CHAPTER 3

Pushdown Automaton

We have seen that the regular languages are precisely those accepted by finite automata. Not every context free language can be recognized by a finite automaton since some context free languages are not regular. Yet the problem of recognizing context free languages is an important one for both theoretical and practical reasons. What sort of more powerful automaton could be used for recognizing arbitrary context free languges? Or to be more specific, what do we need to add a finite automaton so that the construction can be generalized to apply all context free languages ?

If M is a finite automaton accepting L, it is constructed in such a way that states act as a form of primitive memory. The states remember the variables encounted in the course of derivation of a string. Let us consider $L = \{a^n b^n / n \geq 1\}$. This is a context free language but not regular. A finite automaton cannot accept L, *i.e.*, strings of the form $a^n b^n$ as it has to remember the number of a's in a string and so it will require infinite number of states. This difficulty can be avoided by adding an auxillary memory in the form of a stack. The a's in the given string are added to the stack. When the syumbol b is encountered in the input string an a is removed from the stack. Thus the matching of number of $a's$ and number of $b's$ is accomplished. This type of arrangement where a finite automaton has stack leads to the generation of pushdown automaton.

In this chapter we introduce Pushdown Automaton (PDA). We discuss two types of acceptance of sets by pushdown automata. We prove that the sets accepted by pushdown automata are precisely the class of context free languges. We conclude this chapter after proving pumping lemma and giving some decision algorithms.

3.1. INTRODUCTION

In chapter 1 we have introduced the Finite State Automata (FSA) as a class of machines which accept exactly the regular language (regular sets). In this chapter we introduce a class of machines known as Pushdown Automata (PDA) or (pushdown acceptors) which accept exactly the context free languages. In

addition to the components of FSA, like the set of states, finite control, input tape, the PDA has a pushdown tape (or pushdown store) which acts on a first in last out basis. Symbols are added or removed only from the top. When a symbol is added on top, the symbol originally on top becomes the second and so on. On the otherhand, when the symbol on top is removed, the symbol which was originally second becomes the top symbol and so on.

3.1.1 Functioning of PDA

The Pushdown automaton 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 the pushdown store. Depending on the state of the finite control an input

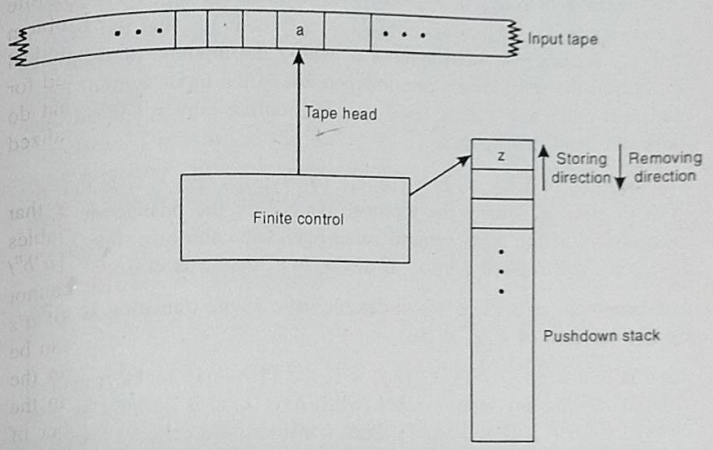

Fig 3.1. Model of pushdown automaton.

Symbol (or λ) just read with a pushdown symbol on the top of the pushdown store, the PDA can (non-deterministically) change state move to the next input symbol to the right (or read the same symbol if it is λ) and replace the top of the pushdown symbol by a string of pushdown symbols or erase it. There is a start state, a start pushdown symbol and a set of final states.

We now give the formal definition of PDA.

3.1.2. Definition

Nordeterministic

A *pushdown automaton* (PDA) is a 7-tuple $M = (Q, \Sigma, \Gamma, q_0, z_0, \delta, F)$

PDS. The same transition is effected by starting with input string xy and processing only x. In this case, y remains to be processed and hence we get (3.2)

In the similar way we can prove (3.2) implies (3.1) Hence the proof.

3.1.9. Property

$$(q, x, \alpha) \overset{*}{\vdash} (q', \lambda, \nu) \quad \ldots(3.3)$$

If

then for every $\beta \in \Gamma^*$,

$$(q, x, \beta\alpha) \vdash (q', \lambda, \beta\nu) \quad \ldots(3.4)$$

Proof. The sequence of moves given by (3.3) can be split as

$$(q, x, \alpha) \vdash (q_i, x_1, \alpha) \vdash (q_2, x_2, \alpha_2) \vdash \cdots \vdash (q', \lambda, \nu)$$

Consider $(q_i, s_i\ \alpha_i) \vdash (q_{i+1}, x_{i+1}, \alpha_{i+1})$. Let $\alpha_i = z_m \cdots z_i$. As a result of this move, z_1, is erased and some string is placed above $z_m \cdots z_2$. So $z_m \cdots z_2$ is not affected. If we have β below $z_m \cdots z_2$, then also $z_m \cdots z_2\beta$ is not affected. so we obtain

$$(q_i, x_i, \beta\alpha_i) \vdash (q_{i+1}, x_{i+1}, \beta\alpha_{i+1})$$

Therefore, we get a sequence of moves

$$(q, x, \beta\alpha) \vdash (q_1, x_1, \beta\alpha_1) \vdash \cdots \vdash (q', \lambda, \beta\nu)$$

i.e.,

$$(q, x, \beta\alpha) \overset{*}{\vdash} (q', \lambda, \beta\nu)$$

Note. In general (3.4) need not imply (3.3)

3.2 ACCEPTANCE BY PDA

3.2.1. Acceptance by Final State

A word w is accepted (or recognized) by a PDA if starting with the start state in the finite control, the start pushdown symbol on the top the pushdown store, by a sequence of moves, the word w is read from left to right letter by letter and the PDA finally enters a final state (after reading the rightmost symbol). This is called *acceptance by final state*. That is

Let A = $(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA. The set accepted by PDA by final state is defined by

$T(A) = \{w \in T^*/(q_0, w, z_0) \overset{*}{\vdash} (q_i, \lambda, \alpha)$ for some $q_i \in F$ and $\alpha \in \Gamma^*\}$

3.2.2 Acceptance by empty store (or stack)

A word w may be accepted by emptying the pushdown store also. Here the

action of the PDA is similar but instead of entering a final state when the last input letter is read, the PDA accepts the word w if the pushdown store is empty after reading the rightmost input letter. That is

Let $A = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA. The set accepted by null store (or empty store) is defined by

$$N(A) = \{w \in \Sigma^*/(q_0, w, z_0) \overset{*}{\vdash} (q, \lambda, \lambda) \text{ for some } q \in Q\}.$$

Note. It can be established that the two kinds of acceptance are equivalent in the sense that if a set of words is accepted by final state by some PDA, it can be accepted by empty store by some other PDA and vice versa.

3.2.3. Theorem

A language L is accepted by a PDA by empty store if and only if L is accepted by a PDA by final state

(or)

If L is $T(M_1)$ for some PDA M_1 if and only if L is $N(M_2)$ for some PDA M_2

Proof. Let $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA accepting L by final state. That is $L = T(M_1)$ then $M_2 = (Q, \Sigma', \Gamma', \delta', q_0', z_0', F')$ where $F_1 = \{\phi\}$

$$Q' = Q \cup \{q_0', q_e\} \text{ where } q_0', q_e \notin Q.$$

$$\Sigma' = \Sigma$$

$$\Gamma' = \Gamma \cup \{z'_0\}, \text{ where } z_0' \notin \Gamma$$

and δ' is defined as follows.

1. $\delta'(q_0', \lambda, z_0')$ contains $(q_0, z_0'\ z_0)$
2. $\delta'(q_i, a, z)$ is same as $\delta(q_i, a, z)$ for all $q_1 \in, Q, a \in \Sigma \cup \{\lambda\}$ and $z \in \Gamma$.
3. $\delta'(q_i, \lambda, z)$ contains (q_e, λ) for $q_i \in F$ and for all $z \in \Gamma \cup \{z_0'\}$
4. $\delta'(q_e, \lambda, z)$ contains (q_e, λ) for all $z \in \Gamma \cup \{z_0'\}$

It can be easily seen that $T(M_1) = N(M_2)$

[For, 1 causes M_2 to enter the intial configuration of M_1 with z'_0 as the leftmost pushdown symbol. The effect of z is to make M_2 simulate. The behaviour of M_1, thus reading an input word w if M_1 reads w and enters a final state. Once w is read and M_2 enter a final state of M_1, the dffect of 3 and 4 is to empty the pushdown store].

Thus w is in $T(M_1)$ iff

$$(q_0', w, z_0') \overset{*}{\underset{M_1}{\vdash}} (q_i, \lambda, r) \text{ for some } q_i \in F$$

$(q_0', w, z_0') \overset{}{\underset{M_2}{\vdash}} (q_0, w, z_0' z_0)$ by 1

$\overset{*}{\vdash} (q_i, \lambda, z_0' r)$ by 2 where $q_i \in F$

$\overset{*}{\vdash} (q_e, \lambda, \lambda)$

by 3 and 4. That is w is in $N(M_2)$. That is L is accepted by empty store by M_2. That is $L = N(M_2)$.

Conversely, let $M_2 = (Q', \Sigma', \Gamma', \delta', q_0', z_0', \phi)$ be a PDA accepting L by empty store is $L = N(M_2)$. Then $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where

$Q = Q^1 \cup \{q_0, q_j\}$ where $q_0, q_i \notin Q'$

$\Gamma = \Gamma' \cup \{z_0\}$, where $z_0 \notin \Gamma'$

$F = \{q_j\}$ and δ is defined as follows.

1. $\delta(q_0, \lambda, z_0)$ contains $(q_0', z_0 z_0')$
2. $\delta(q_i, a, z)$ is the same as $\delta'(q_i, a, z)$ for all $q_i \in Q'$, $a \in \Sigma \cup \{\lambda\}$ and $z \in \Gamma'$
3. $\delta(q_i, \lambda, z_0)$ contains (q_j, z_0) for all $q_i \in Q'$.

It can be easily verified that $N(M_2) = T(M_1)$ [For rule 1 causes M_1 to enter the initial configuration of M_2 with the difference that z_0 is the bottom most symbol on the pushdown store. The effect of 2 is to allow M_1 to simulate the behaviour of M_2. When M_2 erases the pushdown store and accepts w, M_1 will erase the entire pushdown store except the bottom symbol z_0. When z_0 is the topmost pushdown symbol, 3 causes M_1 to enter the final state q_j, there by accepting the input w, hence the theorem.]

3.2.4. Problem

1. Construct a PDA to accept a given language L by final state where $L = \{a^n b^n / n \geq 1\}$.

Solution. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where

$Q = \{q_0, a_1, q_2\}$

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

$\Gamma = \{z_0, z_1, z_2\}$

$F = \{q_2\}$

and δ is defined as follows

$\delta(q_0, a, z_0) = \{(q_0, z_1)\}$

$\delta(q_0, a, z_1) = \{(q_0, z_1 z_2)\}$

$\delta(q_0, b, z_2) = \{(q_0, z_2 z_2)\}$

$\delta(q_0, b, z_1) = \{(q_2, z_1)\}$

$\delta(q_0, b, z_2) = \{(q_1, \lambda)\}$

$\delta(q_1, b, z_2) = \{(q_1, \lambda)\}$

$\delta(q_1, b, z_1) = \{(q_2, z_1)\}$

Then $(q_0, ab, z_0) \vdash (q_0, b, z_1) \vdash (q_2, \lambda, z_1)$ so that $ab \in T(M)$

also for $n > 1$.

$$(q_0, a^n b^n, z_0) \vdash (q_0, a^{n-1} b^n, z_1)$$

$$\vdash (q_0, a^{n-2} b^n, z_1 z_2)$$

$$\vdash (q_0, a^{n-3} b^n, z_1 z_2^2)$$

$$\vdash (q_0, a^{n-4} b^n, z_1 z_2^3)$$

$$\overset{*}{\vdash} (q_0, b^n, z_1 z_2^{n-1})$$

$$\vdash (q_1, b^{n-1}, z_1 z_2^{n-2})$$

$$\overset{*}{\vdash} (q_1, b, z_1 z_2^{n-n}) = (q_1 b z_1)$$

$$\vdash (q_2, \lambda, z_1)$$

So that $a^n b^n \in T(M)$. Hence the PDA, M accepts L by final state.

