Chapter-3 Context Free Language:-

Context Free Grammar:-

→ It is a 4-tuple (V, Σ, R, S) where,

V is set of variables

Σ is set of terminals, $\Sigma \subseteq V$.

R is set of rules or productions.

$R \subseteq (V-\Sigma) \times V^*$

S is start symbol

$V-\Sigma$ is set of non-terminals.

$A \to u$, is written when $(A, u) \in R$.

where, $A \in V-\Sigma$, $u \in V^*$

$u \Rightarrow v$ means u derives v if for some $x, y \in V^*$

$u = xAy$ where,

$A \in V-\Sigma$, $A \to v'$, $v = xv'y$

$\Rightarrow^*$ is reflexive transitive closure of $\Rightarrow$

$L(G)$ is language generated by G, such that,

$$L(G) = \{ w \in \Sigma^* : S \Rightarrow^* w \}$$

Example:-

Let, language $L = \{ a^m b^n ; m \geq n \}$

Then, the grammar G that generates L is;

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

$V = \{a, b, S\}$

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

$R = \{ S \to e,$

$S \to aS,$

$S \to aSb \}$

Eg② $L = a^n b^n ; n \geq 0$

The grammar G that generates L is,

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

$V = \{a, b, S\}$

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

$R = \{ S \to e,$

$S \to asb \}$

Eg③ $L = a^n b^n ; n > 0$

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

such that,

$V = \{a, b, S\}$

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

$R = \{ S \to asb$

$S \to ab \}$

Eg④ $L = \{ w \in \{a, b\}^* ;$ number of 'b' is twice the no. of 'a' $\}$

Corresponding Grammar is;

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

$V = \{a, b, S\}$

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

$R = \{ S \to e, S \to aSbb, S \to bbSa,$

$S \to bSaSb, S \to SS \}$

Deriving string "abbbaabbb" using above Grammar,

$S \Rightarrow SS$

$\Rightarrow aSbbS \Rightarrow abbS \Rightarrow abbbSaSb \Rightarrow abbbaSb$

$\Rightarrow abbbaaSbbb \Rightarrow abbbaabbb.$

Note:-
This is left most derivation

Other examples:-

- $L = a^{2n} b^n ; n \geq 0$
  - ↳ $R = \{ S \to e / aaSb \}$
- $L = a^{2n} b^n ; n > 0$
  - ↳ $R = \{ S \to aaSb / aab \}$
- $L = a^{2n} b^{n+1} ; n \geq 0$
  - ↳ $R = \{ S \to b / aaSb \}$
- $L = a^{2n+1} b^{3n+2} ; n \geq 0$
  - ↳ $R = \{ S \to abb / aaSbbb \}$
- L = Equal number of a & equal number of b.
  (#a = #b)
  - ↳ $R = \{ S \to e / aSb / bSa / SS \}$

eg:- for string "ababba"

$S \Rightarrow \underline{S}S$
$\Rightarrow a\underline{S}bS$
$\Rightarrow ab\underline{S}abS$
$\Rightarrow abab\underline{S}$
$\Rightarrow ababb\underline{S}a$
$\Rightarrow ababba$ #

8) Construct a grammar that generates palindrome:-

$L = \{ w \in \{a,b\}^* : w = w^R \}$

soln:-

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

$V = \{a, b, S\}$

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

$R = \{ S \to e, S \to aSa, S \to bSb, S \to a, S \to b \}$

for $L = WW^R$

$R = \{S \to e \mid aSa \mid bSb\}$

for $L = a^n b^m$; $n, m \geq 0$, $n \geq m$

$R = \{S \to e \mid eSb \mid A$

$A \to aA \mid e\}$

OR

$R = \{S \to e \mid asb \mid aS\}$

for $L = a^n b^m$; $n, m \geq 0$, $n < m$

$R = \{S \to b \mid Sb \mid asb\}$

Q $L = \{w \in \{(,)\}^* : w$ is balanced parenthesis$\}$

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

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

$\Sigma = \{(,)\}$

$R = \{S \to e \mid AB \mid SS$

$A \to (\mid AAB$

$B \to) \mid ABB\}$

To derive "(())()"

Left most derivation:-

$S \Rightarrow \underline{S}S \Rightarrow \underline{A}BS \Rightarrow \underline{A}ABBS \Rightarrow (\underline{A}BBS$

$\Rightarrow ((\underline{B}BS \Rightarrow (()\underline{B}S \Rightarrow (())\underline{S}$

$\Rightarrow (())\underline{A}B \Rightarrow (())(\underline{B} \Rightarrow (())()$.

Right most derivation:-

$S \Rightarrow SS\underline{S} \Rightarrow SA\underline{B} \Rightarrow S\underline{A}) \Rightarrow \underline{S}()$

$\Rightarrow A\underline{B}() \Rightarrow AABB\underline{(}) \Rightarrow AA\underline{B})()$

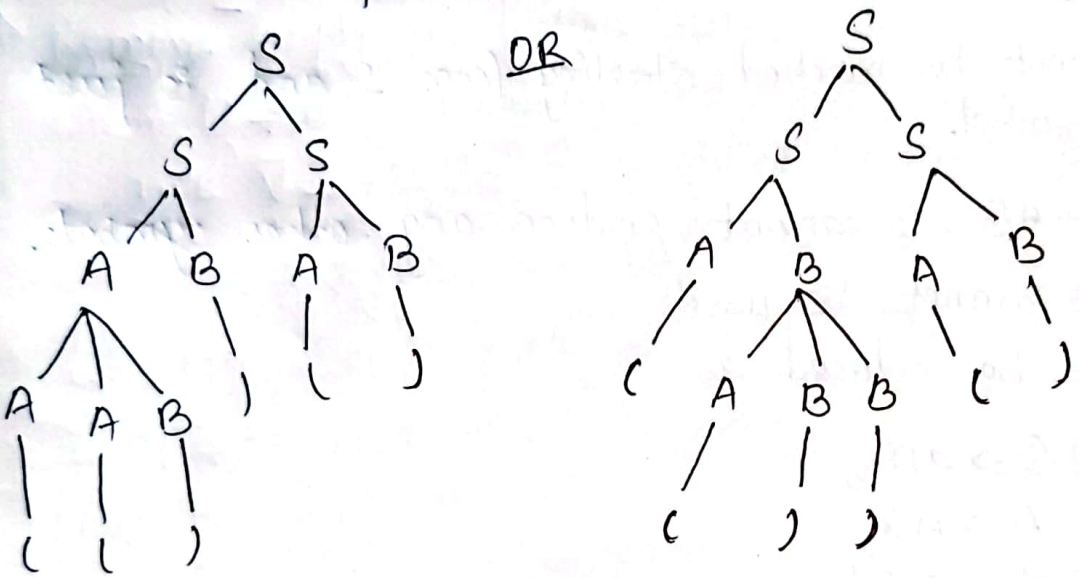
$\Rightarrow AA\underline{B}))() \Rightarrow \underline{A}(()) \Rightarrow (())()$ //

CamScanner

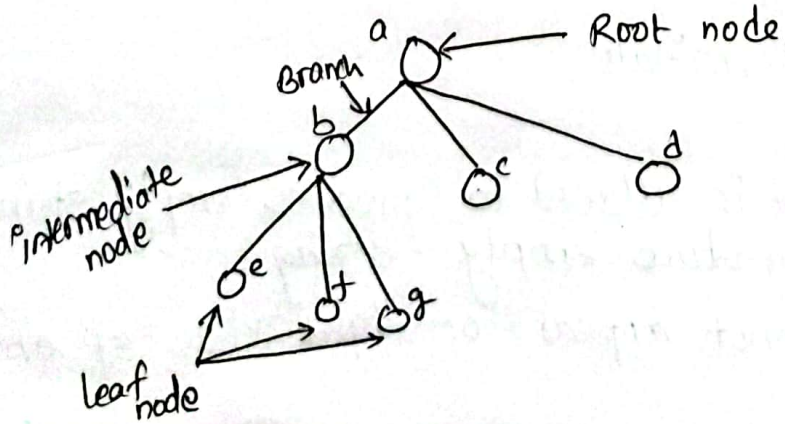
Parse Tree :-

→ A tree structure with nodes and branches connecting the nodes.
→ Top node is called root and bottom nodes are called leaves.
→ leaves are labelled with terminals.
→ Connecting the labels of leaves from left to right, we obtain the string of terminals called "the yield of parse tree".

