→ There is polynomial time algorithm that can decide whether given two DFA's are equivalent or not

→ Deciding given two NFA's are they equal? is exponential time complex

→ Also, deciding equivalence of two regular expression is exponential time complex.

End of chapter 2.

CHAPTER - 3.

CONTEXT FREE LANGUAGE :- (Introduction)

Let us consider a regular expression given as: $a(a \cup b)^* b$. This expression gives strings consisting of a leading 'a' followed by middle part $(a \cup b)^*$ followed by trailing 'b'.

If we define M be a new symbol to represent $(a \cup b)^*$ then, the above expression may be associated with another new symbol 'S' given by,

$S \to aMb$
$M \to aM$
$M \to bM$
$M \to e$

A context free grammar is a language generator which operates on the rules like above. The language generated by context free grammar is known as the context free language.

Now, let, us take generation of aab. For this case, we have to replace M by aM, whatever be the particular surrounding. So it called context free.

This can be obtained as.

$S \to aMb$ [$\because S \to aMb$
$\Rightarrow aab$ $M \to aM$
$M \to e$]

'a' 'a' 'b' Language (string)

↓ Rule [Grammar]

x-Def Context free Grammar (CFG)

def^n Context free grammar is a quadruple (4-tuple) $G = (V, \Sigma, R, S)$ where,

(i) V is a set of variables (alphabets)
(ii) Σ is a set of terminals, $\Sigma \subseteq V$ (symbols)
(iii) R is set of rules or productions it is subset of $(V - \Sigma) \times V^*$
i.e $R \subseteq (V - \Sigma) \times V^*$
(iv) S is start symbol $S \in (V - \Sigma)$

and $V - \Sigma$ is set of non-terminal symbols.

x Some Terminologies & trends to follow in CFG

(1) $A \to u$ is written when $(A, u) \in R$, $A \in V - \Sigma$, $u \in V^*$ — production

(2) $u \Rightarrow v$ (meaning derives) u derives v if for some $x, y \in V^*$, $u = xAy$, $A \in (V - \Sigma)$, $A \to v'$, $v = xv'y$.

(3) $\overset{*}{\Rightarrow}$ is a symbol of transitive reflexive closure of $\Rightarrow$ (derives)

(4) $L(G_1)$ is Language generated by 'G' such that $L(G_1) = \{\omega \in \Sigma^* : S \overset{*}{\underset{G}{\Rightarrow}} \omega\}$

Context free Grammar (CFG)

Context free grammar is a quadruple (4-tuple) $G = (V, \Sigma, R, S)$ where,

(i) V is a set of variables (alphabets)

(ii) Σ is a set of terminals, $\Sigma \subseteq V$ (symbols)

(iii) R is set of rules or productions it is subset of $(V-\Sigma) \times V^*$
i.e. $R \subseteq (V-\Sigma) \times V^*$

(iv) S is start symbol $S \in (V-\Sigma)$

and $V-\Sigma$ is set of non-terminal symbols.

* Some Terminologies & trends to follow in CFG

(1) $A \xrightarrow{\text{production}} u$ is written when $(A, u) \in R$, $A \in V-\Sigma$, $u \in V^*$

(2) $u \Rightarrow v$ (meaning derives) u derives v if for some $x, y \in V^*$, $u = xAy$, $A \in (V-\Sigma)$, $A \to v'$, $v = xv'y$.

(3) $\Rightarrow^*$ is a symbol of transitive reflexive Closure of $\Rightarrow$ (derives)

(4) $L(G)$ is Language generated by 'G' such that
$L(G) = \{\omega \in \Sigma^* : S \overset{G}{\Rightarrow}{}^* \omega\}$

(5). Derivation:-

Let $G = (V, \Sigma, R, S)$ be a Context free grammar (CFG). If $w_1, w_2, w_3, \ldots, w_n$ are strings over V, such that

$w_1 \Rightarrow w_2 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n$, then we say w_n is derivable.

From w_1 and w_n, we may write $w_1 \Rightarrow^* w_n$.

The sequence of steps used to obtain 'w_n' from w_1 is called derivation.

(6) Sentential Form :-

Derivations from the start symbol produce strings that have a special rule. This is called sentential form. That is, if $G = (V, \Sigma, R, S)$ is a CFG, then any string 'α' in $((V-\Sigma) \cup \Sigma)^*$ such that $S \Rightarrow^* \alpha$ is a sentential form.

