Context – Free Grammar (CFG)

A CFG is a way of describing languages by recursive rules or substitution rules called productions.

A CFG consists of a set of variables, a set of terminals symbols and a start variable as well as the productions. Each production consists of a head variable and a body consisting of a string of zero or more variables and / or terminals.

Variable Symbol – represented by capital letters

Terminal Symbol – represented by lower case letters, numbers or special symbols.

Start variable – Occurs on the left – hand side of the topmost rule.

Example: Grammar G1

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

No. of production rules : 3

Variables : A, B

Start Symbol : A

Terminals : 0, 1, #

Definition of Context-Free Grammar (CFG):

Grammar is denoted as G. Which is defined as

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

where,

V = set of variables or Non Terminals.

T = set of Terminals (V & T are disjoint $\therefore V \cap T = \phi$).

P = finite set of productions each production is of the form

$$A \rightarrow \alpha$$

where A is a variable

α is a string of symbols from $(V \cup T)^*$

S = is a special variable called the start symbol.

Example: 1

Construct the context-free grammar representing the set of palindromes, over $(0+1)^*$.

Solution:

Palindromes are the strings which gives the same meaning when we reverse the string.

1) First possibility is

$S \rightarrow 0 \mid 1 \mid \epsilon$ palindrome with length = 1.

2) If length > 1, then

$S \rightarrow 0S0$

or

$S \rightarrow 1S1$

$\therefore$ The CFG for a palindrome is given by

$S \rightarrow 0 \mid 1 \mid \epsilon$

$S \rightarrow 0S0$

$S \rightarrow 1S1$

Example : 2

Construct a context free grammar for the given expression $(a+b)(a+b+0+1)^*$.

Solution:

From the given expression categorize the operators and arguments / identifiers.

Operators => +, *

Identifiers => a, b, 0, 1

Here we need two varrables to represent

(i) The Start State and the expressions

(ii) The identifiers.

The possible Strings generated by the given expression are

a, b, aa^*, ab^*, $a0^*$, $a1^*$, ba^*, bb^*, $b0^*$, $b1^*$.

First we will identify how to represent the identifiers shown above.

I -> a

I -> b

I -> Ia

I -> Ib

I -> I0

I -> I1

In order to formulate the expression. We need Some more productions like

E -> I

E -> E+E

E -> E*E

E -> (E)

Thus we had ten productions

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

Example : 3

Construct a CFG for the language $L = \{a^n \mid n \text{ is odd}\}$

Solution:

The possible strings are $a, a^3, a^5, \ldots$

(i) Expression with length $1 = E \rightarrow a$

(ii) Expression with $2n+1 = E \rightarrow aaE$

∴ The productions are

$E \rightarrow a$

$E \rightarrow aaE$

∴ $CFG = (V, T, P, S)$ where.

$V = \{E\}$

$T = \{a\}$

$S = E$

$P : E \rightarrow a$

$E \rightarrow aaE$

Example : 4

Construct a CFG for the language

$L = \{wcw^R \mid w \text{ is a string in } (a+b)^*\}$

Solution:

The intermediate symbol is C and the expression can be generated by placing the same symbol on both sides of C.

The possible productions are aca, abcba, ...

So the productions are

$E \rightarrow C$

$E \rightarrow aEa$

$E \rightarrow bEb$

Thus the CFG is given by

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

where,

$V = \{E\}$

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

$S = E$

P : $E \rightarrow C$

$E \rightarrow aEa$

$E \rightarrow bEb$.

Application of CFG:

1) Specification and Compilation of programming languages.

2) Document Type Definitions (DTD).

Additional problems:

1) Find the language L(G) generated by the grammar G with variables S, A, B terminals a, b and productions.

$S \rightarrow aB$, $B \rightarrow b$, $B \rightarrow bA$, $A \rightarrow aB$

Solution:

Since only one start symbol is given, apply the productions to observe the form of terminal string

ie) (i) $S \Rightarrow aB$

$\Rightarrow ab \quad (B \to b)$

(ii) $S \Rightarrow aB$

$\Rightarrow abA \quad (B \to bA)$

$\Rightarrow abaB \quad (A \to aB)$

$\Rightarrow abab \quad (B \to b)$

(iii) $S \Rightarrow aB$

$\Rightarrow abA \quad (B \to bA)$

$\Rightarrow ababA \quad (B \to bA)$

$\Rightarrow ababaB \quad (A \to aB)$

$\Rightarrow ababab \quad (B \to b)$

$\therefore L(G_1) = \{(ab)^n = abab \ldots ab : n \geq 1\}$

---

2. Let G_1 be the grammar $S \to aS \mid aSbS \mid \epsilon$. Prove that $L(G_1) = \{x \mid$ each prefix of x has atleast as many a's as b's$\}$

Solution:

$S \Rightarrow aS$

$\Rightarrow aaSbS \quad (S \to aSbS)$

$\Rightarrow aaaSbS \quad (S \to aS)$

$\Rightarrow aaabbb \quad (S \to b)$

$\therefore x$ has atleast as many a's as b's.

---

3. Construct CFG to generate $\{a^n b^n \mid n \in Z^+\}$.

Solution:

$G_1 = (\{S\}, \{a, b\}, P, S)$ where

$P = \{S \to aSb \mid ab\}$

$S \Rightarrow aSB$

$\Rightarrow aaSbb \quad (S \to aSb)$

$\Rightarrow aaabbb \quad (S \to ab)$

$\Rightarrow a^n b^n$ for $n \geq 1$

4. Consider the alphabet $\Sigma = \{a, b, (,), +, *, \ldots \in\}$. Construct a context free grammar that generates all strings in Σ^* that are regular expressions over the alphabet $\{a, b\}$.

Solution:

Context Free Grammar (CFG)

$R \rightarrow R + R$

$R \rightarrow R . R$

$R \rightarrow R^*$

$R \rightarrow (R)$

$R \rightarrow a \mid b \mid \in$

5. Write a CFG to generate the set $\{ a^m b^n c^p \mid m+n = p \text{ and } p \geq 1 \}$.

Solution:

Context Free Grammar

$S \rightarrow aSc \mid bPc$

$S \rightarrow ac \mid bc$

$P \rightarrow bc$

6. Consider the alphabet $\Sigma = \{a, b, (,), +, *, \ldots, \in\}$. Construct a context Free Grammar. that generates all strings in Σ^* that are regular expressions over the alphabet $\{a, b\}$.

Solution:

Context Free Grammar (CFG)

$R \rightarrow R + R$

6. Find out the Context Free Language.

$S \rightarrow aSb \mid aAb$

$A \rightarrow bAa$

$A \rightarrow ba$

Solution:

$S \to aSb \mid aAb$

$A \to bAa$

$A \to ba$

$S \Rightarrow aSb$
$\Rightarrow aaAbb$
$\Rightarrow aababb$

$S \Rightarrow aAb$
$\Rightarrow aababb$

$S \Rightarrow aSb$
$\Rightarrow aaaSbb$
$\Rightarrow aaaAbbb$
$\Rightarrow aaababbb$

$S \Rightarrow aAb$
$\Rightarrow abAab$
$\Rightarrow abbaaab$

$S \Rightarrow aAb$
$\Rightarrow abbaaab$

$L = \{$ The set of strings over $\Sigma = \{a, b\}$ starting with a and ending with b and substring ba $\}$

7. Construct the CFG for the following Languages:

(i) $L(G) = \{a^m b^n \mid m \neq n \quad m, n > 0\}$ and

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

Solution: 1

CFG:

$S \to aSb$

$S \to aC \mid a \mid bD \mid b$

$C \to aC \mid a$

$D \to bD \mid b$

Solution : 2

CFG:

$S \to aSa$

$S \to b$.

