarrow a$

$C_b \rightarrow b$

$S \rightarrow aAb \Rightarrow S \rightarrow C_a A C_b$

$A \rightarrow bS \Rightarrow A \rightarrow C_b S$

$A \rightarrow aAAb \Rightarrow A \rightarrow C_a A A C_b$

$V_1 = \{S, A, C_a, C_b\}$

$P_1 = \{S \rightarrow a, S \rightarrow C_a C_b S C_b, S \rightarrow C_a A C_b,$

$A \rightarrow C_b S, A \rightarrow C_a A A C_b,$

$C_a \rightarrow a, C_b \rightarrow b\}$

Step3 :-

Let $G_2 = (V_2, T, P_2, S)$

a) $P_2 = \{S \rightarrow a, A \rightarrow C_b S, C_a \rightarrow a,$

$C_b \rightarrow b\}$

b) $S \rightarrow C_a C_b S C_b \Rightarrow S \rightarrow C_a C_1$

$C_1 \rightarrow C_b C_2$

$C_2 \rightarrow S C_b$

$S \rightarrow C_a A C_b \Rightarrow S \rightarrow C_a C_3$

$C_3 \rightarrow A C_b$

$A \to C_a A A C_b \Rightarrow A \to C_a C_4$

$C_4 \to A C_3$

Let $G_2 = (\{S, A, C_a, C_b, C_1, C_2, C_3, C_4\},$
$\{a, b\}, P_2, S)$

where P_2 consists of

$S \to a / C_a C_1 / C_a C_3$ $\quad C_1 \to C_b C_2$

~~Step 3~~ $\quad A \to C_b S / C_a C_4$ $\quad C_2 \to S C_b$

~~Let G~~ $\quad C_a \to a$ $\quad C_3 \to A C_b$

$C_b \to b$ $\quad C_4 \to A C_3$

3. Find the grammar in CNF equivalent to the grammar $S \to \sim S / [S \supset S] / b / a$ (S being the only variable)

→ Step 1 :-

No null or unit pdn.

Step 2 :

Let $G = (V_1, T, P_1, S)$

a) $P_1 = \{S \to p, S \to q\}$

$V_1 = \{S\}$

b) $S \to \sim S \implies S \to AS$

$A \to \sim$

$S \to [S \supset S] \implies S \to BSCSD$

$B \to C$

$C \to \supset$

$D \to]$

Now $P_1 = \{S \to p/q\ /AS/BSCSD,$

$A \to \sim,\ B \to [,$

$C \to \supset,\ D \to]\}$

$V_1 = \{S, A, B, C, D\}$

Step 3 :-

Let $G_0 = (V_2, T, P_2, S)$

a) $P_2 = \{S \to p/q/AS,\ A \to \sim,\ B \to [,$

$C \to],\ D \to]\}$

b) $S \rightarrow BSCSD \Rightarrow S \rightarrow BC_1$

$C_1 \rightarrow SC_2$

$C_2 \rightarrow CC_3$

$C_3 \rightarrow SD$

Now $G_2 = (\{S, A, B, C, D, C_1, C_2, C_3\},$
$\{p/q/\sim, [,),]\}, P_2, S)$

P_2 consists of

$S \rightarrow p/q / AS / BC_1$

$A \rightarrow \sim$ $\quad C_1 \rightarrow SC_2$

$B \rightarrow [$ $\quad C_2 \rightarrow CC_3$

$C \rightarrow)$ $\quad C_3 \rightarrow SD$

$D \rightarrow]$

4. Change the ~~to~~ following grammar into CNF

$S \rightarrow bA / aB$

$A \rightarrow bAA / aS / a$

$B \rightarrow aBB / bS / a$

Sol → Step 1 :-

No null pdns & unit pdns.

