GRAMMAR

A grammar defines set of rules, with the help of these rules valid sentences in a lang. are constructed

eg. The smart Teena beats shy Kiran.

<Sentence> ⇒ <Subject> <Predicate>

⇒ <Noun-Phrase> <Predicate>

⇒ <Article> <Noun-Phrase> <Predicate>

⇒ <Article> <Adjective> <Noun-Phrase> <Predicate>

⇒ <Article> <Adjective> <Noun> <Predicate>

⇒ <Article> <Adjective> <Noun> <Verb> <Noun-Phrase>

⇒ <Article> <Adjective> <Noun> <Verb> <Adjective> <Noun>

Article Adjective Noun Verb Adjective Noun

⇒ The smart Teena beats shy Kiran.

Defn:- A grammar can be defined as 4-tuples (V_N, Σ, P, S), where (V_N, Σ, P, S)

(i) V_N → finite non-empty set of variables or non-terminals

(ii) Σ → finite ,, non-empty ,, ,, of input alphabets or terminals

(iii) P → finite set of productions or rewriting rules.

(v) S → start symbol $(S \in V)$ ie $S \in V$ (∈ → belongs to.)

$V_N \cap \Sigma = \phi$ (ie Set V_N and Σ have no common elements)

eg. Consider context free grammar

$S \to 0S1 \mid 1S1$

$S \to \varepsilon$.

$G = \{V, T, P, S\}$. be context free grammar where $V = \{S\}$, $\Sigma = \{0, 1\}$, $S = \{S\}$.

$P \to 0S1 \mid 1S1 \mid \varepsilon$

If $w \in L(G)$

and $S \Rightarrow \beta_1 \Rightarrow \beta_2 \Rightarrow \cdots \Rightarrow \beta_n = w$ is derivation of string w then string $\beta_1, \beta_2, \ldots \beta_n$ which belongs to variables & terminals is called a sentential form of the derivation.

If $S \overset{*}{\underset{L}{\Rightarrow}} \beta$ then it is called left sentential form

$S \overset{*}{\underset{R}{\Rightarrow}} \beta$, then right sentential form

Q1 Derive the language from following productions representing grammar:-

$S \Rightarrow \beta_1 \Rightarrow \beta_2 \Rightarrow \beta_3 \cdots = L$

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

where $P \Rightarrow S \to aSb, S \to \wedge$

Soln From the production, $S \to \wedge$, we can say that the smallest string generated by the grammar is $\wedge$.

$S \to aSb$

$\to ab \quad [S \to \wedge]$

$a\,S\,b$

ab

$S \to aSb$

$aaSbb$

$aabb.$

$S \to aSb$

$\to aaSbb$

$\to aabb.$

ie equal no. of a's and b's.

$S \Rightarrow a^n b^n.$

The value of n must be 0. Therefore language $L(G)$ derived by grammar G is $L(G) = \{a^n b^n \mid n \geq 0\}$.

(smallest)

Q2

$S \to aSb$

$\Rightarrow ab.$

$S \to aSb$

$\Rightarrow aaSbSb$

$G_1 = (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to a, B \to b\})$

P.

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

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

$S \to$ start

Derivations from grammar - strings may be derived from other strings using productions in a grammar. If a grammar G has a production.

$$\alpha \to \beta, .$$

or $A \to \gamma$

where $A \in N$ (Non Terminal)

$\gamma \in (T \cup N)^*$ (string of terminals & non-terminals)

eg: $S \to xa$

$x \to a$

$x \to ax$

$x \to abc$

$x \to \wedge$.

Q1 Let us consider grammar

$G_2 = (\{S, A\}, \{a, b\}, S, \{S \to aAb, aA \to aaAb, A \to$

$S \to \underline{a}\underline{A}b$

$\to aa\underline{A}bb$

$aabb$

$S \to \underline{aA}b$

$S \to aa\underline{A}bb$

$S \to \underline{aabb}$

Sentential form :- If $G = (V, T, P, S)$ is a context-free grammar then any string $\beta \in (V \cup$

$S \overset{*}{\Rightarrow} \beta$ is a sentential form.

If $w \in L(G)$

and $S \Rightarrow \beta_1 \Rightarrow \beta_2 \Rightarrow \cdots \Rightarrow \beta_n = w$ is derivation of string w then string $\beta_1, \beta_2, \ldots \beta_n$ which belongs to variables & terminals is called a sentential form of the derivation.

If $S \overset{*}{\underset{L}{\Rightarrow}} \beta$ then it is called left sentential form

$S \overset{*}{\underset{R}{\Rightarrow}} \beta$, then right sentential form

Q1 Derive the language from following production representing grammar :-

$S \Rightarrow \beta_1 \Rightarrow \beta_2 \Rightarrow \beta_3 = L$

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

where $P \Rightarrow S \to aSb, S \to \wedge$

Soln From the production, $S \to \wedge$, we can say that the smallest string generated by the grammar is $\wedge$.

$S \to aSb$

$\to a\wedge b \quad [S \to \wedge]$

$\to ab$

aSb

ab

$S \to aSb$

$aaSbb$

$aabb.$

$S \to aSb$

$\to aaSbb$

$\to aabb.$

ie equal no. of a's and b's.

$S \Rightarrow a^n b^n.$

The value of n must be 0. Therefore language L(G

smallest

derived by grammar G is $L(G) = \{a^n b^n \mid n \geq 0\}$.

Q2

$S \to aSb$

$\Rightarrow ab.$

$S \to aSb$

$\Rightarrow aaSbSb$

Q2 Let $G = (\{S\}, \{a,b\}, P, S)$ where P $S \to a$, $S \to b$, $S \to aS$, $S \to bS$. Find language

Soln The string generated by grammar G is smallest either a or b.

$S \to aS \mid a$ | $S \to aS \mid ab$ |

$S \to aS$
$\to aaS$
$\to aaa$

Hence, all strings of any combination of a's & b's can be derived except the empty string.

Therefore, $L(G) = (a+b)^+$

Q3 $S \to aS \mid aA \mid a$,
$A \to aAb \mid ab$.

Soln $aa^n b^n$ for all $n \geq 0$ → (rule aA)

The rule aS for S allows any arbitrary no. of a's to be added at beginning

Therefore,
$L(G) = \{a^m b^n \mid m > n \geq 0\}$.

Q4 $S \to aABa$, $A \to baABb$
$B \to Aab$, $aA \to baa$, $bBb \Rightarrow abab$

Test $S = ba^2b^2aba^3b^2aba$ is in lang. L(G)

Soln
$S \to aABa$
$\to baaBa$ (replacing aA by baa)
$\to ba^2Ba$
$\to ba^2Aaba$ (replacing $B \to Aab$)
$\to ba^2baABbaba$ (replacing $A \to baABb$)
$\to ba^2b baaBbaba$
$\to ba^2b^2aaBbaba$

$S \to aABa$, $A \to baABb$

$B \to Aab$, $aA \to baa$, $bBb \to abab$

$ba^2b^2aba^3b^2aba$ is in lang $L(G)$

Soln $S \to \underline{aA}Ba$

$\Rightarrow baa\underline{B}a$ [replace $aA \to baa$

$\Rightarrow ba^2\underline{B}a$

$\Rightarrow ba^2\underline{A}aba$ [$B \to Aab$]

$\Rightarrow ba^2\underline{ba}\underline{A}Bbaba$ [$A \to baABb$]

$\Rightarrow ba^2bbaaBbaba$ [$aA \to baa$]

$\Rightarrow ba^2b^2\underline{a}a\underline{B}baba$ [$B \to Aab$]

$ba^2b^2aa\underline{A}abbaba$

~~$ba^2b^2a\ baABb$~~

$ba^2b^2a\ baa\ abbaba$

$ba^2b^2ab\ a^3b^2aba$.

Therefore $s \in L(G)$

$S \to aS \mid aA \mid a$

$A \to aAb \mid ab$.

$S \to$

$S \to aS \mid aA \mid a$

$A \to aAb \mid ab$.

Q $S \to AB$

$A \to BB$

$B \to AA$ Find language generated by G.

Soln Since $\Sigma = \{\phi\}$ because all 3 productions have no terminals on right hand side

Hence, $L(G) = \{\phi\}$

Q Let $G = (\{S, A\}, \{a, b, c\}, P, S\}$, where P contains $S \to \underline{aSA}c$, $S \to abc$, $cA \to Ac$, $bA \to bb$. Find lang. generated by gramma

Soln $S \to aSAc$, Smallest string $abc \in L(G)$

If we apply $S \to a\underline{S}Ac$, $n-1$ times, we have

$S \Rightarrow a^{n-1}S$

$S \to aS|A$

$A \to bB|b$

$B \to cC|\wedge$

$C \to \wedge$

$S \to aS$

$S \to A$

$S \Rightarrow a^n S$

$S \overset{n+1}{\Rightarrow} a^n S$

$\Rightarrow a^n A$ / $a^n b$

$a^n bB$

$a^n bc$

$a^n bcC$

$a^n bc$

$S \Rightarrow ab$.

$S \Rightarrow aAb$

$\Rightarrow abAab$

$S \to a^n A \to a^n bB$ / $a^n b$

$a^n b$

$a^n bcC - a^n bc$

$ab ab$

$S \to aAb$

$abAab$

$abbAaab$

$abbaaab$

$S \to aAb|ab$.

$a^n b^n$.

$a^n b^n a^n b$.

Q Determine the language generated by grammar having following productions

$S \to aS \mid A$

$A \to bB \mid b$

$B \to cC \mid \wedge$

$C \to \wedge$

$S \to a^n S$

$a^n A \to a^n b$

$\to a^n bB$

Soln The production $S \to aS$ can generate

$S \overset{n-1}{\Rightarrow} a^n S$

where $S \to aS$ is applied $n-1$ times.

$a^n S \to a^n A$

To eliminate S from left side. $S \to A$

$S \Rightarrow a^n A \to a^n b$

$a^n bB \to a^n bc C \to a^n bc$ $(C \to \wedge)$

$a^n b$

$S \Rightarrow a^n bc$ or $a^n b$.

$S \to aS$

$\to a(as)^{n-1}$

$a a^{n-1}$.

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

Q Determine the language generated by grammar

$S \to aAb \mid ab$

$A \to bAa \mid \wedge$

Soln:→ The production $S \to ab$ can generate $S \Rightarrow ab$., which is smallest string in lang.

$S \to aAb$.

ab

$S \Rightarrow aAb$

$\Rightarrow abAab$.

$abab$.

$S \Rightarrow aAb$

$\Rightarrow abAab$.

$ab\,bAa\,ab$

$abbaab$.

$A \Rightarrow b^n a^n$.

by substituting $A \Rightarrow b^n a^n$

$S \Rightarrow ab^n a^n b$

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

Q Determine the language generated by a grammar having productions

$S \to A \mid B \mid C \mid D$

$A \to aA \mid aa$ — aa, aaa, aaaa ... a^n

$B \to bB \mid bb$

$C \to cC \mid cc$

$D \to dD \mid dd$

A generates $\{a^n \mid n \geq 2\}$

B generates $\{b^m \mid m \geq 2\}$

C generates $\{c^l \mid l \geq 2\}$

D —— $\{d^p \mid p \geq 2\}$

$L = \{a^n \cup b^m \cup c^l \cup d^p \mid n, m, l, p \geq 2\}$

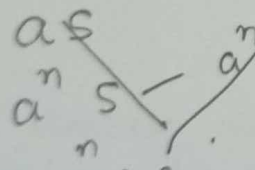
Q $S \to aS \mid bS \mid aa$.

$S \to aa$

$S \to aS$
$\quad a$

$S \to aS \to aaa$

$S \to bS \to baa$



$S \overset{*}{\Rightarrow} (a,b)^* S$

$S \Rightarrow (a,b)^* aa.$

$L = \{$ set of all strings over (a,b) ending with $aa\}$

Q $A \to aA \mid bA \mid \wedge$

$S \to aAb$

$L = \{ s \in (a,b)^* \mid s$

$\{a(a,b)^* b\}$

Q. $S \to 0S11 \mid \wedge$

$S^{n-1} \Rightarrow 0^n S (11)^n$

$\Rightarrow 0^n \wedge (11)^n \Rightarrow 0^n (11)^n$

$\Rightarrow 0^n (1)^{2n}$

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

Q. $S \to aSbb \mid aabbb$

$S^{n-1} \Rightarrow a^n S (bb)^n \qquad (a^n S b^{2n})$

$\Rightarrow a^n aabbb (b)^{2n}$

$\Rightarrow a^{n+2} b^{2n+3}$

$L = \{ a^{n+2} b^{2n+3} \mid n \geq 1 \}$

Grammar generate

Q. $S \to aSB$

$S \to aB$

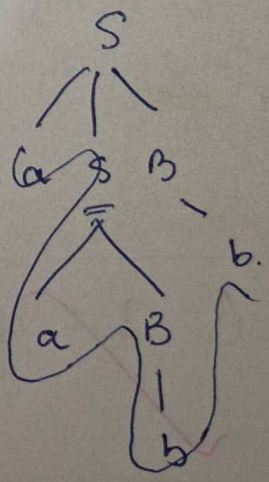
$B \to b.$

$V = \{ S, B \}$

$T = \{ a, b \}$

$S \to \{ S \}$

$S \to a S B$



Derivation:

$S \to a S B$

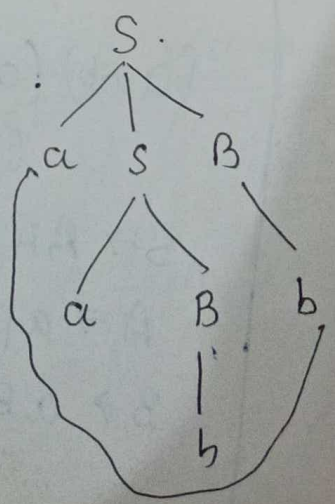
↓ left most symbol

Derivation

$S \to aaBB.$

$S \Rightarrow aabB$ — sentential form

$S \Rightarrow aabb.$



aabb.

Path tree

Parse.

Q1 $L = \{aa, bb, ab, ba\}$

$S \rightarrow aa \mid bb \mid ab \mid ba$

$(a+b)\ (a+b)$

$S \rightarrow AA$

$A \rightarrow a \mid b$

✓✓

Q2 imp

$a^n \mid n \geq 1$

$\{a, aa, aaa, \ldots\}$

$A \rightarrow aA \mid \wedge$

$\downarrow a^*$. $A \rightarrow aA \mid \wedge$

$A \rightarrow SA \mid \varepsilon$

S^*

Q $(a+b)^*$

$S \rightarrow aS \mid bS \mid \wedge$

$S \rightarrow aB \mid bS \mid \wedge$

abab.

$S \rightarrow aS$

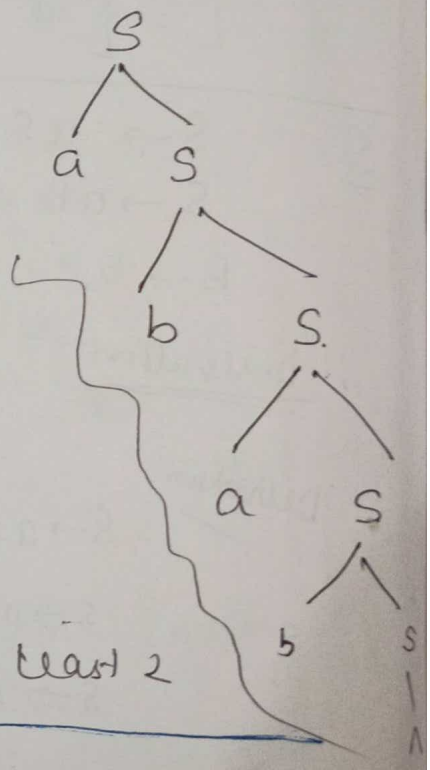
$\rightarrow abS$

$\rightarrow abaS$

$\rightarrow ababS$

$\rightarrow abab$

$\Rightarrow$ abab



Q Set of all strings of length at least 2

$(a+b)\ (a+b)\ (a+b)^*$

$S \rightarrow AAB$.

$A \rightarrow a \mid b$

$B \rightarrow aB \mid bB \mid \wedge$

$(a+b)\ (a+b)\ (a+b)^*$

$S \rightarrow AAB$.

$A \rightarrow a \mid b$

$B \rightarrow aB \mid bB \mid \wedge$.

at most 2

$(a+b+\wedge)\ (a+b+\wedge)$.

$S \to AA$

$A \to a \mid b \mid \wedge$

$S \to AA$

$A \to a \mid b \mid \wedge$

$a\ (a+b)^*\ b$

$S \to a$

Q Set of all strings starting with a & ending with b

$a\ (a+b)^*\ b$

$S \to a\ A\ b$

$A \to aA \mid bA \mid \wedge$

$L = a\ (a+b)^*\ b$

$S \to aAb$. $A \to aA \mid bA \mid \wedge$
$a\ (a+b)^*\ b$.

Q starting & ending with diff symbols

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

$S \to aAb \mid bAa$

$A \to aA \mid bA \mid \wedge$

$S \to aAb \mid bAa$

$A \to aA \mid bA \mid A$ ——— same symbol

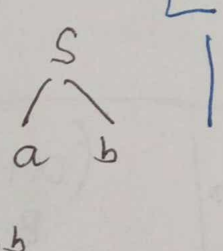
$S \to aAa \mid bAb \mid a \mid b \mid \wedge$

$A \to aA \mid bA \mid \varepsilon$.

Q $a^n b^n \mid n \geq 1$

$a^n b^n$

$S \to aSb \mid ab$.



$ww^R \cup waw^R \cup wbw^R \cup waqw^R \cup wbw^R$.

$S \to aSa \mid bSb \mid a \mid b \mid \wedge$

Set of all palindrom aabb

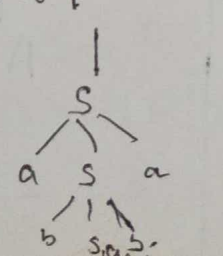
$ww^R \cup waw^R \cup wbw^R$, $w \in (a+b)^*$.

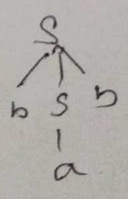
even length

$S \to aSa \mid bSb \mid a \mid b \mid \wedge$

$S \to aSa$

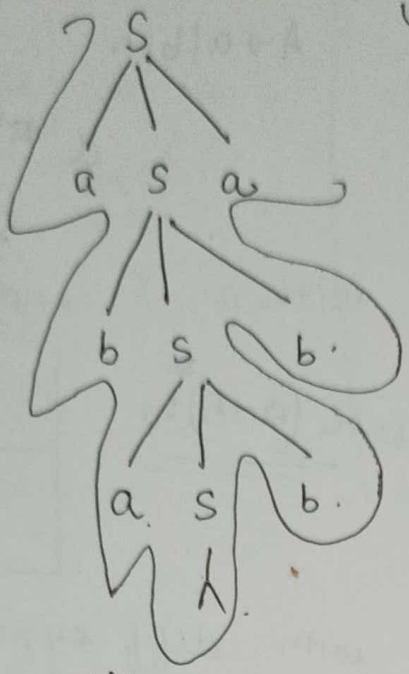
$\to aa$.





abaaba

aba aba



Yield of Parse tree → set of all palindromes

abaaba
Top-down
Left-Right

① $((a+b)(a+b))^*$ → set of even length strings

$S \to BS \mid \wedge$

$B \to AA$

$A \to a \mid b$

② $a^n b^m \mid n, m \geq 1$

Soln

$S \to AB$

$B \to bB \mid b$

$A \to aA \mid a$

③ $a^n b^n c^m \mid n, m \geq 1$

Soln

$S \to AB$

$A \to aAb \mid ab$

$B \to cB \mid c$

$a^n c^m b^n \mid n, m \geq 1$

$S \to aSb \mid aAb$

$A \to cA \mid c$

aa c bb

$S \to aSb$

$a\,a\,S\,b\,b$

$a\,a\,c\,b\,b$

$a^n b^n c^m$ = $n, m \geq 1$

$S \to AB$

$A \to aAb \mid ab$

$B \to cB \mid c$

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

$S \to BS \mid \wedge$

$B \to AA$

$A \to a \mid b$

$S \to aSb$

$a\,a\,A\,b\,b$

$a\,a\,c\,b\,b$

$a^n b^m \mid n, m \geq 1$

$S \to AB$

$B \to bB \mid b$

$A \to aA \mid a$

(1)

$a^n b^n c^m d^n \mid n, m \geq 1$

$S \to AB.$

$A \to aAb \mid ab.$

$B \to cBd \mid cd.$

$a^n b^n c^n \mid n \geq 1.$ ✗

Sol^n Context sensitive grammar.

(2)

$a^n b^{2n} \mid n \geq 1$

$a^3 b^6$ ✓

Sol^n $S \to aSbb \mid abb$

$a^n b^m c^m d^n \mid n, m \geq 1$ ✓

$S \to aSd \mid aAd.$

$A \to bAc \mid bc$

$a^n b^m c^n d^m \mid n, m \geq 1$

Diff to construct a grammar.

$a^{m+n} b^m c^n \mid m, n \geq 1$

Sol^n $a^n a^m b^m c^n$ easy.

$S \to aSc \mid aAc.$

$A \to aAb \mid ab.$

abbccd

$S \to aSd$

$\to ad$ ✗

$S \to aAd$

$\to abAcd$

$abbccd$

$a^n b^{n+m} c^m \mid n, m \geq 1$ ✓

Sol^n $a^n b^n b^m c^m \mid n, m \geq 1$

$S \to AB.$

$A \to aAb \mid ab.$

$B \to bBc \mid bc.$

$a^n b^m c^{n+m} \mid n, m \geq 1$ ✓

Sol^n $a^n b^m c^m c^n$

$S \to aSc \mid aAc$

$A \to bAc \mid bc.$

$a^m b^n \mid n, m \geq 1$

$S \to AB.$

$A \to aA \mid a$

$B \to bB \mid b.$

Classification of grammar

Type 3 :

$A \to \alpha B \mid \beta$ — right linear grammar.
$A, B \in V$
$\alpha, \beta \in T^*$
RLG.

$A \to B\alpha \mid \beta$ — where $A, B \in V$
left linear grammar. $\alpha, \beta \in T^*$.

$A \to aB$.
$B \to aB \mid bB \mid a \mid b$.
} right linear grammar.

$A \to Ba$
$B \to Ba \mid Bb \mid a \mid b$.

$A \to Ba \mid a$ → Left linear
$B \to aB \mid a$ No.
↓
(R.L).

Q Find the grammar generating the language

$$L(G) = \{ a^n b a^n \mid n \geq 1 \}$$

Soln :→

a sa

$A \to aSa$
$S \to aAa$.