Remark. We can also represent the transition function by a table known as transition table.

For example consider the function δ given in problem 1. We can represent it in the form of transition table as follows.

States	*Inputs*	*Stake symbols*	*Move(s)*
q_0	a	z_0	(q_0, z_1)
q_0	a	z_1	$(q_0, z_1 z_2)$
q_0	a	z_2	$(q_0, z_2 z_2)$
q_0	b	z_1	(q_2, z_1)
q_0	b	z_2	(q_1, λ)
q_1	b	z_2	(q_1, λ)
q_1	b	z_1	(q_2, z_1)

2. Construct a PDA to accept a given language L by empty store where $L = \{a^n b^n / n \geq 1\}$

Solution. Let $L = \{a^n b^n / n \geq 1\}$

Let $M' = (Q' \Sigma', \Gamma', \delta', q_0' z_0', F')$ where

$Q' = \{q_0'\}$,

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

$Z' = \{z_0', z_1', z_2'\}$

$F' = \phi$ and δ' is defined as follows

$\delta' (q_0', \lambda, z_0') = \{ (q_0', z_2' z_0' z_1') (q_0', z_2' z_1')\}$

$\delta' (q_0', a z_1') = \{(q_0', \lambda)\}$

$\delta' (q_0', b, z_2') = \{(q_0', \lambda)\}$

Then

$$
\begin{aligned}
(q_0', a^n b^n, z_0') &\vdash (q_0', a^n b^n, z_2' z_0' z_1') \\
&\vdash (q_0', a^{n-1} b^n, z_2' z_0') \\
&\vdash (q_0', a^{n-1} b^n, z_2' z_2' z_0' z_1') \\
&\vdash (q_0', a^{n-2} b^n, z_2' z_2' z_0') \\
&\vdash (q_0', a^{n-2} b^n, z_2'^3 z_0' z_1') \\
&\vdash (q_0', a^{n-3} b^n, z_2'^3 z_0') \\
&\vdash^* (q_0', ab^n, z_2'^{(n-1)} z_0') \\
&\vdash (q_0', ab^n, (z_2')^n z_1') \\
&\vdash (q_0', b^n, (z_2')^n) \\
&\vdash (q_0', b^{n-1}, (z_2')^n) \\
&\vdash (q_0', b^{n-1}, (z_2')^{n-1}) \\
&\vdash^* (q_0', \lambda, \lambda)
\end{aligned}
$$

so that $a^n b^n \in N(M')$

Hence the PDA, M' accepts L by empty store. That is $N(M') = L$

3. Consturct a PDA to accept the language $L = \{wew^R / w$ is in $(a, b)^*\}$ by final state

Solution. Consider a PDA

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where

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

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

$Z = \{z_0, z_1, z_2\}$

$F = \{q_2\}$ and δ is defined as follows.

$\delta (q_0, a, z_0) = \{q_0, z_0 z_1\}$

$\delta (q_0, a, z_1) = \{q_0, z_1 z_1\}$

$\delta (q_0, a, z_2) = \{q_0, z_2 z_1\}$

$\delta (q_1, a, z_1) = \{(q_1, \lambda)\}$

$\delta (q_0, b, z_0) = \{(q_0, z_0 z_2)\}$

$\delta (q_0, b, z_1) = \{(q_0, z_0 z_2)\}$

$\delta (q_0, b, z_2) = \{(q_0, z_1 z_2)\}$

$\delta (q_1, b, z_2) = \{(q_1, \lambda)\}$

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

$\delta (q_0, c, z_1) = \{(q_1, z_1)\}$

$\delta (q_0, c, z_2) = \{(q_1, z_2)\}$

$\delta (q_1, \lambda, z_0) = \{(q_2, z_0)\}$

If w is in $(a,b)^*$ then

$$
\begin{aligned}
(q_0, wcw^R, z_0) &\vdash^* (q_0, cw^R, z_0 W) \\
&\vdash (q_1, w^R, z_0 W) \\
&\vdash^* (q_1, \lambda, z_0) \\
&\vdash (q_2, \lambda, z_0)
\end{aligned}
$$

where W is w with a replaced by z_1 and b by z_2. Thus $T(M) = L$.

4. Construct a PDA to accept the language $L = \{wcw^R / w$ in $(a, b)^*\}$ by empty store.

Solution. $L = \{wcw^R / w$ in $(a, b)^*\}$

Let $M' = (Q', \Sigma', \Gamma', q_0', z_0', \delta', F')$

where $Q_0' = \{q_0', q_1'\}$

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

$\Gamma' = \{z_0', z_1', z_2'\}$

$F' = \phi$

and δ' is defined as follows

$\delta' (q_0', a, z_0') = \{(q_0' z_0' z_1')\}$

$\delta' (q_0', a, z_1') = \{(q_0' z_1' z_1')\}$

$$\delta'(q_1', a, z_2') = \{(q_0' z_2' z_1')\}$$
$$\delta'(q_1', a, z_1') = \{(q_1', \lambda)\}$$
$$\delta'(q_0', b, z_0') = \{(q_0', z_0' z_2')\}$$
$$\delta'(q_0', b, z_1') = \{(q_0', z_1' z_2')\}$$
$$\delta'(q_0', b, z_2') = \{(q_0', z_2' z_2')\}$$
$$\delta'(q_0', b, z_2') = \{(q_1', \lambda)\}$$
$$\delta'(q_0', c, z_0') = \{(q_1', z_0')\}$$
$$\delta'(q_0', c, z_1') = \{(q_1', z_1')\}$$
$$\delta'(q_0', c, z_2') = \{(q_1', z_2')\}$$
$$\delta'(q_1', \lambda, z_0') = \{(q_1', \lambda')\}$$

we see that if w is in $(a,b)^*$, then

$$\begin{aligned}(q_0', wcw^R, z_0') &\overset{*}{\vdash} (q_0', cw^R, z_0' W)\\ &\vdash (q_1', w^R, z_0' W)\\ &\overset{*}{\vdash} (q_1', \lambda, z_0')\\ &\vdash (q_1', \lambda, \lambda)\end{aligned}$$

where W is w where a is replaced by z_1' and b by z_2'. Hence $N(M') = L$.

Now we will see few applications of Theorem 3.2.3.

5. Construct a PDA to accept the language $L = \{wcw^R/w \text{ in } (a, b)^*\}$ by final state.

Solution. Consider the PDA M' given in problem 4. we have proved that M' accepts

$$L = \{wcw^R/w \text{ in } (a, b)^*\} \text{ by empty store.}$$

Now we use Theorem 3.2.3 to construct the required PDA

Let $M = (Q, \Sigma, \Gamma, q_0, z_0, \delta, F)$

where $Q = Q' \cup \{q_0, q_j\} = \{(q_0', q_1', q_0, q_j)\}$

$\Gamma = \Gamma' \cup \{z_0\} = \{z_0', z_1', z_2', z_0\}$

$F = \{q_j\}$ and δ is defined as follows.

$$\delta(q_0, \lambda, z_0) = \{(q_0', z_0 z_0')\}$$
$$\delta(q_0', a, z_0') = \{(q_0', z_0' z_1')\}$$

$$\delta(q_0', a, z_1') = \{(q_0', z_1' z_1')\}$$
$$\delta(q_0', a, z_2') = \{(q_0', z_2' z_1')\}$$
$$\delta(q_1', a, z_1') = \{(q_1', \lambda)\}$$
$$\delta(q_0', b, z_0') = \{(q_0', z_0' z_2')\}$$
$$\delta(q_0', b, z_1') = \{(q_0', z_1' z_2')\}$$
$$\delta(q_0', b, z_2') = \{(q_0', z_2' z_2')\}$$
$$\delta(q_0', b, z_2') = \{(q_0', z_2' z_2')\}$$
$$\delta(q_0', c, z_0') = \{(q_0', z_0')\}$$
$$\delta(q_0', b, z_2') = \{(q_1', \lambda)\}$$
$$\delta(q_0', c, z_1') = \{(q_1', z_1')\}$$
$$\delta(q_0', c, z_2') = \{(q_1', z_2')\}$$
$$\delta(q_1', \lambda, z_0') = \{(q_1', \lambda)\}$$
$$\delta(q_0', \lambda, z_0) = \{(q_j, z_0)\}$$
$$\delta(q_1', \lambda, z_0) = \{(q_j, z_0)\}$$

We can easily seen that $T(M) = L = N(M')$

[For, $$\begin{aligned}(q_0, wcw^R, z_0) &\vdash (q_0', wcw^R, z_0 z_0')\\ &\overset{*}{\vdash} (q_0', cw^R, z_0 z_0' w)\\ &\vdash (q_1', w^R, z_0 z_0' w)\\ &\overset{*}{\vdash} (q_1', \lambda, z_0 z_0')\\ &\overset{*}{\vdash} (q_1', \lambda, z_0)\\ &\vdash (q_j, z_0)]\end{aligned}$$

6. Construct a PDA to accept the language $L = \{a^n b^n / n \geq 1\}$ by empty store.

Solution. Consider the PDA M given in problem 1. Then M accepts L by final state. We now use Theorem 3.2.3 to construct the required PDA.

Let $M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$

where $Q' = Q \cup \{q_e, q_0'\}$, q_e and $q_0' \notin Q$

$= \{q_0, q_1, q_2, q_e, q_0'\}$

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

$\Gamma' = \Gamma \cup \{z_0'\}$

$= \{z_0, z_1, z_2, z_0'\}$

$F = \phi$ and δ' is defined as follows.

$$\delta'(q_0', \lambda, z_0') = \{(q_0, z_0' z_0)\}$$
$$\delta'(q_0, a, z_0) = \{(q_0, z_1)\}$$
$$\delta'(q_0, a, z_1) = \{(q_0, z_1 z_2)\}$$
$$\delta'(q_0, a, z_2) = \{(q_0, z_2 z_2)\}$$
$$\delta'(q_0, b, z_1) = \{(q_2, z_1)\}$$
$$\delta'(q_0, b, z_2) = \{(q_1, \lambda)\}$$
$$\delta'(q_1, b, z_2) = \{(q_1, \lambda)\}$$
$$\delta'(q_1, b, z_1) = \{(q_2, z_1)\}$$
$$\delta'(q_2, \lambda, z_0) = \{(q_e, \lambda)\}$$
$$\delta'(q_2, \lambda, z_1) = \{(q_e, \lambda)\}$$
$$\delta'(q_2, \lambda, z_2) = \{(q_e, \lambda)\}$$
$$\delta'(q_2, \lambda, z_0) = \{(q_e, \lambda)\}$$
$$\delta'(q_e, \lambda, z_0) = \{(q_e, \lambda)\}$$
$$\delta'(q_e, \lambda, z_1) = \{(q_e, \lambda)\}$$
$$\delta'(q_e, \lambda, z_2) = \{(q_e, \lambda)\}$$
$$\delta'(q_e, \lambda, z_0') = \{(q_e, \lambda)\}$$

It is easily seen that $N(M') = L = T(M)$.

3.3. PUSHDOWN AUTOMATA AND CONTEXT FREE LANGUAGES

3.3.1. Theorem

If L is a context free language, then there exists a PDA M such that $L = N(M)$.

(Procedure to construct a PDA for a given language)

Proof. Let $G = (N, T, S, P)$ be a CFG generating L (G).

Let $M = (\{q_0\}, \Sigma, \Gamma, \delta, q_0, z_0, \phi)$

where $\Sigma = T$

$\Gamma = N \cup T$

$z_0 = S$

and δ is defined as follows. For each $a \in \Sigma$ and $A \in N$

(*i*) $\delta(q_0, \lambda, A) = \{(q_0, \alpha^R) / A \to \alpha \text{ is in } p\}$

(*ii*) $\delta(q_0, a, a) = \{(q_0, \lambda)\}$

The pushdown symbols in M are terminals and non-terminals, if the PDA reads a variable A on the top of pushdown store, it match a λ- move by replacing the reverse of R.H.S. of any A-production (after erasing A). If the PDA read, a terminal 'a' on PDS and if it matches with the current input symbol, then the PDA erases a. In other cases PDA halts.

