UNIT III
CONTEXT-FREE GRAMMAR AND LANGUAGES

Context-Free Grammar (CFG) – Parse Trees – Ambiguity in grammars and languages – Definition of the Pushdown automata – Languages of a Pushdown Automata – Equivalence of Pushdown automata and CFG, Deterministic Pushdown Automata.

3.1. CONTEXT FREE GRAMMAR(CFG)

3.1.1. Definition:

A context free grammar is a finite set of variables each of which represents a language. The languages represented by the variable are described recursively in terms of each other and primitive symbols called terminals. The rules relating the variables are called production. A Context Free Grammar(CFG) is denoted by,

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

Where, V – Variables

T – Terminals

P – Finite set of Productions of the form $A \rightarrow \alpha$ where, A is a variable and α is a string of symbols.

S – Start symbol.

A CFG is represented in Backus-Naur Form(BNF). For example consider the grammar,

<expression> → <expression> + <expression>
<expression> → <expression> * <expression>
<expression> → (<expression>)
<expression> → id

Here <expression> is the variable and the terminals are +, *, (,), id. The first two productions say that an expression can be composed of two expressions connected by an addition or multiplication sign. The third production says that an expression may be another expression surrounded by paranthesis. The last says a single operand is an expression.

Ex.1:

A CFG, G = (V, T, P, S) whose productions are given by,

$A \rightarrow Ba$

$B \rightarrow b$

Soln:

$A \rightarrow Ba$

$\rightarrow ba$ $\quad \therefore$ which produces the string ba.

3.1.2. <u>Derivations Using a Grammar:</u>

While inferring whether the given input string belongs to the given CFG, we can have two approaches,

- Using rules from body of head.
- Using rules from head to body.

First approach is called by the **name recursive inference**. Here we take strings from each variables, concatenate them in proper order and infer that the resulting strings is in the language of the variable in the head.

Another approach is called **derivation.** Here we use the productions from head to body. We expand the start symbol using one of its productions till it reached the given string.

<u>PROBLEMS:</u>

Ex.1:

Obtain the derivation for L, with production for the string 01c10.

$E \rightarrow c$

$E \rightarrow 0E0$

$E \rightarrow 1E1$

Soln:

$E \rightarrow 0E0$

$\rightarrow 01E10$ $\quad [\ E \rightarrow 1E1\ ]$

$\rightarrow 01c10$ $\quad [\ E \rightarrow c\ ]$

Ex.2:

Obtain the derivation for L, with production for the strings aaababbb and abbaab.

$S \rightarrow aSb \mid aAb$

$A \rightarrow bAa \mid ba$

Soln:

(i) $S \rightarrow aSb$
$\rightarrow aaSbb$ [$S \rightarrow aSb$]
$\rightarrow aaaAbbb$ [$S \rightarrow aAb$]
$\rightarrow aaababbb$ [$A \rightarrow ba$]

(ii) $S \rightarrow aAb$
$\rightarrow abAab$ [$A \rightarrow bAa$]
$\rightarrow abbaab$ [$A \rightarrow ba$]

3.1.3. Leftmost and Rightmost Derivations:

Leftmost Derivation:

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

Ex.1:

Let G = (V, T, P, S), V = {E}, T = {+, *, id}, S = E, P is given by,
$E \rightarrow E + E \mid E * E \mid id$
Construct leftmost derivation for id+id*id.

Soln:

$E \overset{lm}{\Rightarrow} E+E$
$\Rightarrow$ id+E [E → id]
$\Rightarrow$ id+E*E [E → E*E]
$\Rightarrow$ id+id*E [E → id]
$\Rightarrow$ id+id*id [E → id]

Rightmost Derivation:

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

Ex.1:

Let G = (V, T, P, S), V = {E}, T = {+, *, id}, S = E, P is given by,
$E \rightarrow E + E \mid E * E \mid id$
Construct rightmost derivation for id+id*id.

Soln:

$$E \overset{rm}{\Rightarrow} E*E$$
$$\Rightarrow E*id \quad [E \rightarrow id]$$
$$\Rightarrow E+E*id \quad [E \rightarrow E+E]$$
$$\Rightarrow E+id*id \quad [E \rightarrow id]$$
$$\Rightarrow id+id*id \quad [E \rightarrow id]$$

3.2. PARSE TREES or (Derivation Trees)

The derivations can be represented by trees using "parse trees".

3.2.1. Constructing Parse Trees:

Let G = (V, T, P, S) be a grammar. The parse trees for 'G' is a trees with following conditions.

1. Each interior node is labeled by a variable in V.
2. Each leaf is labeled by either a variable, a terminal or ε.
3. If an interior node is labeled A, and its children are labeled $X_1, X_2, \ldots\ldots X_k$ respectively from left, then $A \rightarrow X_1, X_2, \ldots\ldots\ldots\ldots X_k$ is a production in P.
4. If $A \rightarrow \varepsilon$ then A is considered to be the label.

Ex.1:

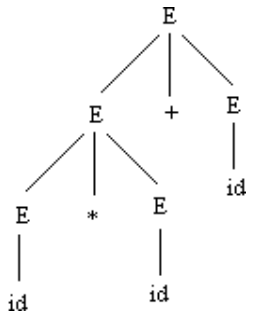
Construct a parse tree for the grammar, $E \rightarrow E + E \mid E * E \mid (E) \mid id$, for the string id*id+id

Soln:

Derivation:

$$E \Rightarrow E+E$$
$$\Rightarrow E*E+E \quad [E \Rightarrow E*E]$$
$$\Rightarrow id*E+E \quad [E \Rightarrow id]$$
$$\Rightarrow id*id+E \quad [E \Rightarrow id]$$
$$\Rightarrow id*id+id \quad [E \Rightarrow id]$$

Parse Tree:



Ex.2:

Construct a parse tree for the grammar, $P \rightarrow \varepsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$, A parse tree showing the derivation of a string 0110.

Soln:

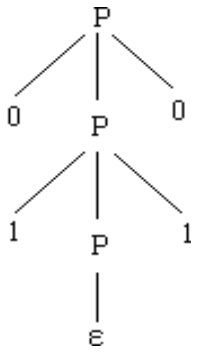
Derivation: Parse Tree:

$P \Rightarrow 0P0$

$\Rightarrow 01P10$ $[P \Rightarrow 1P1]$

$\Rightarrow 01\varepsilon10$ $[P \Rightarrow \varepsilon]$

$\Rightarrow 0110$



3.2.2. Yield of a Parse Tree:

The string obtained by concatenating the leaves of a parse tree from the left is called yield of a parse tree. The yield is always derived from the root. The root is the start symbol.