Dr I Jasmine

Derivations and Languages:-

The derivation of CFG can be represented using trees known as derivation trees or parse tree or S-trees. Thus S-tree is a synonym for "derivation tree". If S is the start symbol.

The derivation trees are used in the Compilation process of programming languages.

The derivation may be.

1. Left most derivation
2. Right most derivation

1) Left most derivation:

If at each step in a derivation, a production is applied to the leftmost variable, then it is called leftmost derivation.

Eg,

$E \rightarrow E+E \mid E*E \mid (E) \mid id$

Deriving a string id + id * using left most derivation

$$E \underset{lm}{\rightarrow} \underline{E} + E$$
$$\underset{lm}{\rightarrow} id + \underline{E} \quad (E \rightarrow id)$$
$$\underset{lm}{\rightarrow} id + \underline{E} * E \quad (E \rightarrow E*E)$$
$$\underset{lm}{\rightarrow} id + id * \underline{E} \quad (E \rightarrow id)$$
$$\underset{lm}{\rightarrow} id + id * id \quad (E \rightarrow id)$$

2) Right most derivation:

If at each step in a derivation, a production is applied to the rightmost variable, then it is called rightmost derivation.

Eg:

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

Deriving a string id + id * id using rightmost derivation

$E \to E*E$

$\underset{rm}{\to} E*id \quad (E \to id)$

$\underset{rm}{\to} E+E*id \quad (E \to E+E)$

$\underset{rm}{\to} E+id*id \quad (E \to id)$

$\underset{rm}{\to} id+id*id \quad (E \to id)$

X. Derivation Trees (Parse Tree)

→ The sequence of substitutions to obtain a string is called a derivation.

→ Derivation can be displayed as a derivation tree.

Definition of derivation Tree:

Let $G = (V, T, P, S)$ be a CFG. A tree is a derivation (or) parse tree for G if:

(i) Every vertex has a label, which is a symbol of $V \cup T \cup \{\epsilon\}$

2) The label of the root is S.

3) If a vertex is interior and has label then A must be in V

4) If vertex has label A and sons vertex A are labeled from left as $X_1, X_2 \dots X_k$ then $A \to X_1, X_2 \dots X_k$ must be a production in P.

5) If a vertex has a label ϵ, then it leaf and is the only son of its father.

Ambiguity in Grammars and Languages:

Sometimes there is an occurrence of ambiguous sentences in language we are using. Like that in CFG there is possibility of having two derivations for the same string.

Definition of Ambiguous Grammar:

Let $G = (V, T, P, S)$ be a context free grammar. A grammar 'G' is ambiguous if and only if there exists atleast one string $w \in T^*$ for which two or more different parse trees exists by applying either the left most derivation or right most derivation.

The Different parse trees can be obtained by the following 4 combinations.

1. (left most, left most)
2. (right most, right most)
3. (left most, right most)
4. (right most, left most)

} ⇒ String 'w' with different parse trees.

	Ambiguous	Unambiguous
1.	There exists more than one Leftmost derivation or Rightmost derivation for a string	Unique Leftmost derivation or Rightmost derivation
2.	LMD & RMD represents different parse trees	LMD & RMD represents same parse trees.
3.	More than one parse tree for a string.	Unique parse tree.

Example: 1

Show that $E \to E+E \mid E*E \mid (E) \mid id$ is ambiguous

String is id + id * id

Solution:

Left most derivation:

$E \to \underline{E}+E$

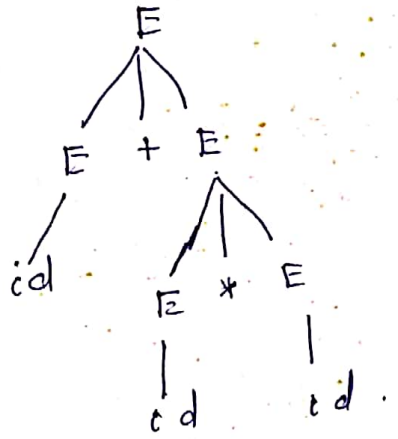
$\to id + \underline{E}$ (E → id)

$\to id + E * E$ (E → E * E)

$\to id + id * E$ (E → id)

$\to id + id * id$ (E → id)

Parse Tree:



Left
~~Right~~ Most derivation

$E \to E*E$

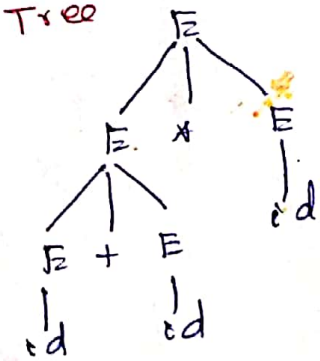
$\to E+E*E$ (E → E + E)

$\to id + E * E$ (E → id)

$\to id + id * E$ (E → id)

$\to id + id * id$ (E → id)

Parse Tree



Ans: The different parse trees for the string id+id*id. So the grammar is ambiguous.

Example:

1) Let G be the grammar $S \to 0B \mid 1A$, $A \to 0 \mid 0S \mid 1AA$, $B \to 1 \mid 1S \mid 0BB$.

For the String 00110101 find

(a) Leftmost derivation

(b) Rightmost derivation

(c) Derivation Tree

(d) For the String 0110 find a rightmost derivation

Solution:

(a) Leftmost Derivation

$S \to 0\underline{B}$

$\to 00\underline{B}B$ ($S \to 0BB$)

$\to 001\underline{B}$ ($B \to 1$)

$\to 0011\underline{S}$ ($B \to 1S$

$\to 00110\underline{B}$ ($S \to 0B$)

$\to 001101\underline{S}$ ($B \to 1S$)

$\to 00110101\underline{B}$ ($S \to 0B$)

$\to 00110101$ ($B \to 1$)

00 1B
001B
0011 0B
00110101

(b) Rightmost Derivation.