$L(G)$ → there must be a production of form

start symbol → (terminal a) start symbol (terminal a)

$S \to aSa$

$S \overset{n-1}{\Rightarrow} a^{n-1} S a^{n-1}$

$S \overset{n-1}{\Rightarrow} a^{n-1} aba\, a^{n-1}$

$S^{n-1} = a^n b a^n$

$S \to aSa \mid aba$

Q set of all strings starting with 00 & ending with 11

Sol $00(0+1)^* 11$

$S \to 00A11$

$A \to 0A \mid 1A \mid \wedge$

$00(0+1)^+ 11$

$S \to 00A11$

$A \to @A' \mid 1A \mid \wedge$

Q Determine grammar that does not generate all strings over $\{0,1\}$ ending with substring 11

$(0+1)^* (00, 01, 10)$

$S' \to S \mid 01 \mid 11 \mid \wedge$

$S \to 0S \mid 1S \mid 00, 01, 10$

Q Determine the grammar G $L = \{ a^n b^m c^n \mid m \geq 2, n \geq 1 \}$

Soln

$S \to aAc \mid aBc$

$A \to aAc \mid ac$

$B \to bB \mid bb$

$S \to aSc \mid aAc$

$A \to bA \mid bb$

Q Determine the grammar G for the language

~~$L = \{ a^n b^m c^n \mid n \geq 1 \} \cup \{ a^n (bc)^n \mid n \geq 1 \}$.~~

$L = \{ a^n b^n c^n \mid n \geq 1 \} \cup \{ a^n (bc)^n \mid n \geq 1 \}$.

Soln

Q Determine the grammar for lang. $L = \{ a^n b^m \mid m+n \text{ is even.} \}$

Soln The value $m+n$ is even if & only if

(i) m & n both are even

(ii) m & n both are odd.

$S \rightarrow AB \mid CD$.

$A \rightarrow aaA \mid \wedge$ (a^{2n})

$B \rightarrow bbB \mid \wedge$ (b^{2n} generates)

$C \rightarrow aaC \mid a$ (generates a^{2n+1})

$D \rightarrow bbD \mid b$ (generates b^{2n+1})

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

$S \overset{*}{\underset{G}{\Rightarrow}} \alpha$, then α is called a sentential form.

Q If $G = (\{S\}, \{0,1\}, \{S \to 0S1, S \to \wedge\}, S)$, find $L(G)$

Soln
$S \to 0S1 \mid \wedge$

$S \to 01$

$S \to 0S1$
$\to 00S11$
0011

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

Q If $G = (\{S\}, \{a\}, \{S \to SS\}, S)$, find language.

Soln
$S \to SS$.

$L(G) = \phi$, since, the only production $S \to SS$ in G has no terminal on right-hand side.

Q $G = (\{S, C\}, \{a, b\}, P, S)$,

$S \to aCa$
$C \to aCa \mid b$

Soln $S \to aCa \Rightarrow aba$. $(\in L(G))$

$S \Rightarrow aCa$
$\Rightarrow aaCaa$
$\Rightarrow aabaa$

$\{a^n b a^n \mid n \geq 1\}$

Q $S \to aS \mid bS \mid a \mid b$, find $L(G)$

$\{a,b\}^+$.

Q $S \to aS \mid a$ then show that $L(G) = \{a\}^+$

Q Let L be set of all palindromes over $\{a, b\}$. Construct a grammar G generating L

Soln Means eg ~~abba~~ abba

~~b~~

abba

that is reverse of string is same string

$S \rightarrow aSa \mid bSb \mid a \mid b \mid \wedge$

Q Consider the grammar G given by

$S \rightarrow 0SA_12 \quad S \rightarrow 012, \quad 2A1 \rightarrow A_12, \quad 1A_1 \rightarrow 11$

Test whether (a) $0011^2 \in L(G)$

(b) $00\,1122 \in L(G)$

$S \Rightarrow W_0, W_1, W_2$ be three sets

$|w| = 5$

$W_0 = \{S\}$

$W_1 = \{012, S, 0SA_12\}$

$W_2 = \{012, S, 0SA_12\}$

(b) $S \Rightarrow 0\underline{S}A_12$ 00 1122

$\Rightarrow 0012\underline{A_1}2$

$\Rightarrow 001\underline{A_1}22$

$\Rightarrow 001122$

Construct a context free grammar generating

(a) $L_1 = \{a^n b^{2n} \mid n \geq 1\}$

(b) $L_2 = \{a^m b^n \mid m \geq n, \; m, n \geq 1\}$

(c) $L_3 = \{a^m b^n \mid m < n, \; m, n \geq 1\}$

(d) $L_4 = \{a^m b^n \mid m, n \geq 0, \; m \neq n\}$

Q If L_1 and L_2 are subsets of $\{a,b\}^*$, prove or disprove

(a) If $L_1 \subseteq L_2$ & L_1 is not regular, then L_2 is not regular → (False)

Sol Let $L_1 = \{a^n b^n \mid n \geq 1\}$ → not regular

$L_2 = \{a, b\}^*$ → regular

So, statement false

(b) If $L_1 \subseteq L_2$ & L_2 is not regular, then L_1 is not regular — False

Soln $L_2 = \{a^n b^n \mid n \geq 1\}$ — not regular

$L_1 = \{a^4 b^4\}$ → regular.

Chomsky classification

Type 3

$A \to \alpha B \mid \beta$

$A, B \in V$

$\alpha, \beta \in T^*$

(if variable is present on right-hand side)

RLG

$A \to B\alpha \mid \beta$

$A, B \in V$

$\alpha, \beta \in T^*$

LLG.

X. → LLG

$A \to Ba \mid a$

$B \to aB \mid a$

↓ RLG.

eg. $A \to aB \mid a$

$B \to aB \mid bB \mid a \mid b$

can't be of Type 3

$A \to Ba \mid a$

$B \to Ba \mid Bb \mid a \mid b$.

either should be complete LLN or RLG.

Regular Lang → Finite automata

Type 2 :- $A \to \alpha$ where $A \in V$ $\alpha \in (VUT)^*$ → Context free Lang

(Context free grammar)

L

Pushdown Automata

[If G is regular, then it must be context free]

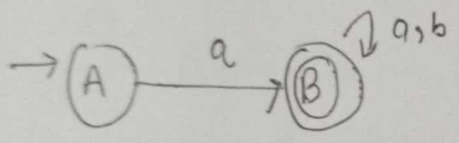
$A \to aAb \mid ab$

$A \Rightarrow aAb$

$\Rightarrow abbb.$

Type 3

Convert FA to Regular Grammar



$A \to aB$

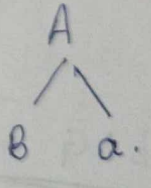
$B \to aB \mid bB \mid \wedge.$

$a(a+b)^*.$

$A \to Ba$

$B \to Ba \mid Bb \mid \wedge.$

Set of all strings starting with a.



Set of all strings ending with a.

$(a+b)^*a.$

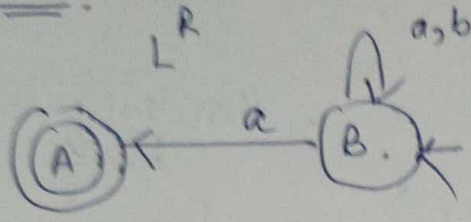
FA → RLG $\xrightarrow{R}$ LLG.

(L) (L) L^R.

FA $\xrightarrow{R}$ FA → RLG $\xrightarrow{R}$ LLR

(L) (L^R) L^R $(L^R)^R = L.$

Consider



$B \to aB \mid bB \mid aA \qquad A \to \wedge \qquad (RLG)$

RLG.

$B \to Ba \mid Bb \mid Aa$

$A \to \wedge$

LLG.

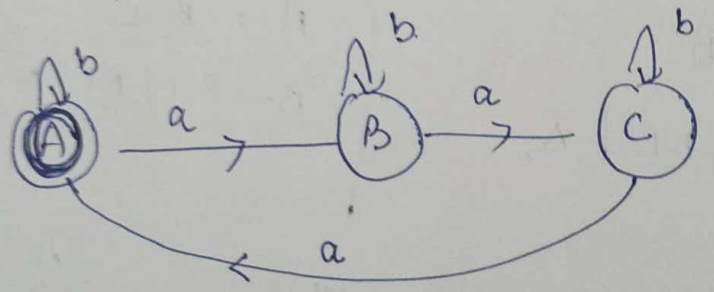
$(L^R)^R = L.$

Convert RLG to FA

$A \to aB \mid bA \mid b$

$B \to aC \mid bB$

$C \to aA \mid bC \mid a$



LLG to FA

$LLG \xrightarrow{R} RLG \; (LL^R) \longrightarrow FA \; L^R \xrightarrow{(R)} FA \; (L^R)^R \to L.$

RG $\rightleftharpoons$ FA (equivalent in power).

RE

RG — R — FA are equivalent.

Type 0 Unrestricted grammars which include all formal grammars. Recognised by Turing Machines
→ any phrase structure grammar without any restrictions.

Type 1 → Context sensitive grammars. A production of form $\phi A \psi \to \phi \alpha \psi$ is called Type 1 production if $\alpha \neq \wedge$ → replacement string.

A → variable
ϕ → ~~right~~ left context
ψ → right context

$A \in N_N$. All natural lang. are context-sensitive

eg.
(1) $abAbcd \to abABbcd$
$\alpha = AB$

(2) $AC \to A$ A → left context
$\wedge$ → right context
$\alpha = \wedge$

(3) $C \to \wedge$ $\wedge$ → left context, right context, $\alpha = \wedge$

(4) $aAbc\triangleright \to abc\triangleright bc\triangleright$
a → left
$bc\triangleright$ → right
$A = bc\triangleright$

(5) $AB \to AbBc$
Left context – A
right context → $\wedge$.

(6) $A \to abA$
Both right & left context are $\wedge$.

- $AB \rightarrow BA$

Type2 : Context-free grammar :

$A \rightarrow \alpha$

$A \in V_N$

$\alpha \in (V_N \cup \Sigma)^*$

No left or right context in left hand side of production.

But no restriction on right hand side of production.

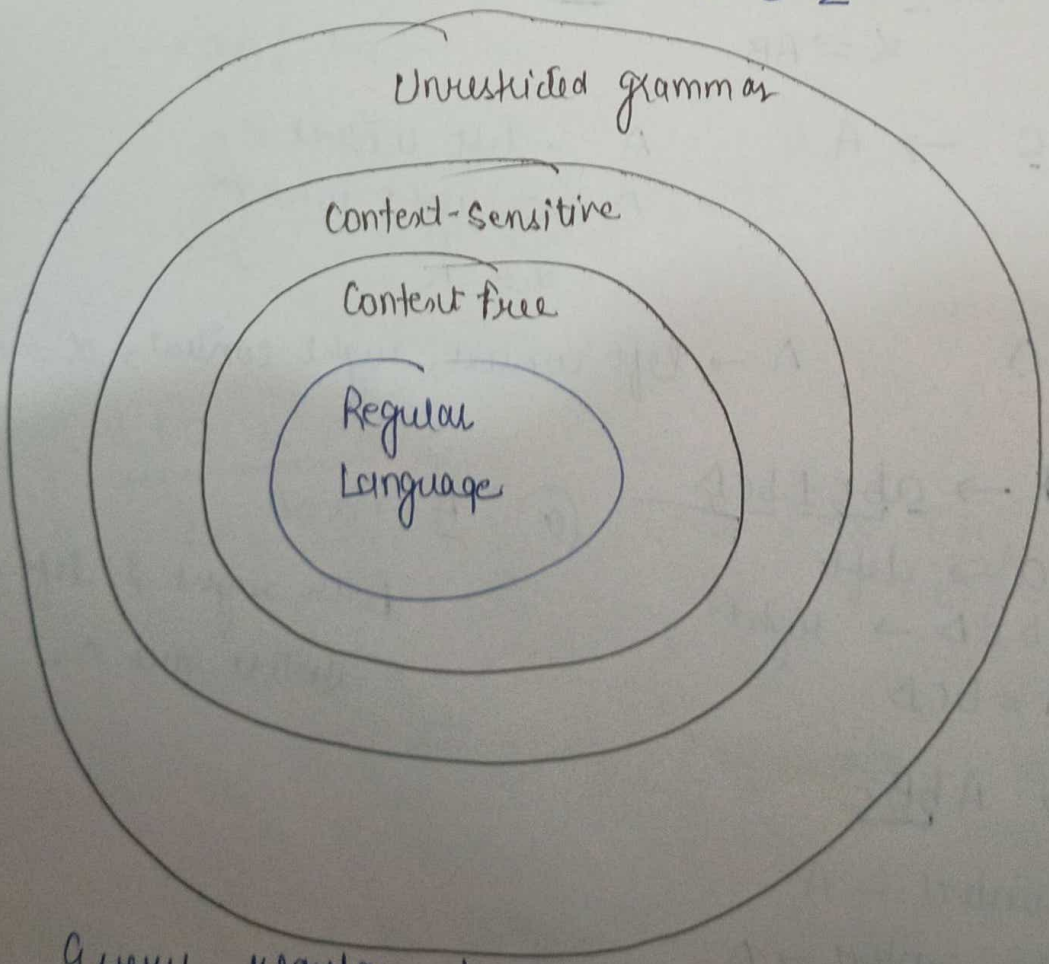
$S \rightarrow aSb \mid \wedge$

$S \rightarrow Aa \quad A \rightarrow a \quad B \rightarrow abC \,,\, A \rightarrow \wedge$

Type3 $A \rightarrow a$

or $A \rightarrow aB$, where $A, B \in V_N$

$a \in \Sigma$



Every regular lang is context free

(1) every regular grammar is context-free
eg $A \to aB$ is context free also.

(2) every context free grammar is context sensitive
$A \to ab \to \wedge$ as left & right context

(3) Every context-sensitive grammar is type 0 or unrestricted grammar

Q Find highest type no. which can be applied to grammar :

(i) $A \to Ba$, $B \to c \mid Ca$, $C \to abc$

Soln $A \to Ba$ $C \to abc$ → Type 2 production
$B \to Ca$
$B \to c$ is type 3.

Highest no → 2

(ii) $B \to aB$, $A \to BAC \mid dab$
type 2. type 2. type 2.

(iii) $A \to aAb$, $Ab \to ab$. → Type 1 production.
Type 2. Type 1.

(iv) $S \to Aa$ $A \to c$ $A \to Ba$ $B \to abc$
Type 2 Type 3. Type 2. Type 2

Highest no → Type 2.

(v) $S \to ASB$, $S \to d$ $A \to aA$ Type 3.
Type 2 Type 3

(vi) $S \to aS \mid ab$
Type 3. Type 2.

$L_{x_1} \subseteq L_{cf_1}$, $L_{cs_1} \subseteq L_0$, $L_{cf_1} \subseteq L_0$.

$L_{x_1} \subseteq L_{cf_1} \subseteq L_{cs_1} \subseteq L_0$.

$L_{RL} \subseteq L_{CFL}$

$L_{CFL} \subseteq L_{CSL}$

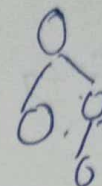
$L_{SL} \subseteq L_0$.

$L_{RL} \subseteq L_{CFL} \subseteq L_{CSL} \subseteq L_0$.

Context- Free Languages ✓ $A \to \alpha$

Context free languages are applied in parsers design. They are useful for describing block structures in programming languages.

$A \to \alpha$ where $A \in V_N$ and $\alpha \in (V_N \cup \Sigma)^*$



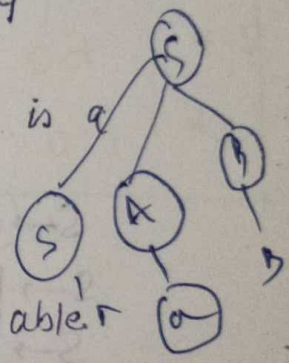
$A \to sabbs$

Derivation Trees → derivations in CFG can be represented using trees such trees representing derivations are called derivation trees 2 marks.

A derivation tree (parse tree) for CFG is a tree satisfying following condn

(1) every vertex has a label which is a variable or terminal or $\wedge$

(2) The root has label S.



(3) The label of an internal node is variable.

(4) If vertices $n_1, n_2, \ldots n_k$ written with labels $X_1, X_2, \ldots X_k$ are sons of vertex n with label A then $A \to X_1 X_2 \ldots X_k$ is a production in P.

(5) A vertex n is a leaf if its label is a $\in$ or $\wedge$

eg. Consider the context free grammar

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

$V = \{E, T, F\}$, $T = \{a, +, *, (,)\}$

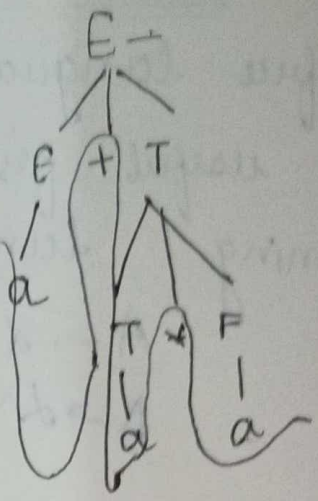
$E \Rightarrow E+T \mid T \mid a$

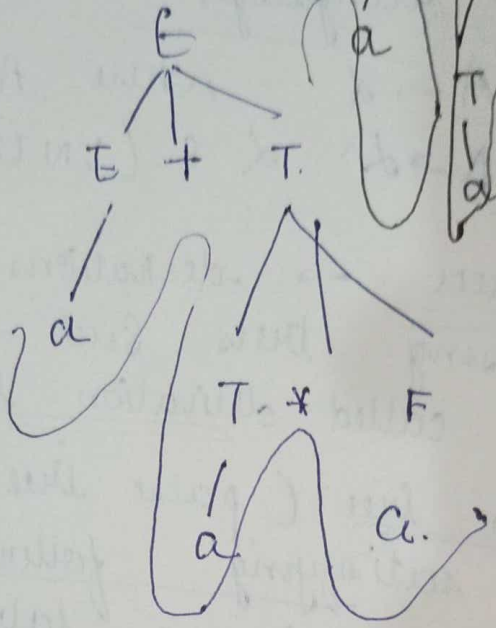
$T \rightarrow T*F \mid F \mid a$

$F \rightarrow (E) \mid a.$

$E \rightarrow E+T$

a*a*a



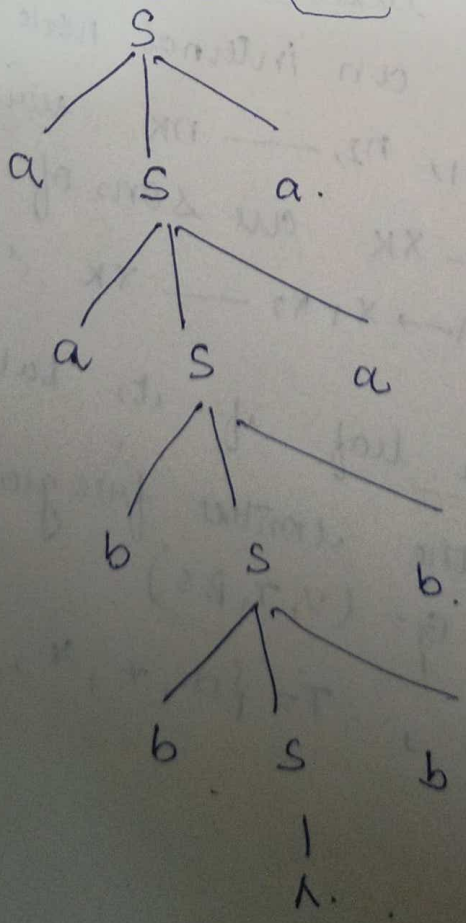


$S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow \wedge$

aabbbbaa.



Leftmost derivation

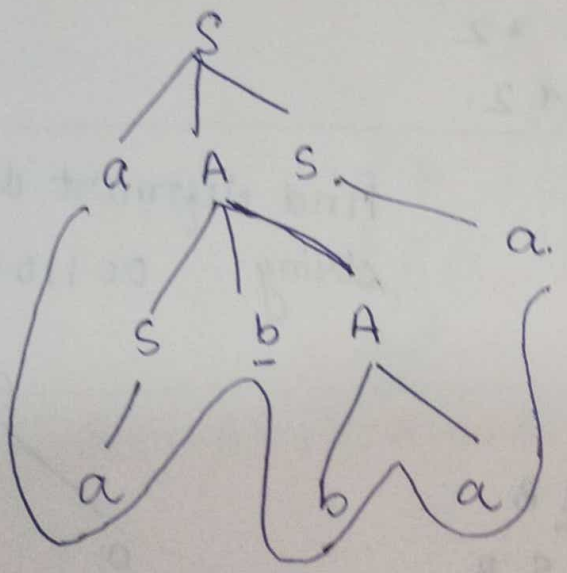
① Consider G whose productions are

$S \to aAS \mid a$

$A \to SbA \mid SS \mid ba$.

$S \overset{*}{\Rightarrow} aabbaa$.

$S \to aAS$



$S \to S+S$

$S+S*S$

$S+S*2$

$S+1*2$

$0+1*2$

$S \to S+S$

$0+S*S$

$0+1*S$

$0+1*2$

Defn leftmost derivation

Rightmost derivation trees

Defn

$A \overset{*}{\Rightarrow} w$ is called leftmost derivation if we apply a production only to the leftmost variable at every step

$A \overset{*}{\Rightarrow} w$ is a _____ to the rightmost variable

$S \to S*S$

$S \to S+S$

$S \to 0$

$S \to 1$

$S \to 2$

Derive using LDT

$0+1*2$

$S \to S+S$

$0+S*S$

$0+1*S$

$0+1*2$.

∴ if production rule is applied to leftmost variable.