Ex.1:

For the grammar G is defined by the production,

$S \rightarrow A \mid B$

$A \rightarrow 0A \mid \varepsilon$

$B \rightarrow 0B \mid 1B \mid \varepsilon$

Find the parse tree for the yields (i) 1001 (ii) 00101

Soln:

(i) 1001

Derivation:

$S \Rightarrow B$

$\Rightarrow 1B$ $[B \rightarrow 1B]$

$\Rightarrow 10B$ $[B \rightarrow 0B]$

$\Rightarrow 100B$ $[B \rightarrow 0B]$

$\Rightarrow 1001B$ $[B \rightarrow 1B]$

$\Rightarrow 1001$ $[B \rightarrow \varepsilon]$

(ii) 00101

Derivation:

$S \Rightarrow B$

$\Rightarrow 0B$ $[B \rightarrow 0B]$

$\Rightarrow 00B$ $[B \rightarrow 0B]$

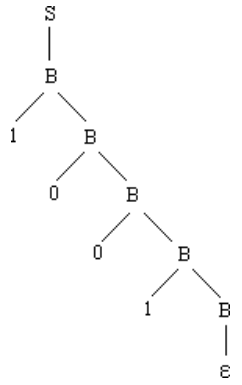
$\Rightarrow 001B$ $[B \rightarrow 1B]$

$\Rightarrow 0010B$ $[B \rightarrow 0B]$

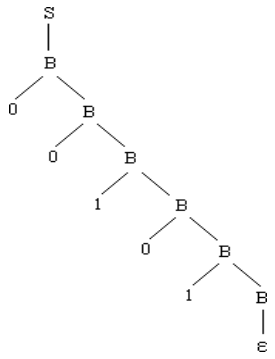
$\Rightarrow 00101B$ $[B \rightarrow 1B]$

$\Rightarrow 00101$ $[B \rightarrow \varepsilon]$

Parse Tree:



Parse Tree:



3.2.3. Relationship Between the Derivation Trees and Derivation:

Theorem:

Let G = (V, T, P, S) be a CFG. Then $S \overset{*}{\Rightarrow} \alpha$ if and only if there is a derivation tree in grammar G with yield α.

Proof:

Suppose there is a parse tree with root S and yield α, then there is a leftmost derivation,

$$S \overset{*}{\underset{lm}{\Rightarrow}} \alpha \text{ in G.}$$

To prove, $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$ in G. Let us prove this theorem by induction on height of the tree.

Basis:

If the height of parse tree is '1' then the tree must be of the form given in figure below with root 'S' and yield 'α'.

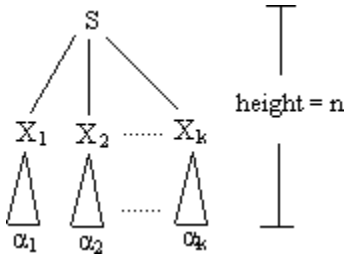
This means that 'S' has only leaves and no subtrees. This is possible only with the production.

$$S \Rightarrow \alpha \text{ in G}$$

$S \underset{lm}{\Rightarrow} \alpha$ is an one step leftmost derivation.

Induction:

If the height of the parse tree is in the parse tree must look like in.



$\alpha = \alpha_1\alpha_2 \ldots\ldots\ldots \alpha_k$. Where $X_1, X_2, \ldots.. X_k$ are all the subtrees of S.

Assume there exist a leftmost derivation $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$ for every parse tree of height less than 'n'. Consider a parse tree of height 'n'. Let the leftmost derivation be,

$$S \overset{*}{\underset{lm}{\Rightarrow}} X_1X_2, \ldots.. X_k$$

The X_i's may be either terminals or variables.

(i) If X_i is a terminal then $X_i = \alpha_i$

(ii) If X_i is a variable then it must be the root of some sub-tree with yield α_i of height less than 'n'.

By applying inductive hypothesis, there is a leftmost derivation,

$$X_i \overset{*}{\underset{lm}{\Rightarrow}} \alpha_i$$

$$S \overset{*}{\underset{lm}{\Rightarrow}} X_1X_2, \ldots.. X_k$$

If X_i is a terminal then no change.

$$S \overset{*}{\underset{lm}{\Rightarrow}} \alpha_1\alpha_2 \ldots\ldots\ldots \alpha_i X_{i+1} \ldots.. X_k$$

If X_i is a variable then derive the string X_i to α_i as,

$$X_i \overset{}{\underset{lm}{\Rightarrow}} w_1 \Rightarrow w_2 \Rightarrow \ldots\ldots \Rightarrow \alpha_i$$

Therefore,

$$S \overset{*}{\underset{lm}{\Rightarrow}} \alpha_1\alpha_2 \ldots\ldots\ldots \alpha_{i-1} X_i X_{i+1} \ldots.. X_k$$

$$\overset{*}{\underset{lm}{\Rightarrow}} \alpha_1\alpha_2 \ldots\ldots\ldots \alpha_{i-1} w_1 X_{i+1} \ldots.. X_k$$

$$\overset{*}{\underset{lm}{\Rightarrow}} \alpha_1\alpha_2 \ldots\ldots\ldots \alpha_{i-1} w_2 X_{i+1} \ldots.. X_k$$

.
.
.

$\underset{lm}{\overset{*}{\Rightarrow}}$ $\alpha_1\alpha_2 \ldots\ldots\ldots \alpha_{i-1}\, \alpha_i\, X_{i+1} \ldots\ldots X_k$

By repeating the process we can get,

$S \underset{lm}{\overset{*}{\Rightarrow}} \alpha_1\alpha_2 \ldots\ldots\ldots \alpha_k$ [$\therefore$ $\alpha = \alpha_1\alpha_2 \ldots\ldots\ldots \alpha_k$]

Thus proved.

3.3. AMBIGUITY IN GRAMMARS AND LANGUAGES

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

3.3.1. Ambiguous Grammars:

A CFG, G = (V, T, P, S) is said to be ambiguous, if there is atleast one string 'w' has two different parse trees.

PROBLEMS:

Ex.1:

Construct ambiguous grammar for the grammar, E → E + E | E * E | (E) | id, and generate a string id+id*id.

Soln:

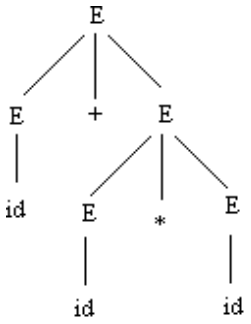
Derivation1:

E ⇒ E+E
⇒ id+E [E → id]
⇒ id+E*E [E → E*E]
⇒ id+id*E [E → id]
⇒ id+id*id [E → id]

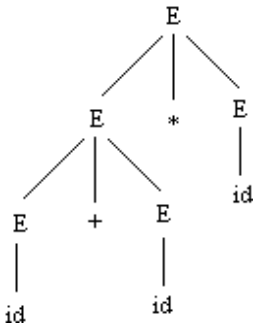
Derivation2:

E ⇒ E*E
⇒ E+E*E [E → E+E]
⇒ id*E+E [E → id]
⇒ id*id+E [E → id]
⇒ id*id+id [E → id]

Parse Tree1:



Parse Tree2:



Ex.2:

Construct ambiguous grammar for the grammar,

E → I | E+E | E*E | (E)

I → a | b | Ia | Ib | I0 | I1

and generate a string a+a*a.