~~step~~

step 2:-

Let $G_1 = (V_1, T, P_1, S)$

a) $V_1 = \{S, A, B\}$

$P_1 = \{$ ~~S~~ $A \rightarrow a, B \rightarrow a\}$

b) $S \rightarrow bA \Rightarrow S \rightarrow C_bA$

$C_b \rightarrow b$

$S \rightarrow aB \Rightarrow S \rightarrow C_aB$

$C_a \rightarrow a$

$A \rightarrow aS \Rightarrow A \rightarrow C_aS$

$A \rightarrow bAA \Rightarrow A \rightarrow C_bAA$

$B \rightarrow aBB \Rightarrow B \rightarrow C_aBB$

$B \rightarrow bS \Rightarrow B \rightarrow C_bS$

Now,

$V_1 = \{S, A, B, C_a, C_b\}$

$P_1 = \{S \rightarrow C_bA \mid C_aB, A \rightarrow C_bAA \mid C_aS \mid a,$

$B \rightarrow C_aBB \mid C_bS \mid a, C_a \rightarrow a,$

$C_b \rightarrow b\}$

step 3 :-

Let $G_2 = (V_2, T, P_2, S)$

a) $P_2 = \{S \rightarrow C_b A \mid C_a B,\ A \rightarrow C_a S \mid a,$
$B \rightarrow C_b S \mid a,\ C_a \rightarrow a,\ C_b \rightarrow b\}$

b) $A \rightarrow C_b A A \Rightarrow A \rightarrow C_b C_1$
$C_1 \rightarrow AA$

$B \rightarrow C_a B B \rightarrow AB \rightarrow C_a C_2$
$C_2 \rightarrow BB$

NOW, $G_2 = (\{S, A, B, C_a, C_b, C_1, C_2\},$
$\{a, b\}, P_2, S)$

$P_2 = \{S \rightarrow C_b A \mid C_a B,\ A \rightarrow C_b C_1 \mid C_a S \mid a,$
$B \rightarrow C_a C_2 \mid C_b S \mid a,\ C_a \rightarrow a,\ C_b \rightarrow b,$
$C_1 \rightarrow AA,\ C_2 \rightarrow BB\}.$

G_2 in CNF is equivalent to the given grammer.

Greibach Normal Form.

If every prdn of CFG is of the form $A \to a\alpha / a$, then it is in GNF where $a \in T$ & $\alpha \in V^*$ is a string of non-terminals with no restrictions on length of V^*.

Procedure for reduction to GNF

Step 1 :-

Eliminate all null prdns & then construct a grammar G in CNF. Rename the variable of G as $A_1 A_2 \dots A_n$ with $S \to A_1$. We write G as $(\{A_1 A_2 \dots A_n\}, T, P, A_1)$. The prdns P are of the form.

$A_i \to A_j\alpha$. where $i<j$ / $i=j$ / $i>j$

Step 2

For the prdns of the form $A_i \to A_j\alpha$ where $i>j$, apply lemma1. so that all

pdns will be in the form $A_i \rightarrow A_j \alpha$
where $i<j$ / $i=j$.

Step 3

For the pdns of the form $A_i \rightarrow A_j \alpha$, where $i=j$, apply lemma 2, so that all pdns will be in the form
$A_i \rightarrow A_j \alpha$ where $i<j$.

Step 4

Starting from the highest no., replace the RHS non-terminal replace the RHS non-terminal with its alternative of the pdn with its alternative until all are in GNF.

Lemma 1 :-

If we have pdns of the form $A \rightarrow B\gamma$ and $B \rightarrow \beta_1 / \beta_2 \ldots / \beta_s$.
This is replaced as

$A \rightarrow \beta_1\gamma / \beta_2\gamma / \ldots / \beta_s\gamma$

This lemma is useful for deleting a variable B appearing as the first symbol on the RHS of some A-pdns provided no. of ~~B~~ B pdns has B as the 1^{st} symbol on RHS.

Lemma 2

If we have the pdns of the form
$A \to A\alpha_1 / A\alpha_2 / \dots / A\alpha_r / \beta_1 / \beta_2 / \dots / \beta_s$
Let 'z' be a new variable.
The above pdn is replaced as follows.

Lemma 2

If we have the pdns of the form.
$A \to A\alpha_1 / A\alpha_2 / \dots / A\alpha_r / \beta_1 / \beta_2 / \dots / \beta_s$
Let z' be a new variable. The above pdn is replaced as follows.