Parse tree for Balanced Parenthesis string "(())()"



→ A grammar is said to be "Ambigious" if there exists two different parse trees that can yield same string.
→ Deciding whether or not the grammar is ambiguous is undecidable. (or partially decidable).



Content Free Grammar with Redundant Symbols:-

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

$V = \{a, b, S\}$

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

$R = \{ S \to aAb,$
$A \to AB \mid aX,$
$X \to aXab,$
$Y \to BX,$
$X \to e$
$\}$

Here, 'Y' cannot be reached starting from S and is just redundant symbol.

Also,
in $A \to AB$, B cannot produce any other symbols.

So, $A \to AB$ cannot be used.

So, R can be reduced as:

$R' = \{ S \to aAb,$
$A \to Ax,$
$x \to aXab,$
$X \to e$
$\}$

Chomsky Normal Form:-

When CFG is in CNF, then every rules has the form,

$A \to TU$
$A \to x$

where,
A, T, U are non-terminals
x is terminal

→ Only starting symbol is allowed to produce empty string if the grammar can produce empty string.
→ Starting symbol cannot appear on right side of any rule.

CamScanner

Theorem:-

Context free grammar in chomsky Normal form generate context free language.

* for context free grammar,

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

The following process converts it to chomsky Normal form (CNF).

i) New start variable S_0 is introduced such that the new rule is $S_0 \to S$.

ii) Remove empty productions rule.

$U \to e$ is removed and

rule $V \to xUyUz$ is modified by adding new rules.

$V \to xyUz$, $V \to xUyz$ and $V \to xyz$

[substitute e in place of U forming new rules].

if $T \to U$ then $T \to e$ is added if $T \to e$ has not already been removed.

iii) Remove unit productions:-

The rules of form $U \to T$ is removed so that whenever $T \to t$ appears, the rule $U \to t$ is added if it has not been added.

iv) Remove long rules:-

Rules of form $U \to U_1 U_2 \dots U_n$ is substituted with

$U \to U_1 V_1$, $V_1 \to U_2 V_2$, $V_2 \to U_3 V_3 \dots V_{n-2} \to U_{n-1} V_{n-1}$

Example:-

Convert given CFG to CNF.

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

$V = \{0, 1, S, A, B\}$.

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

$R = \{ S \to e \mid AB \mid SS,$
$A \to 0 \mid AAB,$
$B \to 1 \mid ABB \}$

Introducing new starting symbol S_0,

$R = \{ S_0 \to S,$
$S \to e \mid AB \mid SS,$
$A \to 0 \mid AAB,$
$B \to 1 \mid ABB$
$\}$

Removing $S \to e$ and $S_0 \to S$,

$R = \{ S_0 \to e \mid AB \mid SS,$
$S \to AB \mid SS,$
$A \to 0 \mid AAB,$
$B \to 1 \mid ABB$
$\}$

Removing long rules, $A \to AAB$ and $B \to ABB$.

$R = \{ S_0 \to e \mid AB \mid SS,$
$S \to AB \mid SS,$
$A \to 0 \mid MB,$
$M \to AA,$
$B \to 1 \mid NB,$
$N \to AB \}$

Example 2

Convert given CFG to CNF.

$S \to AaA \mid BA$
$A \to aB \mid BB \mid e$
$B \to A \mid ab \mid S$

CamScanner

Pushdown Automata:-

→ PDA is a six-tuple

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

where,

Q is finite set of states

Σ is set of input alphabet

T is the set of stack alphabet.

$\delta : Q \times (\Sigma \cup \{e\}) \times (T \cup \{e\}) \to P(Q \times (T \cup \{e\}))$

q_0 is initial state [∵ P is power set]

$F \subseteq Q$ is set of final states

→ PDA starts with initial state q_0 and empty stack.

→ Transition δ is defined as,

$$\delta(q_i, a, \alpha) = (q_j, \beta)$$

Meaning,

When a PDA is in state q_i, process the input a and pops (reads) α from stack, it goes to q_j, and pushes (writes) β in the stack.

→ The configuration of PDA written as (p, w, t) means PDA is in state p, w is the input string to be read and t is string of stack read top-docon.

→ we can write $(p, w, t) \vdash_P (p', w', t')$

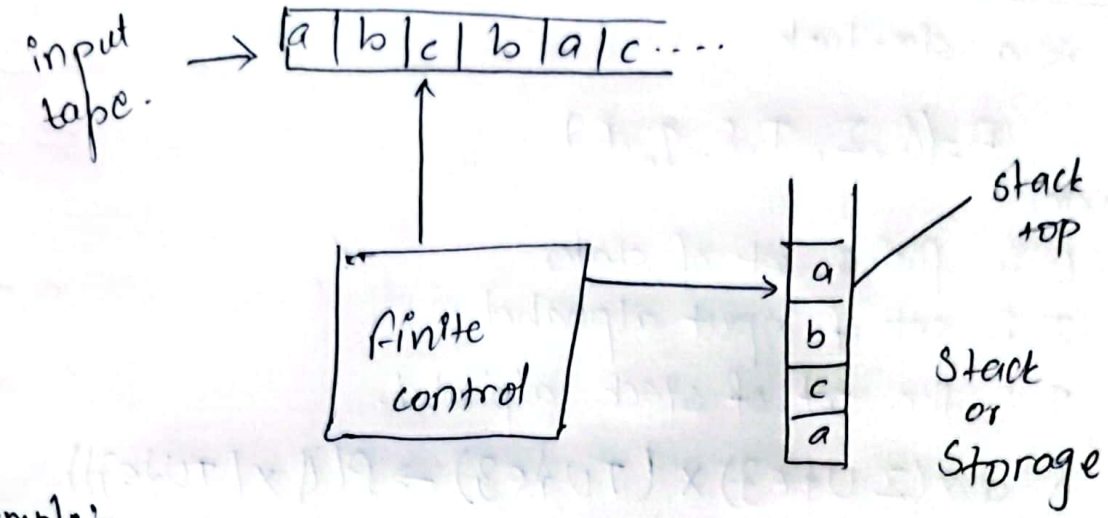
if $\delta(p, a, \alpha) = (p', \beta)$, $w = aw'$, $t = \alpha t$,

$t' = \beta t$, $t \in T^*$

→ A string w is said to be accepted by PDA

if $(q_0, w, e) \vdash_P^* (q, e, e)$; $q \in F$

CamScanner



Example:-

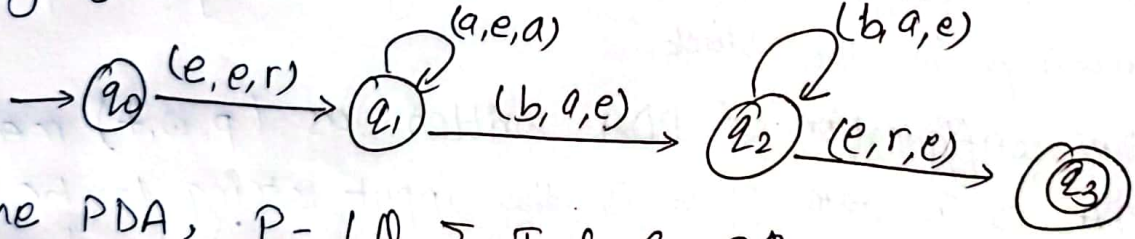
Construct a PDA for language.