Rightmost derivation for $0+1*2$

$S \to S*S$

$S \to S+S$

$S \to 0 \mid 1 \mid 2$

$S \to S+\underline{S}$

$S \to S*\underline{S}$

$S \to S*2$

$S \to S+1*2$

$S \to 0+1*2$

$S \to 0B \mid 1A$

$A \to 0 \mid 0S \mid 1AA$

$B \to 1 \mid 1S \mid 0BB$

Find leftmost derivation for string 00110101

$S \to \underline{0B}$

$\to 0\underline{B}B$

$\to 00\underline{B}BB$

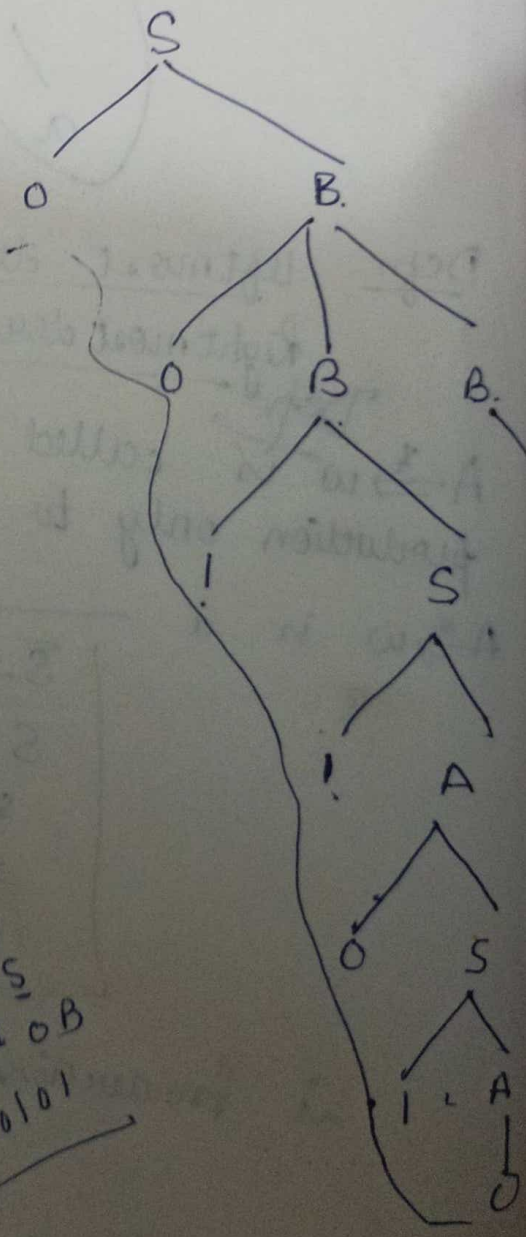
$\to 001S\underline{B}$

$\to 0011\underline{A}B$

$\to 00110\underline{S}B$

$\to 00110\underline{1}A$

$\to 00110101$



Q1 Let G be a CFG whose production rules are

$S \to 0AB$

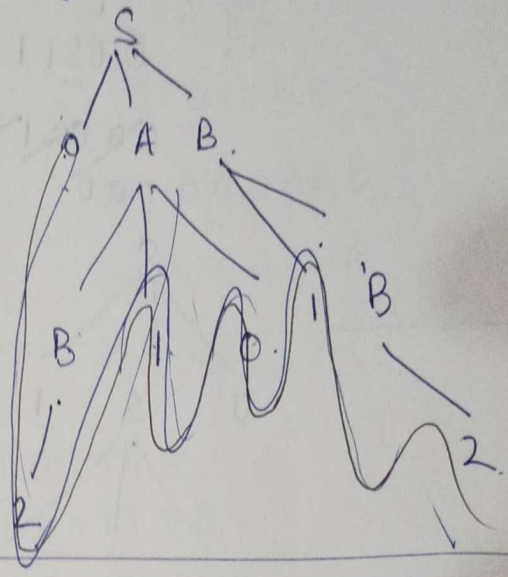
$A \to B|0$

$B \to 1B \mid 2$

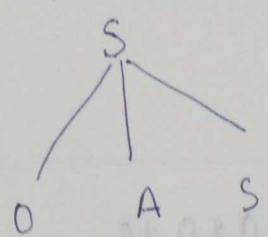
Find derivation tree for string 02.1012

$S \to 0AB$

$\Rightarrow 0$



Q2 $S \to 0$, $S \to 0AS$, $A \to 1S$. 0100100



$S \to 0AS$

$\Rightarrow 01SS$

$\Rightarrow 010S$

$\Rightarrow 0100AS$

$\Rightarrow 0100 1SS$

$\Rightarrow 0100100$

$S \to 0AS$

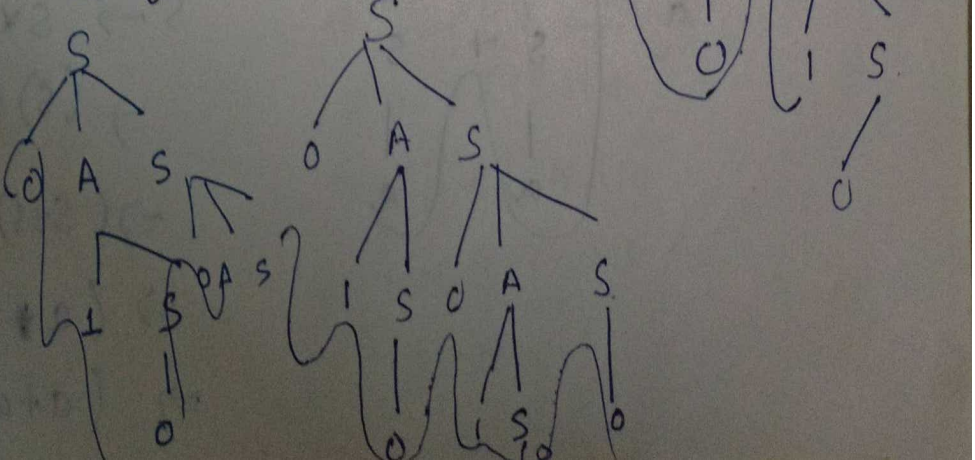
$\Rightarrow 01SS$

$\Rightarrow 010S$

$\Rightarrow 0100AS$

$\Rightarrow 0100 1SS$

$\Rightarrow 0100100$



Let G be a CFG.

$S \to 0S1$

$S \to 01$

Derivation tree for string 000

Also find lang. ~~lang~~ generated by g.

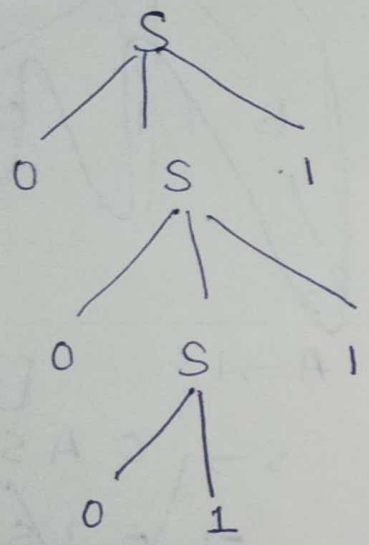
$S \to 0S1$

$0\,0S1\,1 \Rightarrow 000111$

~~$0\,0\,0S1\,1\,1$~~

000

$L(G) = \{0^n 1^n \mid n \geq 1\}$



Q

$S \to S+S$

$S \to S*S$

$S \to (S)$

$S \to a$

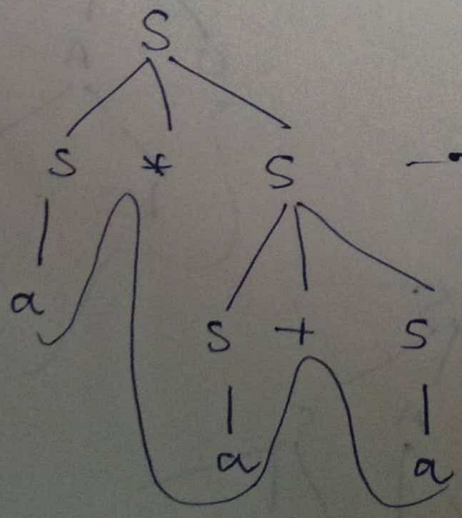
(i) $a*a+a$

(ii) $(a+a)*a$

$S \to S*S$

$a * S+S$

$a * a+a$



$S \to (a+a)*a$

$S \to S*S$

$\Rightarrow (S)*S$

$\Rightarrow (S+S)*S$

$\Rightarrow (a+S)*S$

$\Rightarrow (a+a)*a$

$S \to 0B \mid 1A$

$A \to 0S \mid 1AA \mid 0$

$B \to 1S \mid 0BB \mid 1$

0001101110.

Right most derivation

$S \to 0B$

$\Rightarrow 00BB$

$\Rightarrow 00B1S$

$\Rightarrow 00B11A$

$\Rightarrow 00B110$

$\Rightarrow 00$

Left most derivation

$S \Rightarrow 0B$

$\Rightarrow 00BB$

$\Rightarrow 000BBB$

$\Rightarrow 0001SBB$

$\Rightarrow 00011BB$

$\Rightarrow 000110BB$

$\Rightarrow 0001101B$

$\Rightarrow 00011011S$

$\Rightarrow 000110111A$

$\Rightarrow 0001101110$ ✓

Right most derivation

$S \to 0B$

$\Rightarrow 00BB$

$\Rightarrow 00B1S$

$\Rightarrow 00B11A$

$\Rightarrow 00B110$

$\Rightarrow 000BB110$

$\Rightarrow 000B1S110$

$\Rightarrow 000B11A110$

$\Rightarrow 000B1101110$

$\Rightarrow 00B0B$

Left Most Derivation

Q. $E \to E+E$
$E \to E*E$
$E \to id$

id+id

Soln :- $E \to E+E$
$\Rightarrow id+E$
$\Rightarrow id+id$

(i) id*id+id

$E \to \underline{E}+E$
$\Rightarrow E*E+E$
$\Rightarrow id*E+E \Rightarrow id*id+E$
$\Rightarrow id*id+id.$

abbbb.

Q. $S \to aAB$
$A \to bBb$
$B \to A \mid \wedge$

abbbb.

$S \Rightarrow aAB$
$\Rightarrow abBbB$
$\Rightarrow abAbB$
$\Rightarrow abbBbbB.$
$\Rightarrow abbbb$

$S \Rightarrow aAB$
$\Rightarrow aA\underline{A}$
$\Rightarrow aaAbBb$
$\Rightarrow aAB$
$\Rightarrow aA$
$\Rightarrow abBb$
$\Rightarrow abAb$
$\Rightarrow abbBbb$
$\Rightarrow abbbb.$

Right Most Derivation

$E \to E+E$
$\Rightarrow E+id$
$\Rightarrow id+id$

id*id+id

$E \to \underline{E+E}$
$\Rightarrow E+E*E$
$\Rightarrow E+id$
$\Rightarrow E*E+id$
$\Rightarrow E*id+id$
$\Rightarrow id*id+id$

Q) $G = (N, T, P, S)$ $N = \{S\}$

$T = \{a, b\}$

$P \Rightarrow S \to aSSa$, $S \to b$. abba

$S \Rightarrow aSSa$ (Leftmost derivation)

$\Rightarrow abSa$

$\Rightarrow abba$.

Rightmost derivation :

$S \Rightarrow aSSa$

$\Rightarrow aSba$

$\Rightarrow abba$.

Q)

Soln $G = (N, T, P, S)$, where $N = \{E, I\}$,

$T = \{a, b, c, +, *, (,)\}$,

$P \to E \to E$

$E \to E + E$

$E \to E * E$

$I = a \mid b \mid c$

$S \to aSSa$

$\Rightarrow abSa$

$\Rightarrow abba$

$S \to aSSa$

$\Rightarrow aSba$

$\Rightarrow abba$

Ambiguity in context-free grammars

eg On books selected info is given

Word 'selected' may refer to books or information. So the sentence may be parsed in two different ways.

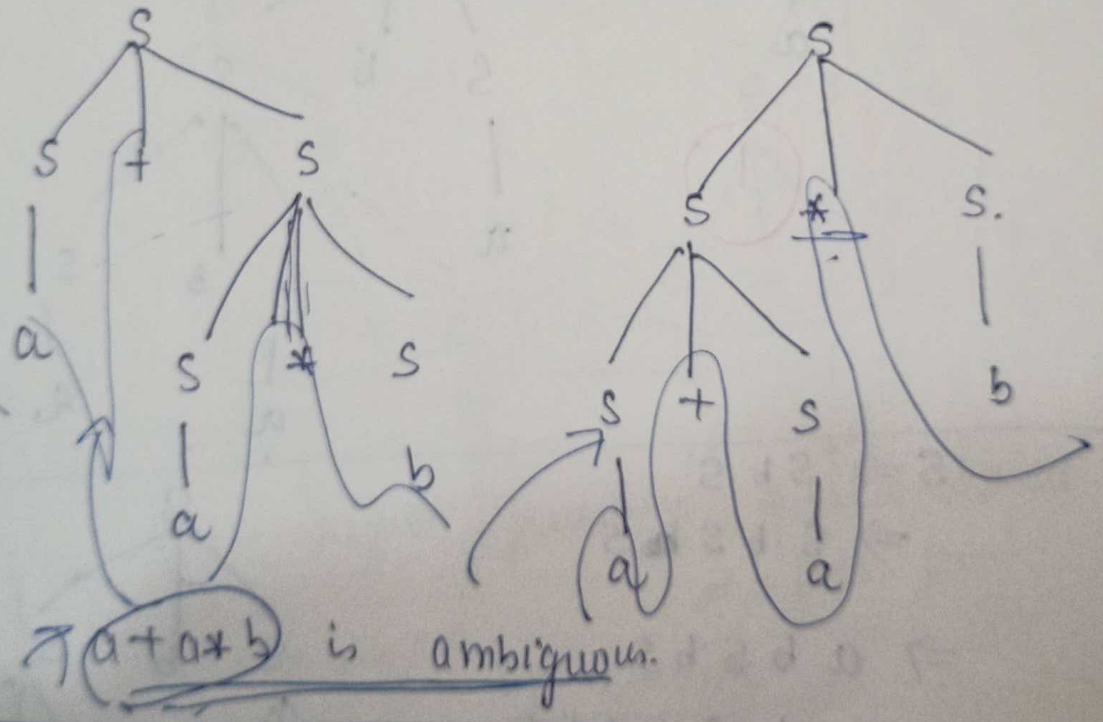
The same terminal string may be yield of two derivation trees

A terminal string $w \in L(G)$ is ambiguous if there exist two or more derivations trees for w. (or there exist two or more leftmost derivation of w)

Ex $G = (\{S\}, \{a, b, +, *\}, P, S)$

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

$a+a*b$



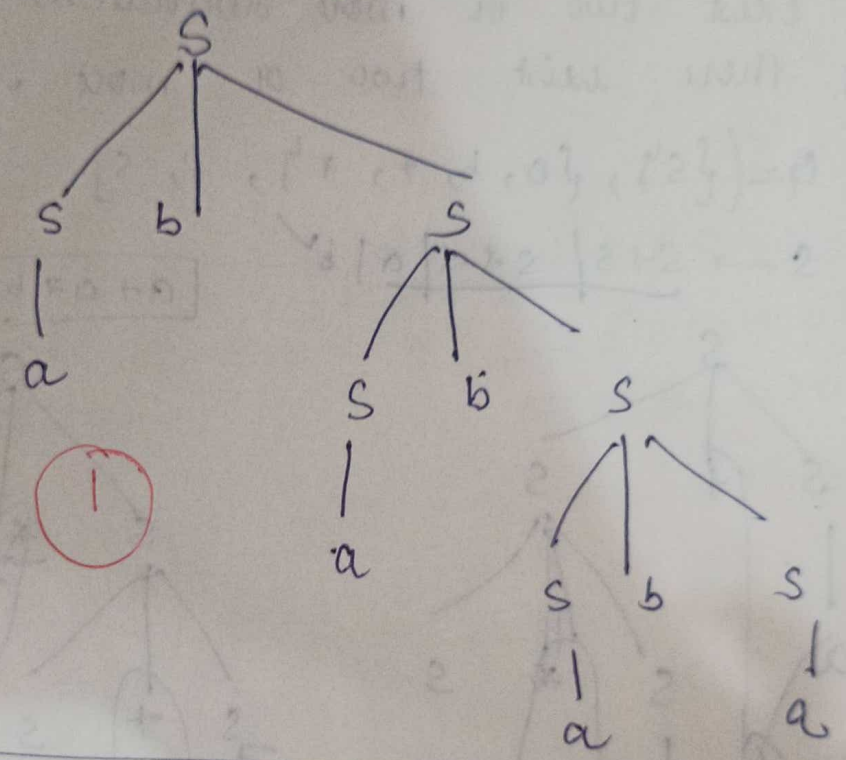
$a+a*b$ is ambiguous.

Q If G is grammar $S \to SbS/a$, show G is ambiguous

Soln To prove G is ambiguous, we have to find $w \in L(G)$ which is ambiguous.

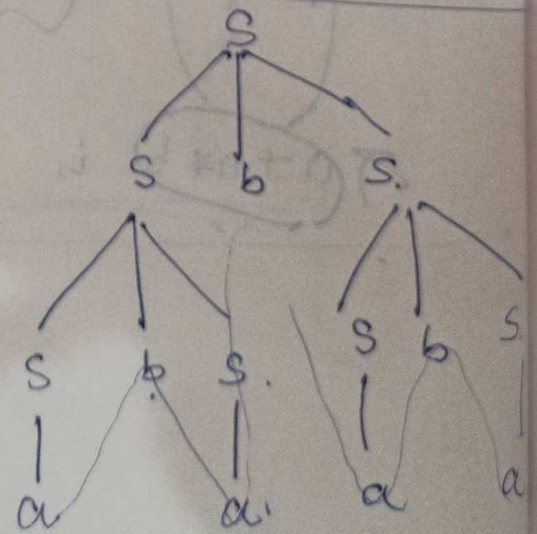
$S \Rightarrow SbS$
$\Rightarrow abSbS$
$\Rightarrow ababa$
$w = abababa \in (LG)$.

$S \Rightarrow SbS$
$\Rightarrow abSbS$
$\Rightarrow ababSbS$
$\Rightarrow abababa$



1

$S \Rightarrow SbS$
$\Rightarrow SbSbS$
$\Rightarrow abSbS$
$\Rightarrow abSbSbS$
$\Rightarrow abababa$



Two derivation trees of abababa.

$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow id$

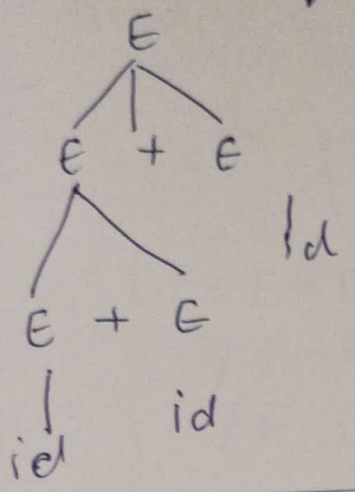
Kiran

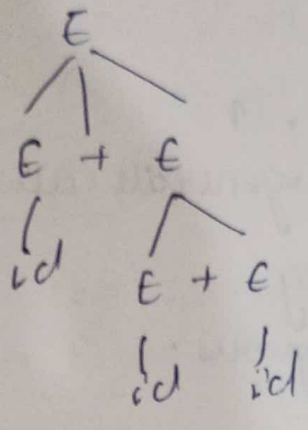
Exceptions A1 A2 B1 B2

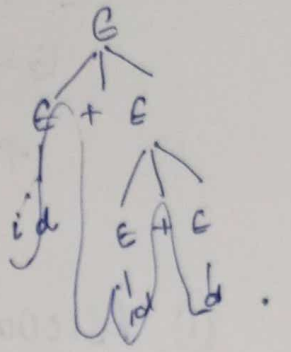
Left Associative ↓

40 Part:

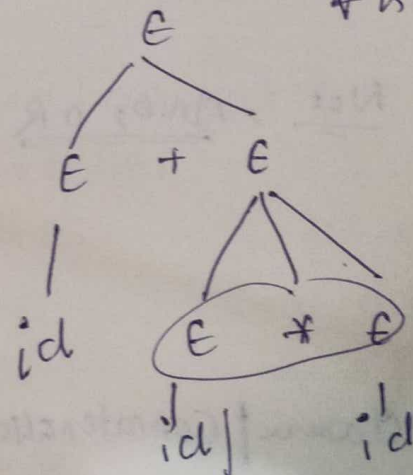
id + id + id ✓

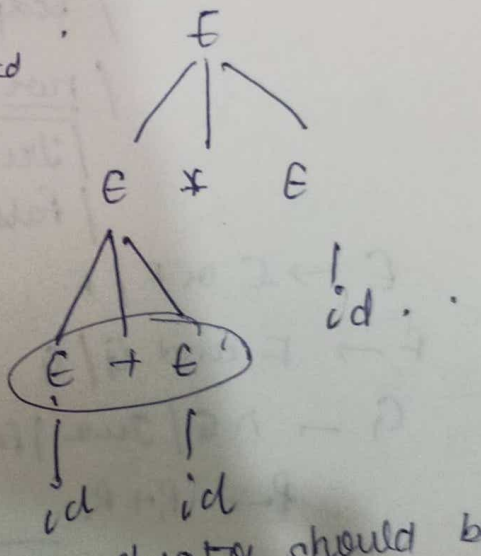






id + id * id ✓ * is evaluated





Highest precedence operator should be at last level.

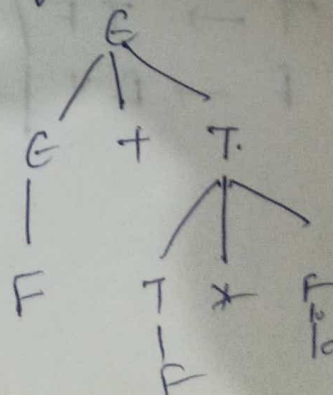
$E \rightarrow E + T / T$ Left Recursive

$T \rightarrow T * F / F$

$F \rightarrow id$

only go for * at last

2^{3^2} $2 \uparrow (3 \uparrow 2)$



$E \to E+T \mid F$

$T \to T * F \mid F$

$F \to G \uparrow F \mid G$ (Eg right recursion).

$G \to id$

$+, *, \uparrow$.

generate all $+$,

2T(312)

① Associativity

② Unambiguous.

bExpression → bexp. or bexp

/ bExp and bExp.

/ not bExp

/ True

/ False.

Not ; AND, OR

$E \to E$ or $F \mid F$.

$F \to F$ and $G \mid G$

$G \to \wedge G \mid$ True $\mid$ False

R → R+R / RR / R^* / a / b / c | Closure | Concatenation | +

$E \to E + T \mid T$.