1. The ~~ϕ~~ set of A pdns are

$$A \to \beta_1 / \beta_2 / \dots \beta_s$$

$$A \to \beta_1 z / \beta_2 z / \dots \beta_s z$$

2. The set of Z pdns are

$Z \rightarrow \alpha_1 / \alpha_2 \dots \alpha_r$

$Z \rightarrow \alpha_1 Z / \alpha_2 Z / \dots \alpha_r Z$.

Problems

1. Reduce the following G into GNF.

$S \rightarrow AB, A \rightarrow BS / b, B \rightarrow SA / a$

→ Step 1

As the given G is in CNF i.e has no null pdns, we can omit step 1 i.e proceed to step 2 after renaming S, A, B as A_1, A_2, A_3 respt.

The pdns are,

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 / b$

$A_3 \rightarrow A_1 A_2 / a$

Step 2:

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 / b$ } This pdns are in req. form ($i<j$)

$A_3 \rightarrow a$

Apply lemma1 to $A_3 \rightarrow A_1 A_2$

$A_3 \rightarrow A_2 A_3 A_2$ ($\because A_1 \rightarrow A_2 A_3$

$i > j$

$A_3 \rightarrow A_3 A_1 A_3 A_2 / b A_3 A_2$ ($\because$ sta $A_2 \rightarrow A_3 A_1 / b$)

$i = j$

Now pdns are :-

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 / b$

$A_3 \rightarrow A_3 A_1 A_3 A_2 / b A_3 A_2 / a$

Step 3

As we have

$A_3 \rightarrow A_3 A_1 A_3 A_2$

Apply lemma 2

$i = j$

$A_3 \rightarrow A_3 A_1 A_3 A_2 / b A_3 A_2 / a$.

Let Z be a new variable.

The resulting pdns are

$A_3 \rightarrow a / b A_3 A_2$

$A_3 \rightarrow aZ / b A_3 A_2 Z$

$Z \rightarrow A_1 A_3 A_2$

$Z \rightarrow A_1 A_3 A_2 Z$

Step 4

4 a) A_3 pdns are:

$A_3 \rightarrow a / bA_3 A_2 / aZ / bA_3 A_2 Z$ — (1)

b) Among A_2 pdns retain $A_2 \rightarrow b$

& eliminate

$A_2 \rightarrow A_3 A_1$ using lemma 1

$A_2 \rightarrow aA_1 / bA_3 A_2 A_1 / aZA_1 / bA_3 A_2 Z A_1$

Modified A_2 pdns are:-

$A_2 \rightarrow aA_1 / bA_3 A_2 A_1 / aZA_1 / bA_3 A_2 Z A_1 / b$ — (2)

c) Eliminate $A_1 \rightarrow A_2 A_3$ using lemma 1

The resulting pdns are,

$A_1 \rightarrow aA_1 A_3 / bA_3 A_2 A_1 A_3 / aZA_1 A_3 / bA_3 A_2$
$Z A_1 A_3 / bA_3$ — (3)

d) The Z-pdns to be modified are.

$Z \rightarrow A_1 A_3 A_2 / A_1 A_3 A_2 Z$

We apply lemma 1 & get, modified Z pdns

$Z \rightarrow aA_1A_3A_2A_3 / aA_1A_3A_3A_2Z$

$Z \rightarrow bA_3A_2A_1A_3A_3A_2 / bA_3A_2A_1A_3A_3A_2Z$

$Z \rightarrow aZA_1A_3A_3A_2 / aZA_1A_3A_3A_2Z$

$Z \rightarrow bA_3A_2ZA_1A_3A_3A_2 / bA_3A_2ZA_1A_3A_3A_2Z$ — (4)

$Z \rightarrow bA_3A_3A_2 / bA_3A_3A_2Z$

The required grammar in GNF is given by eqns. (1) → (4)

2. Construct a grammar in GNF equivalent to the grammar $S \rightarrow AA / a$, $A \rightarrow SS / b$.

→ Step 1 :-

As the given G is in CNF & has no null pdns, we can omit step (1) & proceed to step 2. After renaming S & A as A_1 & A_2 respectively

Now the pdns are :-

$A_1 \rightarrow A_2A_2 / a$

$A_2 \rightarrow A_1A_1 / b$

Step 2 :-

Following pdns are in the required form.