If $w \in L(G)$ is obtained by a leftmost derivation $S \Rightarrow u_1A_1\alpha_1 \Rightarrow u_1u_2A_2\alpha_2\alpha_1 \Rightarrow \cdots \Rightarrow w$, then A can empty the PDS on application of input string w. The first move of M is by a λ move corresponding to $S \to u_1A_1\alpha_1$. The PDA erases the symbols in u_1 by processing prefix of w. Now the topmost symbol in PDS is A_1. Once again by applying the λ-move corresponding to $A_1 \to u_2A_2\alpha_2$, the PDA erases A_1 and stores $\alpha_2A_2u_2$ above α_1 proceeding in this way, the PDA empties the PDS by processing the entire string w.

Now we prove that $L(G) = N(M)$. Let $w \in L(G)$. Then it can be derived by a leftmost derivation. Any sentential form in a leftmost derivation is of the form $u\,A\,\alpha$ where $u \in T^*$, $A \in N$ and $\alpha \in (N \cup T)^*$. We prove the following auxillary result.

If $S \overset{*}{\Rightarrow} uA\alpha$ by a leftmost derivation, then $(q, uv, S) \overset{*}{\vdash} (q, v, \alpha A)$ for every $v \in \Sigma^*$ ...(3.5)

We prove (3.5) by induction on the number of steps in the derivation of $uA\alpha$. If

$S \overset{*}{\Rightarrow} uA$, then $u = \lambda$, $\alpha = \lambda$ and $S = A$.

As $(q, v, S) \overset{*}{\vdash} (q, v, S)$, there is a basis for induction.

Suppose $S \overset{n+1}{\Rightarrow} uA\alpha$ by a leftmost derivation. This derivation can be split as

$S \overset{n}{\vdash} u_1 A_1\alpha_1 \Rightarrow uA\alpha$. If in the A_1 — production we apply in the last step $A_1 \to u_2A\alpha_2$, then $u = u_1u_2$, $\alpha = \alpha_2\alpha_1$.

As $S \overset{n}{\Rightarrow} u_1A_1\alpha_1$, by induction hypothesis,

$$(q_1, u_1u_2v, S) \overset{*}{\vdash} (q_1, u_2v, \alpha_1A_1) \qquad \text{...(3.6)}$$

As $A_1 \to u_2A\alpha_2$ is a production in P, by rule (*i*), we get

$(q, \lambda, A_1) \vdash (q, \lambda, \alpha_2\, Au_2)$. Applying 3.1.8. and 3.1.9, we get

$(q, u_2v, \alpha_1A) \vdash (q, u_2v, \alpha_1\alpha_2\, A\, u_2) \vdash (q, u, A\alpha_2\alpha_1)$ by rule (*ii*)

$(q, u_2v, \alpha_1A_1) \vdash^* (q, v, \alpha_1\alpha_2 A)$ ...(3.7)

Hence

But $u_1u_2 = u$ and $\alpha_1\alpha_2 = \alpha$. So from (3.6) and (3.7), we have

$(q, uv, S) \vdash^* (q, v, \alpha A)$

Thus (3.5) is true for $S \overset{n+1}{\Rightarrow} uA\alpha$. By the principle of induction, (3.5) is true for any derviation. Now we prove that $L(G) \subseteq N(M)$.

Let $w \in L(G)$. Then w can be obtained from a leftmost derivation.

$$S \overset{*}{\Rightarrow} uAv \Rightarrow uu'v = w$$

From (*i*)

$$(q, uu'v, S) \vdash^* (q, u'v, vA)$$

As $A \to u'$ is in P,

$$(q, u'v, vA) \vdash^* (q, u'v, vu')$$

By rule (*ii*)

$$(q, u'v, vu') \vdash^* (q, \lambda, \lambda)$$

Therefore,

$w = uu'v \in N(A)$ proving $L(G) \subseteq N(m)$. Next we prove $N(m) \subseteq L(G)$.

Before proving the inclusion, let us prove the following auxillary result.

$$S \overset{*}{\Rightarrow} u\alpha \text{ if } (q, uv, S) \vdash^* (q, v, \alpha) \quad \text{...(3.8)}$$

we prove (3.8) by the number of moves in $(q, uv, S) \vdash^* (q, v, \alpha)$. If $(q, uv, S) \vdash^* (q, v, \alpha)$, then $u = \lambda$, $S = \alpha$; obviously $S \overset{*}{\Rightarrow} \lambda\alpha$.

Thus there is a basis for induction.

Let us assume (3.8) when the number of moves is n. Assume

$$(q, uv, S) \vdash^* (q, v, \alpha) \quad \text{...(3.9)}$$

The last move in (3.9) is obtained either from $(q, \lambda, A) \vdash (q, \lambda, \alpha')$ or $(q, a, a) \vdash (q, \lambda, \lambda)$. In the first case (3.9) can be split as

$(q, uv, S) \vdash^n (q, v, \alpha^2A) \vdash (q, v, \alpha_2\alpha_1) = (q, v, \alpha)$.

By induction hypothesis, $S \overset{*}{\Rightarrow} uA\alpha_2$, and the last move is induced by $A \to \alpha_1$.

Thus,

$S \overset{*}{\Rightarrow} uA\alpha_2$ implies $\alpha_1\alpha_2 = \alpha$. So $S \Rightarrow uA\alpha_2 \Rightarrow u\alpha_1\alpha_2 = u\alpha$.

In the second case (3.9) can be split as

$$(q, uv, S) \vdash^* (q, av, \alpha a) \vdash (q, v, \alpha)$$

Also $u = u'a$ for some $u' \in \Sigma$. So $(q, u'av, S) \vdash^* (q, av, \alpha a)$ implies (by induction hypothesis) $S \overset{*}{\Rightarrow} u'a\alpha = u\alpha$. Thus in both cases we have shown that $S \overset{*}{\Rightarrow} u\alpha$. By the principle of induction (3.8) is true.

Now we can prove that if $w \in N(M)$ then $w \in L(G)$. As $w \in N(A)$,

$(q, w, S) \vdash^* (q, \lambda, \lambda)$. By taking $u = w$, $v = \lambda$, $\alpha = \lambda$ and applying (3.8), we get $S \Rightarrow w\lambda = w$ *i.e.* $w \in L(G)$.

Thus $L(G) = N(M)$.

3.3.2. Example

Consider a context free grammar given by the production rules $S \to aAA$, $A \to aS/bS/a$.

That is $G = (N, T, S, P)$ where $N = \{S, A\}$, $T = \{a, b\}$ and $P = \{S \to aAA, A \to aS, A \to bS, A \to a\}$.

Let $M = (\{q_0\}, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where $\Sigma = T = \{a, b\}$, $\Gamma = N \cup T = \{S, A, a, b\}$, $F = \phi$ and δ is defined as

$$\delta(q_0, \lambda, S) = \{(q_0, AAa)\}$$

$$\delta(q_0, \lambda, A) = \{(q_0, Sa), (q_0, Sb), (q_0, a)\}$$

$$\delta(q_0, a, a) = \{(q_0, \lambda)\}$$

$$\delta(q_0, b, b) = \{(q_0, \lambda)\}$$

It can be easily verified that $L(G) = N(M)$.

3.3.3. Theorem

If $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ is a PDA, then there exists a context free grammar G such that $L(G) = N(M)$.

Proof. We first give the construction of G and prove that $N(M) = L(G)$.

We define $G = (N, T, S, P)$ where

$N = \{[q, z, q'] / q, q' \in Q, Z \in \Gamma\} \cup \{S\}$

i.e. any element of N is either the new symbol S acting as the start symbol for G or an ordered triple whose first and third elements are states and the second element is a pushdown symbol.

The productions in P are induced by moves of PDA as follows.

(*i*) S-production are given by $S \to [q_0, z_0, q]$ for every q in Q.

(*ii*) Each move erasing a pushdown symbol given by $(q', \lambda) \in \delta(q, a, z)$ induces the production $[q, z, q'] \to a$

(*iii*) Each more not erasing a pushdown symbol given by $(q_1, z_m \ldots z_1) \in \delta(q, a, z)$ induces many productions of the form

$$[q, z, q'] \to a\,[q_1, z_1, q_2]\,[q_2, z_2, q_3] \ldots [q_m, z_m, q']$$

where each of the states $q', q_1, q_2, \ldots, q_m$ can be any state in Q. Each move yields many productions because of (*iii*). We now prove that $L(G) = N(M)$.

We know that a variable $[q, z, q']$ indicates that for the PDA the current state is q and the topmost symbol in PDA is z. In the course of a derivation, a state q' is chosen in such a way that PDS is emptied ultimately. This corresponds to applying (*ii*).

To prove $N(M) = L(G)$, we need an auxillary result.

$$[q, z, q'] \overset{*}{\underset{G}{\Rightarrow}} w \qquad \ldots(3.8)$$

if any only if

$$(q, w, z) \vdash^{*} (q', \lambda, \lambda) \qquad \ldots(3.9)$$

We prove the if part by induction on the number of steps in (3.9).

If $(q, w, z) \vdash (q', \lambda, \lambda)$, then w is either a in Σ or λ. So we have

$$(q', \lambda) \in \delta(q, w, z).$$

By (3.9) we get a prodcution $[q, z, q'] \to w$. So, $[q, z\, q'] \overset{*}{\underset{G}{\Rightarrow}} w$. Thus there is a basis for induction.

Let us assume the result viz., (3.9) implies (3.9) when the former has less than k moves. Consider $(q, w, z) \overset{k}{\vdash} (q', \lambda, \lambda)$. This can be split as

$$(q, aw'\ z) \vdash (q, w', z_m z_2 \ldots z_1) \overset{k-1}{\vdash} (q', \lambda, \lambda) \qquad \ldots(3.10)$$

where $w = aw$; and $a \in \Sigma$ or $a = \lambda$, depending on the first move.

Consider the second part of (31.0). This means that the PDS has $z_m \ldots z_2 z_1$ initially and on application of w, the PDS is emptied. Each move of PDA can either erase the topmost symbol on the PDS or replace the topmost symbol by

some non-empty string. So several moves may be required for getting z_2 on the top of PDS. Let w_1 be the prefix of w such that the PDS has $z_m \ldots z_2 z_1$ after the application of w_1. We can note that $z_m \ldots z_2 z_1$ are not disturbed while applying w_1. Let w_i be the substring of w such that the PDS has $z_m \ldots z_{i+1}$ on application of w_i. $z_m \ldots z_{i+1}$ is not disturbed while applying $w_1, w_2, \ldots, w_i$.

In terms of ID's we have

$$(q_i, w_i, z_i) \overset{*}{\vdash} (q_{i+1}, \lambda, \lambda) \text{ for } i = 1, 2, \ldots, m,\ q_{m+1} = q' \qquad \ldots(3.11)$$

As each move in (3.11) requires less than k steps, by induction hypothesis we have

$$[q_i, z_i, q_{i+1}] \overset{*}{\Rightarrow} w_i \text{ for } i = 1, \ldots, m \qquad \ldots(3.12)$$

The first proof of (3.10) is given by

$(q_1, z_m z_2 \ldots z_1) \in \delta(q, a, z)$. By (3.10) we get a production.

$$[q, z, q'] \to a\,[q_1, z_1, q_2]\,[q_2, z_2, q_3] \ldots [q_m, z_m, q'] \qquad \ldots(3.13)$$

From (3.12) and (3.13) we get

$$[q, z, q'] \overset{*}{\Rightarrow} aw_1 w_2 \ldots w_m = w.$$

By the principle of induction (13.9) $\Rightarrow$ (13.8)

We prove the only if part by induction on the number of steps in the derivation of (3.8). Suppose $[q, z, q'] \Rightarrow q$. Then $[q, z, q'] \to w$ is a production in P. This production is obtained by 2. So $w = \lambda$ or $w \in \Sigma$ and $(q', \lambda) \in \delta(q, w, z)$. This gives the move $(q, w, z) \vdash (q, \lambda, \lambda)$. Thus there is a basis for inudction.