Eg: (1) Let $G = (V, \Sigma, R, S)$ where,

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

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

$R = \{S \to aSb;$
$S \to ab\}$

Then find the language $L(G)$ generated by above CFG.

Sol:

S is only one non-terminal (and a start symbol). 'a' and 'b' are two terminals and rules or productions are given by,

$S \to aSb$, $S \to ab$.

Now, we can find Language of given grammar by using inductive method or by using given rules recursively as follows:

From second rule, we may write.

$S \Rightarrow ab$ [$\because S \to ab$]

Also, from first and second rule, we have,

$S \Rightarrow aSb$ [$\because S \to aSb$]

$\Rightarrow aabb$ [$\because S \to ab$]

i.e. $aabb$ or a^2b^2

By applying first rule $(n-1)$ times, Allowed by an application of second rule we get.

$S \Rightarrow aSb$

$\Rightarrow aaSbb$

$\Rightarrow aaaSbbb$

$\Rightarrow aaaaSbbbb$

⋮ ⋮

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

$\Rightarrow a^n b^n$

Hence, the language of given grammar (CFG) is

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

Pb(2). Let $G = (V, \Sigma, R, S)$ is a grammar, where,

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

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

Rules/productions 'R' = $\{S \to aaS, S \to b\}$

Then find the Language generated by this Grammar.

Sol: From 2nd rule,

$S \Rightarrow b$.

But using 1st rule,

$S \Rightarrow aaS$

$\Rightarrow aaaaS$

$\Rightarrow aaaaaaS$

⋮ ⋮

$\Rightarrow (aa)^n S$

From rule is, $S \Rightarrow b$ we may have.

$S \Rightarrow (aa)^n b$

Hence, the Language of given grammar G is,

$L(G) = \{(aa)^n b : n \geq 1\}$ Ans

Pb(3). Let $G = (V, \Sigma, R, S)$ be a grammar,

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

$R = \{S \to aA$

$A \to bS$

$S \to e\}$

Then find the language generated by above grammar.

Sol:- $S \Rightarrow aA$ [using first production]
$\Rightarrow abS$ [using rule 2 in first]
$\Rightarrow abaA$ similarly using rules 1 & 2 and continuously
$\Rightarrow ababS$
$\vdots$
$\Rightarrow (ab)^n S$
$\Rightarrow (ab)^n e$ [$\because S \to e$ from rule 3rd]
$\Rightarrow (ab)^n$

Therefore, the grammar 'G_1' generates the Language,

$$L(G_1) = \{(ab)^n : n \geq 0\}$$

Pb(4.) Let $G = (V, \Sigma, R, S)$ be a grammar, where,

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

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

$R = \{ S \to aSb$
$S \to bSa$
$S \to SS$
$S \to e \}$

Then find the Language generated by this grammar.

Sol:- $S \Rightarrow aSb$
$\Rightarrow abSab$
$\Rightarrow ab\,SSab$
$\Rightarrow abaSbDSaab$

Hence $L(G_1) = \{a^n b^n : n \geq 0$ gives equal number of a & b$\}$
$= \{w \in \{a, b\}^* : w$ has equal numbers of a's & b's$\}$.

Pb(5.) Let $G_1 = (V, \Sigma, R, S)$ be a grammar, where,

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

Then find the Language generated by this grammar.

Sol:- $S \Rightarrow aSa$ [using Rule 1]
$S \Rightarrow abSba$ [using rule 2 on 1]
$S \Rightarrow abaSaba$
$S \Rightarrow ababSbaba$
$S \Rightarrow ababababa$ [using rule $S \to a$]

studing it, the string w is such that, w^R.

Therefor the Language generated by this grammar 'G_1' is

$$L(G_1) = \{w \in \{a, b\}^* : w = w^R\}$$

* 3.1.1 Writing the Context free Grammars:-

Pb (1) Write a Context free grammar (CFG) for the Language given by: $L = \{ a^n b^n : n \geq 0 \}$.