$A_1 \longrightarrow A_2 A_2 \ / a \ ; \ i < j$

$A_2 \longrightarrow b$

Apply lemma 1 to $A_2 \longrightarrow A_1 A_1$ (i>j)

$A_2 \longrightarrow A_2 A_2 A_1 / a A_1$

Thus A_2 pdns are :

$A_2 \longrightarrow A_2 A_2 A_1 / a A_1 / b$

Step 3

We have to apply lemma 2 to A_2 pdns $A_2 \longrightarrow A_2 A_2 A_1 / a A_1 / b$.

Let z' be a new variable.

The resulting pdns are :-

$A_2 \longrightarrow a A_1 / b$

$A_2 \longrightarrow a A_1 z / b z$

$z \longrightarrow A_2 A_1$

$z \longrightarrow A_2 A_1 z$

Step 4 :-

a) A_2 pdns are.

$A_2 \rightarrow aA_1 / b / aA_1 z / bz$ — (1)

b) Among A_1 pdns, retain $A_1 \rightarrow a$.

Elemenate $A_1 \rightarrow A_2 A_2$ using lemma 1.

The resulting pdns are :

$A_1 \rightarrow aA_1 A_2 / bA_2 / a\, aA_1 z A_2 / bzA_2 / a$ — (2)

c) z-pdns to be modified are

$z \rightarrow A_2 A_1 / A_2 A_1 z$.

Apply lemma 1, we get,

$z \rightarrow aA_1 A_1 / bA_1 / aA_1 z A_1 / bzA_1$

$z \rightarrow aA_1 A_1 z / bA_1 z / aA_1 z A_1 z /$

$bzA_1 z$ — (3)

The eq. grammar in GNF is given by eqn (1) → (3)

3. Consider CFG. $S \rightarrow XY$, $X \rightarrow YS | 1$, $Y \rightarrow SX | 0$. Convert to GNF.

→ Step 1:

The given CFG is in CNF i.e. has no null pdns.

∴ The pdns are.

$$A_1 \rightarrow A_2 A_3$$
$$A_2 \rightarrow A_3 A_1 / 1$$
$$A_3 \rightarrow A_1 A_2 / 0$$

Step 2:-

$A_1 \rightarrow A_2 A_3$
$A_2 \rightarrow A_3 A_1 / 1$ } = are in eq. form ie i<j

$A_3 \rightarrow 0$

Apply lemma 1 to $A_3 \rightarrow A_1 A_2$.

$A_3 \rightarrow A_2 A_3 A_2$ ($\because A_1 \rightarrow A_2 A_3$)

i>j

$A_3 \rightarrow A_3 A_1 A_3 A_2 / 1 A_3 A_2$ ($\because A_2 \rightarrow A_3 A_1 /$

i=j

Now pdns are :-

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 / 1$

$A_3 \rightarrow A_3 A_1 A_3 A_2 / 1 A_3 A_2 / 0$

<u>step 3</u>

As we have,

$A_3 \rightarrow A_3 A_1 A_3 A_2$ ~~$1A_3A_2$~~ apply lemma 2

$i = j$

$A_3 \rightarrow A_3 A_1 A_3 A_2 / 1 A_3 A_2 / 0$

Let Z be a new variable,

pdns are :-

$A_3 \rightarrow 1 A_3 A_2 / 0$

$A_3 \rightarrow 1 A_3 A_2 Z / 0 Z$

$Z \rightarrow A_1 A_3 A_2$

$Z \rightarrow A_1 A_3 A_2 Z$

<u>Step 4</u>

a) A_3 pdns are :-

$A_3 \rightarrow 0 \mid 1A_3A_2 \mid 0Z \mid 1A_3A_2Z$ — (1)

b) In A_2 pdns, retain $A_2 \rightarrow 1$

& modify $A_2 \rightarrow A_3A_1$ using lemma 1.

$A_2 \rightarrow 0A_1 \mid 1A_3A_2A_1 \mid 0ZA_1 \mid 1A_3A_2ZA_1$

∴ Now A_2 pdns are.