Assume the result for derivations where the number of steps is less than k. Consider $[q, z, q'] \overset{k}{\vdash} w$. This can be split as

$$[q, z, q'] \Rightarrow a\,[q, z_m, q_2]\,[q_2, z_{m-1}, q_3] \ldots [q_m, z_1, q'] \overset{k-1}{\vdash} w \qquad \ldots(3.14)$$

As G is context free we can write $w = aw_1 w_2 \ldots w_m$, where

$$[q_i, z_i, q_{i+1}] \overset{*}{\underset{G}{\Rightarrow}} w_i \text{ and } q_{m+1} = q'$$

By induction hypothesis, we have

$$(q_i, w_i, z_i) \vdash^{*} (q_{i+1}, \lambda, \lambda) \text{ for } i = 1, 2, \ldots, m \qquad \ldots(3.15)$$

By applying 3.1.8. and 3.1.9., we get

$$(q_i, w_i, z_m \ldots z_{i+1}\ z_i) \vdash^{*} (q_{i+1}, \lambda, z_m \ldots z_{i+1})$$

$$(q_i, w_i\ w_{i+1} \ldots w_m, z_m \ldots z_i) \vdash (q_{i+1}, w_{i+1} \ldots w_m, z_m \ldots z_{i+1}) \qquad \ldots(3.16)$$

By combining the moves given by (3.15), we get

$$(q_1, w_1w_2 \ldots w_m, z_m \ldots z_1) \overset{*}{\vdash} (q', \lambda, \lambda) \quad \ldots(3.17)$$

The first step in (3.13) is induced by

$$(q_1, z_m \ldots z_1) \in \delta\ (q, a, z).$$

The corresponding move is

$$(q, a, z) \vdash (q_1, \lambda, z_m \ldots z_1)$$

By applying property 3.1.8., we get

$$(q, aw_1, \ldots w_m, z) \overset{*}{\vdash} (q_1, w_1 \ldots w_m, z_m \ldots z_1) \quad \ldots(3.18)$$

From (3.18) and (3.19), we get $(q, w, z) \overset{*}{\vdash} (q', \lambda, \lambda)$. By the principle of induction (3.8) $\Rightarrow$ (3.9).

Thus we have proved the auxillary result. In particular,

$$[q_0, z_0, q'] \overset{\cdot}{\Rightarrow} w \text{ iff } (q_0, w, z_0) \overset{*}{\vdash} (q', \lambda, \lambda) \quad \ldots(3.19).$$

Now $w \in L\ (G)$ iff $S \overset{\cdot}{\Rightarrow} w$

$$S \Rightarrow [q_0, z_0, q'] \overset{\cdot}{\Rightarrow} w \text{ (for some } q' \text{ by } (i)$$

iff $(q_0, w, z_0) \overset{*}{\vdash} (q', \lambda, \lambda)$ by the auxillary result.

iff $w \in N\ (A)$

iff $N\ (M) = L\ (G)$.

Thus

3.3.4. Corrollary

If A is a PDA, then there exists a CFG such that $T\ (A) = L\ (G)$.

Proof. By Theorem 3.2.3. we can find a PDA A' such that $T\ (A) = N\ (A')$. By Theorem 3.3.3. we can construct a grammar G such that $N\ (A') = L\ (G)$. Thus $T\ (A) = L\ (G)$.

3.3.5. Problem

Given a grammar for the language $N\ (M)$ where $M = (\{q_0, q_1\}, \{0, 1\}, \{z_0, X\}, \delta, q_0, z_0\ \phi)$ and δ is given by

$$\delta\ (q_0, 0, z_0) = \{(q_0, xz_0)\}$$

$$\delta\ (q_1, \lambda, X) = \{(q_1, \lambda)\}$$

$$\delta\ (q_0, 1, X) = \{(q_1, \lambda)\}$$

$$\delta\ (q_0, 1, X) = \{(q_1, \lambda)\}$$

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

$$\delta(q_1, \lambda, z_0) = \{(q_1, \lambda)\}$$

Solution. Let $G = (N, T, S, P)$

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

$N = \{[p, A, q]$ / for all $p, q \in k$ and $A \in z\} \cup \{S\}$

$= \{[q_0, z_0, q_1], [q_1, z_0, q_0]\ [q_0, x, q_1], [q_1, x, q_0], [q_0, z_0, q_0], [q_0, x, q_0], [q_1, z_0, q_1], S\}$, S is a new symbol.

P ; $[q_1, X, q_1], S \rightarrow [q_0, z_0, q_0]$.

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

Next we add productions of the variable $[q_0, z_0, q_0]$.

These are $[q_0, z_0, q_0] \rightarrow 0\ [q_0, X, q_0]\ [q_0, z_0, q_0]$

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

These productions are required by

$$\delta\ (q_0, 0, z_0) = \{(q_0, X\ z_0)\}$$

Next the productions for $[q_0, z_0, q_1]$ are

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

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

These are also required by $\delta\ (q_0, 0, z_0) = \{(q_0, Xz_0)\}$

The productions for the remaining variable and the relevant moves of the PDA are

1. $[q_0, X, q_0] \rightarrow 0\ [q_0, X, q_0]\ [q_0, X, q_0]$

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

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

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

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

2. $[q_0, X, q_1] \rightarrow 1$ since $\delta\ (q_0, 1, X) = \{(q_1, \lambda)\}$

3. $[q_1, z_0, q_1] \rightarrow \lambda$ since $\delta\ (q_1, \lambda, z_0) = \{(q_1, \lambda)\}$

4. $[q_1, X, q_1] \rightarrow \lambda$ since $\delta\ (q_1, \lambda, X) = \{(q_1, \lambda)\}$

5. $[q_1, X, q_1] \rightarrow 1$ since $\delta\ (q_1, 1, X) = \{(q_1, \lambda)\}$

It should be noted that there are no productions for the variables $[q_1, X, q_0]$ and $[q_1, z_0, q_0]$. As all the productions for $[q_0, X, q_0]$ and $[q_0, z_0, q_0]$ have $[q_1, X, q_0]$ or $[q_1, z_0, q_0]$ on the right, no terminal string can be derived from $[q_0, X, q_0]$ or $[q_0, z_0, q_0]$ either. Deleting all productions involving one of these four variables on either the right or left , we end up with the following productions.

$$S \rightarrow [q_0, z_0, q_1]$$
$$[q_0, z_0, q_1] \rightarrow 0\,[q_0, X, q_1]\,[q_1, z_0, q_1]$$
$$[q_1, X, q_1] \rightarrow 0\,[q_0, X, q_1]\,[q_1, X, q_1]$$
$$[q_0, X, q_1] \rightarrow 1$$
$$[q_1, z_0, q_1] \rightarrow \lambda$$
$$[q_1, X, q_1] \rightarrow \lambda$$
$$[q_1, X, q_1] \rightarrow 1$$

3.4. Pumping Lemma For CFL

The pumping lemma for regular set states that every sufficiently long string in a regular set contains a short substring that can be pumped. That is, inserting as many copies of the substring as we like always yields a string in the regular set.

The pumping lemma for CFL's states that there are always two short substrings close together that can be repeated, both the same number of times as often as we alike. The construction we make use of in proving pumping lemma yields some decision algorithms regarding CFLs.

3.4.1. Lemma

Let G be a context free grammar in CNF and T be a derivation tree in G. If the length of the longest path in T is less than or equal to k, then the yield of T is of length less than or equal to 2^{k-1}.

Proof. We prove the result by induction on k, the length of the longest path for all A-trees (Recall an A-tree is a derivation tree whose root has a label A).

When the longest path in an A-tree is of length 1, the root has only one son whose label is a terminal (when the root has two sons, the labels are variables). So the yield is of length *1*. Thus there is a basis for induction.

Assume the result for $k-1$ ($k > 1$). Let T be an A-tree with the longest path of length less than or equal to k. As $k > 1$, the root of T has exactly two sons with labels A_1 and A_2. The two subtrees with the two sons are roots have longest paths of length less than or equal to $k-1$.

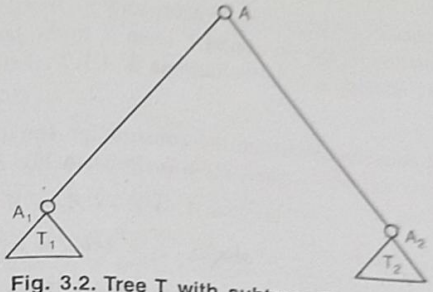

Fig. 3.2. Tree T with subtrees T_1 and T_2.

If w_1 and w_2 are their yields, then by induction hypothesis, $|w| \leq 2^{k-2}$, $|w_2| \leq 2^{k-2}$.So the yield of $T = w_1 w_2$, $|w_1 w_2| \leq 2^{k-2} + 2^{k-2} = 2^{k-1}$. By the priniciple of induction, the result is true for all A-trees, and hence for all derivation trees.

3.4.2. Theorem

(**Pumping lemma for context free languages**). Let L be a context free language. Then we can find a natural number n such that

(*i*) every $z \in L$ with $|z| \geq n$ can be written as $uvwxy$ for some strings u, v, w, x, y.

(*ii*) $|vx| \geq 1$.

(*iii*) $|vwx| \leq n$.

(*iv*) $uv^k wx^k y \in L$ for all $k \geq 0$.

Proof. We can decide whether or not $\lambda \in L$. When $\lambda \in L$, we consider $L - \{\lambda\}$ and construct a grammar $G = (N, T, S, P)$ in CNF generating $L - \{\lambda\}$ (when $\lambda \notin L$, we construct G in CNF generating L).

Let $|N| = m$ and $n = 2^m$. To prove that n is the required number, we start with $z \in L$, $|z| \geq 2^m$, and construct a derivation tree T (parse tree) of z. If the length of a longest path in T is atmost m, by 3.4.1, $|z| \leq 2^{m-1}$ (since z is the yield of T). But $|z| \geq 2^m > 2^{n-1}$. So T has a path say Γ of length greater than or equal to $m+1$. Γ has atleast $m+2$ vertices and only the last vertex is a leaf. Thus in Γ all the labels except the last one are variables. As $|N| = m$, some label is repeated.

We choose a repeated label as follows. We start with leaf of Γ and travel along Γ upwards. We stop when some label, say B is reapeated. (Among several repeated labels B is the first). Let v_1 and v_2 be the vertices with label B, v_1 being nearer the root. In Γ, the portion of the path from v_1 to leaf has only one label viz., B, which is repeated, and so its length is atmost $m+1$.

Let T_1 and T_2 be subtrees with v_1, v_2 as roots and z_1, w as yields respectively. As Γ is a longest path in T, the portion of Γ from v_1 to the leaf is a longest path in T_1 and the length atmost $m + 1$. By Lemma 3. 4.1, $| z_1 | \leq 2^m$ (since z_1 is the yield of T_1).

For better understanding we illustrate the construction for the grammar whose productions are $S \to AB$, $A \to aB/a$, $B \to bA/b$ as in Fig 3.2. In the figure

$$\Gamma = S \to A \to B \to A \to B \to b$$

$$z = ababb, \quad z_1 = bab, \quad w = b$$

$$v = ba, \; x = \lambda, \; u = a, \; y = b$$

As z and z_1 are the yields of T and a proper subtree T_1 of T, we can write $z = uz_1y$. As z_1 and w are the yields of T, and a proper subtree T_2 of T_1, we can write $z_1 = vwx$. Also $| vwx | > | w |$. So $| vx | \geq 1$. Thus we have $z = uvwxy$ with $| vwx | \leq n$ and $| vx | \geq 1$. This proves points (*i*)-(*iii*) of the theorem.