Sol:-

Let the grammar $G = (V, \Sigma, R, S)$ be required CFG, where,

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

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

$R = \{ S \to asb$

$S \to e \}$

Pb (2) Write a Context free grammar (CFG) for the Language $L(G) = \{ a^m b^n : m \geq n \}$

Sol:- Let $G = (V, \Sigma, R, S)$ be required context free grammar where,

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

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

& $R = \{ S \to aS$

$S \to asb$

$S \to e \}$

Pb (3) Write a Context free grammar (CFG) for the Language. $L(G) = \{ w \in \{a, b\}^* :$ no of 'b' is twice the number of 'a' $\}$

Sol^n let $G_1 = (V, \Sigma, R, S)$

Where,

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

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

$R = \{ S \to asbb$
$S \to bbsa$
$S \to bsasb$
$S \to ss$
$S \to e \}$

As a simple analysis

$S \Rightarrow ss$ (left most derivation)
$\Rightarrow asbbs$
$\Rightarrow absasbbbs$
$\Rightarrow abasbbbs$
$\Rightarrow abaasbbbbbs$
$\Rightarrow abaabbbbbs$
$\Rightarrow abaabbbbbasbb$
$\Rightarrow abaabbbbbabb$

Ques(4) Write a context free grammar (CFG) for the language $L(G_4) = \{w \in \{a,b\}^* : w = w^R\}$ (palindrome)

Sol:- let $G = (V, \Sigma, R, S)$

Where,

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

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

$R = \{ S \to asa$
$S \to bsb$
$S \to a$
$S \to b$
$S \to e \}$

Test:- abbaabba

$S \Rightarrow asa \Rightarrow absba$
$\Rightarrow abbsbba$
$\Rightarrow abbasabba$
$\Rightarrow abbaabba$

i.e. $S \overset{*}{\Rightarrow} abbaabba$

Ans(3) Write a context free grammar (CFG) for the Language,

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

Sol^n let $G_3 = (V, \Sigma, R, S)$

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

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

$R = \{ S \to e \mid AB \mid ss$
$A \to (\mid AAB$
$B \to) \mid ABB \}$

Test "(())()"

To derive "(())()" we use the rules

Left Most derivation

$S \underset{LM}{\Rightarrow} SS \Rightarrow ABS \underset{lm}{\Rightarrow} AABBS$
$\Rightarrow (ABBS \Rightarrow ((BBS \Rightarrow (()BS \Rightarrow (($
$(())AB \Rightarrow (())(B$
$\Rightarrow (())()$

Right Most derivation

$S \underset{rm}{\Rightarrow} SS \underset{rm}{\Rightarrow} SAB \underset{rm}{\Rightarrow} SA) \underset{rm}{\Rightarrow} S()$
$\Rightarrow A)() \Rightarrow AAB)() \Rightarrow$

Ans. Write a Context free grammar (CFG) for the Language,

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

Sol? let $G = (V, \Sigma, R, S)$

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

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

$R = \{ S \to e \mid AB \mid SS$
$A \to (\mid AAB$
$B \to) \mid ABB \}$

Test "(())()"

To derive "(())()" we use the rules,

Left Most derivation

$S \underset{LM}{\Rightarrow} SS \Rightarrow ABS \underset{lm}{\Rightarrow} AABBS \Rightarrow (ABBS$

$\Rightarrow ((BBS \Rightarrow (()BS \Rightarrow (())S \Rightarrow$
$(())AB \Rightarrow (())(B$
$\Rightarrow (())()$

Right Most derivation

$S \underset{rm}{\Rightarrow} SS \underset{rm}{\Rightarrow} SAB \underset{rm}{\Rightarrow} SA) \underset{rm}{\Rightarrow} S() \underset{rm}{\Rightarrow} AB()$
$\Rightarrow A)() \Rightarrow AAB)() \Rightarrow AA))()$
$\Rightarrow A(())() \Rightarrow (())()$

3.2 Parse Tree:-

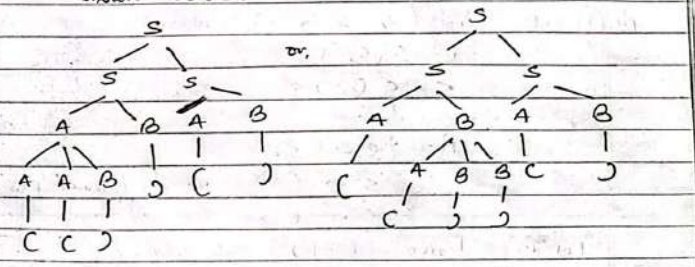
→ A tree structure with nodes and branches connecting the nodes

→ Top node is called root, bottom nodes are called leaves.

→ leaves are labelled with terminals.

→ Connecting the labels of leaves from left to right, we obtain the string of terminals called the "yield of parse tree".

parse tree for balanced parentheis string "(())()" is shown below:



→ Two different parse trees can yield the same string, so the grammar 'G' is "Ambiguous".

Def?