$S \to 0B$

$S \to 00\underline{B}B$ ($S \to 0BB$)

$\to 00B1\underline{S}$ ($B \to 1S$)

$\to 00B10\underline{B}$ ($S \to 0B$)

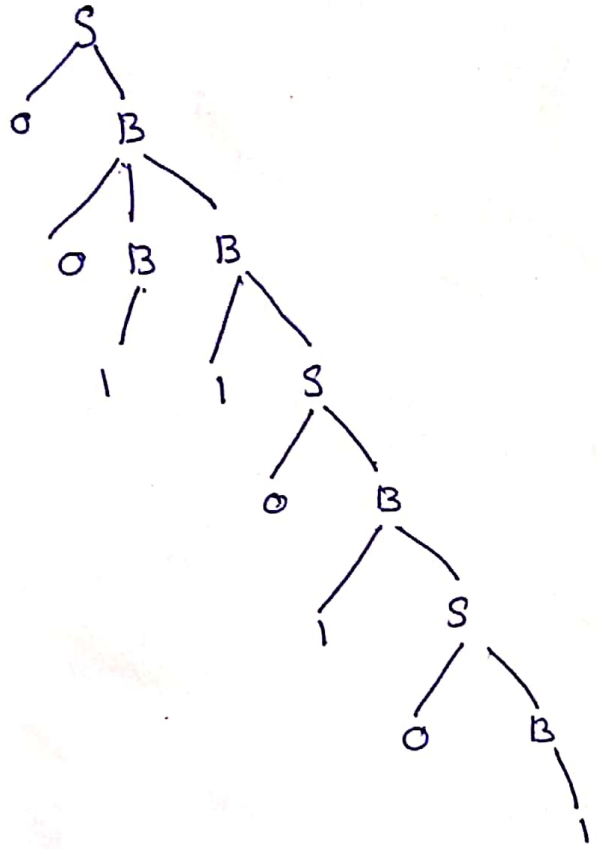
$\to 00B101S$ ($B \to 1S$)

$\to 00B1010B$ ($S \to 0B$)

$\to 00B10101$ ($B \to 1$)

$\to 00110101$ ($B \to 1$)

(c) Derivation Tree:

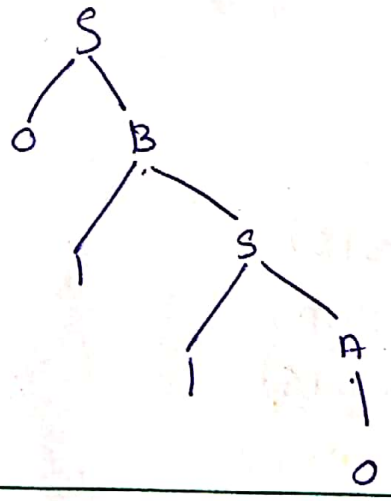


(d) $S \Rightarrow 0B$

$\Rightarrow 01S \quad (B \rightarrow 1S)$

$\Rightarrow 011A \quad (S \rightarrow 1A)$

$\Rightarrow 0110 \quad (A \rightarrow 0)$



Consider the grammar $S \rightarrow aS \mid aSbS \mid \epsilon$. This grammar is ambiguous. Show that the string aab has two (a) parse tree (b) Leftmost derivations (c) Rightmost derivations

Normal Forms for Context Free Grammar (CFG)

Simplification of CFG:

To Simplify a CFG we need to eliminate

1. Variables not deriving terminal.
2. Symbols not appearing in any sentential forms.
3. Null productions.
4. Unit productions.

I. Removing Useless productions:

The productions from a grammar that can never take part in any derivation is called useless productions.

ex. $S \rightarrow aSbS \mid C \mid \epsilon$

$C \rightarrow aC$

The production $S \rightarrow C$ is useless production since 'C' cannot derive any terminal string. Removing this variable and the production will not affect the language generated by this grammar G.

The productions after removing useless symbol and useless productions are,

$S \rightarrow aSbS \mid \epsilon$

II Removing ϵ- productions:

Any productions of CFG of the form $A \rightarrow \epsilon$ is called ϵ- production.

Any variable 'A' for which the derivations, $A \overset{*}{\Rightarrow} \epsilon$ is possible is called as nullable.

Step 1:

Find the set of nullable variables of 'G' for all productions of the form $A \to \epsilon$, put A to Vnullable.

If $B \to A_1, A_2, \ldots A_n$ where $A_1, A_2, \ldots A_n$ are in Vnullable then put B also in Vnullable.

Step 2:

Construct a new set of production 'P'

Eg,

Eliminate ϵ- productions from the grammar

$S \to abB$

$B \to Bb \mid \epsilon$

$B \to \epsilon$ is a null production

$\therefore$ Vnullable = $\{B\}$

Hence the productions will be.

$S \to abB \mid ab$

$B \to Bb \mid b$.

III Removing Unit Productions:

A production in a CFG of the form $A \to B$ where $A, B \in V$ is called a unit productions

Steps:-

For each variable A & B such that $A \overset{*}{\Rightarrow} B$.

The new grammars 'G' is generated by.

1. Putting all non unit productions of 'P' into P.
2. For all A & B, if $A \overset{*}{\Rightarrow} B$

add $A \to x_1 x_2 \ldots x_n$

where $B \to x_1 x_2 \ldots x_n$ where 'x' is not a single variable.

Example:

Eliminate the unit production from the given grammar $G = (V, T, P, S)$

$S \rightarrow AaB \mid c$

$A \rightarrow a \mid bc \mid B$

$C \rightarrow a$

$B \rightarrow A \mid bb$

Unit productions:

$S \rightarrow C$, $A \rightarrow B$, $B \rightarrow A$

Put all non unit productions into P'

P' : $S \rightarrow AaB$

$A \rightarrow a \mid bc$

$B \rightarrow bb$

$C \rightarrow a$

New Rules:

① $S \rightarrow C$

$S \rightarrow a$ [$\because C \rightarrow a$]

2) $A \rightarrow B$

$A \rightarrow bb$ [$\because B \rightarrow bb$]

③ $B \rightarrow A$

$B \rightarrow a \mid bc$ [$\because A \rightarrow a \mid bc$]

The equivalent grammar is

$S \rightarrow a \mid AaB$

$B \rightarrow a \mid bc \mid bb$

$A \rightarrow a \mid bc \mid bb$

$C \rightarrow a$

Example : 1

Remove all unit productions useless Symbols, and all ϵ- productions for the grammar.

$S \rightarrow aA \mid aBB \mid a$

$A \rightarrow aaA \mid \epsilon$

$B \rightarrow bB \mid bbC$

$C \rightarrow B$

I. Remove ϵ- production

$A \rightarrow \epsilon$ is a null production

$V_{nullable} = \{A\}$

$S \rightarrow aA \mid aBB \mid a$ [$\because S \rightarrow aA$, $S \rightarrow a$ [$\because A \rightarrow \epsilon$]]

$A \rightarrow aaA \mid aa$ [$\because A \rightarrow aaA$, $A \rightarrow aa$ [$\because A \rightarrow \epsilon$]]

$B \rightarrow bB \mid bbC$

$C \rightarrow B$

II Remove all unit productions:

Unit production $C \rightarrow B$

Non unit production

$S \rightarrow aA \mid aBB \mid a$

$A \rightarrow aaA \mid aa$

$B \rightarrow bB \mid bbC$

New rules

$C \rightarrow B$

$C \rightarrow bB$ [$\because B \rightarrow bB$]

$C \rightarrow B$

$C \rightarrow bbC$ [$\because B \rightarrow bbC$]

After removing unit production $C \rightarrow B$

$S \rightarrow aA \mid aBB \mid a$

$A \rightarrow aaA \mid aa$

$B \rightarrow bB \mid bbC$

$C \rightarrow bB \mid bbC$

III Removing useless Symbols:

B and C are useless Symbols. Since they will not derive any terminal strings. After removing the useless Symbols B & C, the corresponding grammar is as follows.

$S \rightarrow aA \mid a$

$A \rightarrow aaA \mid aa$

Normal Forms for CFG or Properties of context Free Languages.