$T \to TF \mid F$

$F \to F^* \mid a \mid b \mid c$

A → A $ B / B

B → B # C / C

C → C @ D / D

D → d

Ans:-

If $ ⇒ $ → High precedence

⇒

@ ⇒ @

$ < # < @

E → E * F → Left Associative

/ F + E → Right Recursive

/ F

F → F - F → ambiguous

/ id

* = + * ⇒ *

+ < + + ⇒ +

S → 0 | 0A1 | 01S1

A → 0AA1

A → 1S

Let w = 0101

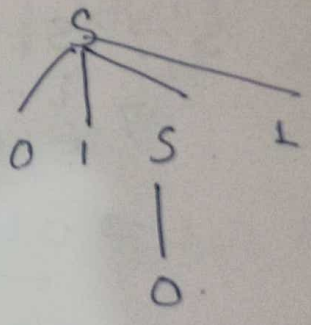
S ⇒ 01S1

⇒ 0101

S → 0A1

⇒ 01S1

⇒ 0101



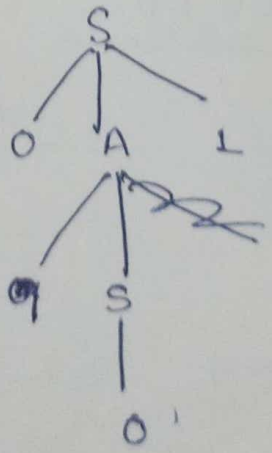
S → 01S1

→ 0101

S ⇒ 0A1

⇒ 01S1

⇒ 0101



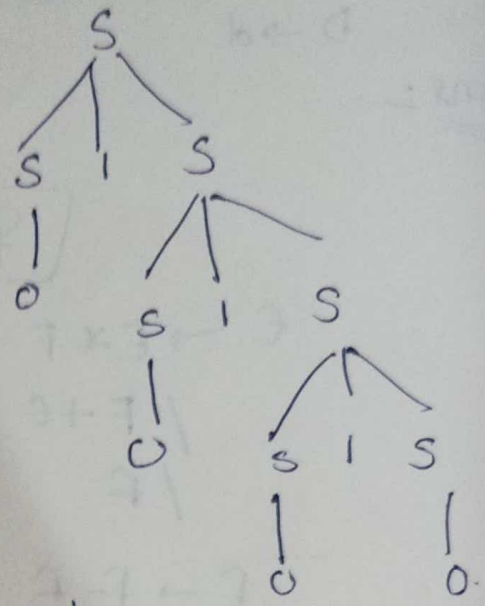
✓ ambiguous

Show that grammar is ambiguous

$S \to SIS \mid 0$

$w = 0\,1\,0\,1\,0\,1\,0$

$S \Rightarrow SIS$
$\Rightarrow 0IS$
$\Rightarrow 0ISIS$
$\Rightarrow 0I0IS$
$\Rightarrow 0I0ISIS$
$\Rightarrow 0I0I0I0$

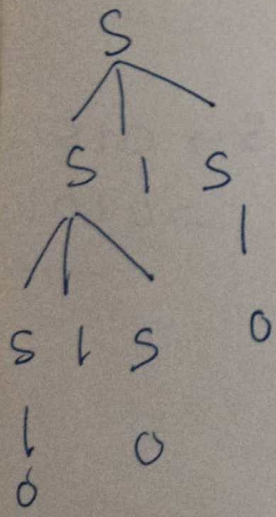


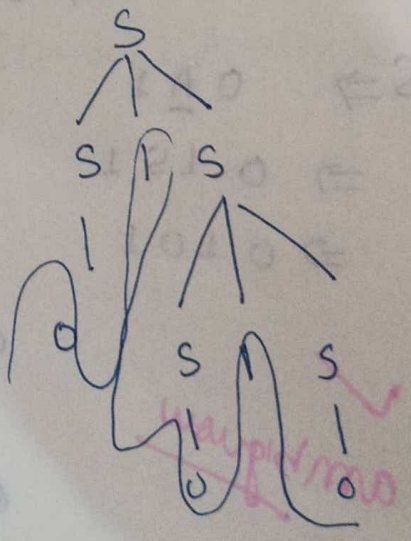
(ii) SIS
$\Rightarrow SISIS$
$\Rightarrow 0ISIS$
$\Rightarrow 0I0IS$
$\Rightarrow 0I0ISIS$
$\Rightarrow 0I0I0IS$
$\Rightarrow 0I0I0I0$

$S \to SIS \mid 0$

$S \Rightarrow SIS$
$\Rightarrow 0IS$
$\Rightarrow 0ISIS$
$\Rightarrow 0I0I0$

$S \Rightarrow SIS$
$\Rightarrow SISIS$
$\Rightarrow 0I0I0$





Simplification of Context Free Grammars

In CFG G, it may not be necessary to use all symbols in $V_N \cup \Sigma$, or all productions in P for deriving sentences. So, when we have to eliminate those symbols and productions in G which are not useful for the derivation of sentences.

$G = (\{S, A, B, C, E\}, \{a, b, c\}, P, S)$

where

(1) Useless productions → Useless
(2) Null productions → Null
(3) Unit productions → Unit

(1) **Useless productions** whose variables (V) & terminals (T) that do not appear in derivation of any string from start symbol

eg $S \to AB$
$A \to a$
$B \to b$
$C \to c$.

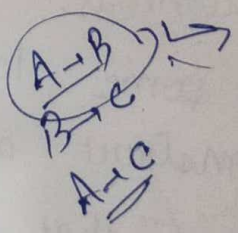
This production can never be used for deriving any string

Here $C \to c$ is useless production because we never use this production in deriving any string from start variable S.

(2) **Null productions** $A \to \Lambda$, where A is variable

eg $S \to AB \mid D$
$D \to \Lambda$
$A \to a$
$B \to b$

$D \to \Lambda$ is null production.



(3) **Unit productions** $A \to B$ (where A & B are variables)

$S \rightarrow AB \mid C$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow D$

$D \rightarrow E$

$E \rightarrow F$

$F \rightarrow a$

$S \rightarrow C$

$C \rightarrow D$, $D \rightarrow E$, $S \rightarrow C$ & $E \rightarrow F$ are unit productions.

The grammar that we obtain after deleting useless production rules are called reduced grammar.

eg $S \rightarrow ABC \mid a$

$A \rightarrow b$

$B \rightarrow c$

$C \rightarrow d$

$E \rightarrow e$

$F \rightarrow f$

$G \rightarrow g$

→ useless production.

$S \rightarrow ABC \mid a$

$A \rightarrow b$

$B \rightarrow c$

$C \rightarrow d$.

For every CFG, G thus exists a reduced grammar G', which is eq. to G.

Step1 → We construct a new grammar G_1 equivalent to given grammar so that every variable in G_1 derives some terminal string

Step2 We construct a grammar $G' = (V', N', \varepsilon, P)$ eq. to G_1 so that every symbol in new grammar G' appears in some sentential form.

$S \rightarrow AB \mid C$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow D$

$D \rightarrow E$

$E \rightarrow F$

$F \rightarrow a$

$S \rightarrow C$

$C \rightarrow D$, $D \rightarrow E$, $S \rightarrow C$ & $E \rightarrow F$ are unit productions.

The grammar that we obtain after deleting useless production rules are called reduced grammar.

eg $S \rightarrow ABC \mid a$

$A \rightarrow b$

$B \rightarrow C$

$C \rightarrow d$

$E \rightarrow e$

$f \rightarrow f$

$g \rightarrow g$

→ useless production.

$S \rightarrow ABC \mid a$

$A \rightarrow b$

$B \rightarrow c$

$C \rightarrow d$.

For every CFG, G there exists a reduced grammar G', which is eq. to G.

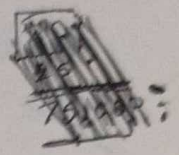
Step1 → We construct a new grammar G_1 eqv to given grammar G so that every variable in G_1 derives some terminal string

Step2 We construct a grammar $G' = (V', N', \epsilon', P', S)$ eq. to G_1 so that every symbol in new grammar G' appears in some sentential form.

If G is a context free grammar such that $L(G) \neq \phi$ then we can find an equivalent grammar G_1 such that each variable in G_1 derives some terminal string

Proof :- Let $G = (V, T, P, S)$ is a context free grammar. We define a new grammar G_1 equivalent to given grammar G such that

$$G_1 = (V', T, P', S)$$



V' and P'

Constn of V' → Let $U_i \subseteq V$ for $i \geq 1$

U_i contains new set of variables for G_1 ($U_i \subseteq V$) U_i recursively as

$U_1 = \{ B \in V \}$ there exists a production $B \to w$ where $w \in T^*$

If $U_1 = \phi$ this means there is no production of from $B \to w$, $w \in T^*$.

$U_{i+1} = U_i \cup \{ B \in V \}$ there exists some productions of form $B \to \alpha$ with $\alpha \in (T \cup U_i)^* \}$

(ii) **Constn of P' :-**

P' as $P' = \{ B \to \alpha \mid B, \alpha \in (V' \cup T)^* \}$

Reduced grammar as $G = (V', T, P', S)$ where $S \in V'$.

Ques $G = (V, T, P, S)$ P is defined as

$S \to AB$ $\quad G \to 1$

$A \to 0$ $\quad H \to 0$

$B \to 1$ $\quad I \to 0$

$A \to C$ $\quad J \to 1$

$F \to 0$

find grammar G_1 such that every variable derives some terminal string

Soln Let G_1 is new grammar equivalent to given grammar G.

Let $G_1 = (V', T, P', S)$.

(a) Consm of V' :-

$U_1 = \{A, B, F, G, H, I, J\}$ since every variable in U_1 derives some terminal like $A \to$ $B \to 1$, $F \to 0$, $G \to 1$, $H \to 0$, $I \to 0$ & $J \to 1$ with terminal string on right hand side

$U_2 = U_1 \cup \{B_1 \in V \mid B_1 \to \alpha$ for some $\alpha \in (T \cup U_1)^*\}$

$= \{A, B, F, G, H, I, J\} \cup \{B_1 \in V / B_1 \to \alpha$.

$= \{A, B, F, G, H, I, J\} \cup \{S\}$

$= \{A, B, F, G, H, I, J, S\}$

$U_3 = U_2 \cup \{B_1 \in V \mid B_1 \to \alpha$ for $\alpha \in (T \cup U_2)^*\}$

$= \{A, B, F, G, H, I, J, S\} \cup \phi$

$= U_2$

$V' = \{ A, B, F, G, H, I, J, S \}$.

(b) Constn of P'

$P' = \{ B_1 \to \alpha \mid B_1, \alpha \in (V' \cup T)^* \}$

$= \{ S \to AB$

$A \to 0$

$B \to 1$

$F \to 0$

$G \to 1$

$H \to 0$

$I \to 0$

$G \to 1 \}$

$G_1 = (V', T, P', S)$ where $V' = \{ S, A, B, F, G, H, I, J \}$

$T = \{0, 1\}$.

$S = \{S\}$.

Every variable in G_1 derives some terminal string.

Ex2 Consider $G = (V, T, P, S)$

$S \to AB$

$A \to b$

$B \to a$

$B \to D$

$E \to a$

Sol Let G_1 is the new grammar equivalent to given grammar G.

Let $G_1 = (V', T, P', S)$,

a) Constn of V'

$U_1 = \{ A, B, E \}$ since every variable in U_1 derives some terminal like ____.

$U_2 = U_1 \cup \{ B_1 \in V \mid B_1 \to \alpha$ for $\alpha \in (T \cup U_1)^* \}$

$= \{ A, B, E \} \cup \{ S \}$

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

$U_3 = U_2 \cup \{ B_1 \in V \mid B_1 \to \alpha \}$

$= \{ A, B, E, S \} \cup P$

$= U_2$

$V' = \{ S, A, B, E \}$ ✓

(b) Constn of P'

$P' = \{ B_1 \to \alpha \mid B_1, \alpha \in (V' \cup T)^* \}$

$= \{ S \to AB$
$A \to b$
$B \to a$
$E \to a \}$

Every variable derives some terminal string

$G_1 = (V', T, P', S)$ —

Theorem :- If we are given $G = (V, T, P, S)$ then we can construct an equivalent grammar $G_1 = (V', T', P', S)$ such that every symbol in $V' \cup T'$ appears in some sentential form

Soln Let $G_1 = (V', T', P', S)$ is a new context free grammar

(i) Constn of U_i for $i \geq 1$

(a) $U_1 = \{S\}$

(b) $U_{i+1} = U_1 \cup \{ Y \in (V \cup T)$ there exists a production $B \to \alpha$ with $B \in U_i$ and α containing symbol

(c) $U_k = U_1 \cup U_2 \cup \dots \cup U_k$.

(ii) Constn of V', T' & P'

$V' = V \cap U_k$

$T' = T \cap U_k$.

$P' = \{ B \to \alpha , B \in U_k \}$

Ex. Let G_1 is context free grammar,

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

$V = \{ S, A, B, F, G, H, I, J \}$

$T = \{ 0, 1 \}$

$S \to \{ S \}$.

P: $S \to AB$ $A \to 0$ $B \to 1$, $F \to 0$, $G \to 1$, $H \to 0$

$I \to 0$, $J \to 1$

Find equivalent grammar G_1 such that every symbol in G_1 appears in some sentential form.

Soln Let $G_1 = (V', T', P', S)$ be a new context free grammar equivalent to a given CFG. G

(a) Constn of U_i :-

(i) $U_1 = \{ S \}$

(ii) $U_2 = \{ S \} \cup \{ Y \in V \cup T \mid B_1 \to \alpha,$ with $B_1 \in U_1$ & α containing $Y \}$

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

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

(iii) $U_3 = U_2 \cup \{ \dots \}$

$= \{ S, A, B \} \cup \{ 0, 1 \}$

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

(iv) $V_4 = V_3 \cup \{Y \in V \cup T \mid$ there exists a production $B_1 \to \alpha$ with $B_1 \in V_3$ & α contains $Y\}$

$= \{S, A, B, 0, 1\} \cup \phi$

$= V_3$

$G_1 = (V', T', P', S)$

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

$T' = \{0, 1\}$

$P' = \{S \to AB, \; A \to 0, \; B \to 1\}$

Theorem 4 For every context free grammar G there exists a reduced context free grammar G_2 which is eq. to G.

To construct reduced grammar, we apply following two steps:

Step 1 :- Using theorem 1, we will find new context free grammar G_1 equivalent to G so that every variable in G_1 derives some terminal string

Step 2 Using theorem 2, we will find new CFG G_2 equivalent to G_1 so that every symbol in G_2 appears in some sentential form.

G_2 is reduced grammar.

In order to get reduced grammar the above steps must be applied in order. If we apply step 2 before step 1, then we will fail to eliminate useless symbols.

Example 1 :- Find reduced grammar equivalent to given grammar G.

$S \rightarrow OBO$

$B \rightarrow S1 \mid 1AA \mid COB$

$A \rightarrow 011 \mid CC$

$E \rightarrow OA$

$C \rightarrow OCB.$

Step 1 Let $G_1 = (V', T, P', S)$ be context free grammar equivalent to given context free grammar.

(a) Consist of V'

$V_1 = \{A\}$ as $A \rightarrow 011$ is only production with terminal on right hand side

$V_2 = \{A\} \cup \{E, B\}$ $E \rightarrow OA$, $B \rightarrow 1AA$ are productions with right hand side in $(TU\{A\})^*$

$V_2 = \{A, E, B\}$

$V_3 = \{A, E, B\} \cup \{S\}$ as $S \rightarrow OBO$

$= (T \cup \{A, E, B\})^*$

$V_4 = V_3 \cup \emptyset$

$V_4 = V_3.$

$V' = \{A, E, B, S\}.$

(b) Constn of P'

$P' = \{ B_1 \rightarrow \alpha \mid \alpha \in (V' \cup T)^* \}$ where B is in b'

$\{ S \rightarrow OBO$
$B \rightarrow S1$
$B \rightarrow 1AA$
$A \rightarrow 01$
$\epsilon \rightarrow OA \}$

Step2 In this we find new context free grammar G_2 equivalent to G_1 such that every symbol in G_2 is in sentential form

Let $G_2 = (V'', T', P'', S)$

(a) Constn of U_i

(i) $U_1 = \{S\}$ where S is start symbol

(ii) $U_2 = \{S\} \cup \{O, B\}$

$= \{S, B, O\}$

(iii) $U_3 = \{S, B, O\} \cup \{S, 1, A\}$

$= \{S, B, A, O, 1\}$

(iv) $U_4 = \{S, B, A, O, 1\} \cup \phi$

$= U_3$.

(b) Constn of V'', T' & P''

$V'' = \{S, B, A\}$, $T' = \{0, 1\}$

$P' = \{ S \rightarrow OBO,$
$B \rightarrow S1$

$B \rightarrow 1AA$
$A \rightarrow 0 \mid 1$

S is start variable

Hence, G_2 is reduced grammar

Elimination of Null Productions

The productions of form $B \rightarrow \wedge$ are called null productions.

eg $S \rightarrow aS$

$S \rightarrow aA$

$S \rightarrow \wedge$

$A \rightarrow \wedge$

→ Null productions.

$S \rightarrow aS \mid aA \mid \wedge$

$A \rightarrow \wedge$.

$S \rightarrow aS \mid aA \mid a$

(A → ∧)

$S \rightarrow aS$

$S \rightarrow a$

$S \rightarrow \wedge$.

This does not change the language defined by given context free grammar. But if we delete $S \rightarrow \wedge$, then we are unable to get language

$L(G_1) = L(G) = \{a^n \mid n \geq 0\}$

Nullable Variable

→ If $A \overset{*}{\Rightarrow} \wedge$.

Def ① Any variable A is nullable if there is production $A \rightarrow \wedge$

② A is also called nullable if

$A \rightarrow B_1 B_2 \text{---} B_n$ and

$B_1, B_2 \text{---} B_n \rightarrow \wedge$

A is nullable

Theorem If $G = (V, T, P, S)$ be any context free grammar with $\wedge$ productions then there is a grammar G_1 that has no null production such that

$$L(G_1) = L(G) - \{\wedge\}$$

Proof Let $G_1 = (V, T, P', S)$ be context free grammar equivalent to given context free grammar G

Step1 :— Constn of set of nullable variables

(i) $V_1 = \{ A \mid A \in V$ and P containing prod $A \to \wedge \}$

(ii) $V_{i+1} = V_1 \cup \{ A \mid A \in V$ and P — $A \to \alpha$ for some $\alpha \in V P^* \}$

Step2 All productions which do not have any nullable variable on right hand side are included in P'.

If $A \to X_1 X_2 \text{ — } X_m$ is in P, then productions of form $A \to \beta_1 \beta_2 \text{ — } \beta_n$ are included in P' where $B_i = x_i$ if $x_i \notin V$ or $\wedge$.

Q Consider context free grammar G whose productions are.

$S \to OS$
$S \to AB$
$A \to \wedge$
$B \to \wedge$
$C \to \wedge$
$E \to \wedge$
$F \to \wedge$
$D \to$ ~~$\wedge$~~

eg. $S \to aS \mid aA \mid A$
$A \to \wedge$
$S \to \wedge$
~~$A \to \wedge$~~

Construct a CF G G_1 without null

productions generating $L(G) - \Lambda$

Soln (i) $U_1 = \{ A_i \in V \mid A_i \to \alpha$ is a production in $G \}$

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

(ii) $U_2 = U_1 \cup \{ A_i \in V \mid$ there exists a production $A_i \to \alpha$ with $\alpha \in U_1^* \}$

$= \{ A, B, C, E, F \} \cup \{ S \}$

$= \{ S, A, B, C, E, F \}$ as $S \to AB$. with $AB \in U_1^*$.

(iii) $U_3 = \{ S, A, B, C, E, F \} \cup \phi$

$= U_2$.

Step2 Constn of P':

(1) $D \to 1$ is included in P'

(2) $S \to 0S$ gives rise to $S \to 0S$ and $S \to 0$

(3) $S \to AB$ gives rise to $S \to AB$, $S \to A$, $S \to B$.

$G_1 = (V, T, P', S)$

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

$T = \{ 0, 1 \}$

$S = \{ S \}$

$P' \to$ $S \to AB$, $S \to A$, $S \to B$, $S \to 0S$, $S \to 0$, $D \to 1$.

(Q) $S \to 0S$, $S \to AB$, $A \to 0$, $B \to 1$.

Soln. Step1 :- Constn of set of nullable variables

(1) $U_1 = \{ A_i \in V \mid A_i \to \Lambda$ is a prodn in $G \}$

$= \{ \phi \}$

(2) $U_2 = U_1 = \{ A_i \in V \mid$ ——— $A_i \to \alpha$.

$\{ \phi \} \cup \{ \phi \} = U_1 \Rightarrow U = \phi$

Q $S \to AB$
$A \to \wedge$
$B \to \wedge$
$A \to 0$
$B \to 1$

Soln Step 1. Constn of set of nullable variables

(i) $U_1 = \{ A_1 \in V \mid A_1 \to \wedge$ is a production $\}$
$= \{A, B\}$

(ii) $U_2 = U_1 \cup \{ A_1 \in V \mid$ there $\longrightarrow A_1 \to \alpha$.
with $\alpha \in U_1^*$.

$= \{ A, B \} \cup \{ S \}$
$= \{ S, A, B \}$ —

(iii) $U_3 = \{ S, A, B \} \cup \phi$
$= U_2$.

$\boxed{U = \{ S, A, B \}}$

Constn of P': (1) $A \to 0$ are included in P'
~~(2)~~ $B \to 1$
(2) $S \to AB$ gives rise to $S \to A$, $S \to$

$G_1 = (V, T, P', S)$
$V = \{S, A, B\}$
$T = \{0, 1\}$
$S = \{S\}$

$S \to AB$
$A \to 0$
$B \to 1$
$S \to A$
$S \to B$

Q. $S \to aS,\ S \to AB,\ S \to cA,\ A \to a$

find a grammar G' such that every variable in G' derive some terminal string & every symbol (non-terminals & terminals) appears in some sentential form.

Soln. First of all we will construct grammar G_1 such that every variable in G_1 derives some terminal string.

Suppose the given grammar $G = (V_N, \Sigma, P, S)$ and constructed grammar G_1

$G_1 = (V_N', \Sigma, P', S)$

(i) Constn of V_N' :-

$U_1 = \{A\}$ because there is a production $A \to a$.

$U_2 = U_1 \cup \{A' \in V_N \mid A' \to \alpha \text{ for some } \alpha \in (\Sigma \cup U_1)^*\}$

$= \{A\} \cup \{S\}$

$= \{A, S\}$

$U_3 = U_2 \cup \{A' \in V_N \mid A' \to \alpha \text{ for some } \alpha \in (\Sigma \cup U_2)^*\}$

$= \{A, S\} \cup \{S\}$.

$= \{A, S\}$

$U_4 = U_3 \cup \{\phi\}$

$V_N' = \{A, S\}$

(ii) Constn of P'

$P' = \{A' \to \alpha \mid A', \alpha \in (V_N' \cup \Sigma)^*\}$

$= \{S \to aS,\ S \to cA,\ A \to a\}$.

$G_1 = (\{S, A\}, \{a, c\}, \{S \to aS,\ S \to cA,\ A \to a\}, S)$.