A parse tree is an ordered rooted tree that shows the essentials of a derivation. The parse trees

from $G = (V, \Sigma, R, S)$ are trees with following properties:

1) Each interior node is labeled by a non-terminal symbol i.e $(V - \Sigma)$
2) Each leaf is labeled by $V \cup \{e\}$
3) If an interior node is labeled A and its children are labeled $A_1, A_2, \ldots, A_n$ respectively from left then $A \to A_1 A_2 \ldots A_n$ is a production in R.

pb (1) Let $G = (V, \Sigma, R, S)$ where,

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

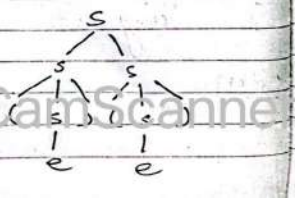
$\Sigma = \{(,)\}$

$R = \{S \to SS,$
$S \to (S),$ properly balanced parenthesis.
$S \to e\}$

Let us derive "()()" as follows.

$S \Rightarrow SS$
$\Rightarrow (S)(S)$
$\Rightarrow ()()$

Now parse tree is:



pb (2) $G = (V, \Sigma, R, S)$ where,

$V = \{A, B, C, S, a, b, c\}$
$\Sigma = \{a, b, c\}$

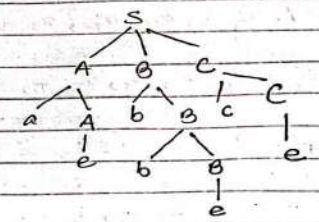
$R = \{S \to ABC$
$A \to aA/e$
$B \to bB/e$
$C \to cC/e\}$

Now give left most derivations (also right most) & parse tree for string "abbc"

Sol: $S \underset{LM}{\Rightarrow} ABC \underset{LM}{\Rightarrow} aABC \underset{LM}{\Rightarrow} aBC \underset{LM}{\Rightarrow} abBC \underset{LM}{\Rightarrow}$

$abbc \underset{LM}{\Rightarrow} abbcC \underset{LM}{\Rightarrow} abbc$

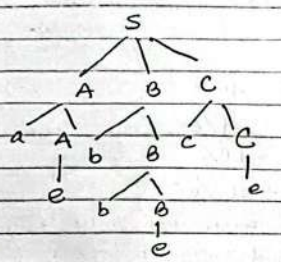
The parse tree is,



The right most $S \underset{RM}{\Rightarrow} ABC \underset{RM}{\Rightarrow} ABcC \underset{RM}{\Rightarrow} ABc \underset{RM}{\Rightarrow}$ Abb

$AbbBc \underset{RM}{\Rightarrow} Abbc \underset{RM}{\Rightarrow} aAbbc \underset{RM}{\Rightarrow} abbc$

The parse tree be.



Same as left Most

(3) Capture the expression $(x_1 + x_2) * x_3$ using principle of Context free grammar and also draw the parse tree.

Sol:- Let $G = (V, \Sigma, R, S)$ be required grammar where,

$V = \{ S, x_1, x_2, x_3, +, *, (,) \}$

$\Sigma = \{ x_1, x_2, x_3, +, *, (,) \}$

$R = \{ S \to (S) \quad$ ["|" indicates or operation]

$S \to S+S$

$S \to S*S$

$S \to x_1 \mid x_2 \mid x_3 \}$

Now for the expression $(x_1 + x_2) * x_3$, it can be written as:

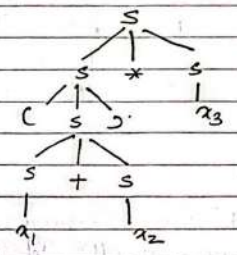
$S \Rightarrow S * S$

$\Rightarrow (S) * S$

$\Rightarrow (S+S) * S$

$\Rightarrow (x_1 + x_2) * x_3$

The parse tree be,



Ambiguous Grammar:-

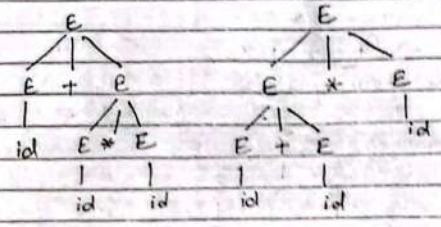
For some CFG's it is possible to find a terminal string with more than one parse tree, or equivalently more than one left-most derivations or right-most derivations. Such a grammar is called ambiguous grammar.

Ex-

Let we take the string "id + id * id" which can be derived as.