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

Also, test your PDA for string "aabb".

Soln:-

Let Pn the notation (q, α, β), first symbol 'a' denote symbol of read, 'α' denote popped item and 'β' denote pushed item into stack. Then the PDA for language L will be,



The PDA, $P = \{ Q, \Sigma, T, \delta, q_0, F \}$

where,

$Q = \{ q_0, q_1, q_2, q_3 \}$ is set of states

$\Sigma = \{ a, b \}$ is set of input alphabet.

$T = \{ a, b, r \}$ is set of stack alphabet

$\delta = \{ ((q_0, e, e), (q_1, r)), ((q_1, a, e), (q_1, a)), ((q_1, b, a), (q_2, e)), ((q_2, b, a), (q_2, e)), ((q_2, e, r), (q_3, e)) \}$ is transition function q_0 is initial state.

$F = \{ q_3 \}$ is final state.

The PDA, P accepts string 'aabb' as:-

$(q_0, aabb, e) \vdash_p (q_1, aabb, r)$

$\vdash_p (q_1, abb, ar)$

$\vdash_p (q_1, bb, aar)$

$\vdash_p (q_2, b, ar)$

$\vdash_p (q_2, e, r)$

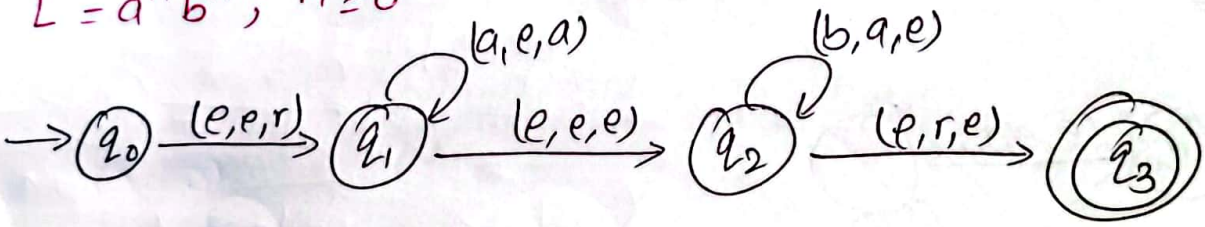
$\vdash_p (q_3, e, e)$

$\Rightarrow (q_0, aabb, e) \vdash_p^* (q_3, e, e)$ and $q_3 \in F$.

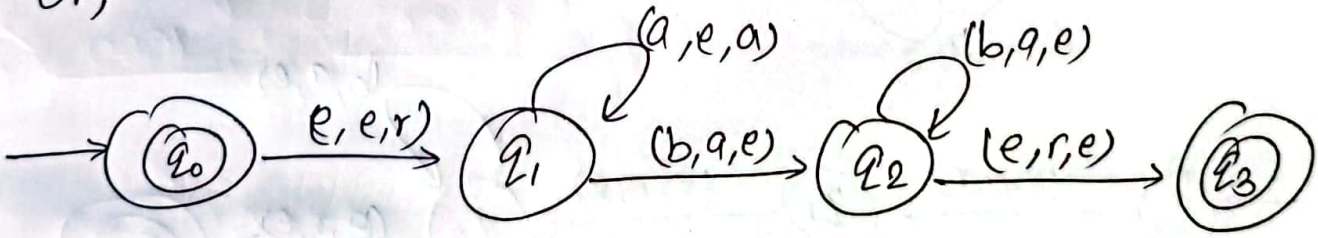
∴ P accepts aabb.