Soln:

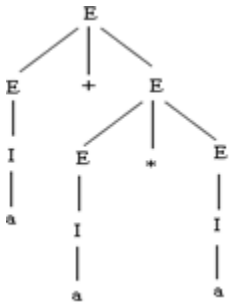
Derivation1:

E ⇒ E+E
⇒ I+E [E → I]
⇒ a+E [I → a]
⇒ a+E*E [E → E*E]
⇒ a+I*E [E → I]
⇒ a+a*E [I → a]
⇒ a+a*I [E → I]
⇒ a+a*a [I → a]

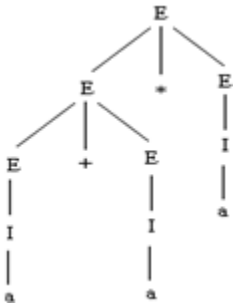
Derivation2:

E ⇒ E*E
⇒ E*I [E → I]
⇒ E*a [I → a]
⇒ E+E*a [E → E+E]
⇒ E+I*a [E → I]
⇒ E+a*a [I → a]
⇒ I+a*a [E → I]
⇒ a+a*a [I → a]

Parse Tree1:



Parse Tree2:



3.3.2. Unambiguous:

If each string has atmost one parse tree in the grammar, then the grammar is unambiguous.

3.3.4. Leftmost Derivations as a way to Express Ambiguity:

A grammar is said to be ambiguous, if it has more than one leftmost derivation.

Ex:

Construct ambiguous grammar for the grammar,

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

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

and generate two leftmost derivation for a string a+a*a.

Soln:

Derivation1:

$E \overset{rm}{\Rightarrow} E+E$

$\Rightarrow I+E \quad [E \rightarrow I]$

$\Rightarrow a+E \quad [I \rightarrow a]$

$\Rightarrow a+E*E \quad [E \rightarrow E*E]$

$\Rightarrow a+I*E \quad [E \rightarrow I]$

$\Rightarrow a+a*E \quad [I \rightarrow a]$

$\Rightarrow a+a*I \quad [E \rightarrow I]$

$\Rightarrow a+a*a \quad [I \rightarrow a]$

Derivation2:

$E \overset{rm}{\Rightarrow} E*E$

$\Rightarrow E+E*E \quad [E \rightarrow E+E]$

$\Rightarrow I+E*E \quad [E \rightarrow I]$

$\Rightarrow a+E*E \quad [I \rightarrow a]$

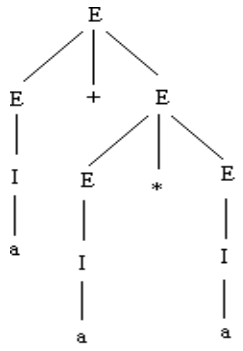
$\Rightarrow a+I*E \quad [E \rightarrow I]$

$\Rightarrow a+a*E \quad [I \rightarrow a]$

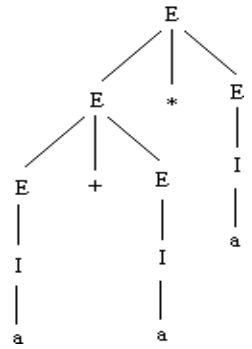
$\Rightarrow a+a*I \quad [E \rightarrow I]$

$\Rightarrow a+a*a \quad [I \rightarrow a]$

Parse Tree1:



Parse Tree2:



3.3.5. Inherent Ambiguity:

A CFL L is said to be inherently ambiguous, if every grammar for the language must be ambiguous.

Ex:

Show that the language is inherent ambiguous L = $\{a^n b^n c^m d^m \mid n\geq 1, m\geq 1\} \cup \{ a^n b^m c^m d^n \mid n\geq 1, m\geq 1\}$ then the production P is given by,

$S \rightarrow AB \mid C$

$A \rightarrow aAb \mid ab$ $\qquad$ $C \rightarrow aCd \mid aDd$

$B \rightarrow cBd \mid cd$ $\qquad$ $D \rightarrow bDc \mid bc$

Soln:

L is a context free language. It separate sets of productions to generate two kinds of strings in L. This grammar is ambiguous For ex, the string aabbccdd has two leftmost derivations.

Derivation1:

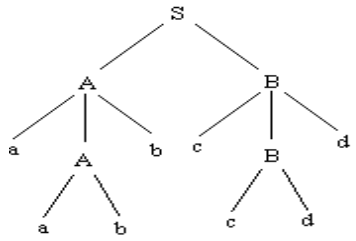
$S \overset{rm}{\Rightarrow} AB$
$\Rightarrow aAbB$
$\Rightarrow aabbB$
$\Rightarrow aabbcBd$
$\Rightarrow aabbccdd$

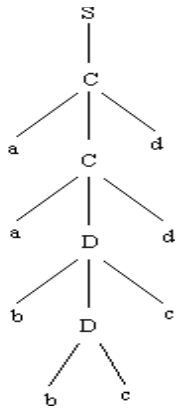
Derivation2:

$S \overset{rm}{\Rightarrow} C$
$\Rightarrow aCd$
$\Rightarrow aaDdd$
$\Rightarrow aabDcdd$
$\Rightarrow aabbccdd$

Parse Tree1:

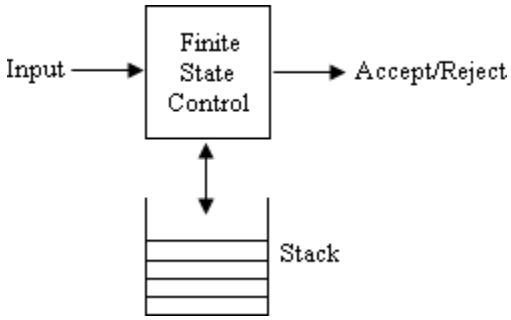
Parse Tree2:





3.4. DEFINITION OF THE PUSH DOWN AUTOMATA

A PushDown Automata(PDA) is essentially a finite automata with control of both an input tape and a stack on which it can store a string of stack symbols. With the help of a stack the pushdown automata can remember an infinite amount of information.



3.4.1. Model of PDA:

- The PDA consists of a finite set of states, a finite set of input symbols and a finite set of pushdown symbols.
- The finite control has control of both the input tape and pushdown store.
- In one transition of the PDA,
  - The control head reads the input symbol, then goto the new state.
  - Replaces the symbol at the top of the stack by any string.

3.4.2. Definition of PDA:

A PDA consists of seven tuples.

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

where, Q – A finite set of states.

Σ – A finite set of input symbols.

Γ – A finite set of stack symbols.