1. $E \Rightarrow E+E \Rightarrow E+E*E \Rightarrow$ id + id * id
2. $E \Rightarrow E*E \Rightarrow E+E*E \Rightarrow$ id + id * id

Now, the corresponding parse trees are:-



* Inherent Ambiguity:-

A context free language L is said to be inherently ambiguous if all of its grammars are ambiguous. If even one grammar for L is un-ambiguous, then L is un-ambiguous language.

Ex: Following Grammar is inherently ambiguous.

S → AB/C
A → aAb/ab
B → cBd/cd
C → aCd/aDd
D → bDc/bc

Find the derivations of the string "aabbccdd".

Sol:- Using the left most derivations.

1. $S \underset{LM}{\Rightarrow} AB \underset{LM}{\Rightarrow} aAbB \underset{LM}{\Rightarrow} aabbB \underset{LM}{\Rightarrow} aabbcBd$

$\underset{LM}{\Rightarrow} aabbccdd$

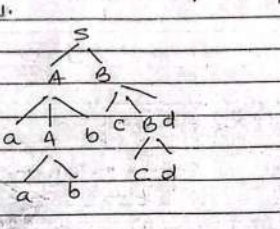
Another

2. $S \underset{LM}{\Rightarrow} C \underset{LM}{\Rightarrow} aCd \underset{LM}{\Rightarrow} aaDdd \underset{LM}{\Rightarrow} aabDcdd$

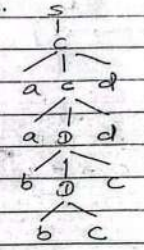
$\underset{LM}{\Rightarrow} aabbccdd.$

Hence the corresponding parse trees are:-

1.



2.



* Application of context free Grammar:-

1. parsers
2. The YACC - Generators
3. Markup - Language
4. XML and Document - Type - Definitions.

HW Solve following

(1) Write a CFG for the language $L(G_1) = \{w \in \{a,b\}^* : w$ has equal number of a's & b's$\}$

(2) $\{ : w$ has even length$\}$

(3) $\{ : w = w w^R \}$

(4) $\{ : w$ has odd length$\}$.

for R:

(1) $R = \{ S \to asb \mid bsa \mid e \mid ss \}$

(2) $R = \{ S \to asa \mid asb \mid bsb \mid bsa \mid e \}$

(3) $R = \{ S \to asa \mid bsb \mid e \}$

(4) $R = \{ S \to asa \mid bsb \mid asb \mid bsa \mid a \mid b \}$

3.2 Regular Grammar:-

A grammar $G = (V, \Sigma, R, S)$ is said to be regular if each production has a right hand side that consists of a string of terminals followed by at most one non-terminal.

Regular grammer may be left-linear or right-linear. That is:

Right linear : $R_1 \subseteq (V - \Sigma) X \Sigma^* ((V - \Sigma) \cup \{e\})$

left linear : $R \subseteq (V - \Sigma) \times ((V - \Sigma) \cup \{e\}) \Sigma^*$

Eg. (1) Let,

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

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

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

$R = \{ S \to abA \mid B \mid baB \mid e$
$A \to bS.$
$B \to aS \mid b \}$

Construction of finite automaton from regular grammar:-

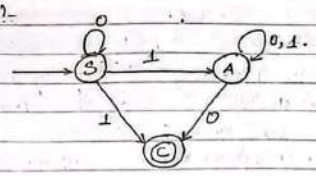
eg

(1) Convert following regular grammer into corresponding finite automaton.

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

CH3-20

Sol:-



Ques (1) For the grammar,

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

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

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

$R = \{ S \to 0AS \mid 0$

$A \to S1A \mid SS \mid 10 \}$

Show the left-most & Right most derivation for the string '00100100'.

Sol:- ✓ left most derivation for 00100100 is :

$S \underset{LM}{\Rightarrow} 0AS \underset{LM}{\Rightarrow} 0S1AS \underset{LM}{\Rightarrow}$ ~~00A1AS~~ (Correct here.) $\underset{LM}{\Rightarrow} 00100AS$

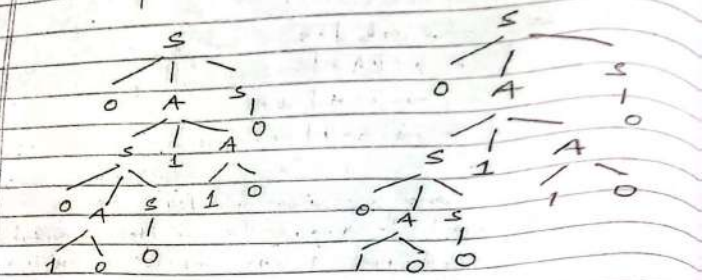
$\underset{LM}{\Rightarrow} 001000AS \underset{LM}{\Rightarrow} 00100100S \underset{LM}{\Rightarrow} 00100100$