More examples:-

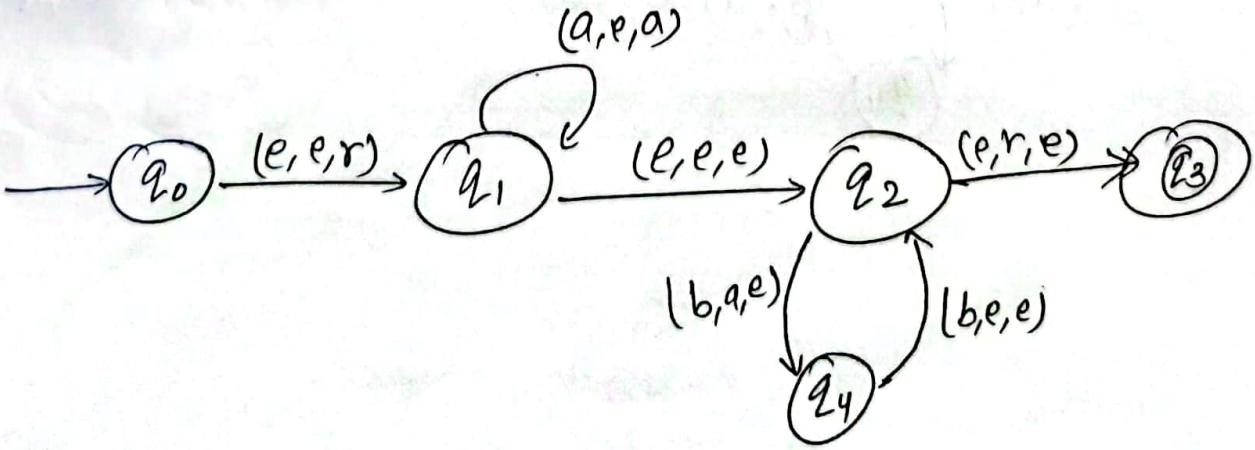
(i) $L = a^n b^n ; n \geq 0$



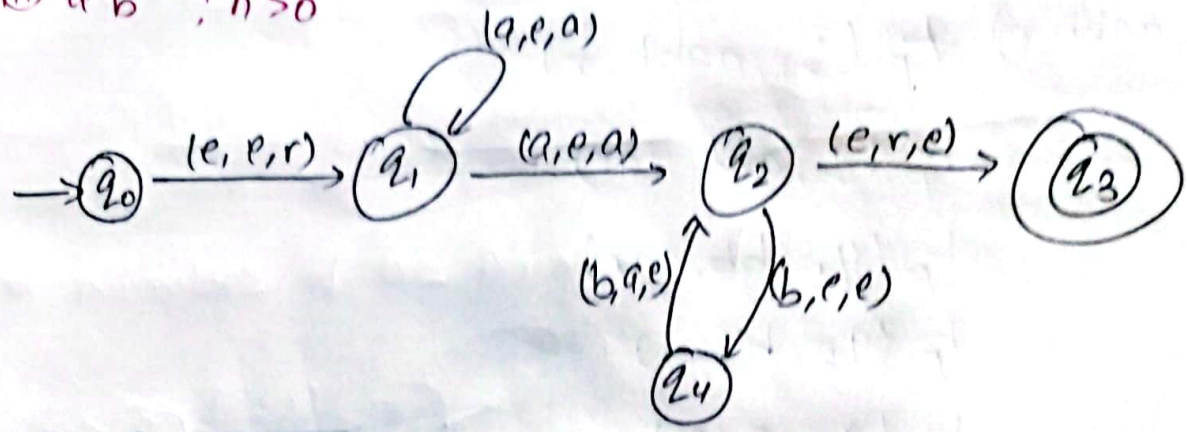
or,



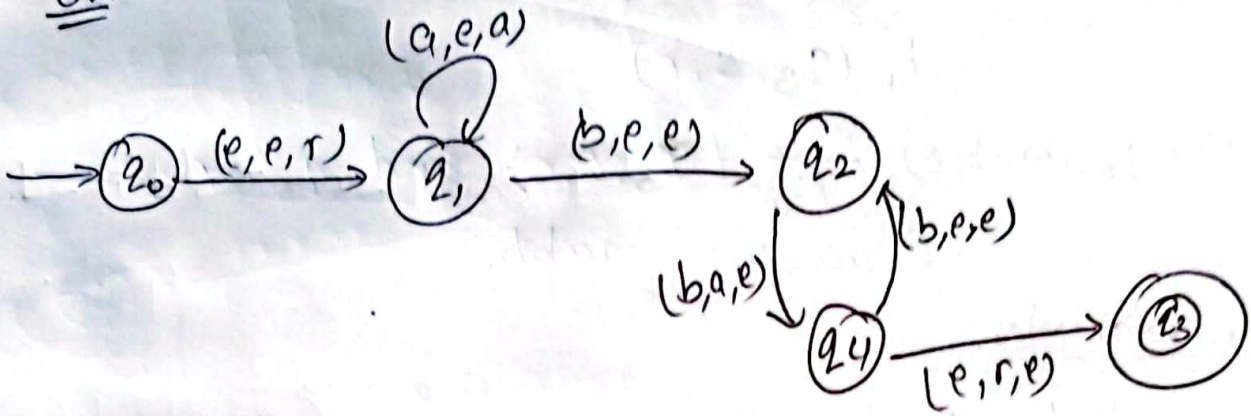
(ii) $L = a^n b^{2n} ; n \geq 0$



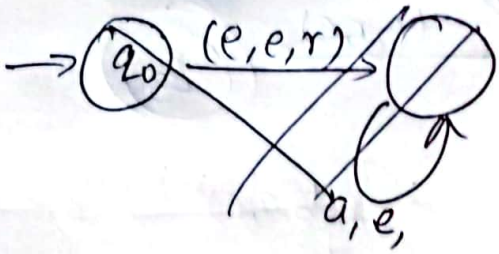
(iii) $a^n b^{2n}$; $n > 0$

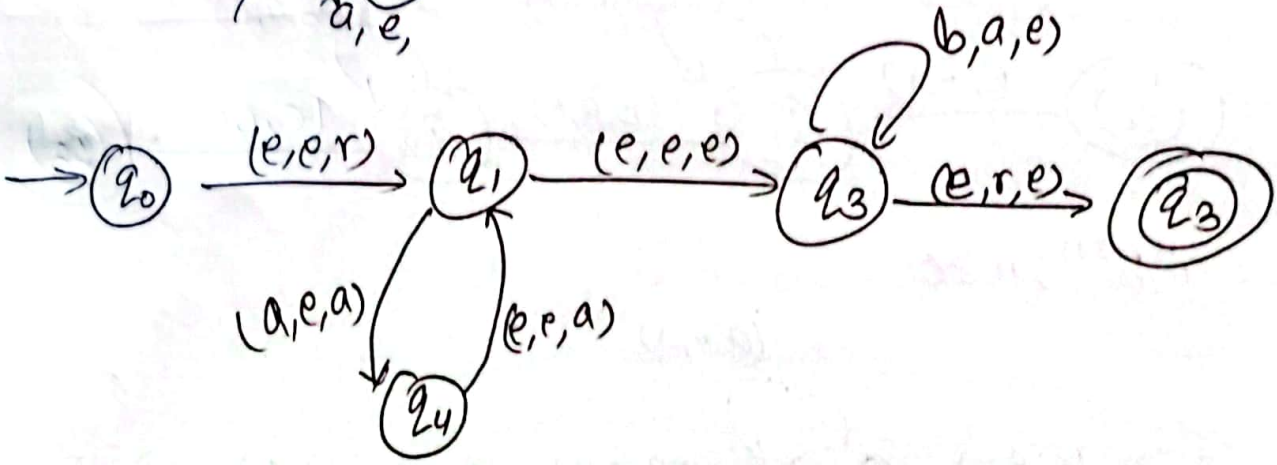


OR

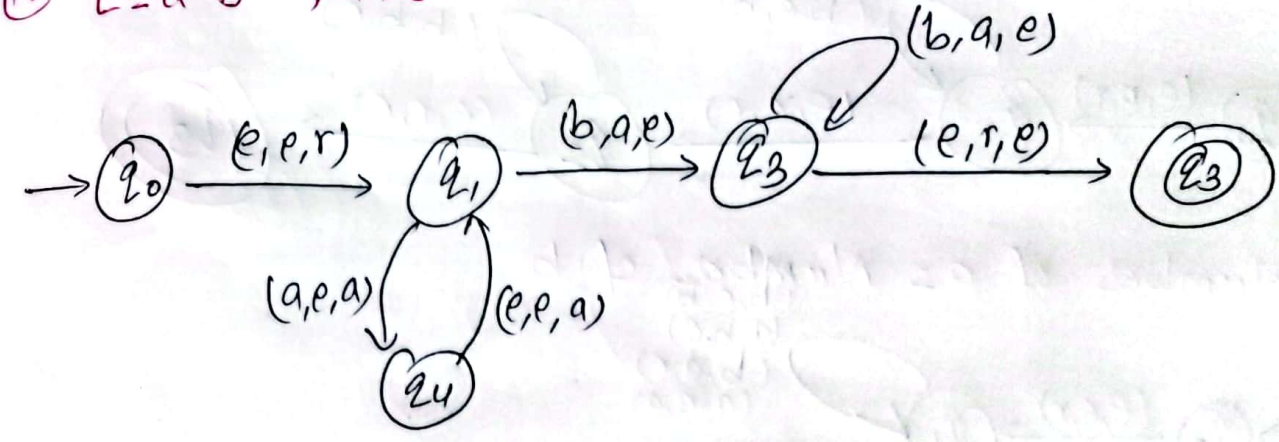


OR

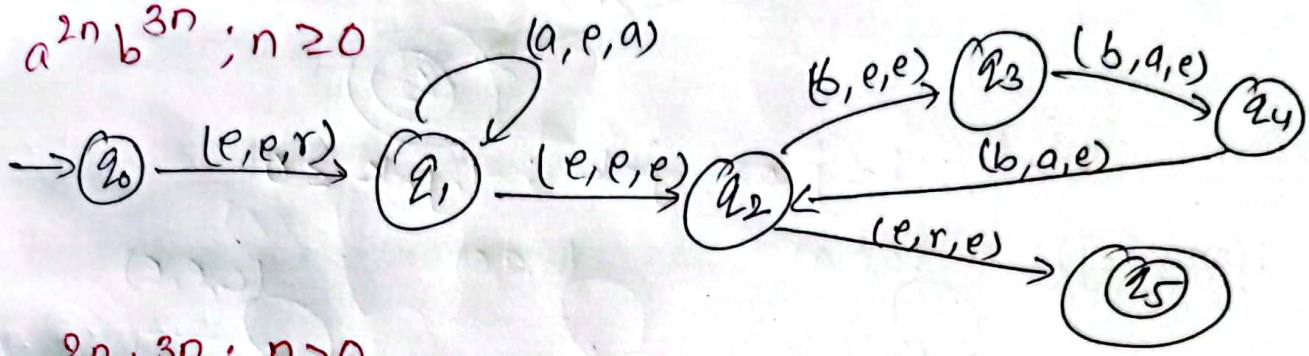




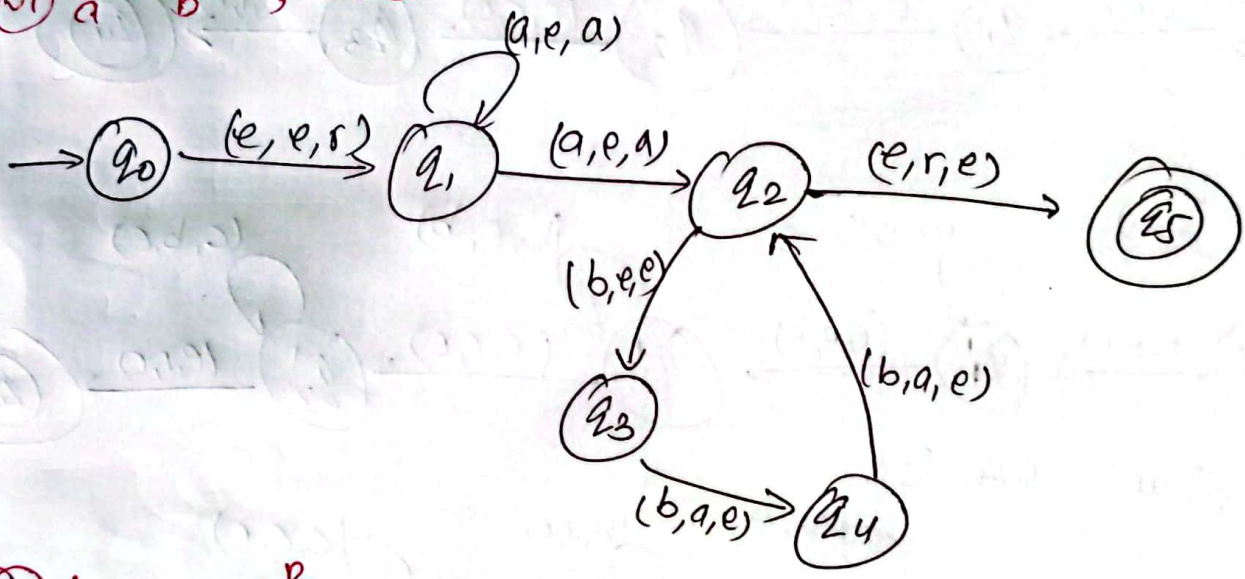
(iv) $L = a^n b^{2n}$; $n > 0$



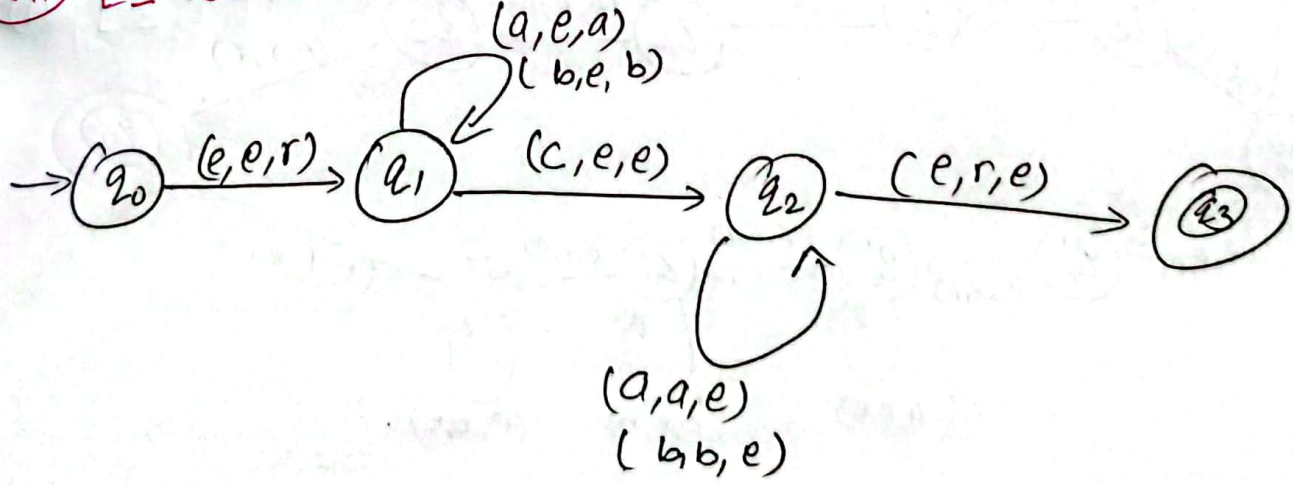
(v) $a^{2n} b^{3n}$; $n \geq 0$



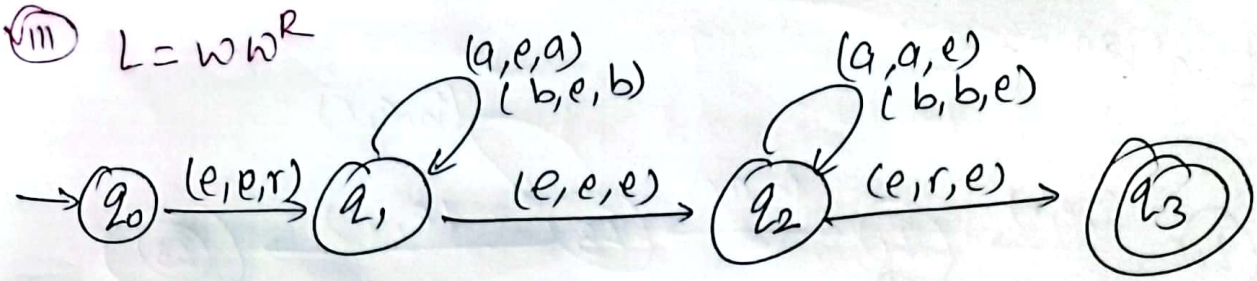
(vi) $a^{2n} b^{3n}$; $n > 0$



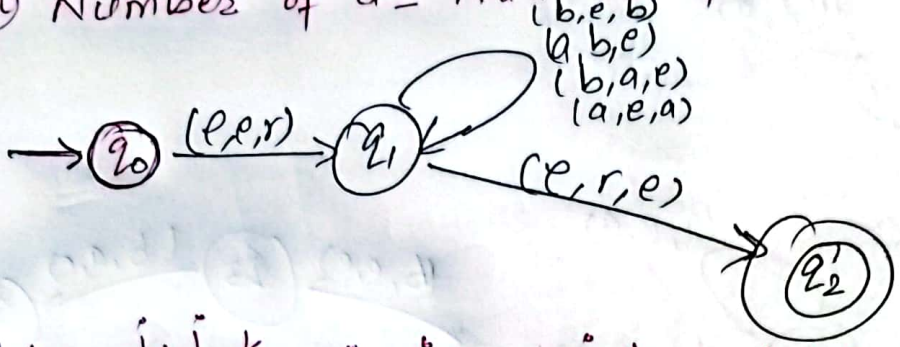
(vii) $L = wcw^R$



(viii) $L = ww^R$

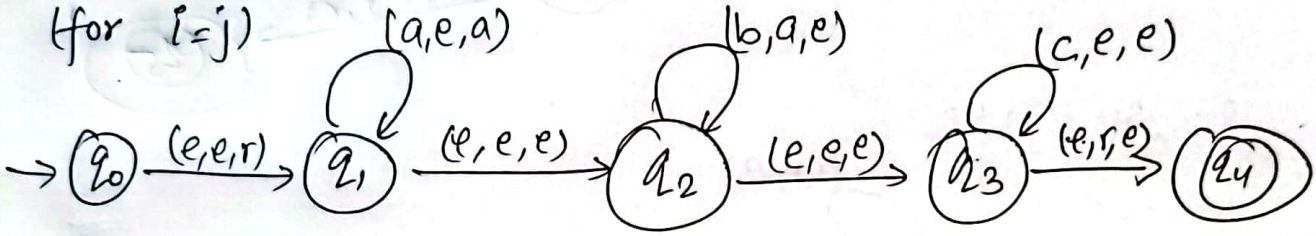