δ - The transition function. Formally, δ takes a argument a triple δ(q, a, X),

where, - 'q' is a state in Q

- 'a' is either an input symbol in Σ or a = ε.
- 'X' is a stack symbol, that is a member of Γ.

Q_0 – The start state.

Z_0 – The start symbol of the stack.

F – The set of accepting states or final states.

Ex:

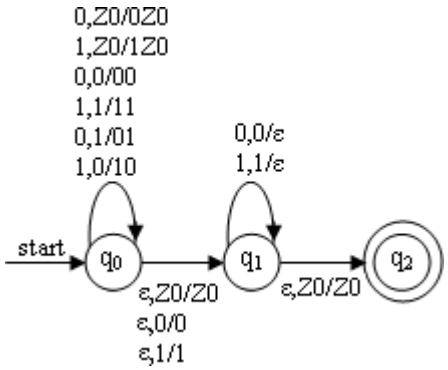
Mathematical model of a PDA for the language, L = {wwR | w is in (0+1)* }, then PDA for L can be described as, P = ({q_0, q_1, q_2}, {0, 1}, {0, 1, Z_0}, δ, q_0, Z_0, {q_2}), where δ is defined by the following rules;

1. $\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$ and $\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$. One of these rules applies initially, when we are in state q_0 and we see the start symbol Z_0 at the top of the stack. We read the first input, and push it onto the stack, leaving Z_0 below to mark the bottom.
2. $\delta(q_0, 0, 0) = \{(q_0, 00)\}$, $\delta(q_0, 0, 1) = \{(q_0, 01)\}$, $\delta(q_0, 1, 0) = \{(q_0, 10)\}$ and $\delta(q_0, 1, 1) = \{(q_0, 11)\}$. These four similar rules allow us to stay in state q_0 and read inputs, pushing each onto the top of the stack and leaving the pervious top stack symbol alone.
3. $\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\}$, $\delta(q_0, \varepsilon, 0) = \{(q_1, 0)\}$, and $\delta(q_0, \varepsilon, 1) = \{(q_1, 1)\}$. These three rules allow P to go from state q_0 to state q_1 spontaneously (on ε input), leaving intact whatever symbol is at the top of the stack.
4. $\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$, and $\delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$. Now in state q1we can match input symbols against the top symbols on the stack, and pop when the symbols match.
5. $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$. Finally, if we expose the bottom-of-stack marker Z_0 and we are in state q_1, then we have found an input of the form ww^R. We go to state q_2 and accept.

3.4.3. A Graphical Notation for DFA:

Sometimes, a diagram generalizing the transition diagram of a finite automaton, will make aspects of the behavior of a given PDA clearer. A transition diagram for PDA indicates,

(a) The nodes correspond to the states of the PDA.
(b) An arrow label start indicates, the start state and doubly circled states are accepting, as for finite automata.
(c) An arc labeled a, X/α from state q to state 'p' means that δ(q, a, X) contains the pair (p,α).



3.4.4. Instantaneous Descriptions(ID) of a PDA:

The ID is defined as a triple (q, w, γ), where,

q – Current state
w – String of input symbols
γ – String of stack symbols

Let P = (Q, Σ, Γ, δ, q_0, Z_0, F) be a PDA. Suppose δ(q, a, X) contains (p,α). Then for all strings 'w' in Σ* and β in Γ*; (q, aw, β) } (p, w, αβ).

PROBLEMS:

Ex.1:

Construct PDA on the input strings 001010c010100, 001010c011100. PDA can be described as, P = ({q_1, q_2}, {0, 1, c}, {R, B, G}, δ, q_1, R, {q_2}), where δ is,

$\delta(q_1, 0, R) = (q_1, BR)$	$\delta(q_1, c, R) = (q_2, R)$
$\delta(q_1, 1, R) = (q_1, GR)$	$\delta(q_1, c, B) = (q_2, B)$
$\delta(q_1, 0, B) = (q_1, BB)$	$\delta(q_1, c, G) = (q_2, G)$
$\delta(q_1, 1, B) = (q_1, GB)$	$\delta(q_2, 0, B) = (q_2, \varepsilon)$
$\delta(q_1, 0, G) = (q_1, BG)$	$\delta(q_2, 1, G) = (q_2, \varepsilon)$

$\delta(q_1, 1, G) = (q_1, GG)$ $\delta(q_2, \varepsilon, R) = (q_2, \varepsilon)$

check whether the string is accepted or not?

Soln:

- w1 = 001010c010100

$(q_1, 001010c010100, R) \vdash (q_1, 01010c010100, BR)$
$\vdash (q_1, 1010c010100, BBR)$
$\vdash (q_1, 010c010100, GBBR)$
$\vdash (q_1, 10c010100, BGBBR)$
$\vdash (q_1, 0c010100, GBGBBR)$
$\vdash (q_1, c010100, BGBGBBR)$
$\vdash (q_2, 010100, BGBGBBR)$
$\vdash (q_2, 10100, GBGBBR)$
$\vdash (q_2, 0100, BGBBR)$
$\vdash (q_2, 100, GBBR)$
$\vdash (q_2, 00, BBR)$
$\vdash (q_2, 0, BR)$
$\vdash (q_2, \varepsilon, R)$
$\vdash (q_2, \varepsilon, \varepsilon)$

$\therefore$ The string is accepted.

- w2 = 001010c011100

$(q_1, 001010c011100, R) \vdash (q_1, 01010c011100, BR)$
$\vdash (q_1, 1010c011100, BBR)$
$\vdash (q_1, 010c011100, GBBR)$
$\vdash (q_1, 10c011100, BGBBR)$
$\vdash (q_1, 0c011100, GBGBBR)$
$\vdash (q_1, c011100, BGBGBBR)$
$\vdash (q_2, 011100, BGBGBBR)$
$\vdash (q_2, 11100, GBGBBR)$
$\vdash (q_2, 1100, BGBBR)$

$\therefore$ There is no transition for $(q_2, 1, B)$. So the string is not accepted.

3.5. LANGUAGES OF A PUSH DOWN AUTOMATA

There are two ways to accept a string a PDA,

(a) Accept by final state that is, reach the final state from the start state.
(b) Accept by an empty stack that is, after consuming input, the stack is empty and current state could be a final state or non-final state.

Both methods are equivalent. One method can be converted to another method and vice versa.

3.5.1. Acceptance by final state:

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. The languages accepted by a final state is defined as,

$$L(M) = \{w \mid (q_0, w, Z_0)\}^* (q, \varepsilon, \alpha), \text{ where } q \in F \text{ and } \alpha \in \Gamma^*.$$

It means that, from the current station q_0 after scanning the input string 'w', the PDA enters into a final state 'q' leaving the input tape empty. Here contents of the stack is irrelevant.