✓ For the Right most:

$S \underset{RM}{\Rightarrow} 0AS \underset{RM}{\Rightarrow} 0A0 \underset{RM}{\Rightarrow} 0S10 \underset{RM}{\Rightarrow} 0S1100$

$\Rightarrow 00AS1100 \Rightarrow 00A01100 \Rightarrow 00100100$

In a parse tree it will be:-



#Simplification of CFG:-

We can find out the simplified grammar by applying:

1) We must eliminate useless / non-generating unreachable symbols from given grammar
2) We must eliminate empty-productions
3) We must eliminate unit productions.

1) Elimination of useless symbols:-

→ We say A is generating if $A \overset{*}{\Rightarrow} w$ for some terminal string w.

→ We say A is reachable if there is a derivation $S \overset{*}{\Rightarrow} \alpha A \beta$ for some α & β.

Prb ① Identify and eliminate useless symbols from following grammar.

$S \to aB \mid bX$
$A \to BAd \mid bsx \mid a$
$B \to asB \mid bBx$
$X \to SBD \mid aBx \mid ad.$

Sol:-

Here, A and X are generating due to the $A \to a$ & $X \to ad$. 'S' is also useful due to production $S \to bX$. But B is useless symbol. So, we must eliminate this production. Then we get,

$S \to bX$
$A \to bsx \mid a$
$X \to ad$

Again, A is unreachable symbol. So, we must eliminate this symbol to get:-

$S \to bX$
$X \to ad$

This is the simplified grammar.

Eg ② $S \to AB \mid CA$
$B \to BC / AB$
$A \to a$
$C \to AB / b.$

Ans $\{ S \to CA,\ A \to a,\ C \to b \}$

2. Elimination of empty-production (i.e. e-production)

Prb ① Eliminate e-production from following grammar.

$S \to AB$
$A \to AaA \mid e$
$B \to b$

Sol:- Here $A \to e$ is an empty production. We can eliminate this production as follows:

$S \to AB / B$
$A \to AaA / aA / Aa / a$
$B \to b$

3. Elimination of unit productions:-

Prb ① Eliminate unit production from following grammar.

$S \to AB$
$A \to a$
$B \to C \mid d$
$C \to D$
$D \to E$
$E \to a$

Here, $B \to C$, $C \to D$, $D \to E$ are unit productio
We can eliminate those as follows:

S → AB
A → a
B → a|d (a/d)
C → a
D → a
E → a

Again, C, D and E are unreachable symbols, so, we have to eliminate those symbols, finally we get,

S → AB
A → a
B → a|d

Pb (2) Eliminate unit production from following grammar

S → aA|B
A → bA|C
B → abA|bC

Sol?

S → B is unit - production.

S → aA|abA|bC
A → bA|C
B → abA|bC ✓

#8 CHOMSKY NORMAL FORM (CNF):-

Simplified grammar G in which all productions in one of the following two forms:-

1. A → TU (where A, T & U all are variable non-terminals. i.e. non-terminal → exactly two terminals.
2. A → a, where 'A' is a variable non-terminal 'a' is a terminal i.e. non-terminal → exactly terminal.

Then the simplified grammar 'CFG' is known as chomsky normal form 'CNF' obeying the above two rules.

Pb (1) Convert the following 'CFG' into 'CNF'.

G = (V, Σ, R, S) where,
V = {S, A, a, b}
Σ = {a, b}
R = { S → aAB|AaB|B
A → aA|e
B → ab|bA }

Sol?

Firstly, we have to simplify this grammar. neither useless symbol nor unreachable one. & the e-production i.e. A → e, we get

S → aAB|aB|B|AaB
A → aA|a
B → ab|bA|b

Again, $S \to B$ is an unit production, So, we can eliminate this unit production to get,

$S \to aAB \mid aB \mid AaB \mid ab \mid bA \mid b$

$A \to aA \mid a$

$B \to ab \mid bA \mid b$

Now,

Converting above simplified grammar into CNF, we get,

Here $S \to b$, $A \to a$ and $B \to b$ are in required form. For other productions, we apply

$S \to CAB \mid CB \mid ACB \mid CD \mid DA \mid b$

$C \to a$

$D \to b$

$A \to CA \mid a$

$B \to CD \mid DA \mid b$

Again, $S \to CAB$ and $S \to ACB$ are not in required form So, we can perform replacement of symbols to get:

$S \to EB \mid CB \mid FB \mid CD \mid DA \mid b$

$E \to CA$

$F \to AC$

$C \to a$

$D \to b$

$A \to CA \mid a$

$B \to CD \mid DA \mid b$, This is the required CNF.