There are two important normal forms.

1. Chomsky Normal Form (CNF)
2. Greibach Normal Form (GNF)

1. Chomsky Normal Form (CNF):

Theorem:

Any context-free language without ϵ is generated by a grammar in which all productions are of the form $A \rightarrow BC$ or $A \rightarrow a$ where A, B and C are variables and a is a terminal.

Non Terminal → Non Terminal. Non Terminal

Non Terminal → Terminal

Proof:

Steps:

1. Eliminate all ϵ- productions
2. Eliminate unit productions
3. Eliminate useless symbols

If all the productions is of form $A \to BC$ or $A \to a$ then the 'G' is in CNF.

Otherwise for grammar $A \to A_1, A_2 \dots A_n$, $n \geq 3$ introduce new productions and new variables. Restrict the length of RHS as 2 variables or a single terminal such as

$A \to AD_1$, $D1 \to A_2D_2$, $D2 \to A_3D_3 \dots D_{m-2} \to A_m$,

If $A \to A_1 \dots A_n$, If 'A_i' is a terminal 't' then introduces a new variable, such that $D_i \to t$, where $D_i \in V$

Example:

Construct a CNF for the following

$S \to AAC$

$A \to aAb \mid \epsilon$

$C \to aC \mid a$

Solution:

I. Remove ϵ- productions

$V_{nullable} = \{A\}$ $\because A \to \epsilon$ is null production

After removing ϵ- productions, the grammar is

$S \to AAC \mid AC \mid C$

$A \to aAb \mid ab$

$C \to aC \mid a$

Dr I Jasmine

II Remove all unit productions

$S \to C$ is unit production

Non unit productions

$S \to AAC \mid AC$

$A \to aAb \mid ab$

$C \to aC \mid a$

New rules

$S \to C$

$S \to aC$ [$\because C \to aC$]

$S \to C$

$S \to a$ [$\because C \to a$]

The grammar after eliminating productions are,

$S \to AAC \mid AC \mid ac \mid a$

$A \to aAb \mid ab$

$C \to aC \mid a$.

III Remove all useless Symbols:

There are no useless Symbols and productions.

IV Reduce the given grammar into CNF.

P_1: $S \to AAC$

$S \to AD_1$

$D_1 \to AC$

P_2: $S \to AC$

P_3: $S \to aC$

$S \to D_2C$

$D_2 \to a$

P_4: $S \to a$.

Dr I Jasmine

P5: A → aAb

P5 $A \rightarrow D_2Ab$

$D_2 \rightarrow a$

P6 $A \rightarrow D_2Ab$

$D_3 \rightarrow A$

P7 $A \rightarrow D_2D_3b$

$D_4 \rightarrow b$

P8 $A \rightarrow D_2D_3D_4$

~~$D_5 \rightarrow D_3D_4$~~

$\therefore A \rightarrow D_2D_5$

$D_5 \rightarrow D_3D_4$

P9.

P10: $A \rightarrow ab$

$A \rightarrow D_2b$ [1

$D_2 \rightarrow a$

$A \rightarrow D_2D_4$ [2

$D_4 \rightarrow b$.

P11 $C \rightarrow aC$

$C \rightarrow D_2C$

$D_2 \rightarrow a$.

Resultant CNG

$S \rightarrow AD_1 \mid AC \mid D_2C \mid a$

$A \rightarrow D_2D_5 \mid D_2D_4$

$C \rightarrow D_2C \mid a$

$D_1 \rightarrow AC$

$D_2 \rightarrow a$

$D_4 \rightarrow b$

$D_5 \rightarrow D_3D_4$

Dr I Jasmine

Examples:

1) Consider the grammar $G(\{S, A, B\}, \{a, b\}, P, S)$ has the production.

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

Convert it into CNF.

Solution:

(i) Find the productions which are already in CNF

$A \rightarrow a$

$B \rightarrow b$

(ii) Replace the terminals on the right by new variables.

$S \rightarrow bA$, $S \rightarrow aB$, $A \rightarrow bAA$, $A \rightarrow aS$

$B \rightarrow aBB$, $B \rightarrow bS$

Change this as follows

$S \rightarrow bA$

$S \rightarrow C_bA$

$C_b \rightarrow b$

$S \rightarrow aB$

$S \rightarrow C_aB$

$C_a \rightarrow a$

$A \rightarrow bAA$

$A \rightarrow C_bAA$

$C_b \rightarrow b$

$A \rightarrow aS$

$A \rightarrow C_a S$

$C_a \rightarrow a$

$B \rightarrow aBB$

$B \rightarrow C_a BB$

$C_a \rightarrow a$

$B \rightarrow bS$

$B \rightarrow C_b S$

$C_b \rightarrow b$

(iii) According to CNF theorem, the right hand side body should contain only two variables.

$A \rightarrow C_b AA$

$A \rightarrow C_b D_1$

$D_1 \rightarrow AA$

$B \rightarrow C_a BB$

$B \rightarrow C_a D_2$

$D_2 \rightarrow BB$

Thus the resultant productions are

$S \rightarrow C_b A \mid C_a B$

$A \rightarrow C_a S \mid C_b D_1 \mid a$

$B \rightarrow C_b S \mid C_a D_2 \mid b$

$D_1 \rightarrow AA$

$D_2 \rightarrow BB$

$C_a \rightarrow a$

$C_b \rightarrow b$

Dr I Jasmine

2) Convert the CFG into CNF

$S \rightarrow AB \mid Aa$

$A \rightarrow aAA \mid a$

$B \rightarrow bBB \mid b$

Solution:

(i) The production which are already in CNF

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$

(ii) Convert the other productions into CNF

$S \rightarrow Aa$, $A \rightarrow aAA$ $B \rightarrow bBB$

$S \rightarrow Aa$

$S \rightarrow ACa$

$Ca \rightarrow a$

$A \rightarrow aAA$

$A \rightarrow CaAA$

$Ca \rightarrow a$

$B \rightarrow bBB$

$B \rightarrow CbBB$

$Cb \rightarrow b$

(iii) According to CNF theorem, the right hand side body should contain only two variables

$A \rightarrow CaAA$

$A \rightarrow CaD1$

$D1 \rightarrow AA$

Dr I Jasmine

$B \rightarrow C_b BB$

$B \rightarrow C_b D_2$

$D_2 \rightarrow BB$.

Thus the resultant production are:

$S \rightarrow AB \mid AC_a \mid$

$A \rightarrow C_a D_1 \mid a$

$B \rightarrow C_b D_2 \mid b$

$C_a \rightarrow a$

$C_b \rightarrow b$

$D_1 \rightarrow AA$

$D_2 \rightarrow BB$

Dr I Jasmine

part C
9

Convert the following CFG into CNF

S -> aaaaS

S -> aaaa

Solution:

As we know the rule for chomsky's Normal Form is

Non terminal -> Non terminal. Non Terminal

Non Terminal -> Terminal

(i) Find the productions which are already in CNF

Nil

(ii) Replace the terminals on the right by new variables.

S -> aaaaS

S -> aaaa

Change this as follows:

S -> aaaaS

S -> AAAAS

A -> a

S -> aaaa

S -> AAAA

A -> a

(iii) According to CNF theorem, the right hand side body should contain only two variables.