As T is an S–tree and T_1, T_2 are B–trees, we get $S \overset{*}{\Rightarrow} u\,By$, $B \overset{*}{\Rightarrow} vBx$ and

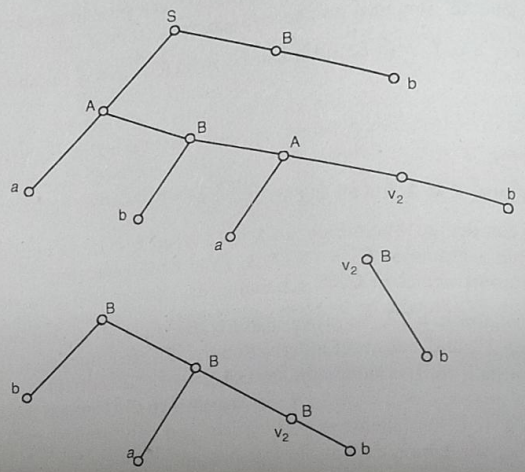

Fig. 3.3.

Tree T and its subtrees T_1 and T_2

$B \overset{*}{\Rightarrow} w$. As $S \overset{*}{\Rightarrow} uBy \Rightarrow Uwy$, $uv^ow\, x^oy \in L$.

For $k \geq 1$,

$S \overset{*}{\Rightarrow} uBy \overset{*}{\Rightarrow} uv^k\, Bx^k y \overset{*}{\Rightarrow} uv^k wx^k y \in L$. This proves (*iv*) of theorem.

3.4.3. Corollary

Let L be a context free language and n be the natural number obtained by using pumping lemma. Then (*a*) $L \neq \phi$ if and only if there exists $w \in L$ with $| w | < n$, and (*b*) L is infinite if and only if there exists $z \in L$ such that $n \leq | z | < 2n$.

Proof. (*a*) We have to prove the 'only if ' part. If $z \in L$ with $| z | \geq n$, we apply pumping lemma to write $z = uvwxy$, where $1 \leq | vx | \leq n$. Also, $uwy \in L$ and $| uwy | < | z |$. Applying pumping lemma repeatedly we can get $z' \in L$ such that $| z' | < n$. Thus (*a*) is proved.

(*b*) If $z \in L$ such that $n \leq | z | < 2n$, by pumping lemma we can write $z = uvwxy$. Also $uv^k wx^k y \in L$ for all $k \geq 0$. Thus we get infinite number of elements in L. conversely, if L is infinite, we can find $z \in L$ with $| z | \geq n$. If $| z | < 2n$, there is nothing to prove. Otherwise, we can apply pumping lemma to write $z = uvwxy$ and get $uwy \in L$. Every time we apply pumping lemma to get a smaller string and the decrease in length is atmost n (being equal to $| vx |$). So we ultimately get a string z' in L such that $n \leq | z' | < 2n$. This proves (*b*).

3.4.3. Application of pumping Lemma

We use pumping lemma to show that a language L is not a context free language. We assume that L is context free. By Applying pumping lemma we get a contradiction.

The procedure can be carried out using the following steps.

Step 1. Assume L is context free. Let n be the natural number obtained by using the pumping lemma.

Step 2. Choose $z \in L$ so that $| z | \geq n$. Write $z = uvwxy$ using the pumping lemma.

Step 3. Find a suitable k so that $uv^k wx^k y \notin L$. This is a contradiction, and so L is not context free.

Example 1. Show that $L = \{a^n b^n c^n / n \geq 1\}$ is not context free but context sensitive.

Solution. We know that

$G = (\{S, B, C\}, \{a, b, c\}, S, \{S \to aSBC/aBC, CB \to BC, aB \to ab, bB \to bb, bC \to bc, cC \to cc\})$ generates L. We note that in every string of L any symbol appears the same number of times as any other symbol. Also, a cannot appear after b and c cannot appear before b and so on.

Step 1. Assume that L is context free. Let n be the natural number obtained by using pumping Lemma.

Step 2. Let $z = a^n b^n c^n$. Then $| z | = 3n > n$. Write $z = uvwxy$, where $| vx | \geq 1$ *i.e.*, atleast one of v or x is not λ.

Step 3. $uvwxy = a^n b^n c^n$. As $1 \leq | vx | \leq n$, v or x contain all the three symbols a, b, c. So (*i*) v or x of the form $a^i b^j$ (or $b^i c^j$) for some i, j such that $i + j \leq n$ or (*ii*) v or x is a string formed by the repetition of only one symbol among a, b, c.

When v or x is of the form $a^i b^j$, $v^2 = a^i b^j a^i b^j$ (or $x^2 = a^i b^j a^i b^j$).As v^2 is a substring of uv^2wx^2y, uv^2wx^2y cannot be of the form $a^m b^m c^m$.So $uv^2wx^2y \notin L$. When both v and x are formed by the repetition of a single symbol (*e.g.* $u = a^i$ and $v = b^j$ for some i and j, $i \leq n$, $j \leq n$), the string uwy will contain the remaining symbol, say a_1. Also a_1^n will be a substring of uwy as a_1 does not occur in v or x. The number of occurrences of one of the other two symbols in uwy is less than n. (recall $uvwxy = a^n b^n c^n$), and n is the number of occurrences of a_1. So $uv^0wx^0y = uwy \notin L$.

Thus for any choice of v or x, we get a contradiction. Therefore L is not context free.

Example 2. Prove that $L = \{0^p / p \text{ is prime}\}$ is not a context free language.

Solution. We use the following property of L. If $w \in L$, the $| w |$ is prime.

Step 1. Suppose $L = L(G)$ is context free. Let n be the natural number obtained by using the pumping lemma.

Step 2. Let p be a prime number greater than n. Then $z = o^p \in L$. we write $z = uvwxy$.

Step 3. By pumping lemma, $uv^0wx^0y = uwy \in L$. so $| uwy |$ is a prime number, say q. Let $| vx | = r$. Then $| uv^q wx^q y | = q + qr$. As $q + qr$ is not a prime, $uv^q wx^q y \notin L$. This is a contradiction. Therefore, L is not context free.

3.4.4. Decision Algorithm for context free language

In this section we give some decision algorithm for context free languages and regular sets.

(*i*) **Algorithm for deciding whether a context free language is empty.** We can apply the construction given in 2.4.1. for getting $N' = w_k$. L is non-empty if and only if $S \in W_k$.

(*ii*) **Algorithm for deciding whether a context free language L is finite.** Construct a non-redundant context free grammar G in CNF generating $L - \{\lambda\}$. We draw a directecd graph whose vertices are variables in Γ.

If $A \to BC$ is a production, there are directed edges from A to B and A to C. L is finite if and only if the directed graph has no cycles.

(*iii*) **Algorithm for deciding whether a regular language L is empty.** Construct a deterministic finite automaton M accepting L. We construct the set of all states reachable from q_0 by applying a single input symbol. These states are arranged as a row under columns corresponding to every input symbol. The construction is repeated for every state appearing in an earlier row. The construction terminates in a finite number of steps. If a final state appears in this tabular column, then L is non-empty. (Actually, we can terminate the construction as soon as some final state is obtained in the tabular column). Otherwise, L is empty.

(*iv*) **Algorithm for deciding whether a regular language L is infinite.** Construct a deterministic finite automaton M accepting L. L is infinite if and only if M has a cycle.

3.4.5. Theorem

The language $L = \{x \in \{a, b\}^* / x = x^r\}$ cannot be accepted by any DPDA.

Proof. Suppose for the sake of contradiction that $M = (Q, \{a, b\}, \Gamma, q_0, z_0, A, \delta)$ is a DPDA accepting L. We can easily modify M if necessary so that every move is either one of the form

$$\delta(p, s, X) = (q, \lambda)$$

or one of the form

$$\delta(p, s, X) = (q, \alpha, X)$$

where $s \in \{a, b, \lambda\}$ and $\alpha \in \Gamma^*$. The effect of modification is that M can still remove a symbol from the stack or place another symbol on the stack, but if cannot do both in the same move.

We observe next that for any string x, M must eventually read every symbol of x. The string xx^r, for example is a palindrome, and M must read it completely in order to accept it. However becuase M is deterministic, the moves it makes while processing x in the course of accepting xx^r are exactly the moves it must make on input x, whether or not x is followed by anything else.

After M has processed a string x, its stack is in a certain height, depending on how much M needs to remember about x. We may consider how much shorter the stack can ever get as a result of reading subsequent input symbols. For each x, let y_x be a string for which this resulting stack height is as small as possible. (It cannot be 0, because M must still be able to process longer strings with prefix xy_x.) In other words,

$$(q_0, xy_x, z_0) \overset{*}{\vdash} (q_x, \lambda, \alpha_x)$$

for some state q_x and some string $\alpha_x \in \Gamma^*$ with $|\alpha_x| > 0$, and for any string y, any state p, and any string $\beta \in \Gamma^*$,

$$\text{if } (q_0, xy, z_0) \overset{*}{\vdash} (p, \lambda, \beta) \text{ then } |\beta| \geq |\alpha_x|$$

Because of our initial assumption about the moves of M, any move that involves removing a stack symbol decreases the stack height. Therefore, once M reaches the configuration (q_x, λ, α_x), as a result of processing the string xy_x, no symbols of the string α_x will ever be removed from the stack subsequently.

Let A_x be the first symbol of the string α_x. The set S of all strings of the form xy_x is infinite (because we may consider x of any length) and the set of all ordered pairs (q_x, A_x) is finite (because the entire set $Q \times \Gamma$ is finite). Therefore, there must be an infinite subset T of S so that for all the elements xy_x of T, the pairs (q_x, A_x) are equal. In particular, we can choose two different strings $u_1 = xy_x$ and $u_2 = wy_w$ so that

$$(q_0, u_1, z_0) \vdash (q, \lambda, A\beta_1)$$

and

$$(q_0, u_2, z_0) \overset{*}{\vdash} (q, \lambda, A\beta_2)$$

for some $q \in Q$, some $A \in \Gamma$, and some strings $\beta_1, \beta_2 \in \Gamma^*$. If we now look at longer strings of the form u_1z and u_2z, we have

$$(q_0, u_1z, z_0) \overset{*}{\vdash} (q, z, A\beta_1)$$

and

$$(q_0, u_2z, z_0) \overset{*}{\vdash} (q, z, A\beta_2)$$

and the symbol A is never removed from the stack as a result of processing the string z. The reason this is useful is that although the stack contents may not be the same in the two cases, the machine can never notice the difference. The ultimate result of processing the string u_1z must be exactly the same as that of processing u_2z. Either both strings are accepted or neither is. Now however, we have the contradiction we are looking for. On the one hand, M treats u_1z and u_2z the same way; on the otherhand, the two strings u_1 and u_2 are distinguishable with respect to L, so that for some z, one of the strings u_1z, u_2z is in L and the other is not. This is impossible if M accepts L.

3.4.6. Theorem

The intersection of a context free language with a regular language is a context free language.

Proof. If L is a context free language and R is a reuglar set, then $L = L(M_1)$ for some $R = L(M_2)$ for some deterministic finite state automata $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$.

The idea is to combine these machines into a single pushdown automaton M that carries out computations by M_1 and M_2 in parallel and accepts only if both would have accepted.

Specifically, let

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

where

$$Q = Q_1 \times Q_2$$

$$\Sigma = \Sigma_1 \cup \Sigma_2$$

$$\Gamma = \Gamma_1$$

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

$$F = F_1 \times F_2$$

$$Z_0 = Z_1$$

and the transition function δ is defined by

$$\delta(\{q_1, q_2\}, u, \beta) = \{(\{p_1, p_2\}, \nu)\} \text{ iff}$$

$$\delta_1(q_1, u, \beta) = \{(p_1, \nu)\} \qquad \ldots(3.20)$$

and

$$\delta(q_2, u) \overset{*}{\underset{M}{\vdash}} (p_2, \lambda) \qquad \ldots(3.21)$$

That is M passes from state (q_1, q_2) to state $\{p_1, p_2\}$ in the same way that M_1, passes from state q_1 to p_1, except that in addition M keeps track of the change in the state of M_2 caused by reading the same input. (3.20) is easy to check, since a deterministic finite automaton reads a single symbol on each move. Thus, in practice , we could construct M by repeatedly choosing a transition $\delta(q_1, u, \beta) = \{(p, \gamma)\}$ of M_1 of a state q_2 and M_2 and simulating M_2 for $|u|$ steps on input u to determine what state p_2 it would reach after reading that input.