Q(2). If $G = (V, \Sigma, R, S)$

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

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

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

$A \to 0 \mid AAB$

$B \to 1 \mid ABB \}$

Express this CFG in 'CNF'.

Sol:- $R = \{ S_0 \to S$

$S \to e \mid AB \mid SS$

$A \to 0 \mid AAB$

$B \to 1 \mid ABB \}$

Again, $A \to AAB$ & $B \to ABB$ to be written so, same to eliminate e.

$R = \{ S_0 \to e \mid AB \mid SS$

$S \to AB \mid SS$

$A \to 0 \mid AAB$

$B \to 1 \mid ABB \}$

Again,

$R = \{ S_0 \to e \mid AB \mid SS$

$S \to AB \mid SS$

$A \to 0 \mid MB$

$M \to AA$

$B \to 1 \mid AN$

$N \to BB \}$

This is the required 'CNF'.

Pb(3) Convert the following CFG to CNF.

$G_1 = (V, \Sigma, R, S)$

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

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

$R = \{ S \to A$
$A \to AAB \mid B \mid b$
$B \to a \}$

Soln.

$R = \{ S_0 \to S$
$S \to A$
$A \to AAB \mid B \mid b$
$B \to a \}$

again, $R = \{ S_0 \to AAB \mid B \mid b$
$A \to AAB \mid B \mid b$
$B \to a \}$

again $R = \{ S_0 \to AAB \mid a \mid b$
$A \to AAB \mid a \mid b$
$B \to a \}$

again $R = \{ S_0 \to AM \mid a \mid b$
$M \to AB$
$A \to AM \mid a \mid b$
$B \to a \}$

This is required CNF.

3.3 Greibach Normal Form (GNF)

GNF is of the form $A \to a\alpha$, where 'A' is a non-terminal and 'a' is exactly one terminal and α is the string of zero or more non-terminals.

Q.(1) Convert the following CFG into GNF.

$$R = \begin{cases} S \to AB \mid BC \\ A \to aB \mid bA \mid a \\ B \to bC \mid cC \mid b \\ C \to c \end{cases}$$

Sol:-

Here, the productions: $S \to AB \mid BC$ is not in GNF on applying substitution rule, we get

$$R = \begin{cases} S \to aBB \mid bAB \mid aB \mid bCC \mid cCC \mid bC \\ A \to aB \mid bA \mid a \\ B \to bC \mid cC \mid b \\ C \to c \end{cases}$$ this is the required form of GNF.

Q.(2) Convert the following Grammar to GNF. where,

$$R = \begin{cases} S \to AB \\ A \to aA \mid bB \mid b \\ B \to b \end{cases}$$

Sol:- Using the substitution, to keep $S \to AB$ in GNF. We have,

$$R = \begin{cases} S \to aAB \mid bBB \mid bB \\ A \to aA \mid bB \mid b \\ B \to b \end{cases}$$ this is in the form $X \to a\alpha$

3.4 Push Down Automata:-

Introduction:-

Push-down automata (PDA) are essentially the finite automata (FA) with auxiliary storage or memory element, known as the stack. The sort of stack is treated as the "pushdown store" allowing read and write access only to the top symbol, would do nicely. In case of PDA, there is control on both the input tape and the stack (stack). The general block-digram of the PDA having input tape, stack, & finite control on both are shown below:

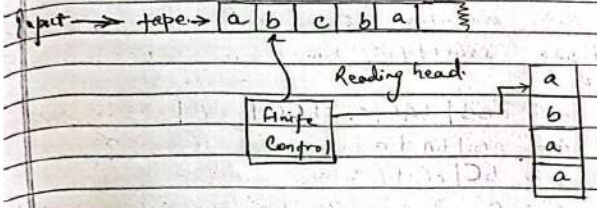


Fig- A PDA.

The stack has two operations namely, push & pop. Push means keep or add the symbol to the top of the stack. Pop means remove the symbol from the top of the stack.