(ix) Number of a = Number of b

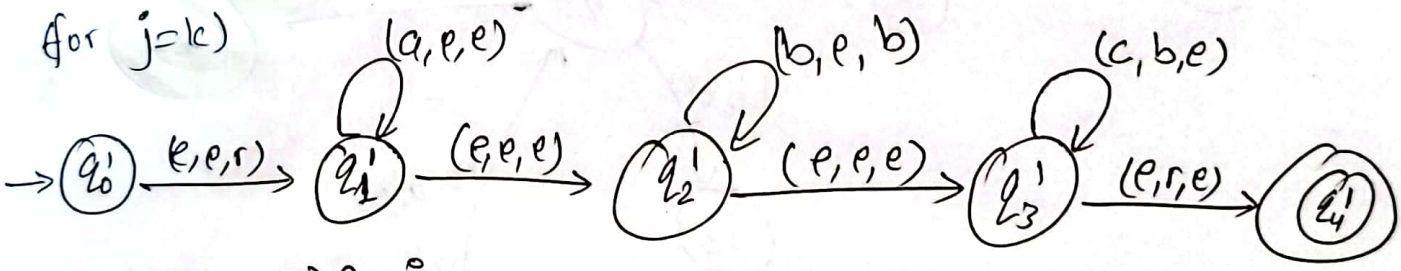


(x) $L = a^i b^j c^k$; $i=j$ or $j=k$; $i, j, k \geq 0$

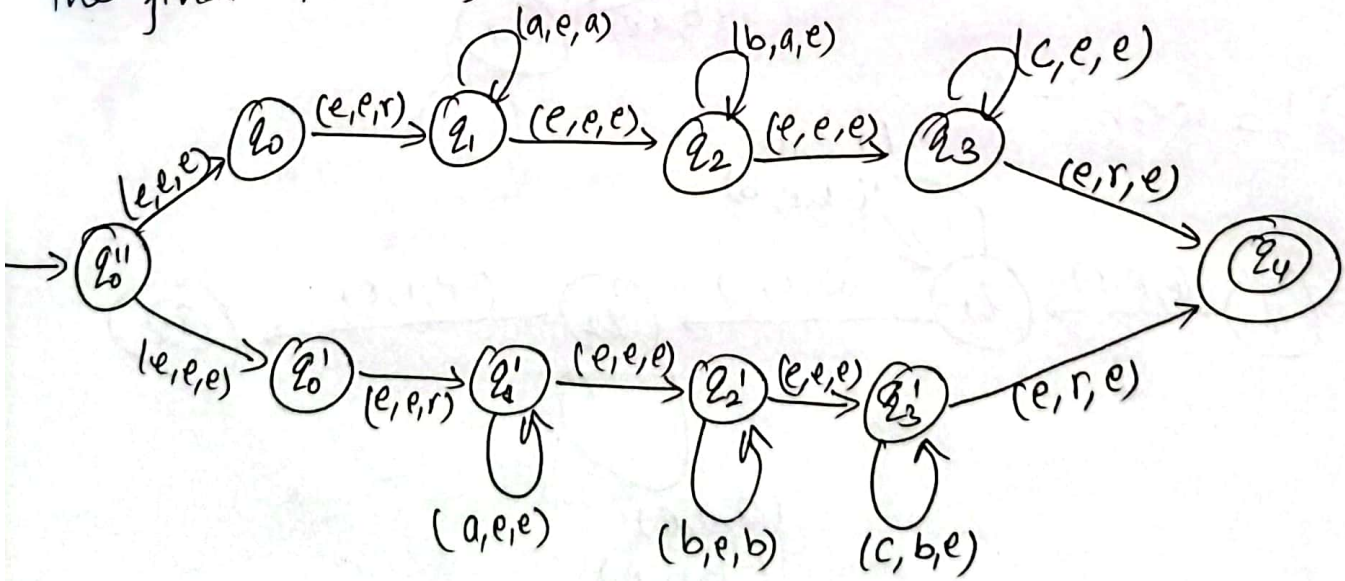
(for $i=j$)



(for $j=k$)



The final PDA is,



Equivalence of context free language (CFL) and Push Down Automata (PDA)

Theorem:-

The language is context free iff some PDA recognize it.

To prove it, we show that

(i) If language is context free, then some PDA recognize it.

(ii) If a PDA recognize a language, then it is context free.

(i) If language is context free, then some PDA recognize it.

We know languages that ~~are~~ context free grammar (CFG) can generate are context free language (CFL)

Let, $G = (V, \Sigma, R, S)$ be a CFG such that language generated by G, $L(G)$ is context free.

For this grammar G, we construct a PDA that simulates left most derivation of G.

Let the PDA be,

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

Q has three states, $Q = \{q_0, q_i, q_f\}$

where,

q_0 is initial state

q_i is intermediate state

q_f is final state; $F = \{q_f\}$

Let P be capable of pushing entire string using transition function $\delta'(p, a, \beta) = (q, \beta)$

where,

$$\beta = b_1 b_2 \dots b_n.$$

$$\delta(p, a, \beta) = (q_1, b_n), \ \delta(q_1, e, e) = (q_2, b_{n-1}), \dots$$

$$\delta(q_{n-1}, e, e) = (q, b_1)$$

CamScanner

- Initialize stack with string S_r,

i.e. $\delta'(q_0, e, e) = (q_i, S_r)$

If the top of stack has non-terminal 'A' and $(A \to w) \in R$, $w \in V^*$ then,