3.4.7. Example

Let L consist of all strings of *a's* and *b's* with equal numbers of *a's* and *b's* but not containing substring *abaa* or *babb*. Then L is context free, since it is the intersection of the language accepted by pushdown automaton and with the regular language $\{a, b\}^* - \{a, b\}^* \{abaa, babb\} \{a, b\}^*$.

3.5. DETERMINISTIC CONTEXT FREE LANGUAGES (DCFL) AND DETERMINISTIC PUSHDOWN AUTOMATA (DPDA)

3.5.1. Definition

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a pushdown automaton. M is deterministic if there is no configuration for which M has a choice of more than one move.

In otherwords, M is deterministic if it statistics both the following conditions

1. For any $q \in Q$, $a \in \Sigma \cup \{\lambda\}$ and $X \in \Gamma$, the set $\delta(q, a, X)$ has atmost one element.
2. For any $q \in Q$ and $X \in \Gamma$, if $\delta(q, \lambda, X) \neq \phi$, then $\delta(q, a, X) = \phi$ for every $a \in \Sigma$.

A language L is said to be a *Deterministic Context Free Language* (DCFL) if and only if there is a deterministic PDA (DPDA) such that $L = L(M)$.

3.5.2. Example

Consider a PDA

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

where $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $\Gamma = \{z_0, a, b\}$, $F = \{q_1\}$

and δ is defined as follows.

$$\begin{aligned}
\delta(q_0, a, z_0) &= \{(q_1, z_0)\} \\
\delta(q_0, b, z_0) &= \{(q_0, bz_0)\} \\
\delta(q_0, a, b) &= \{(q_0, \lambda)\} \\
\delta(q_0, b, b) &= \{(q_0, bb)\} \\
\delta(q_1, a, z_0) &= \{(q_1, az_0)\} \\
\delta(q_1, b, z_0) &= \{(q_0, z_0)\} \\
\delta(q_1, a, a) &= \{(q_1, aa)\} \\
\delta(q_1, b, a) &= \{(q, \lambda)\}
\end{aligned}$$

Clearly M is a DPDA.

The syntax of many programming languages can be described by means of Deterministic context free languages (DCFL's). A compiler be described by means of a context-free grammar of restricted form, which are nothing but the *LR*-grammars. The *LR*-grammars have the property that they generate exactly the DCFL's.

DPDA's are in *normal form*, is like a PDA's are in normal form, where the only stack operations are to erase the top symbol or to push one symbol.

3.5.3. Lemma

Every DCFL is $L(M)$ for a DPDA

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ such that if $\delta(q, a, X) = (p, v)$ then $|v| \leq 2$.

Proof. Suppose $\delta(q, a, X) = (r, v)$ and $|v| > 2$. Let $v = X_1 X_2 \ldots X_n$, with $n \geq 3$,

create new non-accepting state $p_1, p_2, \ldots, p_{n-2}$ and redefine

$$\delta(q, a, X) = (p_1, y_{n-1}, y_n)$$

$$\delta(p_i, \varepsilon, y_{n-i}) = (p_{i+1}, y_{n-i-1}, y_{n-i})$$

for all $1 \leq i \leq n-3$, and $\delta(p_{n-2}, \varepsilon, y_2) = (r, y_1, y_2)$.

Now $(q, a, X) \vdash (p_1, \varepsilon, y_{n-1}\, y_n) \vdash (p_2, \varepsilon, y_{n-2} y_{n-1}\, y_n)$

$\vdash (p_3, \varepsilon, y_{n-3}\, y_{n-2}\, y_{n-1}\, y_n) \overset{*}{\vdash} (p_{n-2}, \varepsilon, y_2\, y_3 \ldots y_n)$

$\vdash (r, \varepsilon, y_1\, y_2 \ldots y_n)$.

That is, in state q, on input a, with X on top of the stack, the revised DPDA still replaces X by $v = y_1 y_2 \ldots y_n$ and enters state r but it takes $(n-1)$ moves to do so. Hence a contradiction.

The above lemma shows that the DPDA can never push more than one symbol per move.

3.5.4. Lemma

Every DCFL is $L(M)$ for a DPDA

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ such that if $\delta(q, a, X) = (p, v)$, then v is either ε, (a, pop), X (no stack move), or of the form $Y X$ (a push) for some stack symbol y.

Proof. Assume $L = L(M')$, where $M' = (Q', \Sigma \Gamma, \delta', q_0', x_0, F')$ satisfies Lemma 3.5.3. Construct M simulate M' while keeping the top stack symbol of M' in M's control. Formally, let

$Q = Q' \times \Gamma'$, $q_0 = [q_0', X_0]$, $F = F \times \Gamma'$ and $\Gamma = \Gamma' \cup \{Z_0\}$, where Z_0 is a new symbol not in Γ'. Define δ by:

(*i*) $\delta([q, X], a, y) = ([p, Y], \varepsilon)$, if $\delta'(q, a, X) = (p, \varepsilon)$, for all y. That is, if M' pops its stack, M also pops its stack.

(*ii*) $\delta([q, X], a\, Z) = ([p, Y], Z)$, if $\delta'(q, a, X) = (p, Y)$, for all Z. That is, of M' changes its top stack symbol, then M records the change in its own control, but does not alter the change.

(*iii*) $\delta([q, X], a, W) = ([p, y], Z\, W)$ if

$\delta'(q, a, X) = (p, y\, W)$, for all W.

By induction it can be shown that

$$(q_0', w, X_0) \overset{*}{\underset{M'}{\vdash}} (q, \varepsilon, X_1 X_2 \ldots X_n)$$

if and only if

$$([q_0', X_0)], W, Z_0) \overset{*}{\vdash}_M ([q, X_1], \varepsilon, X_2 \ldots X_n Z_0).$$

$$L(M) = L(M').$$

Thus

3.5.5. Lemma

Let M be a DPDA. There exists an equivalent DPDA M' such that on every input, M' scans the entire input.

Proof. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Define $M' = (Q \cup \{q_0', d, f\}, \Sigma, \Gamma \cup \{X_0\}, \delta', q_0', X_0, F \cup \{f\})$ d is a dead state, where no next move is possible. On any input symbol, the only transition from d to d occurs, but no change of stack occurs. The action of δ' is as follows:

(*i*) $\delta'(q_0', \varepsilon, X_0) = (q_0, Z_0 X_0)$.

(*ii*) If $\delta(q, a, Z) = \phi$, $\delta(q, \varepsilon, Z) = \phi$, for some $q \in Q$, $a \in \Sigma$, $Z \in \Gamma$, then $\delta'(q, a Z) = (d, Z)$.

Also for all $q \in Q$, $a \in \Sigma$, $\delta'(q, a X_0) = (d, X_0)$

(*iii*) $\delta'(d, a, Z) = (d, Z)$, for all $a \in \Sigma$, $Z \in \Gamma \cup \{X_0\}$

(*iv*) If $(q, \varepsilon, Z) \overset{*}{\vdash} (q_i, \varepsilon, \nu_i)$, then

$$\delta'(q, \varepsilon, z) = (d, z), \text{ provided no } q_i \text{ is final}$$

and $\delta'(q, \varepsilon, z) = (f, z)$, whenever one or more of the q_i's one final.

(*v*) $\delta'(f, \varepsilon, z) = (d, z)$ for all $z \in \Gamma \cup \{X_0\}$.

(*vi*) For any $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $z \in \Gamma$, if $\delta'(q, a, z)$ has not been defined by (*ii*) or (*iv*), then define $\delta'(q, a, z) = \delta(q, a, z)$.

Then $L(M) = L(M')$. M' uses all its input. Suppose, some input y is not used. Then we have an ID of the form $(q, y, Z_1, Z_2 \ldots Z_k X_0)$, such that

$$(q_0', xy, X_0) \overset{*}{\vdash}_M (q, y, Z_1 Z_2, \ldots Z_k X_0).$$

By (*ii*) it is not possible that M' halts. By (*iv*) it is not possible that M' makes an infinite sequence of ε-moves without erasing z_1. M' erases eventually enter an ID (q', y, X_0).

By rule (*ii*), $(q', y, X_0) \vdash_M (\alpha, y', X_0)$, where $y = ay'$; $a \in \Sigma$.

Hence M' reads all the inputs.

3.5.6. Theorem

The complement of DCFL is a DCFL.

Proof. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a DPDA satisfying Lemma 3.5.5. Let $M' = (Q', \Sigma, \Gamma, \delta', q_0, Z_0, F')$ be a DPDA simulating M, where

$$Q' = \{[q, k], q \in Q, k = 1, 2, 3\}$$

$$F' = \{[q, 3] = q \in Q\},$$

$$q_0' = \begin{cases} [q_0, 1] \text{ if } q_0 \in F \\ [q_0, 2] \text{ if } q_0 \notin F \end{cases}$$

Construct Boolean-valued tables T_1, T_2, T_3 such taht

$T_i(q, z, p)$ is true $\Leftrightarrow (q, \varepsilon, z) \overset{*}{\vdash}_M (p, \varepsilon, z)$ for $i = 1, 2, 3$. T_i can be made true by induction.

Basis Step. $T_3(q, z, p)$ = true if $\delta(q, \varepsilon, z) = (p, yz)$

$$T_1(q, z, p) = T_3(q, z, p) = \textit{true} \text{ if}$$

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

$$T_2(q, z, p) = \textit{true} \text{ if } \delta(q, \varepsilon, z) = (p, \varepsilon).$$

Induction Step. Whenever $\delta(q, \varepsilon, z) = (r, y, yz)$,

then

(*a*) If $T_2(r, y, s)$ and $T_2(s, z, p)$ are *true*, then set $T_2(q, z, p)$ = true.

(*b*) If $T_2(r, y, s)$ and $T_1(s, z, p)$ are *true*.

set $T_1(q, z, p)$ = true.

(*c*) If $[T_2(r, y, s)$ and $T_3(s, z, p)]$ or $[T_1(r, y, s)$ and $T_3(s, y, p)]$ are true, set $T_3(q, z, p)$ = true.

(*d*) If $T_3(r, y, p)$ is true, set $T_3(q, z, p)$ = true.

2. Whenever $\delta(q, \varepsilon, z) = (r, z)$, then

(*a*) If $T_1(r, z, p)$ is true, set $T_1(q, z, p)$ = true.

(*b*) If $T_2(r, z, p)$ is true, set $T_2(q, z, p)$ = true.

(*c*) If $T_3(r, z, p)$ is true, set $T_3(q, z, p)$ = true.

Here k in $[q, k]$ record between true inputs, whether or not M has entered an accepting state. If M has entered an accepting state since the last true input, then $k = 1$. If M does not enter an accepting state since the last true input, then $k = 2$.

If $k = 1$, when M reads a true input symbol, then M' simulates M and changes k to 1 or 2 depending on the new state of M is in F or not in F. If $k = 2$, M' changes k to 3, and then simulates M and changes k to 1 or 2, depending on the new state of M is in F or not in F. Thus δ' is defined as follows:

(*i*) $\delta'([q, k], \varepsilon, z) = ([p, k'], v)$
if $\delta(q, \varepsilon, z) = (p, v)$, for $k = 1$ or 2, and $k' = 1$ if $k = 1$; $k' = 2$ if $k = 2$.

(*ii*) $\delta'([q, 2], \varepsilon, z) = \delta'([q, 3], z)$

$\delta'([q, 1], a, z) = \delta'([q, 3], a, z) = ([p, k], v)$

if $\delta(q, a, z) = (p, v)$, $a \in \Sigma$, and

$k = 1$ or 2 for $p \in F$ or not in F respectively.

$L(M')$ is the complement of $L(M)$. For, if $a_1 a_2 \ldots a_n \in L(M)$. Then M enters an accepting state after reading a_n. Correspondingly, the second component of the state of M' is 1 before it is possible for M' to use a true input after a_n.

Hence M' does not accept by entering a state whose second component is 3, and a_n was the last true input used.

Suppose $a_1 a_2 \ldots a_n \notin L(M)$. Then by Lemma 3.5.5, M' after reading a_n have no ε-moves and will have to use a true input symbol. At this time, the second component of M' is 2, since $a_1 a_2 \ldots a_n \notin L(M)$. By rule (2), M' will accept before using a true input symbol.