Eg- The transition $((p,4,e),(q,a))$ pushes a but $((p,4,a),(q,e))$ pops- a.

Definition of PDA:-

A push down automata PDA $M=(K,\Sigma,\Gamma,\Delta,s,F)$ is a sextuple (six-tuple) where,

1. K = a finite set of states
2. Σ is an alphabet (the input symbol) [finite set]
3. Γ is an alphabet (finite set of stack symbols)
4. Δ is a transition relation, is a finite subset of $(K\times(\Sigma\cup\{e\}))\times\Gamma^*)\times(K\times\Gamma^*)$
5. $s\in K$ is an initial state.
6. $F\subseteq K$ is the set of final states.

Note:- (i) Instead of using K we use Q
(ii) " " " s " " q_0 } for solving
(iii) " " " Δ " " δ problems.

→ PDA starts with initial state s/q_0 and empty stack.
→ Transition δ/Δ is defined as:

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

meaning,

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

→ The configuration of PDA written as (p,w,t) means PDA is in state p, w is the input string to be

read and 't' is string of stack, read- top-down.

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

$$\delta(p, a, \alpha) = (p', \beta),\ w = aw',\ t = \alpha t_1,$$

$$t' = \beta t_1,\quad t_1 \in T^*$$

→ A string w is said to be accepted by PDA if $(q_0, w, e) \vdash_P^* (q, e, e)$; $q \in F$.

pb (i.) Design a PDA to accept the language $L = \{ww^R : w \in \{a, b\}^*\}$.

Sol:-

Let, the PDA be $M = (K, \Sigma, \Gamma, \Delta, s, F)$

Where,

$K = \{s, f\}$

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

$F = \{f\}$

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

and Δ contains the following transitions -

(i.) $((s, a, e), (s, a))$ — push 'a' } write in stack

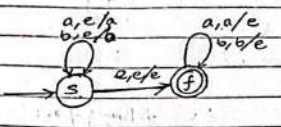
(ii.) $((s, b, e), (s, b))$ — push 'b' }

(iii.) $((s, e, e), (f, e))$ -- switch the states with empty or null string.

(iv.) $((f, a, a), (f, e))$ pop- a (Read 'a') from top of

(v.) $((f, b, b), (f, e))$. pop- b (Read 'b') from top of

Below given figure shows the transition diagram of this PDA.



pb (ii.)

Test that the string "abbbbba" is accepted by above PDA or not

Sol:- To test the string acceptance, following entails a move relation is carried out. If "abbbbba" is read then we find (f, e, e) finally, if it is found then it is not accepted. Say 's' be the starting state & 'f' be the final then,

$(s, abbbbba, e) \vdash_P (s, bbbbba, a)$

$\vdash_P (s, bbbba, ba)$

$\vdash_P (s, bbba, bba)$

$\vdash_P (f, bba, bba)$ → f, bba, bba

$\vdash_P (f, ba, a)$

$\vdash_P (f, a, a)$

$\vdash_P (f, e, e)$

$\vdash (f, e, e)$ ∴ accepted

(5.) Design the PDA 'M' to accept the language $L = \{ w c w^R \mid w \in \{a,b\}^* \}$.

Sol:- Let $M = (K, \Sigma, \Gamma, s, \Delta, F)$

where,

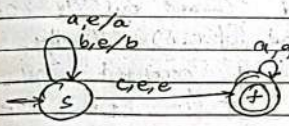
$K = \{s, f\}$ $\quad F = \{f\}$

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

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

Δ contains the following transitions,

1. $((s, a, e), (s, a))$
2. $((s, b, e), (s, b))$
3. $((s, c, e), (f, e))$
4. $((f, a, a), (f, e))$
5. $((f, b, b), (f, e))$



(i) Test that the string "abbcbba" is accepted by above automaton.

✓

Sol:- Now, s is the initial state, & $f \in F$ is the final state, then entail & move relation is given as,

$(s, abbcbba, e) \vdash_P (s, bbcbba, a)$

$\vdash_P (s, bcbba, ba)$

$\vdash_P (s, cbba, bba)$

$\vdash_P (f, bba, bba)$

$\vdash_P (f, ba, ba)$

$\vdash_P (f, a, a)$

$\vdash_P (f, e, e)$

Now, $(s, abbcbba, e) \vdash_P^* (f, e, e)$ where $f \in F$

So, it is accepted. ✓

(6.) Construct a PDA for the language, $L = \{a^n b^n \mid n \geq 1\}$

Also test your PDA for string "aabb".

Sol:-

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