$$\delta'(q_i, e, A) = (q_i, w)$$

- If the top of stack has terminal 'a' then,

$$\delta(q_i, a, a) = (q_i, e)$$

- If 'r' is at the top of stack, then

$$\delta(q_i, e, r) = (q_f, e)$$

Example:-

$G = (V, \Sigma, R, S)$ PDA equivalent to grammar G is,

Where,

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

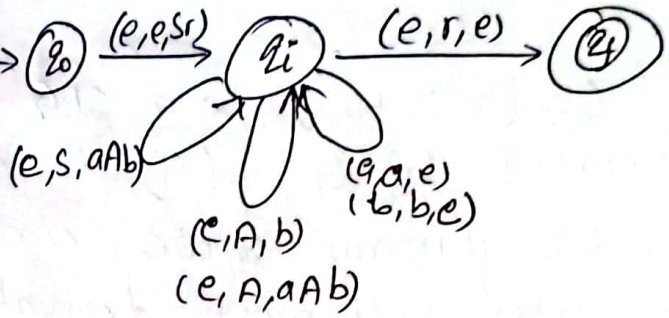
$\Sigma = \{a, b\}$

$R = \{ S \to aAb$

$A \to b | aAb$

$\}$

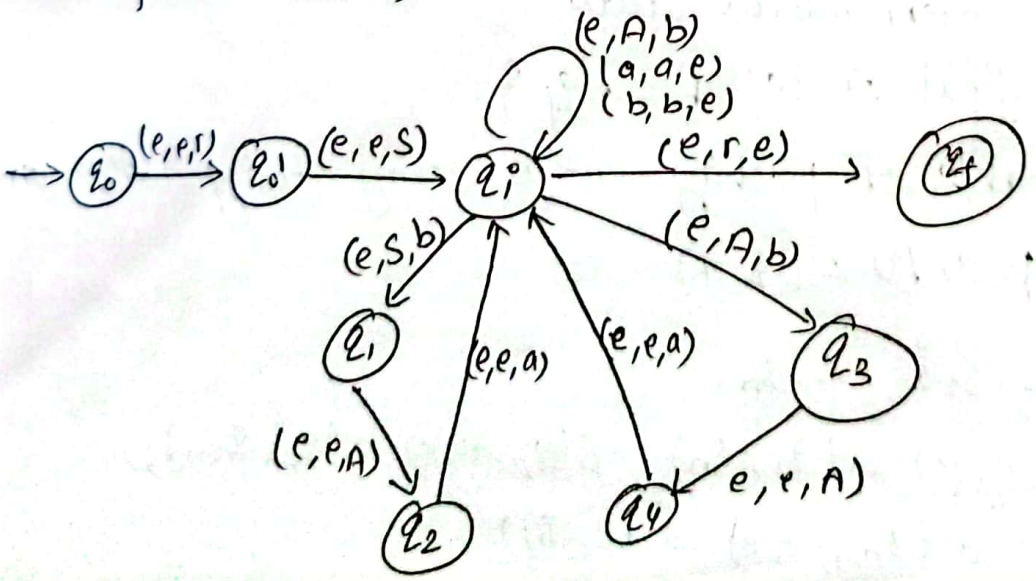
S is start symbol.



The transition functions are:-

$\delta = \{((q_0, e, e), (q_i, S_r)), ((q_i, e, S), (q_i, aAb)),$
$((q_i, e, A), (q_i, b)), ((q_i, e, A), (q_i, aAb)),$
$((q_i, a, a), (q_i, e)), ((q_i, b, b), (q_i, e)), (q_i, e, r), (q_f, e)\}$

* The final PDA is;



(ii) If a PDA recognizes a language, it is context free.

Let PDA, $P = (Q, \Sigma, T, \delta, q_0, F)$ be PDA (with only one final state).

Let G be CFG, $G = (V, \Sigma, R, S)$

$$V - \Sigma = \{ A_{pq} ; p, q \in Q \}$$

$$S = A_{q_0 q_f}$$

$$a, b \in \Sigma, \quad t \in T$$

$R = \{ A_{pq} \to a A_{p'q'} b$ if $\delta(p, a, e) = (p', t)$

$\delta(q', b, t) = (q, e)$

$p, q, p', q' \in Q$

$A_{pq} \to A_{pp'} A_{p'q}$ when $p, p', q \in Q$

$A_{pp} \to e$ when $p \in Q$

$\}$

Then if A_{pq} generates string w, w by construct will take PDA from P with empty stack to q with empty stack, which means if PDA recognize some language then there is a CFG that generates the language. So, the language is context free.

Pumping Lemma for CFL :-

For a context free language L, there is a number n for string $w \in L$, $|w| \geq n$ such that, the string will be divided into five parts.

$w = uvxyz$, fulfilling the condition,

for $i \geq 0$, $uv^i xy^i z \in L$

$|vy| > 0$

$|vxy| \leq n$

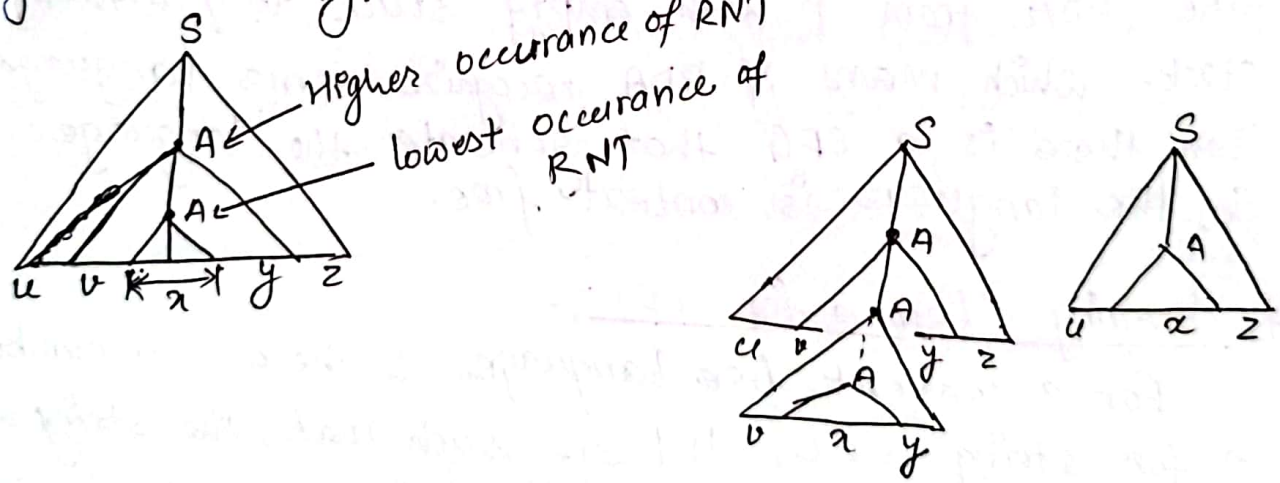
Proof:

Let l be maximum numbes of symbols on the right hand side of the rule. $|V-\Sigma|$ be the numbes of non-terminals, then if pumping length

$$n = l^{|V-\Sigma|} + 1$$

Then any string of length n or greates will have a parse tree with path length atleast $(|V-\Sigma| + 1)$ long. If we choose path with path length $|V-\Sigma|+1$, it will have $|V-\Sigma|+2$ nodes. As the final node is terminal, there will be $|V-\Sigma|+1$ non-terminals. But, there are only $|V-\Sigma|$ non-terminals. Therefore, at least one non-terminals must be a repeat non-terminal (RNT).

Now, dividing w into uvxyz and considering lowest occurance of RNT, we see that it has subtree, that generates x and the higher occurance of RNT generates vxy.



Since, each subtree has been generated by same variable, we can substitute one subtree with another and still have valid parse tree. Therefore, RNT can be substituted many times and still parse tree would be valid which means, uv^ixy^iz will be strings generated by such substitutions.

Also, when smallas subtree substitutes larger one, then we get uxz, which is also part of language.

If v and y are both empty strings, then we will have smaller parse tree, which is not possible because we started with smallest parse tree. Therefore,

$|vy| > 0$.

Also, from construction, there can be almost n branches between two occurance of RNT

i.e, $|vxy| \leq n$

⑤ Using pumping lemma for CFL, show whether the language $L = \{a^n b^n c^n ; n \geq 0\}$ is context free or not.

→ Let L be CFL, then for any string, $w \in L$, $|w| \geq n$,

n = pumping length.