$G' = (V''_N, \Sigma'', P'', S)$

$U_1 = \{S\}$ · As S is start symbol

$U_2 = \{S\} \cup \{a, S, c, A\}$

$= \{S, A, a, c\}$

$U_3 = U_2 \cup \{a\}$

$= \{S, A, a, c\}$

$U_4 = U_3 \cup \{\emptyset\}$

$= \{S, A, a, c\}$.

$V_N'' = V_N' \cap U_4 = \{S, A\}$

$\Sigma'' = \Sigma \cap U_4 = \{a, c\}$.

$P'' = \{A \to \beta \mid A \in W\} = \{S \to aS,\ S \to cA,\ A \to a\}$

G' .

Elimination of Unit Productions

eg $S \to B$, $B \to C$, $C \to D$, $D \to F$, $F \to G$, $G \to 01$

$L(G) = 01$.

$S \overset{*}{\Rightarrow} 01$

$S \to AB$, $S \to AX$, $S \to 1$, $A \to 0$, $A \to BX$, $B \to CX$, $B \to 1$, $C \to DX$, $C \to 0$, $D \to EX$, $D \to 0$, $E \to 0$, $E \to SX$

$G_1 = (V, T, P', S)$

(1) Add to P' all non unit productions of P

eg.

$P' = \{ A \to 0, B \to 1, C \to 0, D \to 0, E \to 0, S \to 1, S \to AB \}$

(b) For every pair (B, C), where C is B-derivable.

S-derivable $= \{ A, B, C, D, E \}$

A-derivable $= \{ B, C, D, E, S \}$

B-deri $= \{ C, D, E, S, A \}$

C-derivable $= \{ D, E, S, A, B \}$

D- $\{ E, S, A, B, C \}$

E- $\{ S, A, B, C, D \}$

for $(S, A) \longrightarrow S \to 0$

for $(S, B) \longrightarrow B \to 1$

$(S, C) \longrightarrow S \to 0$

$(S, D) \longrightarrow S \to 0$

$(S, E) \longrightarrow S \to 0$

$(A, B) \to A \to 1$

$(A, C) \to A \to 0$

$(A, D) \to A \to 0$

$(A, E) \to A \to 0$

$(A, S) \to A \to 1$

$S \to AB$

$S \to 1$ $\quad A \to 0$ $\quad B \to 1$ $\quad C \to 0$

$S \to 0$ $\quad A \to 1$ $\quad B \to 0$ $\quad C \to 1$

$D \to 0$ $\quad D \to 1$ $\quad E \to 0$ $\quad E \to 1$

(1) Eliminate all Null productions ✓

(2) Eliminate all unit productions ✓

(3) Eliminate all useless symbols. ✓

Q Find a reduced grammar equivalent to grammar G whose productions are

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

Sol Step1 :- $U_1 = A \to a, C \to b$.

$V_1 = \{A, C\}$. are productions with terminals on R.H.S.

$W_2 = \{U_1\} \cup \{A_1 \mid A_1 \to \alpha$ with $\alpha \in (\Sigma \cup V(A,C)\}$

$= \{A, C\} \cup \{S\}$ $\quad - \quad S \to CA$

$U_3 = \{A, C, S\} \cup \{A_1 \mid A_1 \to \alpha$ with $\alpha \in (\Sigma \cup V(S, A, C)\}\}$

$= \{A, C, S\} \cup \phi$.

$W_2 = U_3$.

$V_N' = \{S, A, C\}$

$P' = S \to CA$

$A \to a$

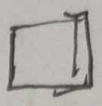
$C \to b$

(2) $V_1 = \{S\}$

$V_2 = \{S\} \cup \{A, C\}$

$V_3 = \{S, A, C\} \cup \{a, b\}$

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



in, $V_3 = V_N' \cup \Sigma$, $P'' = \{S \to a \mid A_1 \in V_3\} = P'$

$G' = \{S, A, C\}, \{a, b\}, \{S \to CA, A \to a, C \to b\}, S)$

Construct Reduad grammar equivalent to grammar

$S \to aAa$

$A \to Sb \mid bCC \mid DA$, $C \to abb \mid DD$, $E \to aC$, $D \to aDA$

Soln Step1 $V_1 = \{C\}$ $C \to abb$ is only production with terminal string on R.H.S.

$V_2 = \{C\} \cup \{A_1 \mid A_1 \to \alpha$ with $\alpha \in (\Sigma \cup \{A, C\})^*\}$

$\{C\} \cup \{A, E\}$.

$V_3 = \{A, E, C\} \cup \{S\}$ $\to$ $S \to aAa$.

$V_4 = V_3$.

$V_N' = \{S, A, C, E\}$

$P' =$

$S \to aAa$

$A \to bCC \mid Sb$

$C \to abb$

$E \to aC$

$V_4 = \{S, A, C, a, b\} \cup \{a, b\}$.

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

$= V_3$.

$P'' = \{ S \to aAa$

$A \to Sb \mid bCC$

$C \to abb \}$

---

(2) (2) $V_1 = \{S\}$

$V_2 = \{S\} \cup \{A, a\}$.

$V_3 = \{S, A, a\} \cup \{b, C\}$

$= \{S, A, C, a, b\}$ ✓

Q $S \to aS \mid AB$

$A \to \wedge$

$B \to \wedge$

$D \to b.$

Soln

Step 1 Constn of set W of all nullable variables

$W_1 = \{A, B\}$

$W_2 = \{A, B\} \cup \{S\}$ as $S \to AB$ is a production with $AB \in W_i^*$

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

$W_3 = W_2 \cup \phi = W_2$

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

Step 2 Constn of P'

$D \to b$

$S \to aS$ → $S \to aS$ and $S \to a$.

$S \to AB$ $S \to A$ & $S \to B$.

$G_1 = (\{S, A, B, D\}, \{a, b\}, P, S)$

$D \to b$

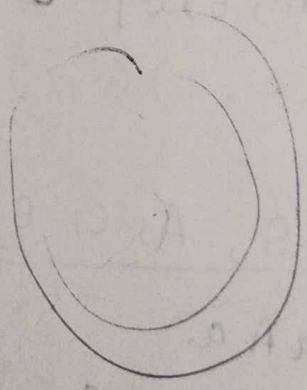
$S \to aS$

$S \to AB.$

$S \to a$

$S \to A$

$S \to B.$



Normal forms for context-free grammar

$A \to \alpha$.

where A is a variable & α is a string of symbols from $(V \cup T)^*$.

Left hand side → one variable

R.H.S any no. of variables or terminals

But if we impose restriction on right hand side of production rule, then context free grammar is said to be in "normal form".

(1) Chomsky Normal form (CNF)

(2) Greibach Normal form (GNF)

$A \to BC$
$A \to a$

(1) Chomsky Normal Form → restrictions on the length of right hand side and type of symbols in R.H.S. of production.

A context-free grammar G is in Chomsky Normal form if every production is of form.

$A \to BC$ or $A \to a$ & $S \to \wedge$ is in G if $\wedge \in L(G)$. ie S does not appear on right hand side of any productions.

$S \to AB \mid \wedge$
$A \to a$
$B \to b$.

eg. $S \to AB \mid \wedge$
$A \to a$
$B \to b$.

Then G is in Chomsky Normal form.

eg $S \to AB$
$S \to BC$
$A \to a$
$B \to b$
$C \to c$.

but $S \to ABC$
$A \to CD$
$C \to EFGH$.
$A \to a$
$B \to b$

Reduction To chomsky Normal form

Step1 Elimination of Null productions & unit productions → $G_1 = (V, T, P, S)$

Step2 Elimination of terminals on R.H.S.

In this step we construct the grammar G_2 equivalent to G_1.

$$G_2 = (V_1, T, P_1, S)$$

V_1 & P_1 are constructed as follows:-

(a) Every production in P_1 should be of form $A \to a$ or $A \to BC$ & all variables in V should be in V_N

(b) Consider $A \to X_1 X_2 \dots X_n$ with some terminal on R.H.S. If X_i is a terminal say a_i add a new Variable C_{a_i} to V_1 and $C_{a_i} \to a_i$ to P_1

eg. $A \to abc$.

$X_a \to a$

$X_b \to b$

$X_c \to c$

$A \to X_a X_b X_c$

Step3 Limiting the number of variables on right hand side of productions.

$A \to a$ right hand side contain only one terminal or right hand side contain two or more variables.

All productions in G_2 are of $A \to a$ or $A \to BC$ are in required form ie in CNF.

If we have productions in G_2 of form:-
$A \to B_1 B_2 \ldots B_n$ where $n > 2$ then we introduce new production equivalent to $A \to B_1 B_2 \ldots B_n$ with two variables on right hand side.

Let $n = 3$

$A \to B_1 B_2 B_3$

$A \to B_1 Y_1$

$Y_1 = B_2 B_3$

Example Convert the following CFG to CNF :⇒

$S \to 0AADD$ ✓

$A \to 0BB$ ✓

$A \to 1AB$ ✓

$B \to 1$ ✓

$D \to 1$ ✓

Soln. 1) Elimination of Null production

2) Elimination of unit production

(3) Elimination of terminals on R.H.S.

$S \to C_0 AADD$

$A \to C_0 BB$

$A \to C_1 AB$

$B \to 1$

$D \to 1$

where. $C_0 \to 0$

$C_1 \to 1$.

$S \to C_0 AADD$

$S \to C_0 A_1 D_1$

$S \to C_0 D_2$ ✓

$A_1 \to AA$

$D_1 \to DD$

$D_2 \leftarrow A_1 D_1$

$A \to C_0 BB$.

$A \to C_0 C_b$ ✓

$C_b \leftarrow BB$

(4) Restricting the no. of variables on R.H.S.

$S \to C_0 A_1 D_1$

$A_1 = AA$

$D_1 = DD$

$D_2 \to A_1 D_1$ ✓

$S \to C_0 D_2$ ✓

$A \to C_0 C_b$ ✓

$A \to C_1 C_c$ ✓

$C_b \to BB$

$C_c \to AB$

$A \to C_1 AB$

$A \to C_1 C_3$ ✓

$C_3 \leftarrow AB$

$B \to 1$ ✓

$D \to 1$ ✓

$C_0 \to 0$ ✓

$C_1 \to 1$ ✓

(1) $S \to *S$ ✓
$S \to (S*S)$
$S \to 1$
$S \to 2$

Soln :- (1) Elimination of Null productions
(2) Elimination of unit productions
(3) Elimination of terminals on R.H.S.

$C \to *$
$S \to CS$

$S \to CS$
$S \to C_1 S C S C_2$
$S \to 1$
$S \to 2$

$C \to *$
$C_1 \to ($
$C_2 \to)$

(4) Restricting the no. of variables on R.H.S

$S \to CS$
$S \to C_1 S C S C_2$
~~$C_3 \to C_1 C_2$~~
$S \to C_1 C_3$
$S \to C_1 S C S C_2$

$\Rightarrow$

$S \to CS$
$S \to C_1 C_3$
$C_3 \to SC S C_2$
$C_4 \to C S C_2$
$C_3 \to S C_4$
$C_4 \to C S C_5$ ~~$C C_5$~~
$C_5 \to S C_2$

---

(1) $S \to aAD$
$A \to aB \mid bAB$
$B \to b$
$D \to d$

Soln
$B \to b$
$D \to d$

$S \to C_a A D$
$A \to C_a B$
$A \to C_b AB$

$C_a \to a$
$C_b \to b$

$A \to C_a B$, $B \to b$, $D \to d$, $C_a \to a$, $C_b \to b$

$S \to C_a AD$
$S \to C_a C_1$

$C_1 \to AD$
$A \to C_b C_2$
$C_2 \to AB$

Q. find a grammar in CNF equivalent to grammar $L \to b$.

$S \to \sim S \mid [S \supset S] \mid p \mid q$ (S being only variable)

Soln (1) No unit or null productions

(2) $S \to p \mid q$

$S \to \sim S$ induces $S \to AS$ and $A \to \sim$

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

$B \to [$

$C \to \supset$

$D \to]$

$S \to BC_1$

$C_1 \to SC_2$

$S \to BC_1$

$C_1 \to SC_2$

$C_2 \to CS_2$

$C_1 \to SCSD$

$C_2 \to CSD$

$S_2 \to SD$

Discrete Mathematics

TOC

8:30 to 8:30

TOC

4:30

TOC

43

Java

Greibach Normal form. (Sheila Greibach)

In Chomsky normal form every production must be of form $A \to a$ or $A \to BC$ where A, B & C are variables and a is Terminal.

In Greibach normal form there is restriction on position in which terminals & variables can appear on right hand side of production rules.

Every production in GNF must start with a single terminal followed by Variables. A production $A \to \alpha B Y$ can be eliminated from grammar if we replace this production with new set of productions.

A CFG is in Greibach normal form if every production is of form

$A \to a\alpha$ where $\alpha \in V_N^*$ and $a \in \Sigma$ (α may be $\wedge$)

eg.

$S \to aAB \mid \wedge$
$A \to bG$
$B \to b$
$C \to c$ is in GNF.

$A \to abCDE$
$C \to bbAb$
$D \to aa$
$E \to ab$
where $A, C, D, E \in V$
$a, b \in T$.
are not in GNF

Substitution Rule

$A \to \alpha B \gamma$
$B \to \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$

$A \to \alpha\beta_1\gamma \mid \alpha\beta_2\gamma \mid \alpha\beta_n\gamma$

Left Recursion

A grammar G is said to be left recursive if there is a variable A whose productions are of form

$A \to Ax$ for any $x \in (V \cup T)^*$.

We can eliminate left recursion from given grammar G.

Suppose we want to eliminate left recursion from β_1 following A-productions

$A \to A\beta_1 \mid A\beta_2 \mid \dots \mid A\beta_m \mid \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$

$A \to \alpha_1 Z \mid \alpha_2 Z \mid \dots \mid \alpha_n Z$

$Z \to \beta_1 Z \mid \beta_2 Z \mid \dots \mid \beta_m Z$ $Z \to \wedge$.

Reduction to Greibach Normal form. CFG to GNF

Let $G = (V, T, P, S)$ be given grammar. To find new grammar in Greibach Normal form, we apply following steps

Step1. (a) given grammar must be in chomsky Normal form, if not in CNF then convert it into CNF.

(b) Now rename the variables in grammar G (which are in CNF).

$G = (\{A_1, A_2 \dots A_n\}, T, P, A_1)$

where $S = A_1$ is start symbol.

$\{A_1, A_2 \dots A_n\}$ is set of variable after renaming.

Step2. Make the production in G of form

$A_i \to a\gamma$ or

$A_i = A_j\gamma$ where $j > i$.

If productions are not satisfying this cond^n, then we will modify the grammar in following manner

If the production is of form

$A_1 \to a A_2 A_3$

$A_2 \to A_3 A_1 \mid b$

$A_3 \to A_2 A_1 \mid a$

$i=3, j=2 \quad (j<i)$ ✓

$A_3 \to A_3 A_1 A_1 \mid bA_1 \mid a$.

Step3 After step 2, we get productions of form

$A_i \to a\gamma$ or

$A_i \to A_j\gamma$ when $j > i$

If we have productions

$A_3 \rightarrow A_3 A_1 A_1 \mid b A_1 \mid a$

$A_3 \rightarrow bA_1 \mid a$

$A_3 \rightarrow bA_1 Z \mid aZ$

$Z \rightarrow A_1 A_1$

$Z \rightarrow A_1 A_1 Z$

Step 4.

$A_1 \rightarrow a A_2 A_3$

$A_2 \rightarrow A_3 A_1 \mid b$ ✗

$A_3 \rightarrow bA_1 \mid a \mid bA_1 Z \mid aZ.$

$Z \rightarrow A_1 A_1 \mid A_1 A_1 Z$

$A_2 \rightarrow bA_1 A_1 \mid aA_1 \mid bA_1 Z A_1 \mid aZ A_1 \mid b.$

The resulting productions are in req. form

Step 5:-

$A_1 \rightarrow a A_2 A_3$

$A_2 \rightarrow bA_1 A_1 \mid aA_1 \mid bA_1 Z A_1 \mid aZ A_1 \mid b.$

$A_3 \rightarrow bA_1 \mid a \mid bA_1 Z \mid aZ$

$Z \rightarrow A_1 A_1 \mid A_1 A_1 Z.$

Modify Z- productions

$Z \rightarrow aA_2 A_3 A_1 \mid aA_2 A_3 A_1 Z.$

Q Construct a grammar in Greibach normal form equivalent to grammar

$S \to AA \mid a$

$A \to SS \mid b$

10 marks.

Soln The given grammar is in CNF.

$A_1 \to A_2 A_2 \mid a$ ✓ $S \to A_1$

$A_2 \to A_1 A_1 \mid b$ $A \to A_2$

Step 2 :- (i) $A_1 \to A_2 A_2 \mid a$.

$A_2 \to A_2 A_2 A_1 \mid a A_1 \mid b$. ✓

j > i.

Step 3 To remove left recursion from A_2 production

$A_2 \to A_2 A_2 A_1 \mid a A_1 \mid b$.

α β_1 β_2

~~$A_2 \to a A_1 Z \mid b Z$~~ $A_2 \to a A_1 \mid b \mid a A_1 Z \mid bZ$ ~~i > j~~ i = j

~~$A_2 \to$~~

$Z \to A_2 A_1 \mid A_2 A_1 Z$.

Step 4. $A_2 \to a A_1 \mid b \mid a A_1 Z \mid bZ$. ✓

$A_1 \to A_2 A_2 \mid a$

$A_1 \to a A_1 A_2 \mid b A_2 \mid a A_1 Z A_2 \mid b Z A_2 \mid a$. ✓

$A_2 \to a A_1 \mid b \mid a A_1 Z \mid bZ$.

$Z \to A_2 A_1 \mid A_2 A_1 Z$

Step 5 $Z \to a A_1 A_1 \mid b A_1 \mid a A_1 Z A_1 \mid bZ A_1$ ~~$\mid A_2$~~

$Z \to a A_1 A_1 Z \mid b A_1 Z \mid a A_1 Z A_1 Z \mid bZ A_1 Z$

Q2 $S \to XX$, $S \to a$ $X \to SS$, $X \to b$

$S \to A_1$

$X \to A_2$

Sol. Step1 Rename variables S and X.

$A_1 \to A_2 A_2$

$A_1 \to a$

$A_2 \to A_1 A_1$

$A_2 \to b$.

Step2 :- $A_1 \to A_2 A_2 \mid a$ (A_1's productions are in required form, they are $A_i \to A_j \gamma$ where $i < j$

$A_i \to \alpha \gamma$ where $\gamma = \Lambda$

The production

$A_2 \to A_1 A_1$ is not in required form.

$A_2 \to A_2 A_2 A_1 \mid a A_1 \mid b$

Step 3 $A_2 \to a A_1 \mid b \mid a A_1 Z \mid b Z$

$Z \to A_2 A_1 \mid A_2 A_1 Z$

Step.4 $A_1 \to A_2 A_2 \mid a$

$A_1 \to a A_1 A_2 \mid b A_2 \mid a A_1 Z A_2 \mid b Z A_2 \mid a$

$A_2 \to a A_1 \mid b \mid a A_1 Z \mid b Z$

$Z \to a A_1 A_1 \mid b A_1 \mid a A_1 Z A_1 \mid b Z A_1$

$Z \to a A_1 A_1 Z \mid b A_1 Z \mid a A_1 Z A_1 Z \mid b Z A_1 Z$.

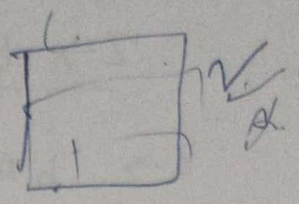
Q Convert this

$S \to AB,\quad A \to BS \mid b,\quad B \to SA \mid a$ into GNF.

Soln $S \Longrightarrow$ $A_1 \to S$

$A_2 \to A$

$A_3 \to B$



(1) $A_1 \to A_2 A_3$ ✓

$A_2 \to A_3 A_1 \mid b$ ✓

$A_3 \to A_1 A_2 \mid a$

$A \to a\alpha.$

$\alpha \in V_N^*$

$a \in \Sigma$ (α may be $\wedge$)

Step2: (i) The production $A_1 \to A_2 A_3$ is in required form.

(ii) $A_2 \to A_3 A_1 \mid b$ are in required form

(iii) $A_3 \to a$ is in required form

$A_3 \to A_1 A_2 \quad i > j$

$A_3 \to A_2 A_3 A_2$, Applying lemma once again

$A_3 \to A_3 A_1 A_3 A_2 \mid b A_3 A_2.$

Step3 $A_3 \to A_3 A_1 A_3 A_2 \mid b A_3 A_2 \mid a.$

$A_3 \to a \mid bA_3A_2 \mid aZ \mid bA_3A_2Z.$

$Z \to A_1 A_3 A_2 \mid A_1 A_3 A_2 Z$

Step4 (i) $A_3 \to a \mid b A_3 A_2 \mid aZ \mid bA_3A_2Z.$

(ii) $A_2 \to b.$ $\quad A_2 \to A_3 A_1$

$A_2 \to b \mid aA_1 \mid bA_3A_2A_1 \mid aZA_1 \mid bA_3A_2ZA_1$

(iii) $A_1 \to A_2 A_3$

$A_1 \to bA_3 \mid aA_1A_3 \mid bA_3A_2A_1A_3 \mid aZA_1A_3 \mid bA_3A_2ZA_1A_3.$

Steps $Z \to A_1A_3A_2 \mid A_1A_3A_2Z$.

~~$Z \to bA_3A_3A_2 \mid aA_1A_3A_3A_2$.~~

$Z \to bA_3A_3A_2 \mid bA_3A_3A_2Z$.

$Z \to aA_1A_3A_3A_2 \mid aA_1A_3A_3A_2Z$.

$Z \to bA_3A_2A_1A_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2Z$

$Z \to aZ_3A_1A_3A_3A_2 \mid aZA_1A_3A_3A_2Z$

$Z \to bA_3A_2Z_3A_1A_3A_3A_2 \mid bA_3A_2Z_3A_1A_3A_3A_2Z$

Q $E \to E+T \mid T$

$T \to T*F \mid F$

$F \to (E) \mid a$

Soln Step1 we first eliminate unit prodn.

$E \to E+T$

$T \to T*F$ $\{E, T, F\}$ $F \to (E)$

$F \to a$.

E derivable $E \to \{E T_3 F\}$.

$T \to \{T, F\}$

$F \to \{F\}$.