S -> AAAAS

S -> AC1

C1 -> AAAS

C1 -> C2C3

C2 -> AA

C3 -> AS

(ii) $S \rightarrow aaaa$

$S \rightarrow AAAA$

$S \rightarrow C_2C_4$

$C_2 \rightarrow AA$

$C_4 \rightarrow AA$

Thus the resultant productions are

$S \rightarrow AC_1 \mid C_2C_4$

$C_1 \rightarrow C_2C_3$

$C_2 \rightarrow AA$

$C_3 \rightarrow AS$

$C_4 \rightarrow AA$

$A \rightarrow a$

---

part B 10 (i) Convert the given CFG to CNF

$S \rightarrow AB \mid aB$

$A \rightarrow aab \mid \epsilon$

$B \rightarrow bbA$

Solution:

(i) Remove ϵ- production

$V_{nullable} = \{A\}$ ∴ [$A \rightarrow \epsilon$ is null production]

After removing ϵ- productions, the grammar is

$S \rightarrow AB \mid B \mid aB$

$A \rightarrow aab$

$B \rightarrow bbA \mid bb$

(ii) Remove all unit production

$S \rightarrow B$ is unit production.

Non unit productions

$S \rightarrow AB \mid aB$

$A \rightarrow aab$

$B \rightarrow bbA \mid bb$

Dr I Jasmine

The new rules

$S \rightarrow B$

$S \rightarrow bbA$ [$B \rightarrow bbA$]

$S \rightarrow B$

$S \rightarrow bb$ [$B \rightarrow bb$]

After eliminating unit production, the Grammar

$S \rightarrow AB \mid aB \mid bbA \mid bb$

$A \rightarrow aab$

$B \rightarrow bbA \mid bb$

(iii) Reduce the given Grammar into CNF

$S \rightarrow AB$ ✓ [Already in CNF)

$S \rightarrow aB$

$S \rightarrow C_{a_1}B$ ✓

$C_{a_1} \rightarrow a$ ✓

$S \rightarrow bbA$

$S \rightarrow C_{b_1}A$ ✓

$C_{b_1} \rightarrow bb$

$C_{b_1} \rightarrow C_{b_2}C_{b_2}$ ✓

$C_{b_2} \rightarrow b$ ✓

$S \rightarrow bbA$

$S \rightarrow C_{b_1}A$ ✓

$C_{b_1} \rightarrow bb$

$C_{b_1} \rightarrow C_{b_2}C_{b_2}$

$C_{b_2} \rightarrow b$

$S \rightarrow bb$

$S \rightarrow C_{b_1}$ ✓

$C_{b_1} \rightarrow bb$

$C_{b_1} \rightarrow C_{b_2}C_{b_2}$ ✓

$A \rightarrow aab$

$A \rightarrow C_{a_2} C_{b_1}$ ✓

$C_{a_2} \rightarrow aa$

$C_{a_2} \rightarrow C_{a_1} C_{a_1}$ ✓

$B \rightarrow bbA$

$B \rightarrow C_{b_1} A$ ✓

$C_{b_1} \rightarrow bb$

$C_{b_1} \rightarrow C_{b_2} C_{b_2}$ ✓

$C_{b_2} \rightarrow b$ ✓

$B \rightarrow bb$

$B \rightarrow C_{b_1}$

$C_{b_1} \rightarrow bb$

$C_{b_1} \rightarrow C_{b_2} C_{b_2}$

The result production are. (CNF):

$S \rightarrow AB \mid C_{a_1} B \mid C_{b_1} A \mid C_{b_1}$

$A \rightarrow C_{a_2} C_{b_1}$

$B \rightarrow C_{b_1} A$

$C_{a_1} \rightarrow A$

$C_{b_1} \rightarrow C_{b_2} C_{b_2}$

$C_{b_2} \rightarrow b$

$C_{a_2} \rightarrow C_{a_1} C_{a_1}$

Dr I Jasmine

Greibach Normal Form

Theorem:

Every context-free language L without ϵ can be generated by a grammar for which every production is of the form $A \to ad$, where A is a variable, a is a terminal and d is a string of variables.

The rule for GNF is

Non-terminal → one terminal. Any no. of non terminal

Non-terminal → terminal

Example

$S \to aA$
$S \to a$
} GNF

$S \to AA$
$S \to Aa$
} not in GNF.

Step 1: Remove unit productions, Null productions and useless symbols.

Step 2: Check if it is in CNF & Convert into CNF if not.

Step 3: Change the names of the non terminal symbols into some A_i in ascending order of i.

Step 4: Check for productions that i value in the LHS < RHS, if not remove the left recursion $A_i \to A_j X$, $i < j$ & should never be $i \geq j$

part-c 10.

Example.

Convert the given CFG to GNF

$S \to ABA$
$A \to aA \mid \epsilon$
$B \to bB \mid \epsilon$

Dr I Jasmine

Solution:

Step 1: Remove ϵ production.

$A \to \epsilon$

$B \to \epsilon$.

$\therefore S \to ABA \mid AB \mid BA \mid AA \mid B \mid A$

$A \to aA \mid a$

$B \to bB \mid b$

Step 2: Remove unit production.

$\therefore S \to A$, $S \to B$

a) $S \to A$ $\quad S \to B$

$S \to aA$ $\quad S \to bB$

$S \to A$ $\quad S \to B$

$S \to a$ $\quad S \to b$

After removal

$S \to ABA \mid AB \mid BA \mid AA \mid aA \mid bB \mid a \mid b$

$A \to aA \mid a$ $\quad B \to bB \mid b$

Not in GNF

$S \to ABA \mid AB \mid BA \mid AA$

(i) $S \to ABA$

Replace A by $aA \mid a$ (First variable)

$S \to aABA \mid aBA$

(ii) $S \to AB$

Replace A by $aA \mid a$

$S \to aAB \mid aB$

(iii) $S \to BA$ (Replace B by $bB \mid b$.

$S \to bBA \mid bA$

Dr I Jasmine

IV $S \rightarrow \underline{A}A$

Replace A by $aA \mid a$

$S \rightarrow aA\bar{A} \mid a\bar{A}$

Equivalent GNF

$S \rightarrow aABA \mid aBA \mid aAB \mid aB \mid bBA \mid bA \mid aAA \mid aA \mid bB \mid a \mid b$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b.$

---

part B 8. Obtain the Greibach Normal Form.

$S \rightarrow AB$

$A \rightarrow BS \mid b$

$B \rightarrow SA \mid a$

Solution:

The given Grammar is in CNF so we apply Greibach Normal Form. Rename the variable $A_1 A_2 A_3$

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 \mid b$ $\Rightarrow$ $S = A_1$, $A = A_2$, $B = A_3$

$A_3 \rightarrow A_1 A_2 \mid a$

Check $A_i \rightarrow A_j$ where $i < j$

$A_1 \rightarrow A_2 A_3 \Rightarrow i < j \rightarrow$ ∴ The production is in required form.

$A_2 \rightarrow A_3 A_1 \mid b \Rightarrow i < j$ ∴ The production is in required form.

$A_3 \rightarrow A_1 A_2 \mid a \Rightarrow i > j$ ∴ The production is not in required form.