3.5.2. Acceptance by Empty Stack:

For each PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, language accepted by empty stack can be defined as,

$$N(P) = \{w \mid (q_0, w, Z_0)\}^* (q, \varepsilon, \varepsilon), \text{ where } q \in Q.$$

It means that when the string 'w' is accepted by an empty stack, the final state is irrelevant, the input tape should be empty and stack also should be empty.

3.5.3. From Empty Stack to Final State:

Theorem:

If $L = N(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$, then there is a PDA P_F such that $L = L(P_F)$.

Proof:

Initially, change the stack content from Z_0 to Z_0X_0. So consider a new stack start symbol X_0 for the PDA P_F. Also need a new start state P_0, which is the initial state of P_F. It is to push Z_0 the start symbol of P_N, onto the top of the stack and enter state q_0. Finally, we need another new state $P_{f.}$, that is the accepting state of P_F.

The specification of P_F is as follows:

$$P = (Q \cup \{P_0, P_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, P_0, X_0, \{P_f\})$$

where δ_F is defined by,

1. $\delta_F(P_0, \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$. In its start state, P_F makes a spontaneous transition to the start state of P_N, pushing its start symbol Z_0 onto the stack.
2. For all states 'q' in Q, inputs 'a' in Σ or $a = \varepsilon$, and stack symbols Y in Γ, $\delta_F(q, a, Y) = \delta_N(q, a, Y)$.
3. $\delta_F(q, \varepsilon, X_0) = (P_f, \varepsilon)$ for every state 'q' in Q.

We must show that 'w' is in $L(P_F)$ if and only if 'w' is in $N(P_N)$. The moves of the PDA P_F to accept a string 'w' can be written as,

$$(P_0, w, X_0) \} \ \overset{*}{P_F} \ (q_0, w, Z_0X_0) \}^* \ \overset{*}{P_F} \ (q_0, \varepsilon, X_0) \} \ \overset{*}{P_F} \ (P_f, \varepsilon, \varepsilon).$$

Thus P_F accepts 'w' by final state.

3.5.4. From Final State to Empty Stack:

Theorem:

Let L be $L(P_F)$ for some PDA, $P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Then there is a PDA P_N such that $L = N(P_N)$.

Proof:

Initially, change the stack content from Z_0 to Z_0X_0. So we also need a start state P_0, and final state P, which is the start and final of P_N.

The specification of P_N is as follows:

$$P_N = (Q \cup \{P_0, P\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, P_0, X_0)$$

where δ_N is defined by,

1. $\delta_N(P_0, \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$ to change the stack content initially.
2. $\delta_N(q, a, Y) = \delta_F(q, a, Y)$, for all states 'q' in Q, inputs 'a' in Σ or $a = \varepsilon$, and stack symbols Y in Γ.
3. $\delta_N(q, \varepsilon, Y) = (P, \varepsilon)$, for all accepting states 'q' in F and stack symbols Y in Γ or $Y = X_0$.
4. $\delta_N(P, \varepsilon, Y) = (P, \varepsilon)$, for all stack symbols Y in Γ or $Y = X_0$, to pop the remaining stack contents.

Suppose $(Q_0, w, Z_0) \}^* \ \overset{*}{P_F} \ (q, \varepsilon, \alpha)$ for some accepting state 'q' and stack string α. Then P_N can do the following:

$$(P_0, w, X_0) \} \ \overset{*}{P_N} \ (q_0, w, Z_0X_0) \}^* \ \overset{*}{P_N} \ (q, \varepsilon, \alpha X_0) \}^* \ \overset{*}{P_N} \ (P, \varepsilon, \varepsilon).$$

PROBLEMS:

Ex.1:

Construct a PDA that accepts the given language, $L = \{x^m y^n \mid n<m\}$.

Soln:

Language L accepted by the strings are, L = {xxy, xxxy, xxxyy, xxxxyy,……}

First find the grammar for that language. The grammar for the language can be,

$S \rightarrow xSy \mid xS \mid x$

The corresponding PDA for the above grammar is,

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

where, $Q = \{q\}$

$\Sigma = \{x, y\}$

$\Gamma = \{S, x, y\}$

$q_0 = \{q\}$

$Z_0 = \{S\}$

$F = \phi$

and δ is defined as,

$\delta(q, \varepsilon, S) = \{(q, xSy), (q, xS), (q, x)\}$

$\delta(q, x, x) = (q, \varepsilon)$

$\delta(q, y, y) = (q, \varepsilon)$

To prove the string xxxyy is accepted by PDA,

(q, xxxyy, S) } (q, xxxyy, xSy) } (q, xxyy, Sy) } (q, xxyy, xSyy) } (q, xyy, Syy)

} (q, xyy, xyy) } (q, yy, yy) } (q, y, y) } (q, ε, ε)

Hence the string is accepted.

Ex.2:

Construct a PDA that accepts the given language, $L = \{0^n1^n \mid n \geq 1\}$.

Soln:

Language L accepted by the strings are, L = {01, 0011, 000111, 00001111,……}

First find the grammar for that language. The grammar for the language can be,

$S \rightarrow 0S1 \mid 0A1$

$A \rightarrow 01 \mid \varepsilon$

The corresponding PDA for the above grammar is,

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

where, $Q = \{q\}$

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

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

$q_0 = \{q\}$

$Z_0 = \{S\}$

$F = \phi$

and δ is defined as,

$\delta(q, \varepsilon, S) = \{(q, 0S1), (q, 0A1)\}$
$\delta(q, \varepsilon, A) = \{(q, 01), (q, \varepsilon)\}$
$\delta(q, 0, 0) = (q, \varepsilon)$
$\delta(q, 1, 1) = (q, \varepsilon)$

To prove the string 000111 is accepted by PDA,

(q, 000111, S) } (q, 000111, 0S1) } (q, 00111, S1) } (q, 00111, 0S11) } (q, 0111, S11)
} (q, 0111, 0A111)} (q, 111, A111)} (q, 111, 111)} (q, 11, 11) } (q, 1, 1)
} (q, ε, ε)

Hence the string is accepted.

3.6. EQUIVALENCE OF PUSHDOWN AUTOMATA AND CFG

3.6.1 From Grammars to PushDown Automata:

It is possible to convert a CFG to PDA and vice versa.

Input:

Context Free Grammar 'G'.

Output:

PDA – P that simulates the leftmost derivations of G. Stack contains all the symbols (variables as well as terminals) of CFG.

Let G = (V, T, P, S) be a CFG. The PDA which accepts L(G) is given by,

$P = (\{q\}, T, V \cup T, \delta, q, S, \phi)$ where δ is defined by,

1. For each variable 'A' include a transition $\delta(q, \varepsilon, A) = (q, b)$ such that $A \rightarrow b$ is a production of P.
2. For each terminal 'a' include a transition $\delta(q, a, a) = (q, \varepsilon)$.