$A_2 \rightarrow 0A_1 \mid 1A_3A_2A_1 \mid 0ZA_1 \mid 1A_3A_2ZA_1 \mid 1$ — (2)

c) In A_1 pdns, modify $A_1 \rightarrow A_2A_3$ using lemma 1

Pdns are.

$A_1 \rightarrow 0A_1A_3 \mid 1A_3A_2A_1A_3 \mid 0ZA_1A_3 \mid 1A_3A_2ZA_1A_3 \mid 1A_3$ — (3)

d) Modyfing Z pdns :-

$Z \rightarrow A_1A_3A_2 \mid A_1A_3A_2Z$

$Z \rightarrow 0A_1A_3A_3A_2 \mid 0A_1A_3A_3A_2Z \mid 1A_3A_2A_1A_3A_3A_2 \mid 1A_3A_2A_1A_3A_3A_2Z \mid 0ZA_1A_3A_3A_2 \mid 0ZA_1A_3A_3A_2Z \mid 1A_3A_2ZA_1A_3A_3A_2 \mid$

$1A_3A_2 \; zA_1A_3A_3A_2z \,/\, 1A_3A_3A_2 \,/$

$1A_3A_3A_2z$

The req. grammer in GNF is given by eqns ①-③.

Pumping Lemma for context free Languages

Let 'L' be a CFL. Then we can find a natural no n such that

i) Every $z \in L$ with $|z| \geq n$ can be written as $uvwxy$ for some strings $u\,v\,w\,x\,y$.

ii) $|vx| \geq 1$

iii) $|vwx| \leq n$

iv) $uv^iwx^iy \in L$ for all $i \geq 0$.

Application of pumping lemma.

It is used to P.T ~~certain~~ certain languages are not context free.

steps for proving the given language is not context free

First we take a language 'L' is CFL. By applying pumping lemma, we get contradiction.

The steps considered for this are:

Assume 'L' is CFL. Let 'n' be a natural no obtained by using ~~PLT~~ pumping lemma.

2. choose a string $z \in L$ such that $|z| \geq n$. Using pumping lemma principle, $z = uvwxy$.

3. Find a suitable integer i, so that $uv^iwx^iy \notin L$. This is a contradiction. So L is not a CFL.

2. ~~A~~ Show that $L = \{a^i b^i c^i \mid i \geq 1\}$ is not a CFL.

→ Step 1 :-

Suppose L is a CFL & n is a constant

Step 2 :-

Assume $z = a^n b^n c^n$ where

where $|z| = |a^n b^n c^n| = 3n > n$.

Let $z = uvwxy$ where $|vx| \geq 1$ & $|vwx| \leq n$.

Here v & x the strings that get pumped could lie in $a^n b^n c^n$

Since $|vwx| \leq n$, it is not possible for vx to contain instances of a's & c's. If v & x consists of a's only, then uwy has n b's & n c's but fewer than n a's since $|vx| \geq 1$

Step 3:

Then $z = uv^i wx^i y$

$= \underset{u}{a^j}\,\underset{v}{a}\,\underset{w}{a^{n-j-2}}\,\underset{x}{a}\,\underbrace{b^n c^n}_{y}$

Let $i = 2$, then.

$uv^2wx^2y = a^j a^2 a^{n-j-2} a^2 b^n c^n$

$= a^{n+2} b^n c^n \notin L$

This is a contradiction

$L = \{a^i b^i c^i \mid i \geq 1\}$ is not CFL

2. show that $L = \{a^p \mid p$ is prime$\}$ is not CFL.

→ Step 1

Assume that L is context free.

Let 'n' be a natural no. obtained by using pumping lemma.

Step 2

Consider a string

$z = aaaaa \in L$.

where $|z| = |aaaaa| = 5 \geq n$

Using pumping lemma principle

$z = uvwxy$

<u>step 3 :-</u>

Let $i = 3$, then

$uv^iwx^iy = uv^3wx^3y = aa^3aa^3a$.

Here $|aa^3aa^3a|$, which is not a prime no.

This is a contradiction. So,

$L = \{a^P / P$ is a prime no $\}$ is not a CFL.