3.5.7. Corollary

Every DCFL is accepted by some DPDA.

Proof. Follows from Theorem 3.5.4.

3.5.8. Example

The language $L = \{0^i 1^j 2^k : i = j \text{ or } j = k\}$

is not DCFL.

Let L be generated by a grammar whose productions are

$S \to AB/CD$, $A \to 0A^1/\varepsilon$, $B \to 2B/\varepsilon$, $C \to 0C/\varepsilon$, $D \to 1D2/\varepsilon$.

Using odgen's Lemma, we can prove $L_1 = \{0^i 1^j 2^k : i \neq j, j \neq k\}$ is not a CFL, hence not a DCFL. Also

$L_1 = \overline{L} \cap 0^* 1^* 2^*$, As L_1 is not DCFL, $\overline{L}$ is not DCFL, and hence L is not DCFL.

3.5.9. Lemma

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a DPDA. Let B be any subset of $Q \times \Gamma$. Define $L = \{w : (q_0, w, Z_0) \vdash^* (q, \varepsilon, Z\,v)$ for some $(q, z) \in B\}$.

Then L is DCFL.

Proof. Define a DPDA M' accepting L as follows:

$$M' = (Q', \Sigma, \Gamma, \delta'\ q_0, z_0, F')$$

where

$$Q' = \{q, q', q'' : q \in Q\}$$

$$F' = \{q''\}$$

Define δ' as follows.

(*i*) If $\delta(q, a, z) = (p, v)$, then

$$\delta'(q, a, z) = (p', v)$$

(*ii*) If $(q, z) \notin B$, then

$$\delta'(q', \varepsilon, z) = (q, z).$$

(*iii*) If $(q, z) \in B$

$$\delta'(q', \varepsilon, z) = (q'', z) \text{ and}$$

$$\delta'(q'', \varepsilon, z) = (q, z).$$

Recall that the quotient of L_1 with respect to L_2, denoted by L_1/L_2 is $\{x$: there exists w in L_2 such that xw is in $L_1\}$.

3.5.10. Theorem

Let L be a DCFL and R be a regular set. Then L/R is a DCFL.

Proof. Let $L = L(M)$ for some DPDA M. Let $R = L(A)$ for some finite state automaton A. Suppose $M = (Q_M, \Sigma, \Gamma, \delta_M, q_0, Z_0, F_M)$ and $A = (Q_A, \Sigma, \delta_A, p_0, F_A)$. Then let $M' = \pi(M, A) = (Q_M, \Sigma, \Gamma \times \Delta, \delta, q_0, [Z_0, \mu_0], F_M)$ be the predicting machine for M and A, where Δ is set of subset of $Q_M \times Q_A$. If $\pi(M, A)$ is in $ID(r, x, [z, \mu]\, v)$, the μ consists of exactly those paris (q, p) such that there is a $w \in \Sigma^*$ for which $\delta_A(p, w) \in F_A$ and $(q, w, z_\beta) \overset{*}{\underset{M}{\vdash}} (s, \varepsilon, \alpha)$ for some $s \in F_M$, $\alpha, \beta \in \Gamma^*$, where β is the string of first components of v.

Let Mq, z be M with q and z as the starting state and start stack symbol respectively. Let Ap be A with p as the starting state. Let

$$M_0 = \{(q, p) : L(Mq, z_0) \cap L(Ap) \neq \phi\}$$

Then $(q_0, x, [z_0\ \mu_0]) \overset{*}{\underset{\pi(M,A)}{\vdash}} (r, y, [z_1, \mu_1]\ [z_2, \mu_2] \ldots [z_n, \mu_n]$

if and only if

(a) $(q_0, x, z_0) \overset{*}{\underset{M}{\vdash}} (r, y, z_1 z_2, \ldots z_n)$

(b) $\mu_i = \{(q, p)$: for some w, $(q, w, z_i z_{i+1} \ldots z_n \overset{*}{\underset{M}{\vdash}} (s, \varepsilon, v)$ for some s in F_M, $v \in \Gamma^*$, and $\delta_A (p, w) \in F_A\}$ for $1 \le i \le n$.

Now, let B be the subset of $Q_M \times (\Gamma \times \Delta)$ containing all $(q, [z, \mu])$ with (q, p_0) is in μ. Then by Lemma 3.5.9.

$L_1 = \{x : (q_0, x, [z_0, \mu_0] \overset{*}{\underset{M}{\vdash}} (q, \varepsilon, [z, \mu]\ v$ and $(q, p_0) \in \mu\}$ is a DCFL. Also

$L_1 = \{x$: for some $w \in \Sigma^*$, $(q_0, x, z_0) \overset{*}{\underset{M}{\vdash}} (q, \varepsilon, Z\ v^1)$

$(q, w, Zv') \overset{*}{\underset{M}{\vdash}} (s, \varepsilon, \beta)$ $s \in F_M$, v' is the first component of v, $\delta_A (p_0, w) \in F_A\}$.

That is, $L_1 = \{x$: for some $w \in \Sigma^*$, $x\ w \in L\ (M)$ and $w \in L\ (A)\}$.

Hence $L_1 = L/R$. Thus L_1 is a DCFL.

Now we introduce two operations that preserve DCFL's but not CFL's

MIN$(L) = \{x : x \in L$ and no w in L is a proper prefix of $x\}$

MAX $(L) = \{ x : x \in L$ and x is not a proper prefix of any word in $L\}$.

3.5.11. Theorem

If L is a DCFL, the MIN (L) and MAX (L) are DCFL's.

Proof. Let $M = (Q_M, \Sigma, \Gamma, \delta_M, q_0, Z_0, F_M)$ be a DPDA that accepts L and always scans its entire input. Modify M to make no move in a final state. Then the resulting DPDA M_1, accepts MIN (L). If $w \in$ MIN (L), then let

$$(q_0, w, Z_0) = I_0 \vdash_M I_1 \vdash_M \ldots \vdash_M I_M \quad (* *)$$

be the sequence of ID's entered by M, where $I\mu = (q, \varepsilon, v)$ for some v, and q is in F_M. Furthermore, since w is in MIN (L), none of $I_0, I_1, \ldots, I_{m-1}$, has an accepting state. Thus $(* *)$ is also a computation of M_1, so w is in $L\ (M)$.

Conversely if $(q_0, w, Z_0) = I_0 \vdash_{M_1} I_1 \vdash_{M_1} I_2 \vdash_{M_1} \ldots \vdash_{M_1} I_m$ is an accepting computation of M_1, then none of $I_0, I_1, \ldots, I_{m-1}$ has an accepting state. Thus w is in MIN (L).

To show MAX(L) is a DCFL's. Let $A = (Q_A, \Sigma, \delta_A, p_0, F_A)$ be the simple FSA accepting Σ^+.

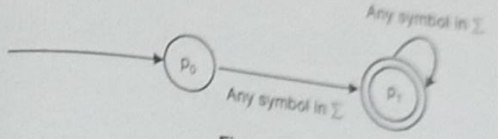

Fig. 3.4.

Let $M = (Q_M, \Sigma, \Gamma \times \Delta, \delta, q_0, [Z_0, \mu_0], F_M)$ be $\pi\ (M, A)$. Let $B = \{(q, [z, \mu])$: q is in F_M, $(q, p)\mu\}$

Then

$L_1 = \{x : (q_0, z, z_0) \overset{*}{\underset{M}{\vdash}} (q, \varepsilon, v)$ for some q in F_M, and for no $w \ne \varepsilon$ does $(q, w, v) \overset{*}{\vdash} (s, \varepsilon, \beta)$ for s in $F_M\}$ is a DCFL. But $L_1 =$ MAX (L), so MAX (L) is a DCFL.

3.6. LR (0) GRAMMAR

Next we introduce a restricted type of CFG called an *LR (0) grammar*. It stands for "left-to-right scan of the input producing a right most derivation and using 0 symbols of look ahead on the input".

3.6.1 Definition

L is said to have the *prefix property* if whenever w is in L, no proper prefix of w is in L.

3.6.2. Definition

An *item* for a given CFG is a production with a dot anywhere in the right side, including the beginning or end.

In the case of an ε-production, $B \to \varepsilon$, $B \to 0$ is an *item*.

3.6.3. Example

The *items* for grammar with the produciton $S \to Sc$, $S \to SA/A$, $A \to ab$.

The *items* are,

$$S' \to \cdot Sc,\ S \to \cdot SA \quad A \to \cdot ab$$
$$S' \to S \cdot c,\ S \to S.A \quad A \to a \cdot b$$
$$S' \to Sc.,\ S \to SA.\ \ A \to ab \cdot$$
$$S \to \cdot A$$

$$S \to A.$$

Use the symbols $\overset{*}{\underset{rm}{\Rightarrow}}$ and $\overset{*}{\underset{rm}{\Rightarrow}}$ to denote right most derviations and single steps in a right most derivation, respectively.

3.6.4. Definiton

A right-sentential form is a sentential form that can be derived by a right most derivation. A *handle* of a right sentential form ν for CFG *G* is a substring β, such that

$$S \overset{*}{\underset{rm}{\Rightarrow}} \delta Aw \overset{*}{\underset{rm}{\Rightarrow}} \delta \beta w$$

and $\delta \beta w = v$. A *viable prefix* of a *right-sentential form* ν is any prefix of ν ending no further right than the right end of a *handle* of *u*.

For example, $S' \to SC, S \to SA|A, A \to aSb|ab$ produces a rightmost derivation.

$$S' \Rightarrow Sc \Rightarrow SA\ c \Rightarrow SaSbc.$$

Thus *SaSbc* is a *right-sentential form*, and its *handle* is *aSb*, ε, *S*, *Sa*, *SaS*.

An *item* $A \to \alpha \cdot \beta$ is said to be *valid* for a viable prefix *v* if there is a right most derviation

$$S \overset{*}{\underset{rm}{\Rightarrow}} \delta Aw \overset{*}{\underset{rm}{\Rightarrow}} \delta \alpha\beta w$$

and $\delta\alpha = v$. An *item* is said to be *complete* if the dot is the right most symbol in the *item*. For example, $A \to \alpha.$ is a *complete item*.

3.6.5. Definition

G is said to be an *LR (0) grammar* if

(*i*) its start symbol does not appear on the right side of any production, and

(*ii*) for every viable prefiex *v* of *G*, whenever $A \to \alpha\cdot$ is a complete item valid for *v*, then no other *complete item* nor any item with a terminal to the right of the dot is valid for *v*.

Now we construct a DPDA from an *LR* (0) *grammar* which differs from the non-deterministic PDA which are constructed from an arbitrary CFL. We introduce a new notation for ID's of a DPDA. Here we use $[q, \alpha, w]$ instead of (q, w, α^R). To simulate right most derivations in an *LR(0) grammar* not only do we keep a viable prefix on the stack, but above every symbol we keep a state

of the DFA recogninzing viable prefixes. If viable prefix $X_1 X_2 \ldots X_k$ is on the stack, then the complete stack contents will be $s_0 X_1 s_1, \ldots X_k s_k$ where s_i is $\delta(q_0, X_1 X_2 \ldots X_i)$ and δ is the transition function of the DFA. The top state s_k provides the valid items for $X_1, X_2 \ldots X_k$.

If s_k contans $A \to \alpha\cdot$, then $A \to \alpha\cdot$ is valid for $X_1 X_2 \ldots X_k$. Thus α is a suffix of α to A by replacing $X_{i+1} X_{i+2} \ldots X_k$ on top of the stack by *A*. The DPDA will enter a sequence of ID's.

$$[q_1, \text{ so } X_1 \ldots s_{k-1} X_k s_k, w] \overset{*}{\vdash} [q, s_0 X_1 \ldots s_{i-1} X_i s_i A s, w] \qquad (3\text{-}22)$$

where $s = \delta(s_i, A)$. Note that if the grammar is *LR* (0), s_k contains only $A \to \alpha\cdot$, unless $\alpha = \epsilon$, in which case s_k may contain some incomplete items.