For. $(E,T) \to (a), a$ (i) $E \to E+T \mid T*F \mid$

$E \to (E), a$ (ii) $T \to T*F \mid (E) \mid a$

$(E,F) \to a$ (iii) $F \to (E) \mid a$

(1) $E \to E+T \mid T*F \mid (E) \mid a$

$T \to T*F \mid (E) \mid a$

$F \to (E) \mid a$

$A \to +$

$B \to *$

$C \to)$

(A, B) $\boxed{A \to B}$

(i) $E \to EAT \mid TBF \mid (EC \mid a$

(ii) $T \to TBF \mid (EC \mid a$

(iii) $F \to (EC \mid a$

A

A, B, C, F, T, E

$A_1, A_2, A_3, A_4, A_5, A_6.$

Q Convert the following to GNF

Soln $S \to abSb \mid aa$

$S \to aBSB$

$S \to aA$

$A \to a$

$B \to b$

$A \to a$

$B \to b$

Q

$S \to ASB$

$S \to AB$

$A \to a$

$B \to b$

$S \to A_1$

$A \to$

$A_1 \to S$

$A_2 \to A$

$A_3 \to B.$

(1) $A_1 \to A_2 A_1 A_3$

$A_1 \to A_2 A_3$

$A_2 \to a$

$A_3 \to b.$

(2) The rules are already in required form

$A_i \to A_j Y$ where $j > i$

$A_1 \to A_2 A_1 A_3 \mid A_2 A_3$

$A_2 \to a$ ✓

$A_3 \to b$ ✓

$A_1 \to aA_1A_3 \mid aA_3$ $\quad A_2 \to a$

$A_3 \to b$

Algorithm

$A \to a$

$A \to a\alpha \qquad \alpha \in (V)^*$

1. Create a new symbol and new prodn $S' \to S$
2. Step Remove Null production & unit prodn.
3. Remove all direct & indirect left recursion.
4. Do proper substitution of prodn. to convert it into proper form of GNF

Rough Work

$0^n 1^n 2^n$

$x\ y\ z$

$n = n = n$

$z \in L$.

$|z| \geq n$

w ≥ p

$w = uvxyz$

$vy \neq \varepsilon$

$|vxy| \leq p$.

$uvwxy$ $\quad |vx| \geq 1$

$|vwx| \leq n$

$uv^k wx^k y \in L$ KTP

Pumping Lemma for context free grammars

Step1 Assume L is a context-free

Let n be natural number obtained by using pumping lemma

Step2 Choose $z \in L$ so that $|z| \geq n$.

Write $z = uvwxy$ using pumping lemma

Step3 Find suitable k so that $uv^k wx^k y \notin L$. This contradiction & L is not context-free language.

Push Down Automata.

We need to construct a m/c whose computing power is higher than finite automata. Push Down Automata recognize the language called context free language.

We use PDA because of weakness of finite automata.

eg. $\{0^n 1^n \mid n \geq 0\}$ which is not regular. equal no. of 0's and 1's and all 0's are followed by 1's ie 0011. To accept any string $w \in L$, It has to remember how many 0's it has read before reading 1's. To remember no. of 0's in input string, finite automata need some auxiliary memory when it can store strings of infinite length.

NPDA ✓
DPDA ✓
CFL & PDA ✓
Comp NDFA PDA.

PDA = FA + Auxiliary Memory (Stack).

Higher computational power than finite automata because it can do everything that can be done by finite automata & can also do some extra that can't be done by finite automata.

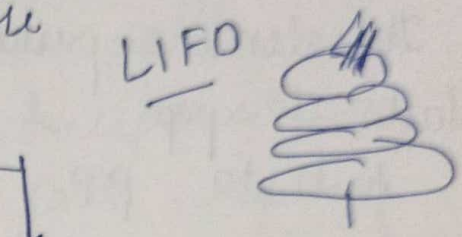
Closure prop

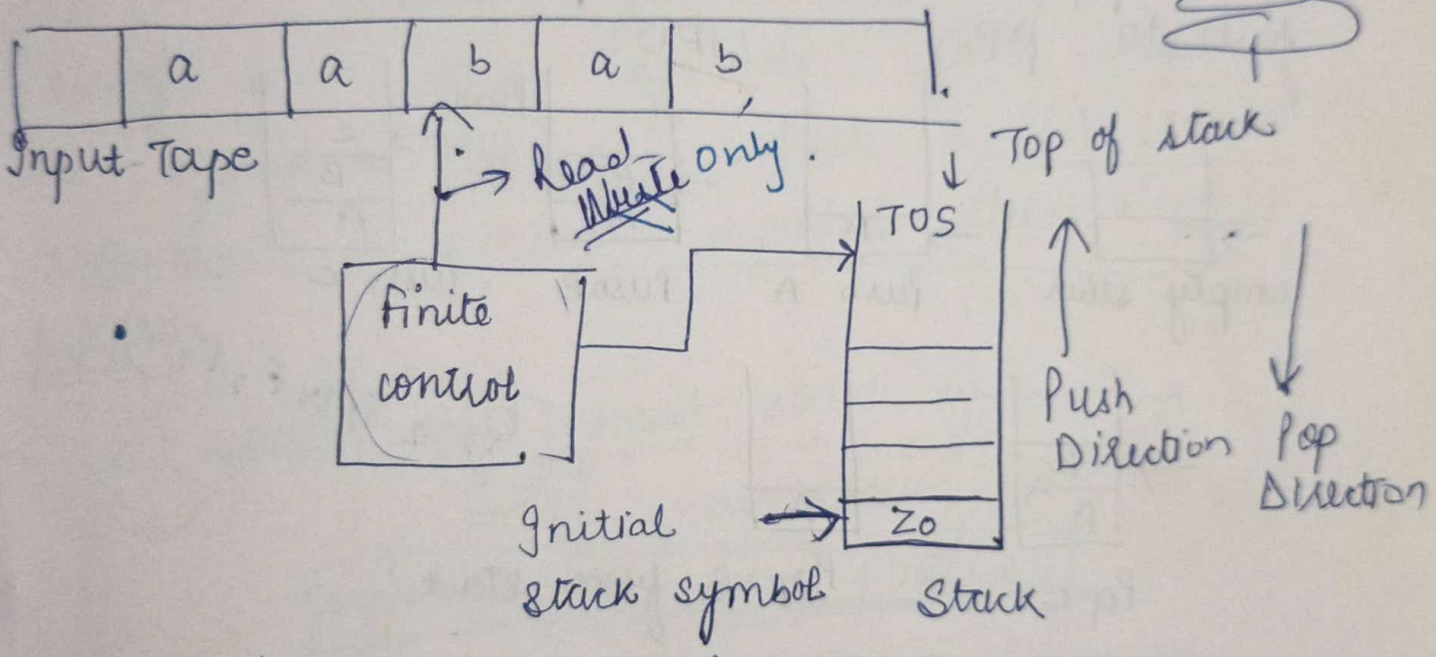
Push down automata as an automata coupled with a stack that can be used to store string of arbitrary length

1. Read only input tape ✓
2. Finite control ✓
3. Input alphabet ✓
4. Set of final states ✓
5. Initial state

) Stack (PDS) Push Down Store

LIFO





(Model of PDA)

* The input tape is read only. ie the head can read the symbols from tape but can't write on it. The stack of PDA allows read and write operations ie we can insert elements into stack or delete elements from stack.

Stack is the memory where insertion and deletion of data can be done only at the top of stack. PDA choose next move based on

(1) current state (Q)
(2) the input symbol (Σ)
(3) Symbol at top of its stack

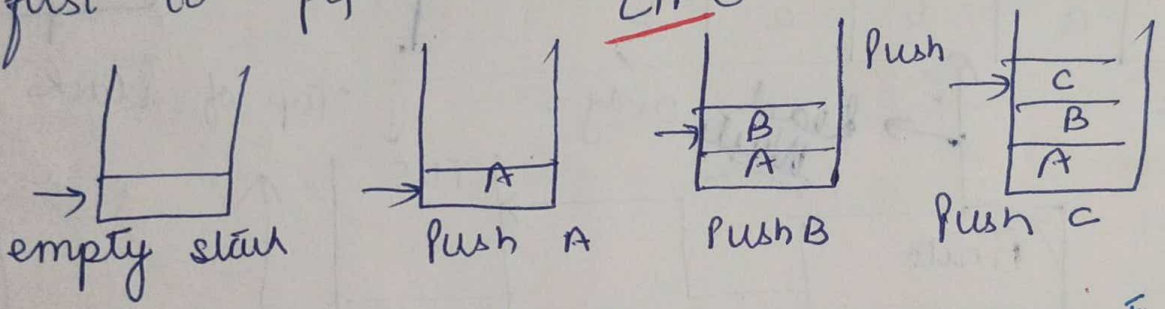
Operations on Stack

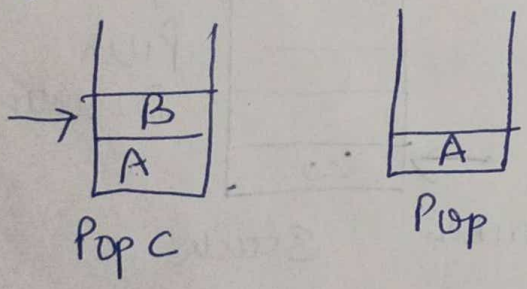
Push open → Insertion in stack

Pop → deletion from stack

The element pushed first into stack will be last to pop & element pushed last will be first to pop.

LIFO.





$(Q, \Sigma, \Gamma, \delta, q_0, F, \Gamma, Z_0)$

C is LIFO

Formal Defn of PDA

PDA can be defined as consisting of seven components or seven tuples.

$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F).$$

- Σ: Stack alphabet
- q_0: Initial state, $q_0 \in Q$.
- Z_0: Stack start symbol

F: set of final states $F \subseteq Q$.

δ: transition function.

javac -d .

javac -

$\delta(q, b, x)$ where

q is a state in Q

b is either an input symbol or $b = \wedge$

x is stack symbol (symbol on top of stack)

$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to$ finite subset of $Q \times \Gamma^*$

$Q \times \Sigma \times \Gamma \to Q \times \Gamma^*$

The output of δ contains two arguments. The first argument corresponds to new state & second argument corresponds to string of stack symbols that replaces the symbol at top of stack. If second argument is empty symbol then stack is popped.

δ(state, input symbol, stack symbol) →

(new state, $\wedge$)

symbol will be popped.

(ii) If the second argument contains same symbol as third symbol of δ then stack is unchanged.

δ(state, input symbol, stack symbol)

→ (new state, same stack symbol)

Stack is unchanged.

(iii) If the second argument contains other symbol that will be push on stack.

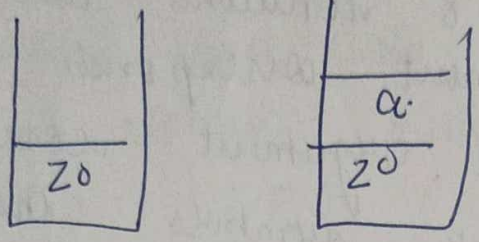
δ (state, input symbol, stack symbol) → (new state, other stack symbol)

(symbol will be pushed)

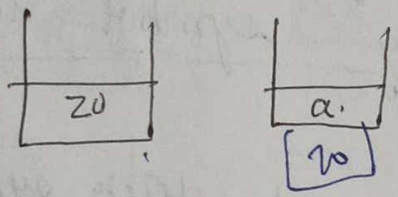
PDA moves

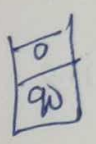
$$(Q \times (\Sigma^*) \times \Gamma) \to (Q \times \Gamma^*)$$

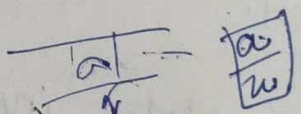
I. If $\delta(q_0, a, Z_0) = (q_1, aZ_0)$



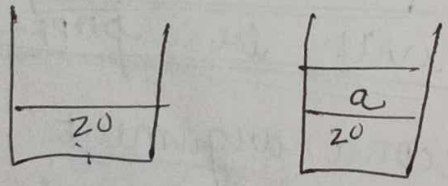
II $\delta(q_0, a, Z_0) = (q_1, a)$







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

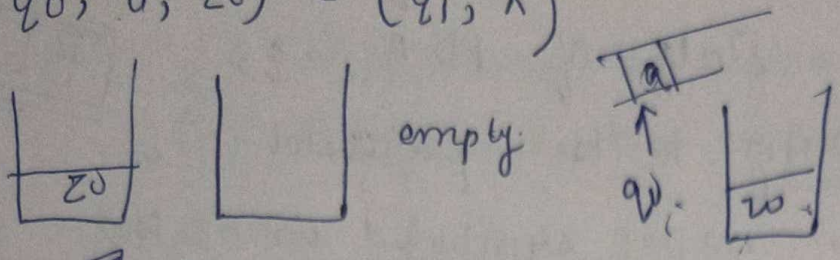


(IV) $\delta(q_0, a, Z_0) = (q_1, \varepsilon)$

PDA is in state q_0, after reading input symbol a and top of stack contains symbol Z_0 i.e. initial stack symbol, the PDA enters in new state q_1 and pop symbols from stack

After this transition, the stack becomes empty

$\delta(q_0, a, z_0) = (q_1, \wedge)$

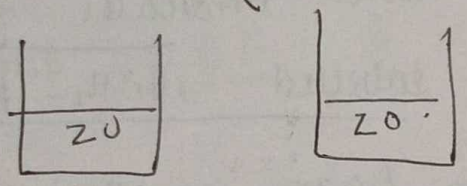


(v) $\delta(q_0, a, z_0) = (q_0, z_0)$

PDA is in state q_0, after reading input symbol a and the top of stack contains the symbols z_0 i.e initial stack symbol, the PDA remains on same state q_0 and make no change in contents of stack.

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

$\delta: Q \times [\Sigma \cup \epsilon] \times \Gamma \to Q \times \Gamma^*$



Sentential form

ID (Instantaneous Description) PDA can be represented with the help of instantaneous description abbreviated as ID. The moves in PDA can be represented with the help of instantaneous description or we can say configuration of PDA at given instant is called (ID).

ID

(state of m/c, input symbols to be read, symbols in stack) $\longrightarrow$

(Q, Σ^*, Γ^*)

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

ID for this (q, a, z_0) where $q \in Q$, $a \in \Sigma^*$, $z_0 \in \Gamma^*$.

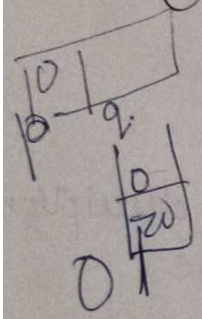
eg $(q, 001\_\_\_\_00, Z_0)$ is an ID

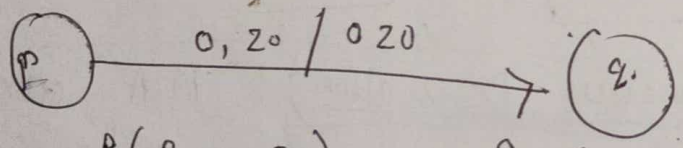
The current state of PDA is q, the input string to be processed is 001_____00, Z_0 is symbol in stack PDA scan string 001_ _00 from left to right.

Transition Diagram

PDA can also be represented with the help of Transition diagram. The transition diagram for PDA contains the following:-

1. States of PDA represented with circle.
2. Arrow to circle indicates final state.
3. Each arc is labelled with tripple





$\delta(P, 0, Z_0) = \{(q, 0Z_0)\}$.

$\delta(q_0, 0, Z_0) = (q_0, 0Z_0)$

$\delta(q_1, 1, 0) = (q_1, \wedge)$

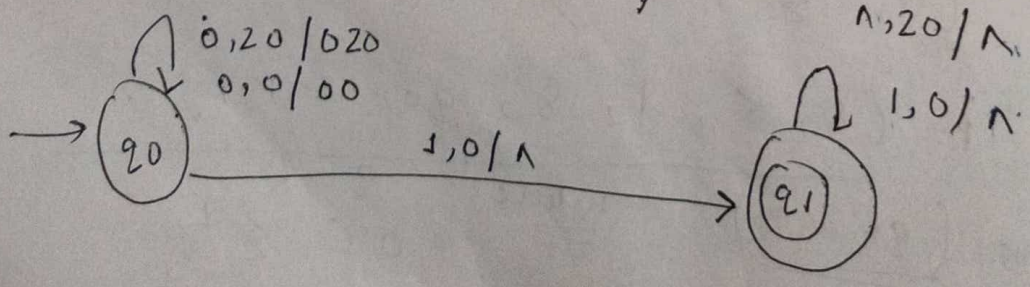
$\delta(q_0, 0, 0) = (q_0, 00)$

$\delta(q_1, \wedge, Z_0) = (q, \wedge)$

$\delta(q_0, 1, 0) = \{q_1, \wedge\}$

$0^n 1^n$.

$a^n b^n$



Acceptance of PDA

2 marks

1. Acceptance by final state
2. Acceptance by empty stack

(1) Acceptance by FINAL STATE

$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be PDA.

$\delta(q_0, w, Z_0) \rightarrow (q_f, \wedge, \alpha)$

$q_f \in F$

$\alpha \in \Gamma^*$.

Let $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be PDA. aabb...

Set accepted by final state

$$T(A) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q_f, \wedge, \alpha)$$

for some $q_f \in F$ & $\alpha \in \Gamma^*$.

(2) Let $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be PDA.

Set $N(A)$ accepted by null state is defined by.

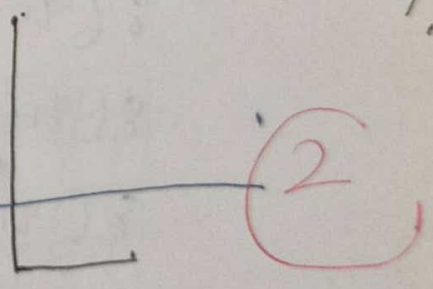
$$N(A) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \wedge, \wedge) \text{ for some } q \in Q\}$$

(using something on top of stack)

$\delta: Q \times \{\Sigma \cup \wedge\} \times \Gamma \rightarrow Q \times \Gamma^*$.

$\delta: Q \times \{\Sigma \cup \wedge\} \times \Gamma \rightarrow 2^{(Q \times \Gamma^*)}$

(more than one state).



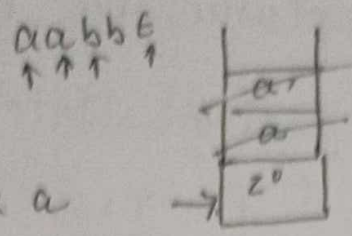
$\Sigma = \{a, b\}$ abaa

$(q_0, aabb, Z_0) \vdash$

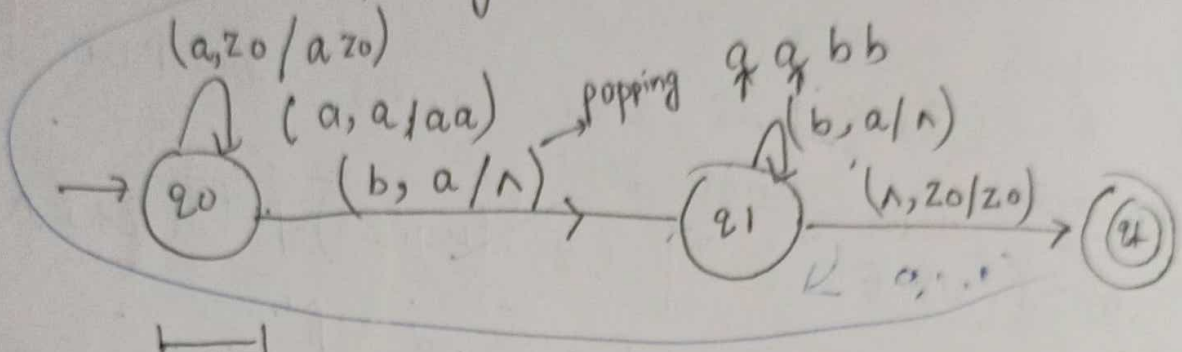
$Z_0 \Rightarrow$

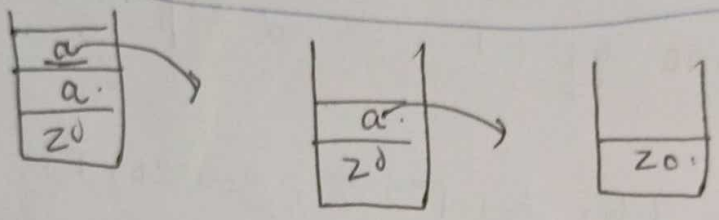
Q. Design a PDA that accept the following language

$a^n b^n \mid n \geq 1$ count a's and match a's with b's



As long as u see a keep on pushing a's and as long as u see b keep on popping a.