$\therefore A_3 \rightarrow \underline{A_1} A_2 \mid a$

$A_3 \rightarrow \underline{A_2} A_3 A_2 \mid a \quad [A_1 \rightarrow A_2 A_3]$

$A_3 \rightarrow A_3 A_1 A_3 A_2 \mid b A_3 A_2 \mid a \quad [A_2 \rightarrow A_3 A_1]$

The above is in Left Recursion.

$\therefore$

$A \rightarrow A\alpha \mid \beta$

$A' \rightarrow \alpha A' \mid \alpha$

$A \rightarrow \beta A' \mid \beta$

$\therefore A_3 \rightarrow \underset{\downarrow A}{A_3} \underbrace{A_1 A_3 A_2}_{\alpha} \mid \underbrace{b A_3 A_2 \mid a}_{\beta}$

$A' \rightarrow \alpha A' \mid \alpha$

$A' \rightarrow A_1 A_3 A_2 A' \mid A_1 A_3 A_2$

$A \rightarrow \beta A' \mid \beta$

$\therefore A_3 \rightarrow b A_3 A_2 A_1 A_3 A_2 A' \mid b A_3 A_2 A_1 A_3 A_2 \mid$

$a A_1 A_3 A_2 A_1 \mid a A_1 A_3 A_2 \mid b A_3 A_2 \mid a$

$\therefore$ The above production is in GNF.

Consider $A_2 \rightarrow \underline{A_3} A_1 \mid b$

Apply A_3 in A_2.

$A_2 \rightarrow b A_3 A_2 A_1 A_3 A_2 A_1 A_1 \mid b A_3 A_2 A_1 A_3 A_2 A_1$

$a A_1 A_3 A_2 A_1 A_1 \mid a A_1 A_3 A_2 A_1 \mid b A_3 A_2 A_1 \mid$

$a A_1 \mid b.$

$\therefore$ The above production in GNF

Now consider $A_1 \rightarrow \underline{A_2} A_3$ Apply A_2 in A_1

$\therefore A_1 \rightarrow b A_3 A_2 A_1 A_3 A_2 A_1 A_1 A_3 \mid b A_3 A_2 A_1 A_3 A_2 A_1 A_3 \mid$

$a A_1 A_3 A_2 A_1 A_1 A_3 \mid a A_1 A_3 A_2 A_1 A_3 \mid$

$b A_3 A_2 A_1 A_3 \mid a A_1 A_3 \mid b A_3$

Now the above production is in GNF.

Now Consider A' which is not in GNF

$A' \rightarrow \underline{A_1} A_3 A_2 A' \mid \underline{A_1} A_3 A_2$

$A' \rightarrow bA_3A_2A_1A_3A_2A_1A_1A_3A_3A_2A' \mid bA_3A_2A_1A_3A_2A_1A_3A_3A_2A' \mid aA_1A_3A_2A_1A_1A_3A_3A_2A' \mid aA_1A_3A_2A_1A_3A_3A_2A' \mid bA_3A_2A_1A_3A_3A_2A' \mid aA_1A_3A_3A_2A' \mid bA_3A_3A_2A' \mid bA_3A_2A_1A_3A_2A_1A_1A_3A_3A_2 \mid bA_3A_2A_1A_3A_2A_1A_3A_3A_2 \mid aA_1A_3A_2A_1A_1A_3A_3A_2 \mid aA_1A_3A_2A_1A_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2 \mid aA_1A_3A_3A_2 \mid bA_3A_3A_2$

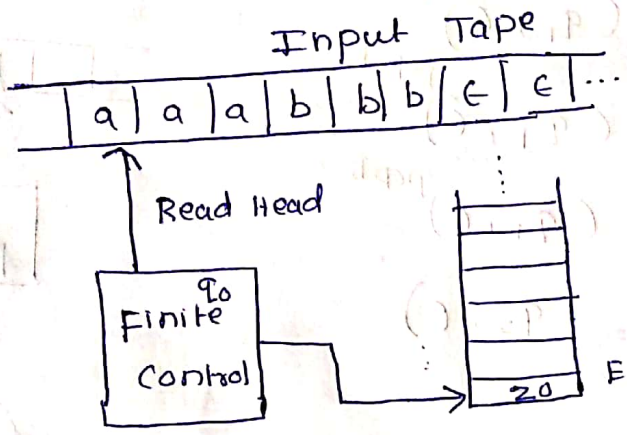
Construct a PDA for the language $L = \{a^n b^n \mid n \geq 1\}$

Solution:

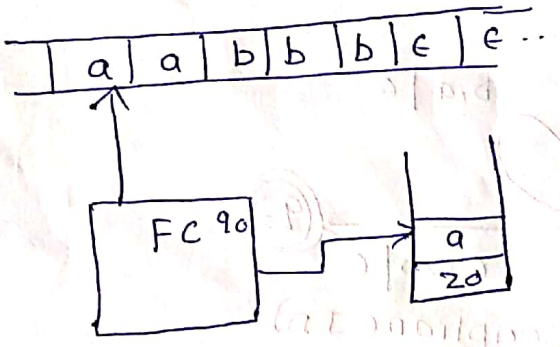
$$L = \{a^n b^n \mid n \geq 1\}$$

$$\therefore L = \{ab, aabb, aaabbb, \ldots\}$$

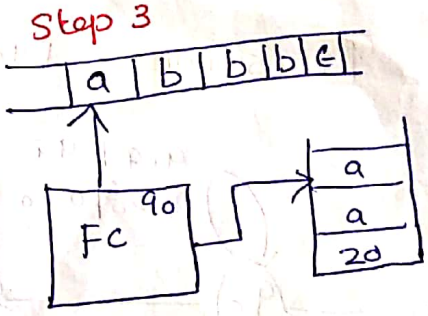
Step 1:



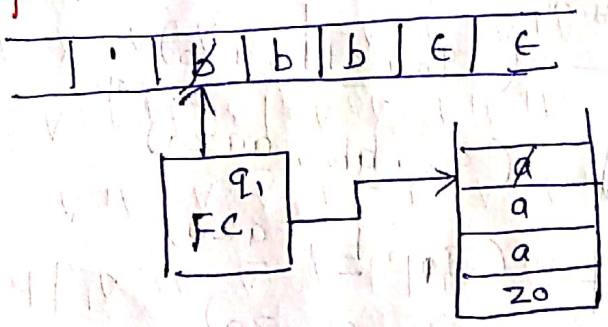
Step : 2



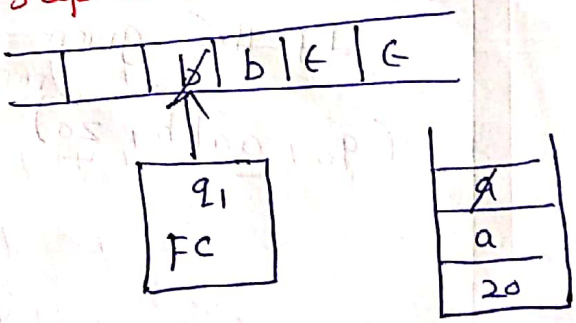
Step 3



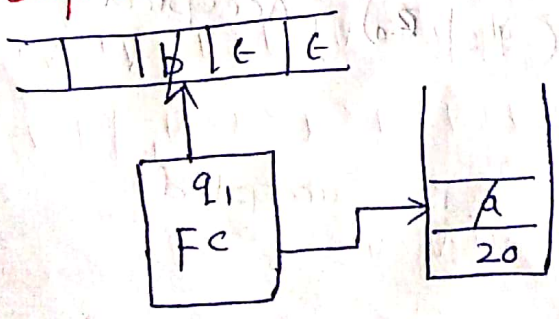
Step 4:



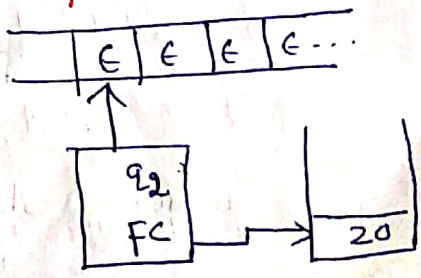
Step 5:



Step 6:



Step 7:



Reached Empty Stack or Reached the Final State

It is accepted.