Then, using pumping lemma for CFL,

w can be divided into five parts,

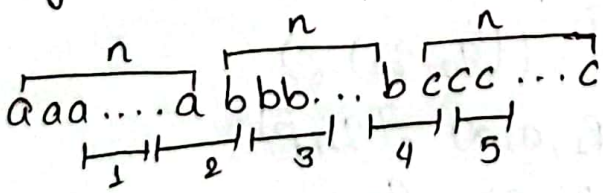
$w = uvxyz$ such that,

$uv^i xy^i z \in L ; i \geq 0$

$|vy| > 0$

$|vxy| \leq n$

Let us choose a string $w = a^n b^n c^n$ and try to find u, v, x, y, z such that pumping lemma holds for w.



Case I:

as $|vxy| \leq n$, taking vxy part in a's only,

$v = a^p ; y = a^q ; p, q \leq n, |vy| > 0$

then for $i>1$, No. of a exceeds no. of b and c.
i.e, $uv^ixy^iz \notin L$ for $i>1$.

Case II :-

Taking vxy part in the border between a's and b's.
then for $i>1$,
No. of a and no. of b will increase but no. of c can't increase. So, $uv^ixy^iz \notin L$ for $i>1$.
Similarly, for cases when vxy part lies totally in b, border between b's and c's and totally in c, $uv^ixy^iz \notin L$ for $i>1$.

Therefore, w cannot be pumped.
So, L is not context free.

Intersection of CFL, L and Regular Language L, is a CFL.
(CFL ∩ Regular Language = CFL).

→ Let PDA corresponding to CFL, L be $P_1 = (Q_1, \Sigma_1, T_1, \delta_1, q_0', F_1)$ and DFA corresponding to R be $D = (Q_2, \Sigma_2, T_2, \delta_2, q_0'', F_2)$.
Then, we construct PDA,

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

such that,

$$Q = Q_1 \times Q_2$$
$$F = F_1 \times F_2$$
$$\delta((P_1, P_2), a, \alpha) = ((q_1, q_2), \beta)$$
iff $\delta_1(P_1, a, \alpha) = (q_1, \beta)$
$\delta_2(P_2, a) = q_2$

Then, the string will take p from start state to final state iff it simultaneously takes P_1 and D from initial state to final state (with empty stack of P_1).

This means, P will recognize language which is intersection of L and R(L∩R). Since, a PDA recognizes L∩R, therefore L∩R is context free language.

For eg:-

CFL, $L_1 = a^n b^n = \{e, ab, aabb, aaabbb, \ldots\}$

R.L, $L_2 = a^* b^* = \{e, a, b, ab, aa, bb, aabb \ldots\}$

$L_1 \cap L_2 = \{e, ab, aabb, aaabbb, \ldots\}$

$= a^n b^n \; ; n \geq 0$

So, it is CFL.

Closure Property of CFL:-

① CFL is closed under UNION operation :-

Let L_1, L_2 be CFL's and G_1, G_2 be their corresponding grammars, such that,

$L_1 = L(G_1)$, $L_2 = L(G_2)$,

$G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$

Let G be grammar,

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

such that,

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

$\Sigma = \Sigma_1 \cup \Sigma_2$

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

for any string $w \in (L_1 \cup L_2)$

$S \Rightarrow_G^* w$ because,

$S \Rightarrow_G S_1 \Rightarrow_{G_1}^* w_1 \; ; w_1 \in L$

$S \Rightarrow_G S_2 \Rightarrow_{G_2}^* w_2 \; ; w_2 \in L$

Since, there is CFG that generates $w \in (L_1 \cup L_2)$, $(L_1 \cup L_2)$ is also CFL.

Hence, CFL are closed under union operation.

CamScanner

(ii) CFL are closed under concatenation operation :-

Let L_1 and L_2 be CFL's and G_1, G_2 be their corresponding grammars, such that,

$L_1 = L(G_1)$; $G_1 = (V_1, \Sigma_1, R_1, S_1)$

$L_2 = L(G_2)$; $G_2 = (V_2, \Sigma_2, R_2, S_2)$

Let, G be grammar,

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

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

$\Sigma = \Sigma_1 \cup \Sigma_2$

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

for any string $w \in L$, $L = L_1 L_2$

$S \Rightarrow_G^* w$ because,

$S \Rightarrow_G S_1 S_2$, $S_1 \Rightarrow_{G_1}^* w_1$, $S_2 \Rightarrow_{G_1}^* w_2$

$\therefore S \Rightarrow_G^* w_1 w_2$

i.e. $S \Rightarrow_G^* w$; $w = w_1 w_2 \in L_1 L_2$

Since, G is CFG that generates $L = L_1 L_2$, L is CFL.

Hence, CFL are closed under concatenation operation.

(iii) CFL are closed under kleene star operation :-

Let L_1 be CFL and G_1 be its CFG.

i.e. $L_1 = L(G_1)$ & $G_1 = (V_1, \Sigma_1, R_1, S_1)$

Let us define a grammar G_1,

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

$V = V_1 \cup \{S\}$, $\Sigma = \Sigma_1$

$R = R_1 \cup \{S \to e, S \to SS_1\}$

then, for any string,

$w = w_1 w_2 w_3 \cdots w_n$; $w \in L$, $n \geq 0$

for $1 \leq i < n$, $w_i \in L_1$

CamScanner

i.e, $L = (L_1)^*$

$S \Rightarrow_G^* w$ because,

$$S \Rightarrow_G SS_1 \Rightarrow_G SS_1S_1 \Rightarrow \dots \Rightarrow \underset{(1)\ (2)\ \dots\ (n)}{SS_1S_1\dots S_1}$$

Since,

$$S_1 \Rightarrow G_1^* w_1, \dots, S_1 \Rightarrow_{G_1}^* w_1, \dots, S_1 \Rightarrow G_1^* w_n$$

Therefore,

$$S \Rightarrow_G^* Sw_1w_2\cdots w_n \Rightarrow_G w_1w_2\cdots w_n$$

i.e., $S \Rightarrow_G^* w$

Since, there is CFG that generate $L = (L_1)^*$. Therefore, L is also CFL.

Hence, CFL are closed under kleene star operation.

(iv) CFL are not closed under intersection operation

Let L_1 and L_2 be context free languages, such that,

$$L_1 = \{a^n b^n c^m : m, n \geq 0\}$$

$$L_2 = \{a^m b^n c^n : m, n \geq 0\}$$

then, $L_1 \cap L_2 = \{a^n b^n c^m ; m, n \geq 0\} \cap \{a^m b^n c^n : m, n \geq 0\}$

$$= \{a^n b^n c^n : n \geq 0\}$$

But, $\{a^n b^n c^n : n \geq 0\}$ is not context free.

(can be proved using pumping lemma)

$\therefore L_1 \cap L_2$ is not context free.

Hence, CFL are not closed under operation of intersection.

(v) CFL are not closed under complementation:-

Let L_1 and L_2 be context free languages, then $L_1 \cap L_2$ is given by,

$$L_1 \cap L_2 = \Sigma^* - ((\Sigma^* - L_1) \cup (\Sigma^* - L_2))$$

$$= \overline{\bar{L}_1 \cup \bar{L}_2} \quad [A \cap B = \overline{\bar{A} \cup \bar{B}} \text{ (De-morgan's law)}]$$

CamScanner

Here, We know,

LHS $= L_1 \cap L_2$ is not context free $\Rightarrow$ RHS is not context free.

RHS has union & complementation operations.

As, CFL are closed under union operation & LHS is not context free, this leads to the conclusion that complementation is the reason for RHS not being context free.

Hence, CFL are not closed under complementation.