PROBLEMS:

Ex.1:

Construct a PDA that accepts the language generated by the grammar,

$S \rightarrow aSbb \mid abb$

Soln:

PDA – P is defined as follows:

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

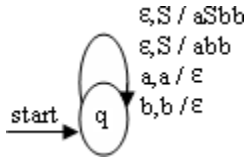
Where, $Q = \{q\}$

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

Γ = {S, a, b}
q_0 = {q}
Z_0 = {S}
F = ϕ

and δ is defined as,

$\delta(q, \varepsilon, S) = \{(q, aSbb), (q, abb)\}$
$\delta(q, a, a) = (q, \varepsilon)$
$\delta(q, b, b) = (q, \varepsilon)$



Ex.2:

Construct a PDA equivalent to the CFG,

$S \rightarrow aABB \mid aAA$
$A \rightarrow aBB \mid a$
$B \rightarrow bBB \mid A$

Soln:

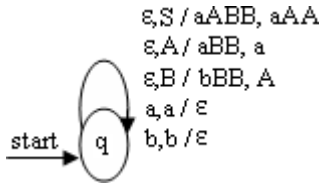
PDA – P is defined as follows:

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

Where, Q = {q}
Σ = {a, b}
Γ = {S, A, B, a, b}
q_0 = {q}
Z_0 = {S}
F = ϕ

and δ is defined as,

$\delta(q, \varepsilon, S) = \{(q, aABB), (q, aAA)\}$
$\delta(q, \varepsilon, A) = \{(q, aBB), (q, a)\}$
$\delta(q, \varepsilon, B) = \{(q, bBB), (q, A)\}$
$\delta(q, a, a) = (q, \varepsilon)$
$\delta(q, b, b) = (q, \varepsilon)$



Ex.3:

Construct a PDA equivalent to the CFG,

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

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Soln:

PDA – P is defined as follows:

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

Where, $Q = \{q\}$

$\Sigma = \{a, b, 0, 1, +, *, (,)\}$

$\Gamma = \{E, I, B, a, b, 0, 1, +, *, (,)\}$

$q_0 = \{q\}$

$Z_0 = \{E\}$

$F = \phi$

and δ is defined as,

$\delta(q, \varepsilon, E) = \{(q, I), (q, E+E), (q, E*E), (q, (E))\}$

$\delta(q, \varepsilon, I) = \{(q, a), (q, b), (q, 0), (q, 1), (q, +), (q, *), (q, ( ), (q,))\}$

$\delta(q, a, a) = (q, \varepsilon)$

$\delta(q, b, b) = (q, \varepsilon)$

$\delta(q, 0, 0) = (q, \varepsilon)$

$\delta(q, 1, 1) = (q, \varepsilon)$

$\delta(q, +, +) = (q, \varepsilon)$

$\delta(q, *, *) = (q, \varepsilon)$

$\delta(q, (, ( ) = (q, \varepsilon)$

$\delta(q,),)) = (q, \varepsilon)$

3.6.2. From PDA's to Grammars:

Theorem:

If L is N(M) for some PDA M, then L is a context free grammar.

Construction:

Let M = (Q, Σ, Γ, δ, q_0, Z_0, F) be the PDA . Let G = (V, T, P, S) be a CFG.

Where, - V is the set of objects of the form [q, A, p], 'q' and 'p' in Q and A in Γ.
- New symbol S.
- P is the set of productions.

The productions are,

(1) $S \rightarrow [q_0, Z_0, q]$, q in Q

(2) If $\delta(q, a, A) = (q_1, B_1B_2 \ldots\ldots B_n)$ then,
$[q, A, q_{m+1}] = a[q_1, B_1, q_2][q_2, B_2, q_3] \ldots\ldots [q_m, B_m, q_{m+1}]$, for each 'a' in $\Sigma \cup \{\varepsilon\}$ and A, $B_1B_2 \ldots. B_m$ in Γ.

Proof:

If m = 0; $\delta(q, a, A) = (q_1, \varepsilon)$

$[q, A, q_1] \rightarrow a$

Let 'x' be the input string, to show that, $[q, A, p] \overset{*}{\Rightarrow} x$, iff $(q, x, A) \}^* (p, \varepsilon, \varepsilon)$

We show by induction on 'i' that, if $(q, x, A) \}^i (p, \varepsilon, \varepsilon)$ then $[q, A, p] \overset{*}{\Rightarrow} x$

Basis: when i = 1,

$\delta(q, x, A) = (p, \varepsilon)$

here 'x' is a single input symbol. Thus $[q, A, p] \rightarrow x$ is a production of G.

Induction: when i > 1, let x = ay.

$(q, ay, A) \} (q1, ay, B_1B_2 \ldots\ldots B_n)$

The string 'y' can be written as $y = y_1y_2 \ldots y_n$, where y_j has the effect of popping B_j from the stack possibly after a long sequence of moves.

Let y_1 be the prefix of 'y' at the end of which the stack first becomes short as n-1 symbols. Let y_2 be the symbols of 'y' following y_1 such that at the end of y_2 the stack is a short as n-2 symbols and so on. That is,

$(q_1, y_1y_2 \ldots\ldots y_n, B_1B_2 \ldots\ldots B_n) \} (q_2, y_2y_3 \ldots\ldots y_n, B_2B_3 \ldots\ldots B_n) \} (q_3, y_3y_4 \ldots\ldots y_n, B_3B_4 \ldots\ldots B_n)$

There exist states $q_2, q_3, \ldots q_{n+1}$.

Where $q_{n+1} = p$

$(q_1, y_1, B_1) \} (q_2, \varepsilon, \varepsilon)$

$(q_2, y_2, B_2) \} (q_3, \varepsilon, \varepsilon)$

$\vdots \qquad \vdots$

$(q_j, y_j, B_j) \} (q_j, \varepsilon, \varepsilon)$

In CFG, $[q_j, B_j, q_j+1] \overset{*}{\Rightarrow} y_j$

The original move,

$(q, ay, A) \} (q_1, y, B_1B_2\ldots\ldots B_n)$

$(q, ay_1y_2 \ldots\ldots y_n, A) \} (q_1, y_1y_2\ldots\ldots y_n, B_1B_2\ldots\ldots\ldots\ldots B_n)$

CFG is,

$[q, A, p] \Rightarrow a[q_1, B_1, q_2] [q_2, B_2, q_3]\ldots\ldots\ldots\ldots [q_n, B_n, q_{n+1}]$

$[q, A, p] \overset{*}{\Rightarrow} ay_1y_2\ldots\ldots\ldots\ldots y_n$

$[q, A, p] \overset{*}{\Rightarrow} ay$

$[q, A, p] \Rightarrow x$ iff $(q, x, A) \}^* (p, \varepsilon, \varepsilon)$, where $q_{n+1} = p$.