Transition Function

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

$\delta(q_0, a, a) = (q_0, aa)$ (push, stack top)

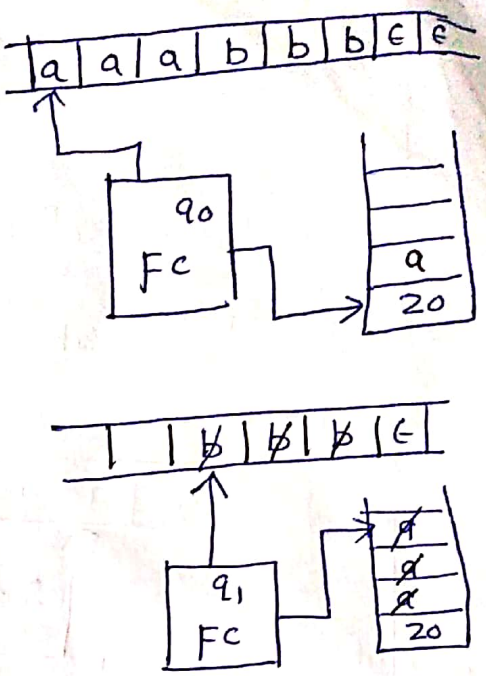
$\delta(q_0, a, a) = (q_0, aa)$ Recursive

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

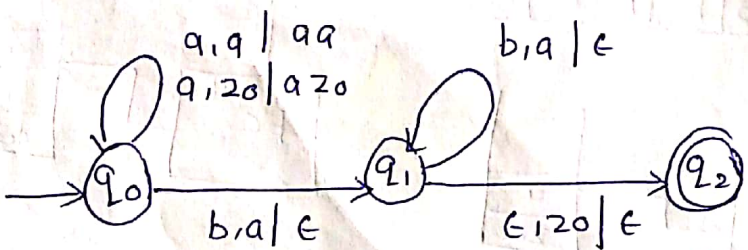
$\delta(q_1, b, a) = (q_1, \epsilon)$ pop a

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

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



Transition Diagram:



Instantaneous Description (ID):

If the given string is accepted or not

$(q_0, \underline{a}abb, \underline{z}_0)$ (sample, Empty stack)

$\vdash (q_0, \underline{a}bb, \underline{a}z_0)$

$\vdash (q_0, \underline{b}b, \underline{a}az_0)$ pop

$\vdash (q_0, \underline{b}, \underline{a}z_0)$

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

$\vdash (q_2, \epsilon)$ Accepted.

∴ The given string is accepted.

part B, 12

Construct a PDA for the Languages $L = \{ wcw^R \mid 'w' \text{ in } (a+b)^* \}$. For the given PDA write the instantaneous description for the string 'aabcbaa'.

Solution

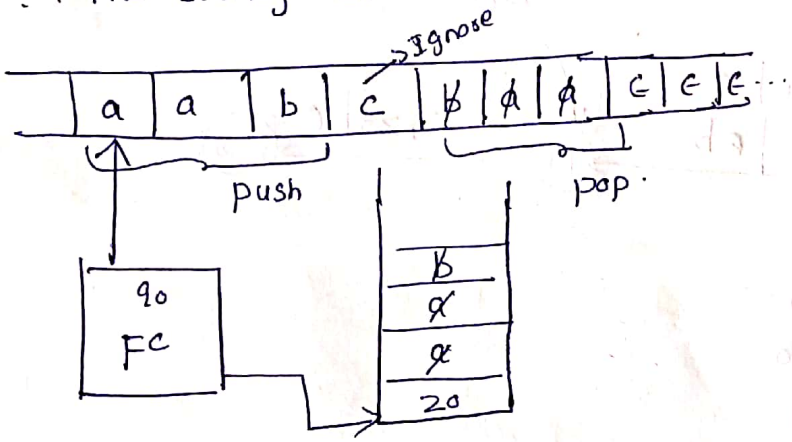
$L = \{ wc\,w^R \mid w \in (a+b)^* \}$

$w = \{ \epsilon, a, b, aa, ab, bb, aba, bba, aab, \ldots \}$

Given String $w = aab$

$w^R = baa$

$\therefore$ The String $wcw^R = aabcbaa$



Transition Function:

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ — push, Stack, push, Top

$\delta(q_0, b, Z_0) = (q_0, bZ_0)$ } Initial condition.

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

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

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

$\delta(q_0, b, b) = (q_0, bb)$ } Recursive condition pushed

$\delta(q_0, c, b) = (q_1, b)$ → Ignore Top of the stack.

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

$\delta(q_1, b) =$

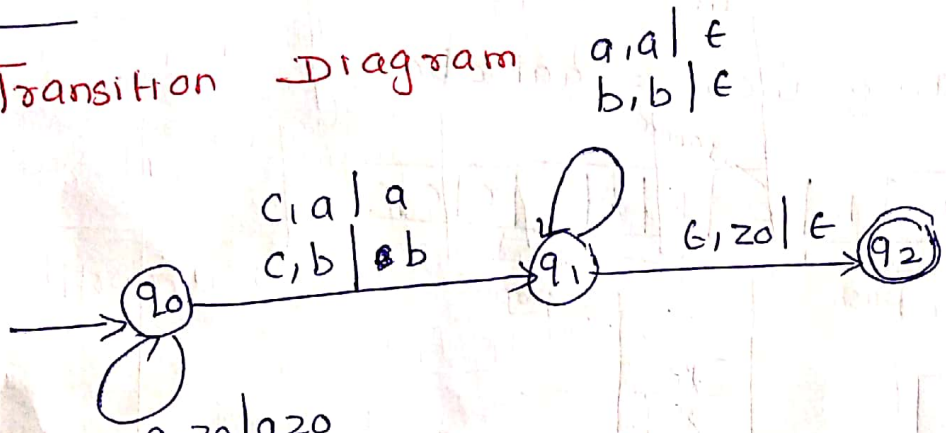
$\delta(q_1, a) =$

$\delta(q_1, b, b) = (q_1, \epsilon)$ } Recursive condition

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

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

Transition Diagram



Instantaneous Description (ID)

The given String 'aabcbaa' accepted or not

Empty stack

$(q_0, \underline{a}abcbaa, z_0) \vdash (q_0, \underline{a}bcbaa, az_0)$

$\vdash (q_0, \underline{b}cbaa, baz_0)$

$\vdash (q_0, \underline{c}baa, baaz_0)$

$\vdash (q_1, \underline{b}aa, baaz_0)$

$\vdash (q_1, \underline{a}a, aaz_0)$

$\vdash (q_1, a, az_0)$

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

$\vdash (q_2, \epsilon)$ Accepted.