Transition function

q a b b

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

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

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

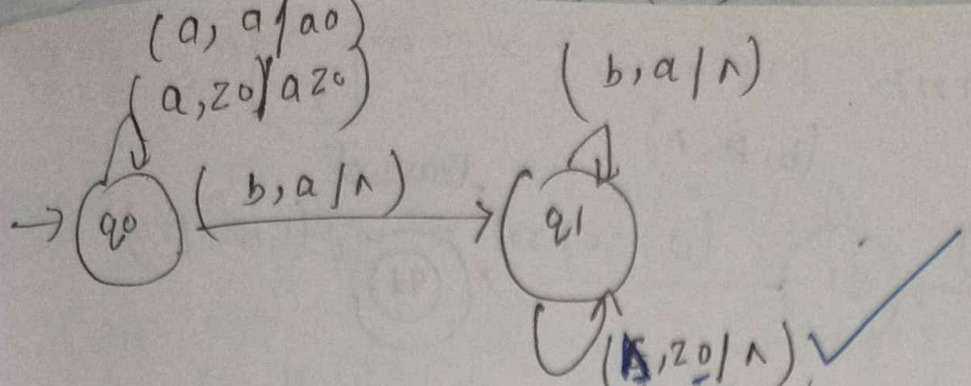
$\delta(q_1, b, a) = (q_1, \wedge)$

$\delta(q_1, \wedge, Z_0) = (q_f, Z_0)$ or

acceptance by empty

$(q_1, \wedge)$

{ PDA by final state }



baba

$n_a = n_b$

abab

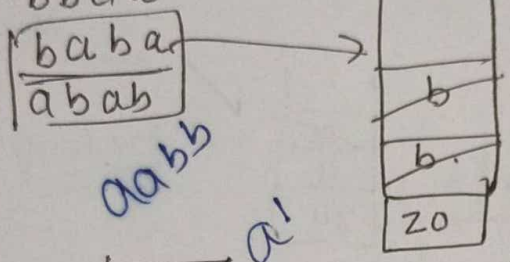
$a^n b^n$ ≠.

a_1 (q_1, ϵ)

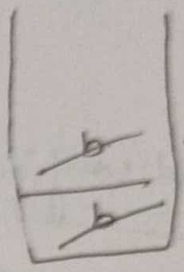
L(a): Set of all strings such that $n_a(w) = n_b(w)$

ab bbaa

aabb



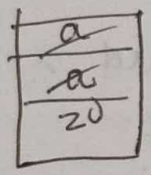
① baba



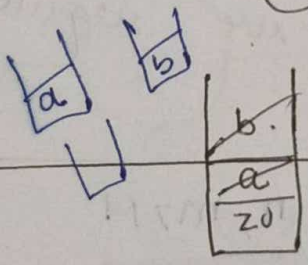
aabb

a'

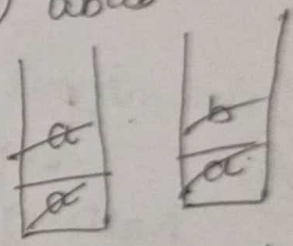
② abab



③ abba

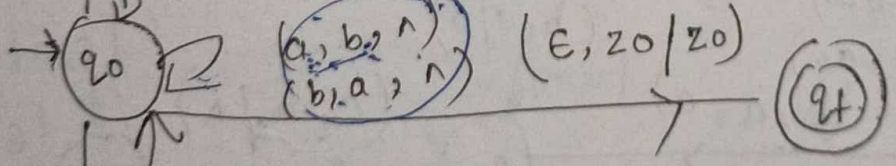


② abab



(a, z0/az0)

(b, z0/bz0)

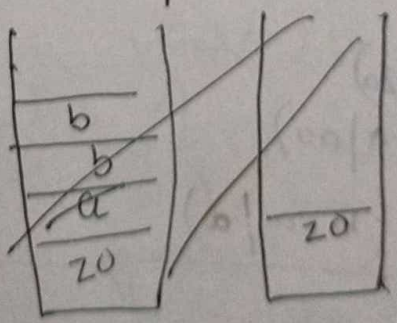


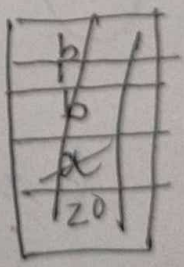
abbbaa

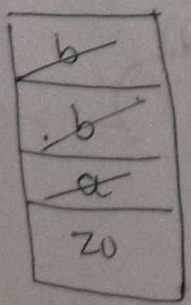
(a, a/aa)

(b, b/bb)

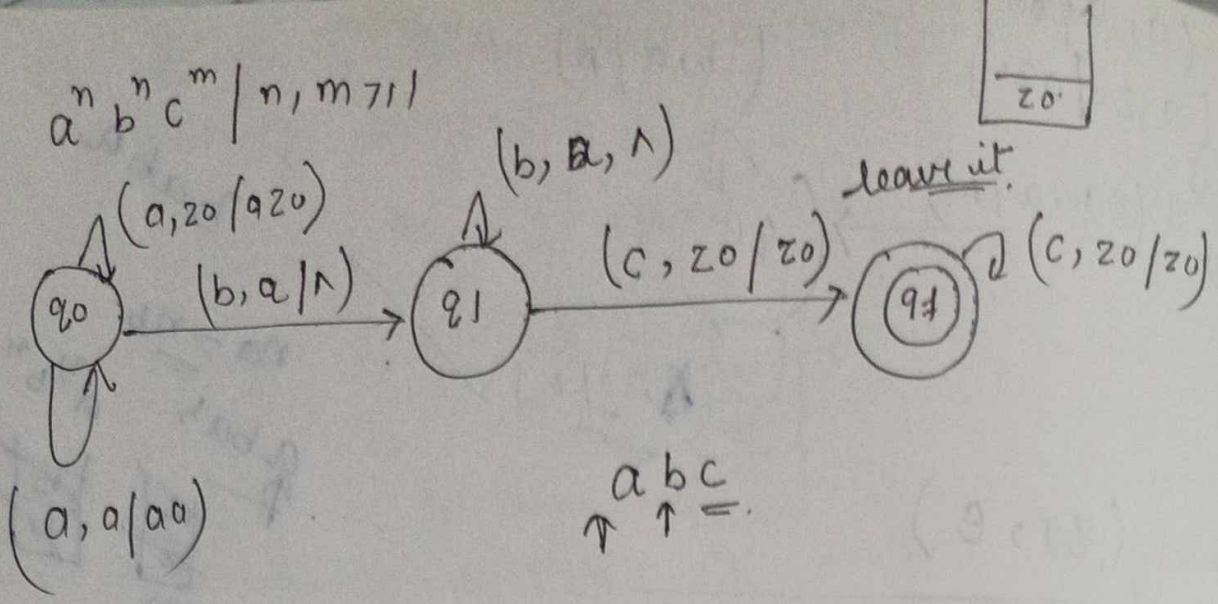
ab bbbaa







Q3. $a^n b^n c^m \mid n, m \geq 1$

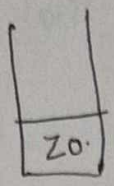


a b c

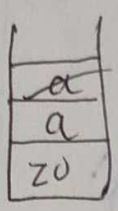
↑ ↑ =

b c

a a b c c

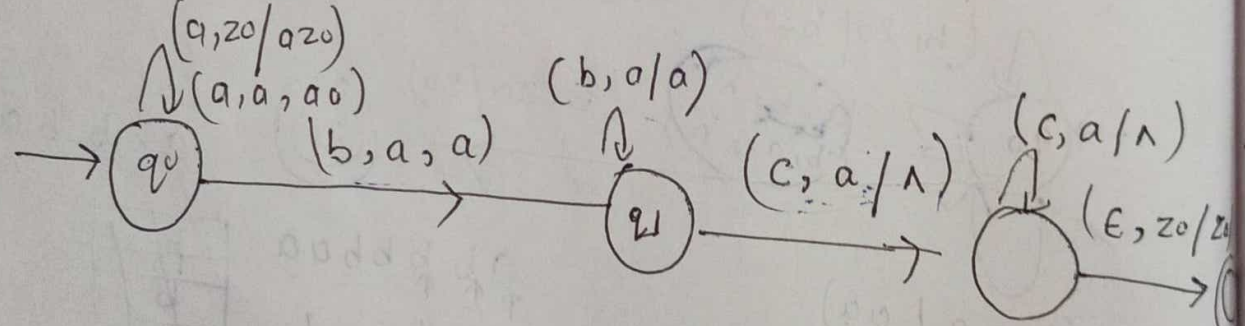


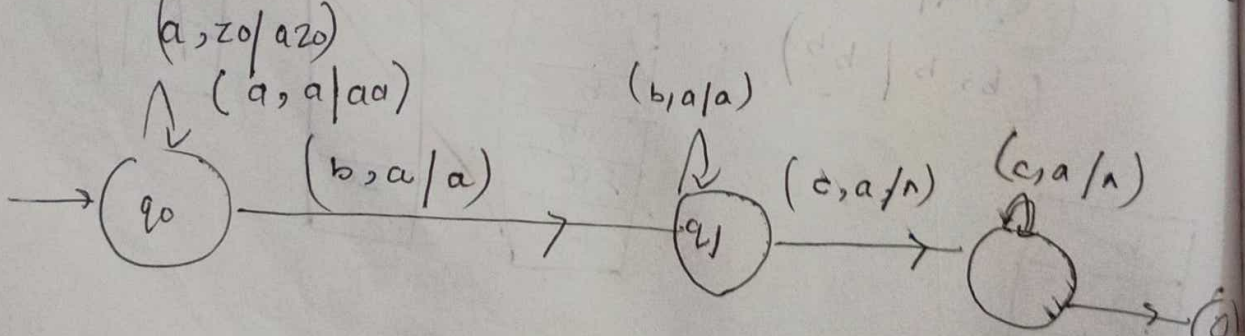
Read configuration



Input is not in seq. which we required so, it is halting.

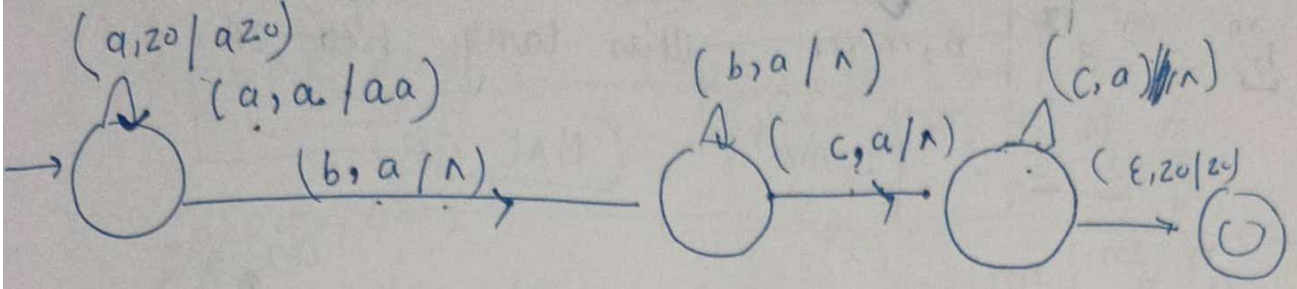
Q4 $a^n b^m c^n \mid n, m \geq 1$





Q $a^{m+n}\ b^m\ c^n \mid m,n \geq 1$

Soln: $a^m\ a^n\ b^m\ c^n \mid m,n \geq 1$

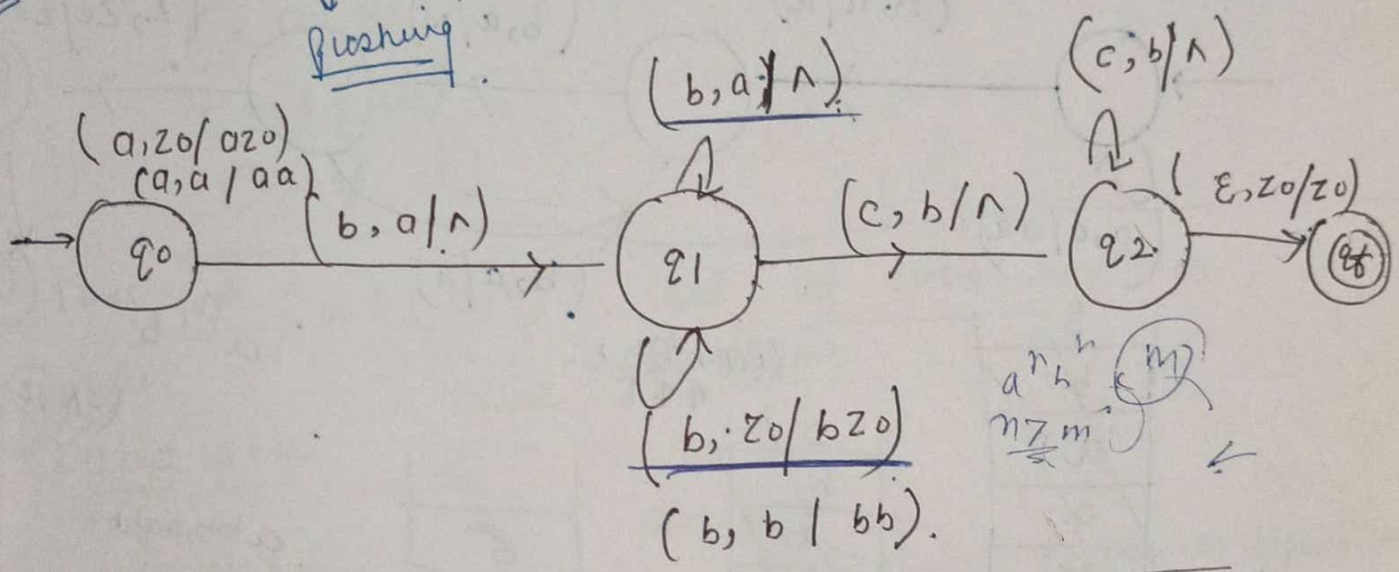


$a^n\ b^{m+n}\ c^m \mid n,m \geq 1$ Important

$a^n\ b^n\ b^m\ c^m \mid n,m \geq 1$

Soln

Pushing

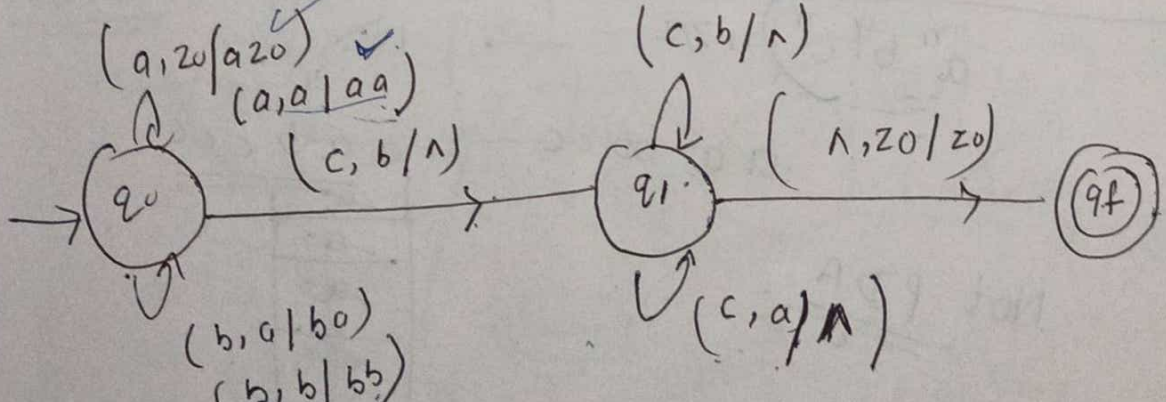


$a^n\ b^m\ c^{n+m} \mid n,m \geq 1$

$a^n\ b^m\ c^m\ c^n$

$a^n\ \underline{b}^m\ \underline{c}^n\ c^m$

aabbba



$a^m b^n c^m d^m$

$a^n b^m c^m d^n \mid m, n \geq 1$ ✓

~~$a^n b^m c^m d^m \mid n, m \geq 1$~~ this lang. not CF.

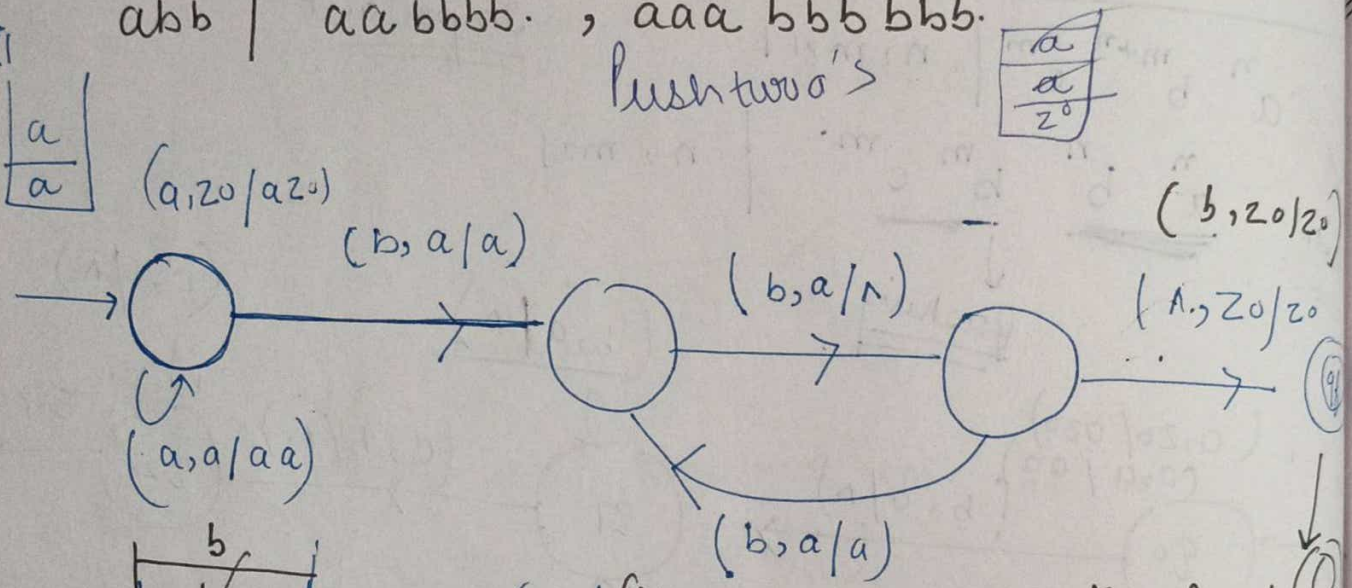
Soln $a^n b^m c^n d^m \mid n, m \geq 1$ (Not CF).

Q $a^n b^{2n} \mid n \geq 1$

$aabbbbbb$

$abb \mid aabbbb$, $aaabbbbbb$

Push two a's



abb, aabbbb

ab^2, a^2b^4 —

a^3b^6 —

Q $a^n b^n c^n \mid n \geq 1$ → NPDA

X aa bb cc

aa bbb cccc

Not PDA.

Q $wcw^R \mid w \in (a,b)^+$

Set of all palindromes

odd ✓ abcba

dbcba

abbcbba

odd palindromes

aabcbaa

(a, z0 / az0)

(b, z0 / bz0)

(a, a / aa)

(b, a / ba)

(a, b / ab)

(b, b / bb)

(c, a / a)

(c, b / b)

(b, b / ∧)

(a, a / ∧)

(∧, z0 / z0)

q0, q1, q2

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

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

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

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

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

DPDA

NDPDA

Q $ww^R \mid w \in (a,b)^+$ set of even length palindromes

ab ba

aba aba

aaaa.

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

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

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

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

(for same configuration)

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

(Designing two moves).

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

a

b

a

z0

abbcbba

a

a

NDPDA:

(a, z0 / az0)

(b, z0 / bz0)

(a, a / ∧)

(b, b / ∧)

(a, b / ab)

(b, a / ba)

(a, a / aa)

(b, b / bb)

q0, q1, q2

abba

(∧, z0 / z0)

aaaa

(q0, a, z0) →

abaaba

Centre (Not come)

ID

$(q_s, aaaa, z_0)$

$(q_s, aaa, a z_0)$

Nocenter | center

(q_s, aa, aa)

(q_1, aa, z_0) ×

No center | center

(q_s, a, aaa) ×

(q_1, a, az_0)

(q_1, ϵ, z_0)

q_2

$(q_s, \wedge, aaaa)$ ×

$(q_f, \wedge, aaz_0)$ ×

$(q_s, aaaa, z_0)$

(q_s, aaa, az_0)

(q_s, aa, aa)

(q_1, aa, z_0) ×

$(q_f, \wedge, z_0)$

$\delta : (q, a, a) = (-, -)$

δ : { — }

NDPA ∈ DPDA

q_s, a, aaa q_s, a, az_0

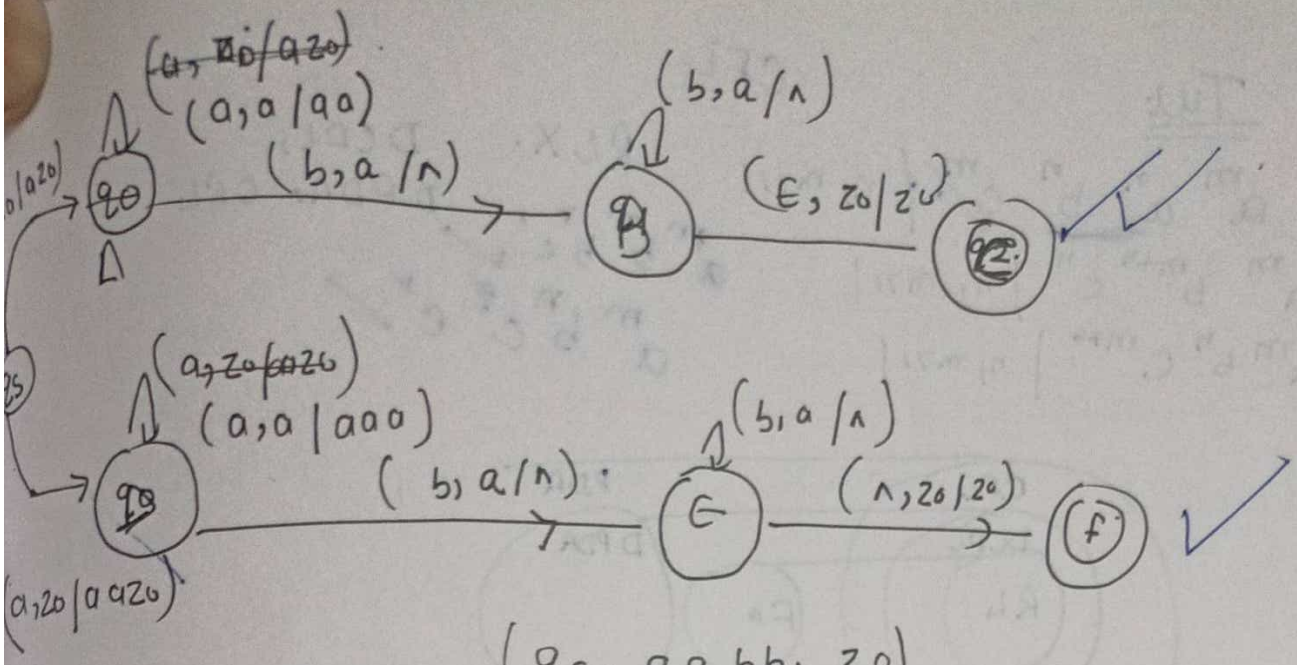
$q_s, \wedge, aaa$ $q_1, \wedge, az_0$

NDPDA

Q $L = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$.

Soln aa bb

(Not looking at a, i can't decide whether to push on a



abb

aabb ∉

(qs, aabb, Z0)

(qs

(A, abb, aZ0)

(A, bb/aaZ0)

(A, b, aZ0)

(A, ∧, Z0)

(C)

(D, abb, aaZ0)

(D, bb, aaaaZ0)

(∈, b, aaaZ0)

(∈, ∧, aaZ0)

X

□

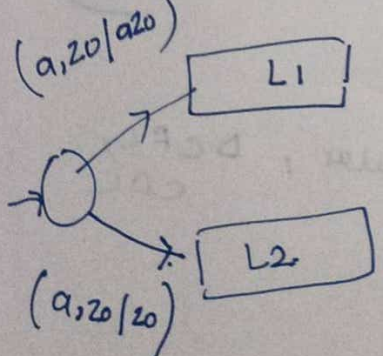
(3) $a^i b^j c^k d^l / i = k$ or $j = l$

Soln

$i=k$ $j=l$ $a^m b^n c^m d^n$.