3.6.2.1. Algorithm for getting production rules of CFG:

1. The start symbol production can be, $S \rightarrow [q_0, Z_0, q]$
   where, q indicates the next state.
   q_0 is a start state
   Z_0 is a stack symbol
   q and $q_0 \in Q$
2. If there exist a move of PDA, $\delta(q, a, Z) = \{(q', \varepsilon)\}$, then the production rule can be written as, $[q, Z, q'] \rightarrow a$
3. If there exist a move of PDA as, $\delta(q, a, Z) = \{(q_m, Z_1Z_2\ldots\ldots Z_n)\}$, then the production rule can be written as, $[q, Z, q_m] \rightarrow a[q_1, Z_1, q_2] [q_2, Z_2, q_3] [q_3, Z_3, q_4] \ldots\ldots [q_{m-1}, Z_{n-1}, q_m]$

PROBLEMS:

Ex.1:

Construct a CFG for the PDA, $P = (\{q_0, q_1\}, \{0, 1\}, \{S, A\}, \delta, q_0, S, \{q_1\})$, where δ is,

$\delta(q_0, 1, S) = \{(q_0, AS)\}$ $\quad\quad$ $\delta(q_0, 0, A) = \{(q_1, A)\}$

$\delta(q_0, \varepsilon, S) = \{(q_0, \varepsilon)\}$ $\quad\quad$ $\delta(q_1, 1, A) = \{(q_1, \varepsilon)\}$

$\delta(q_0, 1, A) = \{(q_0, AA)$ $\quad\quad$ $\delta(q_1, 0, S) = \{(q_0, S)\}$

Soln:

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

Where, V = {[q_0, S, q_0], [q_0, S, q_1], [q_1, S, q_0], [q_1, S, q_1], [q_0, A, q_0], [q_0, A, q_1],

[q_1, A, q_0], [q_1, A, q_1]}

T = {0, 1}

S = {S} [Start stack symbol]

To find production, P;

(1) Production for S,

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

$S \rightarrow [q_0, S, q_1]$ [q_0 – Start state, S – Initial stack symbol]

(2) $\delta(q_0, 1, S) = \{(q_0, AS)\}$ we get,

For q_0, $[q_0, S, q_0] \rightarrow 1[q_0, A, q_0] [q_0, S, q_0]$

$[q_0, S, q_0] \rightarrow 1[q_0, A, q_1] [q_1, S, q_0]$

For q_1, $[q_0, S, q_1] \rightarrow 1[q_0, A, q_0] [q_0, S, q_1]$

$[q_0, S, q_1] \rightarrow 1[q_0, A, q_1] [q_1, S, q_1]$

(3) $\delta(q_0, \varepsilon, S) = \{(q_0, \varepsilon)\}$

$[q_0, S, q_0] \rightarrow \varepsilon$

(4) $\delta(q_0, 1, A) = \{(q_0, AA)$

For q_0, $[q_0, A, q_0] \rightarrow 1[q_0, A, q_0] [q_0, A, q_0]$

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

For q_1, $[q_0, A, q_1] \rightarrow 1[q_0, A, q_0] [q_0, A, q_1]$

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

(5) $\delta(q_0, 0, A) = \{(q_1, A)\}$

For q_0, $[q_0, A, q_0] \rightarrow 0[q_1, A, q_0]$

For q_1, $[q_0, A, q_1] \rightarrow 0[q_1, A, q_1]$

(6) $\delta(q_1, 1, A) = \{(q_1, \varepsilon)\}$

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

(7) $\delta(q_1, 0, S) = \{(q_0, S)\}$

For q_0, $[q_1, S, q_0] \rightarrow 0[q_0, S, q_0]$

For q_1, $[q_1, S, q_1] \rightarrow 0[q_0, S, q_1]$

Since [q_1, A, q_0], [q_1, A, q_1] does not have any productions we can leave them. After eliminating the unwanted productions,

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

$[q_0, S, q_0] \rightarrow 1[q_0, A, q_1] [q_1, S, q_0]$
$[q_0, S, q_0] \rightarrow \varepsilon$
$[q_0, A, q_1] \rightarrow 1[q_0, A, q_1] [q_1, A, q_1]$
$[q_0, A, q_1] \rightarrow 0[q_1, A, q_1]$
$[q_1, A, q_1] \rightarrow 1$
$[q_1, S, q_0] \rightarrow 0[q_0, S, q_0]$

Finally P is given by,

$S \rightarrow [q_0, S, q_0]$
$[q_0, S, q_0] \rightarrow 1[q_0, A, q_1] [q_1, S, q_0] \mid \varepsilon$
$[q_0, A, q_1] \rightarrow 1[q_0, A, q_1] [q_1, A, q_1] \mid 0[q_1, A, q_1]$
$[q_1, A, q_1] \rightarrow 1$
$[q_1, S, q_0] \rightarrow 0[q_0, S, q_0]$

Ex.2:

Construct a CFG for the PDA, $P = (\{q_0, q_1\}, \{0, 1\}, \{X, Z_0\}, \delta, q_0, Z_0, \{q_1\})$, where δ is,

$\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$ $\quad$ $\delta(q_1, 1, X) = \{(q_1, \varepsilon)\}$
$\delta(q_0, 0, X) = \{(q_0, XX)\}$ $\quad$ $\delta(q_1, \varepsilon, X) = \{(q_1, \varepsilon)\}$
$\delta(q_0, 1, X) = \{(q_1, \varepsilon)$ $\quad$ $\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$

Soln:

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

Where, $V = \{[q_0, X, q_0], [q_0, X, q_1], [q_1, X, q_0], [q_1, X, q_1], [q_0, Z_0, q_0], [q_0, Z_0, q_1],$
$[q_1, Z_0, q_0], [q_1, Z_0, q_1]\}$
$T = \{0, 1\}$
$S = \{S\}$ [Start stack symbol]

To find production, P;

(1) Production for S,
$S \rightarrow [q_0, Z_0, q_0]$
$S \rightarrow [q_0, Z_0, q_1]$ [q_0 – Start state, Z_0 – Initial stack symbol]

(2) $\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$ we get,
For q_0, $[q_0, Z_0, q_0] \rightarrow 0[q_0, X, q_0] [q_0, Z_0, q_0]$
$[q_0, Z_0, q_0] \rightarrow 0[q_0, X, q_1] [q_1, Z_0, q_0]$
For q_1, $[q_0, Z_0, q_1] \rightarrow 0[q_0, X, q_0] [q_0, Z_0, q_1]$

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

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

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

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

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

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

(4) $\delta(q_0, 1, X) = \{(q_1, \varepsilon)$

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

(5) $\delta(q_1, 1, X) = \{(q_1, \varepsilon)\}$

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

(6) $\delta(q_1, \varepsilon, X) = \{(q_1, \varepsilon)\}$

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

(7) $\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$

$[q_1, Z_0, q_1] \rightarrow \varepsilon$

After eliminating the unwanted productions, we get;