∴ The given String is accepted.

Dr I Jasmine

part B 4

Let Gr be the Grammar $S \rightarrow 0B \mid 1A$

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

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

Convert this grammar into PDA and check the string "00110101" is accepted or not.

Solution:

Step: 1

The given productions are exist in GNF form only, therefore the PDA can be constructed directly.

Step 2:

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

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

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

$\Gamma = \{S, A, B, z_0\}$

$F = \{q_2\}$

$q_0 \rightarrow$ Start State.

1. First the Start Symbol S is put on the stack.

$\delta(q_0, \epsilon, z_0) = \{(q_1, Sz_0)\}$

2. The productions $S \rightarrow 0B \mid 1A$ is simulated by PDA as per $A \rightarrow \alpha$ rule.

$\delta(q_1, \epsilon, S) = \{(q_1, 0B), (q_1, 1A)\}$

3. In a similar manner, the other productions are:

$\delta(q_1, \epsilon, A) = \{(q_1, 0) (q_1, 0S), (q_1, 1AA)$

$\delta(q_1, \epsilon, B) = \{(q_1, 1) (q_1, 1S), (q_1, 0BB)$

4. For each $a \in \Sigma$ the corresponding productions are:

$$\delta(q_1, 0, 0) = \delta(q_1, \epsilon)$$

$$\delta(q_1, 1, 1) = (q_1, \epsilon)$$

5. Then the end of the derivations is identified by the stack symbol.

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

Check whether the given string "00110101" is accepted or not.

$(q, 00110101, S) \vdash (q, 00110101, \not{B})$ $S \to 0B$

$\vdash (q, 0110101, B)$ $B \to 0BB$

$\vdash (q, 110101, \not{B}BB)$ $B \to 1SB$

$\vdash (q, 10101, \not{S}BB)$ $S \to 1A$

$\vdash (q, 0101, \not{A}BB)$ $A \to 1AA$

$\vdash (q, 101, \not{A}ABB)$ $A \to 0$

$\vdash (q, 01, \not{A}BB)$ $A \to 0$

$\vdash (q, 1, \not{B}B)$ $B \to 1$

$\vdash (q, 1, \not{B})$

$\vdash (q, \epsilon)$ Accepted.

∴ The given string is accepted.

part B
9.

Construct the grammar for the following PDA.

$M = (\{q_0, q_1\}, \{0,1\}, \{X, z_0\}, \delta, q_0, z_0, \phi)$ and where δ is given by.

$\delta(q_0, 0, z_0) = \{(q_0, Xz_0)\}$
$\delta(q_0, 0, X) = \{(q_0, XX)\}$
$\delta(q_0, 1, X) = \{(q_1, \epsilon)\}$
$\delta(q_1, 1, X) = \{(q_1, \epsilon)\}$
$\delta(q_1, \epsilon, X) = \{q_1, \epsilon)\}$
$\delta(q_1, \epsilon z_0) = \{(q_1, \epsilon)\}$

Solution:

Step: 1

$T = \{0,1\}$

$V = \{S, [q_0, X, q_0], [q_0, X, q_1], [q_0, z_0, q_0], [q_0, z_0, q_1], [q_1, z_0, q_0], [q_1, z_0, q_1], [q_1, X, q_0], [q_1, X, q_1]\}$

Here V is a start symbol, whose first & third elements are states and the second element is a push down symbol.

Step: 2

The production in P' are:

R1: S - productions are given by $S \to [q_0, z_0, q]$ for every q in Q.

The productions for S

P1: $S \to [q_0, z_0, q_0]$

P2: $S \to [q_0, z_0, q_1]$

where q_0 is the start state and z_0 is the initial stack symbol.

Step: 3

R_2: Each move, not erasing a pushdown Symbol, given by. for each state (q_0, q_1)

$\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$

For q_0

$P_3: [(q_0, Z_0, q_0] \rightarrow 0[q_0, X, q_0][q_0, Z_0, q_0]$

$P_4: [q_0, Z_0, q_0] \rightarrow 0[q_0, X, q_1][q_1, Z_0, q_0]$

For q_1

$P_5: [q_0, Z_0, q_1] \rightarrow 0[q_0, X, q_0][q_0, Z_0, q_1]$

$P_6: [q_0, Z_0, q_1] \rightarrow 0[q_0, X, q_1][q_1, Z_0, q_1]$

Similarly:

$\delta(q_0, 0, X) = \{(q_0, XX)\}$ yields.

for q_0

$P_7: (q_0, X, q_0] = 0[q_0, X, q_0][q_0, X, q_0]$

$P_8: (q_0, X, q_0] = 0[q_0, X, q_1][q_1, X, q_0]$

for q_1

$P_9: (q_0, X, q_1] = 0[q_0, X, q_0][q_0, X, q_1]$

$P_{10}: (q_0, X, q_1] = 0[q_0, X, q_1][q_1, X, q_1]$

Step4: R_3: Each move erasing pushdown Symbol given by $(q', \epsilon) \in \delta(q, a, Z)$ produce. the production $[q, Z, q'] = q$

$\therefore$ P/N $\delta(q_0, 1, X) = \{(q_1, \epsilon)\}$ gives

$P_{11}: [q_0, X, q_1] \rightarrow 1$

Similarly $\delta(q_1, 1, X) = \{(q_1, \epsilon)\}$ gives

$P_{12}: [q_1, X, q_1] \rightarrow 1$

Similarly. $\delta(q_1, \epsilon, X) = \{(q_1, \epsilon)\}$

$P_{15}: [q_1, X, q_1] \rightarrow \epsilon$

Similarly $\delta(q_1, \epsilon, z_0) = \{(q_1, \epsilon\}$

P16: $[q_1, z_0, q_1] \rightarrow \epsilon$

After analyzing all the productions, it is clear that these are no productions for the variables.

$[q_1, X, q_0]$, $[q_1, z_0, q_0]$

And the variables $[q_0, X, q_0]$ $[q_1, z_0, q_0]$ have $[q_1, X, q_0]$ or $[q_1, z_0, q_0]$ on the right, which do not derive any terminal string. Like so for the variables $[q_0, X, q_0]$, $[q_1, z_0, q_0]$.

∴ Deleting all the productions which involves these variables.

∴ The final productions are.

$S \rightarrow [q_0, z_0, q_1]$

$[q_0, z_0, q_1] \rightarrow 0\,[q_0, X, q_1]\,[q_1, z_0, q_1]$

$[q_1, X, q_1] \rightarrow 0\,[q_0, X, q_1]\,[q_1, X, q_1]$

$[q_0, X, q_1] \rightarrow 1$

$[q_1, z_0, q_1] \rightarrow \epsilon$

$[q_1, X, q_1] \rightarrow \epsilon$

$[q_1, X, q_1] \rightarrow 1$

Dr I Jasmine