$\{a^m b^j c^m d^l\} \cup \{a^i b^m c^k d^m\}$

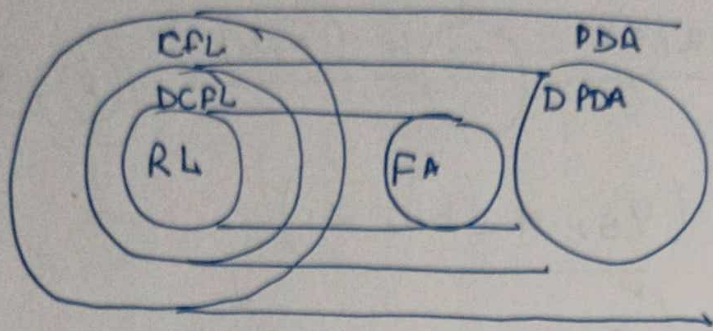
L_1 L_2



CFL

Tut

1. $a^m a^n b^n c^m \mid n, m \geq 1$ — RL ✗. DCFL, DPDA, CFL
2. $a^m b^{m+n} c^n \mid n, m \geq 1$ — $a^m b^m b^n c^n$ ✓
3. $a^m b^n c^{m+n} \mid n, m \geq 1$ — $a^m b^n c^n c^m$ ✓



4. $a^m b^m c^n d^n \mid m, n \geq 1$ — DCFL
5. $a^m b^n d^m d^n \mid m, n \geq 1$ → Not CFL
6. $a^m b^n c^n d^m \mid m, n \geq 1$. DCFL
7. $a^m b^i c^m d^k \mid m, n \geq 1$ → DCFL
8. $a^m b^n \mid m > n$ → Regular. $a^m b^n$
9. $a^n b^{2n} \mid n \geq 1$ DCFG. CFL
10. $a^n b^{n^2} \mid n \geq 1$ ✓ Not PDA ✓ — $a^n b^{n^2}$: ab, aabbbb, aaabbbb
11. $a^n b^{2^n} \mid n \geq 1$ ✗.
12. $ww^R \mid w \in (a,b)^*$ NDPDA. CFL — abab, baba, ww^R, aabb
13. $ww \mid w \in (a,b)^*$ ~~NDPDA~~. ✗
14. $a^n b^n c^m \mid n, m \geq 1$ ~~DPDA~~ ✗.
15. $a^n b^n c^n d^n \mid n \leq 10^{10}$ — Regular, DCFL, CFL.
16. $a^n b^{2n} c^{3n} \mid n \geq 1$ ✗.
17. $xcy \mid x, y \in (0,1)^*$ → h, c

CFL, RL

$S \to OBB$

$B \to OS \mid 1S \mid 0$

$S \to OBB$

$\to 000$

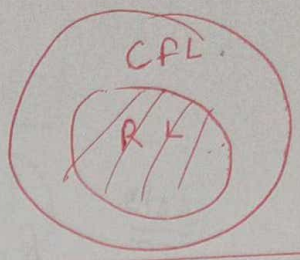
$0\ B\ B$

$0\ 0SB$

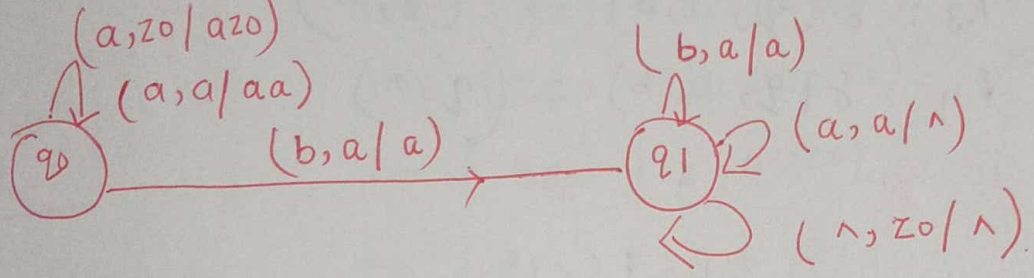
$0\ 00BB\ B$

$000\ 000$

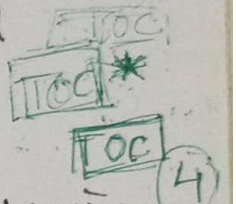
$(000)^*$



$a^n b^m a^n \mid m, n \geq 1$



The intersection of context free language and regular language is context free language

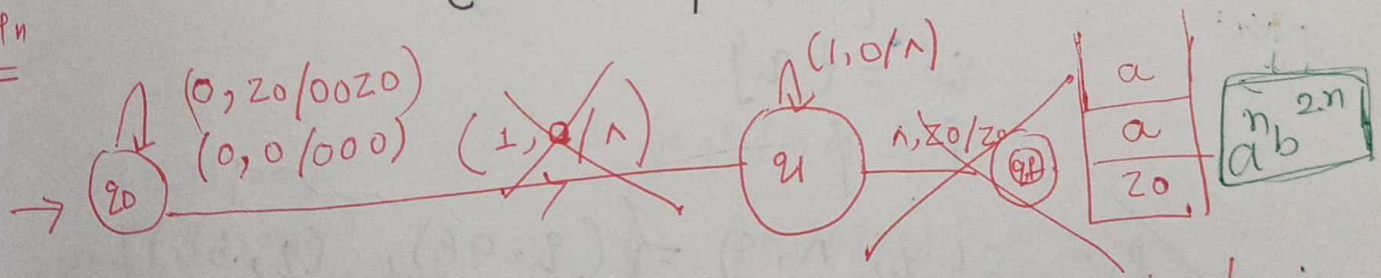


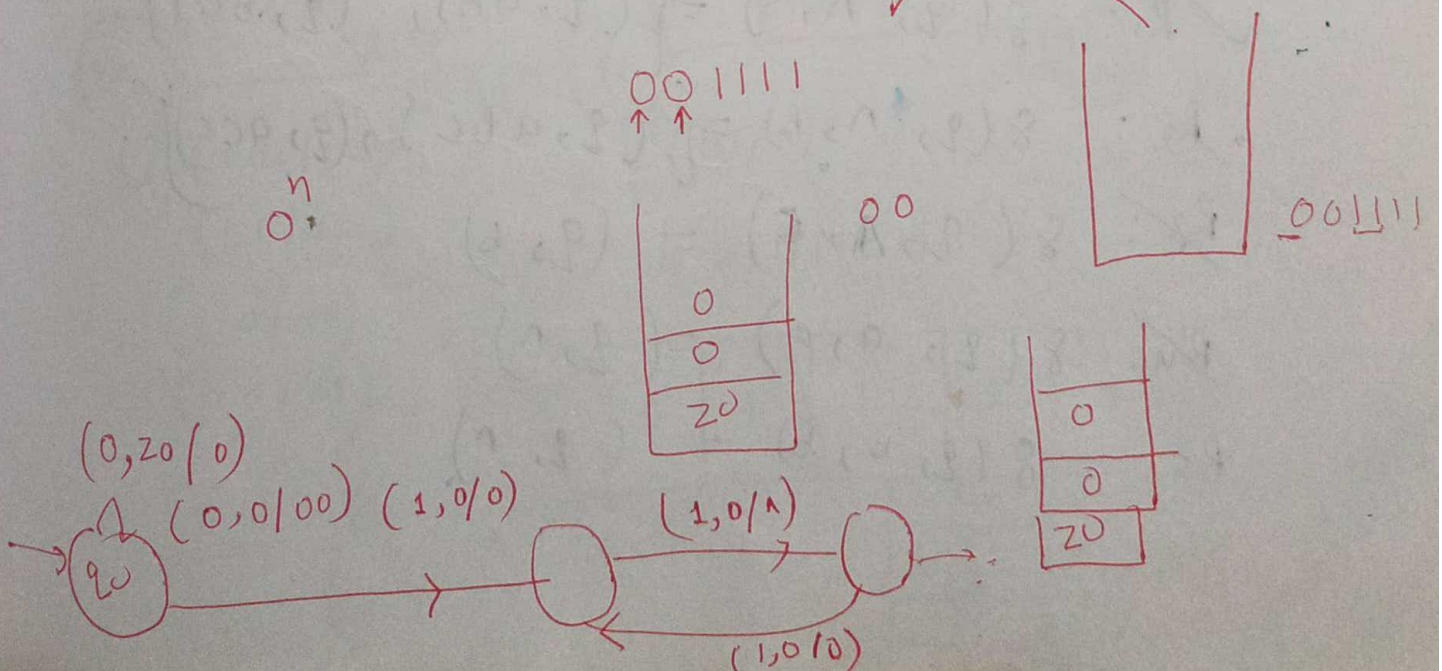
Q. Design a PDA that accepts the following language by final state $a^n b^{2n}$.

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

$a^n b^{2n}$ aabb

Soln





Q Design a PDA equivalent to CFG.

$S \rightarrow aBB$

$B \rightarrow aS \mid bS \mid b$

Soln

$R_1: \delta(q, \wedge, S) = (q, aBB)$

$R_2: \delta(q, \wedge, B) = \{(q, aS), (q, bS), (q, b)\}$

$R_3: \delta(q, a, a) = (q, \wedge)$

$R_4: \delta(q, b, b) = (q, \wedge)$

Q Design PDA for

$S \rightarrow aB \mid bC$

$B \rightarrow aBC \mid aCC$

$C \rightarrow b$

Soln $P = (Q, \Sigma, \Gamma, \delta, q_0, S, F)$ be PDA

$Q = \{q\}$

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

$\Gamma = \{S, B, C, a, b\}$

$q_0 = \{q\}$

$F = \phi$

$R_1: \delta(q, \wedge, S) = \{(q, aB), (q, bC)\}$

$R_2: \delta(q, \wedge, B) = \{(q, aBC), (q, aCC)\}$

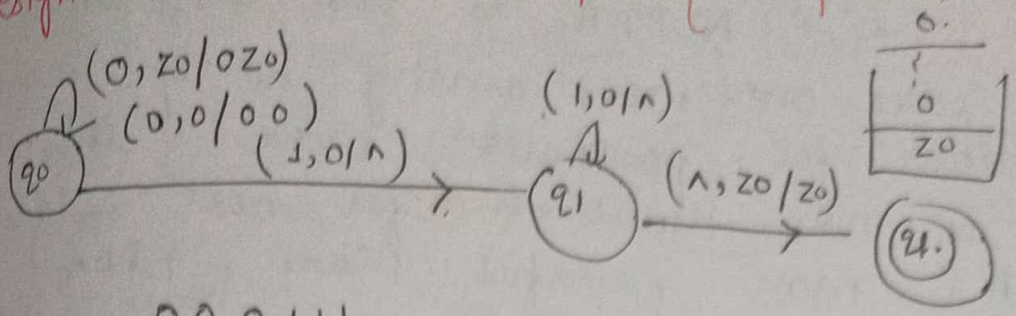
$R_3: \delta(q, A, C) = (q, b)$

$R_4: \delta(q, a, a) = (q, \wedge)$

$R_5: \delta(q, b, b) = (q, \wedge)$.

① Design a PDA that accepts $\{0^n 1^n \mid n \geq 1\}$

Soln



000111

$(q_0, 000111, z_0) \vdash (q_0, 00111, 0z_0)$ by rule 1

$\vdash (q_0, 0111, 00z_0)$ rule 2

$\vdash (q_0, 111, 000z_0)$ rule 2

$\vdash (q_0, 11, 00z_0)$

$\vdash (q_0, 1, 0z_0)$

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

$\vdash (q_f, \wedge, z_0)$.

WcW^R

WW^R

abcba

abbbbba

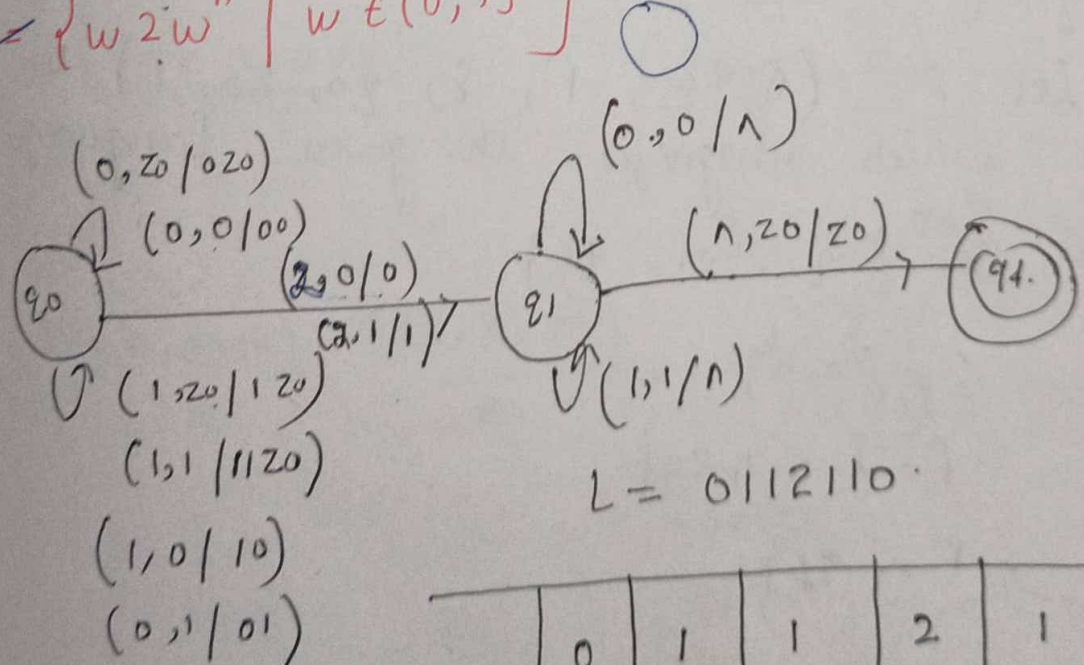
$(q_0, 000111, z_0) \vdash^* (q_f, z_0)$

---

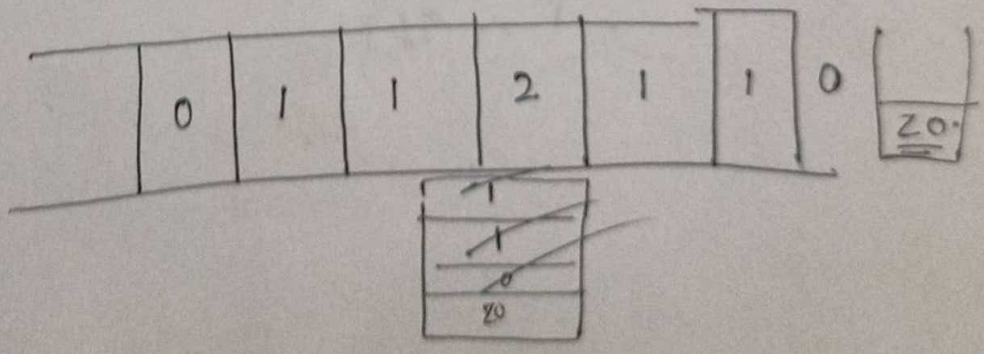
② Design a PDA that recognizes the following lang.

$L = \{w 2 w^R \mid w \in (0,1)^*\}$

Soln

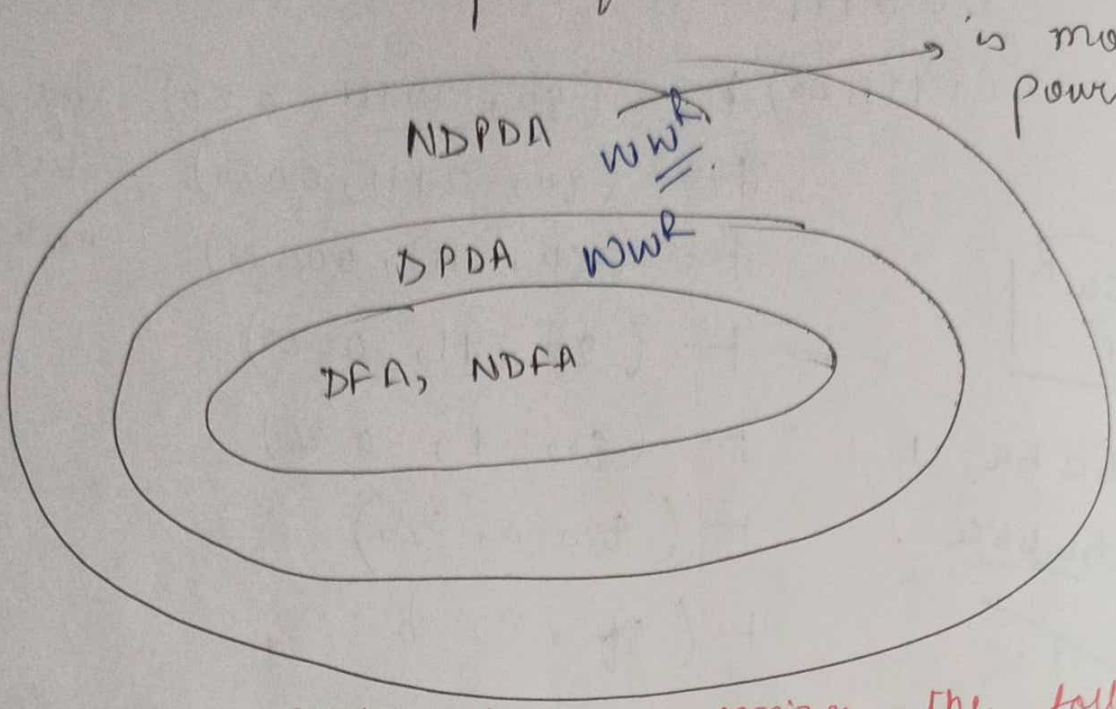


$L = 0112110$.



NDPDA

A NDPDA is one in which we have to select one move among no. of moves. This means for particular move, the NDPDA gives two or more transitions for next state. NDPDA is more powerful than DPDA.



Q Design a PDA that recognizes the following language. $L = \{ w w^R \mid w \in (a,b)^* \}$

eg. ($\wedge$, aa, abba, baab, ———)

Soln

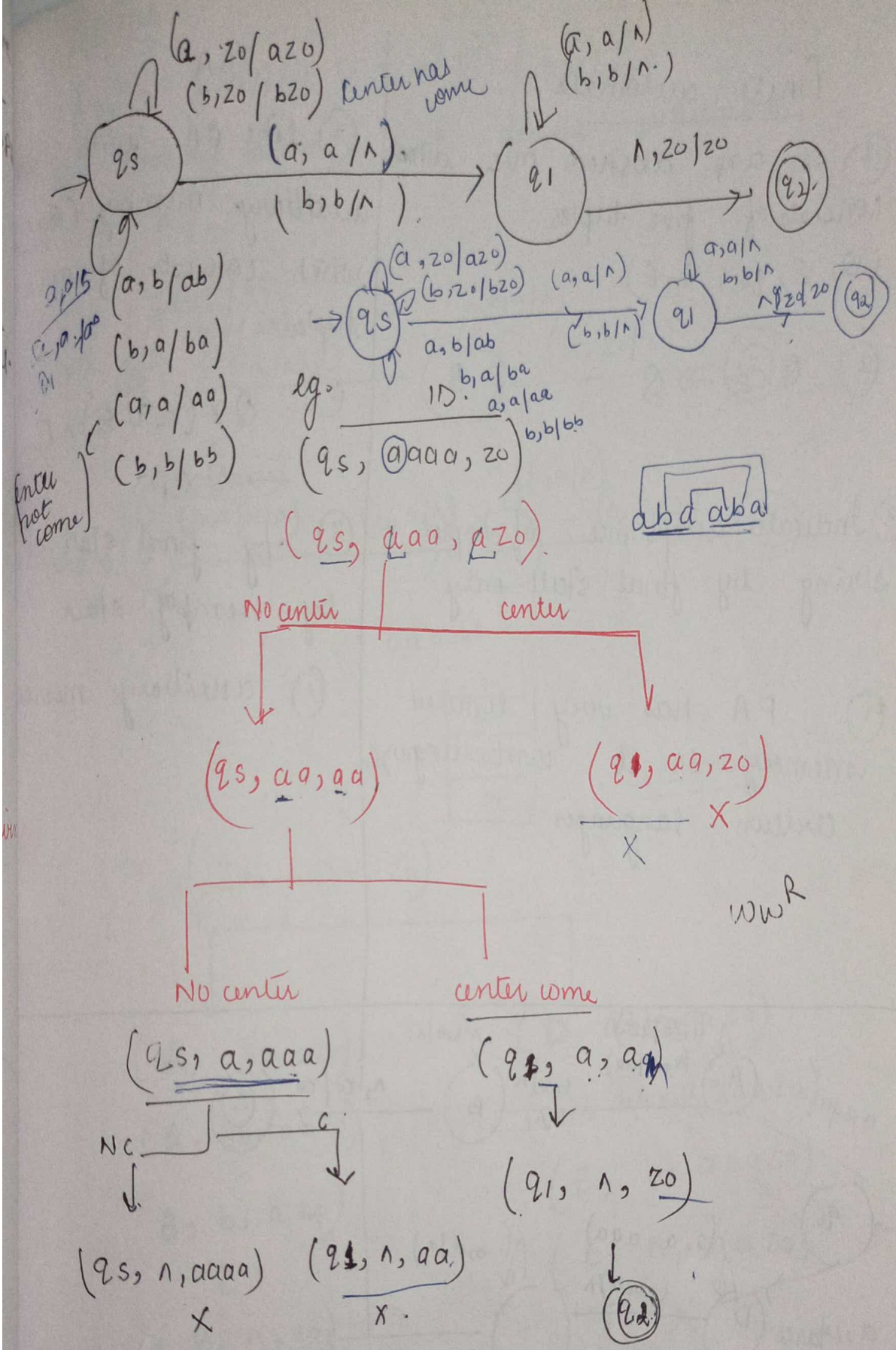
Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be PDA which recognizes the given language where

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

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

$\Gamma = \{ 0, 1, Z_0 \}$

$F = \{ q_f \}$


(a, z0/ az0)
(b, z0/ bz0)
centro nas vezes
(a, a/ λ)
(b, b/ λ)
qs
q1
q2
λ, z0/ z0
(a, b/ ab)
(b, a/ ba)
(a, a/ aa)
(b, b/ bb)
entra pra vezes
eg.
(qs, aaaa, z0)
(qs, aaa, az0)
(qs, aa, aa)
(q1, aa, z0)
X
No centro
centro
(qs, a, aaa)
(q1, a, a)
NC
C
(qs, λ, aaaa)
(q1, λ, aa)
(q1, λ, z0)
abba
abba
wwR