$S \rightarrow [q_0, Z_0, q_1]$

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

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

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

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

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

$[q_1, Z_0, q_1] \rightarrow \varepsilon$

Finally P is given by,

$S \rightarrow [q_0, Z_0, q_1]$
$[q_0, Z_0, q_1] \rightarrow 0[q_0, X, q_1]\ [q_1, Z_0, q_1]$
$[q_0, X, q_1] \rightarrow 0[q_0, X, q_1]\ [q_1, X, q_1] \mid 1$
$[q_1, X, q_1] \rightarrow 1 \mid \varepsilon$
$[q_1, Z_0, q_1] \rightarrow \varepsilon$

Ex.3:

Construct a CFG for the PDA, P = ({q_0, q_1}, {0, 1}, {X, Z_0}, δ, q_0, Z_0, {q_1}), where δ is,

$\delta(q_0, b, Z_0) = \{(q_0, ZZ_0)\}$ $\quad\quad$ $\delta(q_0, \varepsilon, Z_0) = \{(q_0, \varepsilon)\}$

$\delta(q_0, b, Z) = \{(q_0, ZZ)\}$ $\quad\quad$ $\delta(q_0, a, Z) = \{(q_1, Z)\}$

$\delta(q_1, b, Z) = \{(q_1, \varepsilon)$ $\quad\quad$ $\delta(q_1, a, Z_0) = \{(q_0, Z_0)\}$

Soln:

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

Where, V = {$[q_0, Z_0, q_0]$, $[q_0, Z_0, q_1]$, $[q_1, Z_0, q_0]$, $[q_1, Z_0, q_1]$, $[q_0, Z, q_0]$, $[q_0, Z, q_1]$,

$[q_1, Z, q_0]$, $[q_1, Z, q_1]$}

T = {a, b}

S = {S} [Start stack symbol]

To find production, P;

(1) Production for S,

$S \rightarrow [q_0, Z_0, q_0]$

$S \rightarrow [q_0, Z_0, q_1]$ [q_0 – Start state, Z_0 – Initial stack symbol]

(2) $\delta(q_0, b, Z_0) = \{(q_0, ZZ_0)\}$ we get,

For q_0, $[q_0, Z_0, q_0] \rightarrow b[q_0, Z, q_0] [q_0, Z_0, q_0]$

$[q_0, Z_0, q_0] \rightarrow b[q_0, Z, q_1] [q_1, Z_0, q_0]$

For q_1, $[q_0, Z_0, q_1] \rightarrow b[q_0, Z, q_0] [q_0, Z_0, q_1]$

$[q_0, Z_0, q_1] \rightarrow b[q_0, Z, q_1] [q_1, Z_0, q_1]$

(3) $\delta(q_0, b, Z) = \{(q_0, ZZ)\}$

For q_0, $[q_0, Z, q_0] \rightarrow b[q_0, Z, q_0] [q_0, Z, q_0]$

$[q_0, Z, q_0] \rightarrow b[q_0, Z, q_1] [q_1, Z, q_0]$

For q_1, $[q_0, Z, q_1] \rightarrow b[q_0, Z, q_0] [q_0, Z, q_1]$

$[q_0, Z, q_1] \rightarrow b[q_0, Z, q_1] [q_1, Z, q_1]$

(4) $\delta(q_1, b, Z) = \{(q_1, \varepsilon)$

$[q_1, Z, q_1] \rightarrow b$

(5) $\delta(q_0, \varepsilon, Z_0) = \{(q_0, \varepsilon)\}$
$[q_0, Z_0, q_0] \rightarrow \varepsilon$

(6) $\delta(q_0, a, Z) = \{(q_1, Z)\}$
For q_0, $[q_0, Z, q_0] \rightarrow a[q_1, Z, q_0]$
For q_1, $[q_0, Z, q_1] \rightarrow a[q_1, Z, q_1]$

(7) $\delta(q_1, a, Z_0) = \{(q_0, Z_0)\}$
For q_0, $[q_1, Z_0, q_0] \rightarrow a[q_0, Z_0, q_0]$
For q_1, $[q_1, Z_0, q_1] \rightarrow a[q_0, Z_0, q_1]$

After eliminating the unwanted productions, we gt;

$S \rightarrow [q_0, Z_0, q_0]$
$[q_0, Z_0, q_0] \rightarrow b[q_0, Z, q_1] [q_1, Z_0, q_0]$
$[q_0, Z_0, q_0] \rightarrow \varepsilon$
$[q_0, Z, q_1] \rightarrow b[q_0, Z, q_1] [q_1, Z, q_1]$
$[q_0, Z, q_1] \rightarrow a[q_1, Z, q_1]$
$[q_1, Z_0, q_0] \rightarrow a[q_0, Z_0, q_0]$
$[q_1, Z, q_1] \rightarrow b$

Finally P is given by,

$S \rightarrow [q_0, Z_0, q_0]$
$[q_0, Z_0, q_0] \rightarrow b[q_0, Z, q_1] [q_1, Z_0, q_0] \mid \varepsilon$
$[q_0, Z, q_1] \rightarrow b[q_0, Z, q_1] [q_1, Z, q_1] \mid a[q_1, Z, q_1]$
$[q_1, Z_0, q_0] \rightarrow a[q_0, Z_0, q_0]$
$[q_1, Z, q_1] \rightarrow b$

3.7. DETERMINISTIC PUSHDOWN AUTOMATA (DPDA)

3.7.1. Definition of a Deterministic PDA:

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ to be deterministic, if and only if the following conditions are met;

(1) $\delta(q, a, X)$ has at most one member for any 'q' in Q, 'a' in Σ or $a=\varepsilon$ and X in Γ.
(2) If $\delta(q, a, X)$ is nonempty, for some 'a' in Σ then $\delta(q, \varepsilon, X)$ must be empty.

3.7.2. Regular Languages and DPDA's:

The DPDA's accept a class of language that is between the regular languages and the CFL's. We shall first prove that the DPDA languages include all the regular languages.

3.7.2.1. Theorem:

If L is a regular language, then L = L(P) for some DPDA P.

Proof:

A DPDA can simulate a deterministic finite automaton. The PDA keeps some stack symbol Z_0 on its stack because a PDA has to have a stack, but really the PDA ignores its stack and just uses its state. Formally, let A = (Q, Σ, δ_A, q_0, F) be a DFA construct DPDA.

$$P = (Q, \Sigma, \{Z_0\}, \delta_P, q_0, Z_0, F)$$

By defining $\delta_P(q, a, Z_0) = \{(p, Z_0)\}$ for all states 'p' and 'q' in Q, such that $\delta_A(q, a) = p$.

$$\therefore (q_0, w, Z_0) \}^*_P (p, \varepsilon, Z_0) \text{ if and only if } \delta A(q_0, w) = p.$$

3.7.3. DPDA's and Context Free Languages:

The languages accepted by DPDA's by final state properly include the regular languages, but are properly included in the CFL's.