Suppose s_k contains only incomplete items. Then the right sentential form previous to $X_1 X_2 \ldots X_k$ *w* could not be formed by reducing a suffix of $X_1 X_2 \ldots X_k$ to some variable else there would be a complete item valid for $X_1 X_2, \ldots X_k$. There must be a handle ending to the right of X_k in $X_1 \ldots X_k w$, as $X_1 \ldots X_k$ is a viable prefix. That is,

$$[q, s_0 X_1 \ldots s_{k-1} X_k s_k, ay) \vdash [q, So\ X_i \ldots s_{k-1} X_k s_k \text{ at } y] \qquad \ldots(3.23)$$

where $t = \delta(s_k, a)$. If *t* is not the empty set of items. $X_1 \ldots X_k a$ is a viable prefix.

3.6.6. Theorem

If *L* is *L* (*G*) for an *LR*(0) grammar *G*, then *L* is *N* (*M*) for a DPDA *M*.

Proof. Construct from *G*, the DFA 'A' with transition function δ, that recognizes *G*'s viable prefizes. Let the stack symbols of *M* be the grammar symbols of *G* and the states of *A*. *M* has starting state *q* together with some states used by reductions as in (3· *) & (3· **).

If *G* is *LR* (0), reductions are the only possible way to get the previous right-sentential form when the state of the DFA on the top of *M*'s stack contains a complete item.

When *M* starts with *w* in *L* (*G*) on its input and only so on its stack, then it will construct a rightmost derivation for *w* in reverse order.

When a shift is, as in (3.22), because the top DFA state on *M*'s stack has only incomplete items, then there could not be a handle among the grammar symbols $X_1 X_2, \ldots X_k$ found on the stack at that time. If there were such a handle, then some DFA state on the stack, below the top, would have a complete item.

If the top state on the stack is $\{S \to \alpha\}$ where *S* is *G*'s start symbol, then *G* pops its stack, accepting. In this case we have completed the reverse of a rightmost derivation of the original input. Note that as *S* does not appear on the right of any production, it is impossible that there is an item of the form $A \to$

S- α valid for viable prefix S. Thus there is never a need to shift additional input symbols when S alone appears on the stack. We proved if w is in $L(G)$, M finds a right most derivation of w, reduces w to S, and accepts. Conversely, if M accepts w, the sequence of right sentential forms represented by the ID's of M provides a derivation of w from S. Thus $N(M) = L(G)$.

3.6.7. Lemma

Let $M = (Q, \Sigma, \Gamma, \delta\ q_0, Z_0, \phi)$ be a DPDA. If L is $N(M)$, then L has an $LR(0)$ grammar.

Proof. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \phi)$ be a DPDA. Define grammar $G_M = (V, \Sigma, P, S)$ such that $L(G_M) = N(M)$. V consists of the symbol S, the symbols $[q \times p]$ for q and p in Q and X in Γ, and the symbols A_{qay} for q in Q, a in $\Sigma \cup (\varepsilon\}$ and Y in Γ.

Symbol A_{qay} indicates that the production is obtained from the move of M in $\delta(q, a, y)$. The productions of G_M are as follows.

(*i*) $S \to [q_0, Z_0\ p]$ for all p in Q

(*ii*) If $\delta(q, a, y) = (p, \varepsilon)$, then there is a production $[q, y\ p] \to A_{qay}$

(*iii*) If $\delta(q, a, y) = (p_1, X_1 X_2 \ldots X_k)$ for $k \geq 1$, then for each sequence of states $p_2, p_3, \ldots, p_{k+1}$, there is a production $[qy\ p_{k+1}] \to A_{qay}\ [p_1 X_1\ p_2]$ $\ldots\ [p_k\ X_k\ p_{k+1}]$

(*iv*) For all q, a and y, $A_{qay} \to a$.

The right most derivation in G_M starts with $S \Rightarrow [q_0\ z_0\ p]$ for some state p. The only productions for $[q_0\ z_0\ p\ ]$ that derive strings beginning with right sides.

$Aq_0 az_0\ [r\ X\ s]\ [s\ y\ t]\ [t\ z\ p]$ for some states s and t. Suppose that the right most derivation finally derives some string w from $[t\ z\ p]$. If $\delta(s, b, y) = (u, Vw)$, then

$$S \underset{rm}{\overset{*}{\Rightarrow}} Aq_0az_0[\ rXs]\ [s\ y\ t]\ w \underset{rm}{\Rightarrow} A\ q_0\ a\ z_0\ [rXs]\ A_{sby}\ [uVv)\ [vWt]w. \qquad (3.24)$$

The input corresponding to the above derivation is of the form $ax_1\ bx_2\ x_3\ w$, where $[rXs] \overset{*}{\Rightarrow} x_1$, $[u\ V\ v] \overset{*}{\Rightarrow} x_2$, $[vWt] \overset{*}{\Rightarrow} x_3$. The move is of the form

$$\begin{aligned}(q_0,\ ax_1bx_2x_3w_1Z_0) &\vdash (r,\ x,\ bx_2x_3w,\ XYZ)\\ &\overset{*}{\vdash} (s,\ bx_2\ x_3w,\ YZ)\\ &\vdash (u,\ x_2x_3\ w,\ VWZ)\\ &\vdash (v,\ x_3w,\ WZ)\\ &\vdash (t,\ w,\ Z).\end{aligned}$$

From (3.24) and (3.23) we conclude that given any viable prefix α of G_M, there corresponds ID I of M in which the stack contains all and only the stack symbols that were introduced in a right most derivation of some αw and then replaced by a string of terminals.

3.6.8. Lemma

Let v be a viable prefix of G_M. Then $(q_0, h(v), Z_0) \overset{m(v)}{\vdash} (p, \varepsilon, \beta)$, for some p and β by the sequence of moves $m(v)$.

Proof. As v is a viable prefix, there is some y in Σ^* such that vy is a right sentential form. Then for some state γ, $[q_0, Z_0\ r] \underset{rm}{\overset{*}{\Rightarrow}} h(v)\ y$. The last $N(v)$ expansions of $A's$ in that derivation take place after the right sentential form vy is reached. There is a unique sequence of move $(q_0, h(v)\ y, z_0 \overset{*}{\vdash} (r, \varepsilon, \varepsilon)$, and the first $N(v)$ of these must be $m(v)$.

3.6.9. Lemma

If I is a set of items of G_M, and $B \to \beta.$ is in I, then there is no item $A_{qa}v \to a$ in I.

Proof. Let I be the set of items for viable prefix v.

Case 1. If $B \to \beta$ is a production from rule (*i*) then $v = \beta$, then v is a single vairbale $[q_0\ z_0\ p]$, since S appears on no right side. If $A_{qay} \to a$ valid for v, then there is a derivation $S \overset{*}{\Rightarrow} v\ A_{qay}\ y \underset{rm}{\Rightarrow} vay$. However, no right sentential form begins with a variable $[q_0\ z_0\ p]$ unless it is the first step of a derivation. All subsequent right-sentential forms begin with a subscripted A, until the last, which begins with a terminal. Thus v could not be followed $A_{qa}v$ in a right sentential form.

Case 2. If $B \to \beta$ is introduced by rules (*ii*) or (*iii*) then $v'\beta\ A_{qa}v$ is a viable prefix, where $v = v'\ \beta$. In any rightmost derivation, when $B \to \beta$ is applied the last symbol of β is immediately expanded by rules (*ii*), (*iii*) or (*iv*), so β could not appear in a right-sentential form followed by $Aqav$.

Case 3. If $B \to \beta$ is $A_{pbz} \to b$ introduced by rule (*iv*), and $A_{qay} \to \cdot\ a$ is valid for v, then b must be ε, else $v\ A_{qay}$, which is a viable prefix, has a terminal in it. As $A_{pez} \to$ is valid for v, it follows that vA_{pez} is a viable-prefix. Thus by previous lemma applied to vA_{qay} and vA_{pez}, the first $N(v) + 1$ moves made by M when given input $h(v)a$ are both $m(v)\ (p\ \varepsilon\ z)$and in $m(v)\ (q\ a\ y)$, contradicting the determinism of M, where m is a homomorphism from v^* to moves defined by $m(A_{qay}) = qay$.

3.6.10 Lemma

If I is a set of items of G_M, and $B \to \beta.$ is in I, then there is no other item $C \to \alpha \cdot$ in I.

Proof. Let v be a viable prefix with set of valid items I.

Case 1. The productions $B \to \beta$, $C \to \alpha$ are not introduced by rule (*iv*). The productions of type (1) are applied only at the first step tell us that as α and β are both suffixes of v, we must have $\beta = \alpha$. If the productions are of type (2) or (3), then $B = C$. But rule (*ii*) requires that $\delta(q, a, Y) = (p, \varepsilon)$, so the determinism of M assures that p is unique.

Case 2. $B \to \beta$, $C \to \alpha$ are type (4) productions. Then vB and vC are viable prefixes and by Lemma, M is not deterministic. That is if $\beta = \alpha = \varepsilon$, then the first $N(v) + 1$ moves of M on input $h(v)$ must be $m(vB)$ and $m(vC)$. If $\beta = a \neq \varepsilon$, $\alpha = b \neq \varepsilon$, then $a = b$, and the first $N(v) + 1$ moves of M on input $h(v)$ a must be $m(vB)$ and $m(vC)$. If $\beta = a \neq \varepsilon$, $\alpha = \varepsilon$, then the first $N(v) + 1$ moves of M on input $h(v)$ a provides a contradiction.

Case 3. $B \to \beta$ is from rule (1) (2), or (3) and $C \to \alpha$ is from rule (4) or vice versa. Then vC is a right-sentential form, and v ends in β.

3.6.11. Theorem

If M is a DPDA, then G_M is an $LR(0)$ grammar.

Proof. Proof follows from Lemma 3.6.9 and 3.6.10.

3.6.12. Theorem

As language L has an $LR(0)$ grammar if and only if L is a DCFL with the prefix property.

Proof. Suppose L is a DCFL with the prefix property. Then L is $L(M')$ for some DPDA M' we can make M' accept L by empty stack by putting a bottom-of-stack marker on M' and causing M' to enter a new state that erases the stack whenever it enters a final state. As L has the prefix property, we do not change the language accepted, k and L is accepted by empty stack by the new DPDA M. Hence $L = L(G_M)$.

Conversely, if L is $N(M)$ for some DPDA, M. Then we know that L is $L(M')$ for some DPDA M'. L has the prefix property follows from the fact that a DPDA 'dies' when it empties its stack.

3.6.13. Corollary

$L\$$ has $LR(0)$ grammar if and only if L is a DCFL, where \$ is not a symbol of L's alphabet.

Proof. $L\$$ surely has the prefix property. If $L\$$ is a DCFL, then $L = L\$/\$$ is a DCFL.

Conversely, if L is a DCFL, then it is easy to construct a DPDA for $L\$$.

3.7 PARSING

Parsing becomes important in the case of programming languages. If a statement in a programming language is given, only the derivation of the statement can give the meaning of the statement (semantics). There are two types of parsing.

(*i*) Top-down parsing.

(ii) Bottom-up parsing.

In top down parsing we try to construct the derivation (or the corresponding parse tree) of the input string, starting from the root (with label S) and end in the given input string. This is equivalent to a left most derivation. In bottom-up parsing we try the derivation from the given input string to the top.

3.7.1 Top Down Parsing

The language $L = \{a^n b^n : n \geq 0\}$ is generated by the context free grammar

$$G = (\{a, b, S\}, \{a, b\}, R, S,)$$ where R contains the two rules $S \to aSb$, $S \to \lambda$. This L is accepted by a pushdown automaton.

$$M_1 = (\{p, q\}, \{a, b\}, \{a, b, S\}, \delta_1, p, \{q\}),$$

where

$$\delta_1(p, \lambda, \lambda) = (q, S)$$
$$\delta_1(q, \lambda, S) = (q, aSb)$$
$$\delta_1(q, \lambda, S) = (q, \lambda)$$
$$\delta_1(q, a, a) = (q, \lambda)$$
$$\delta_1(q, b, b) = (q, \lambda)$$

Since M_1 has two different transitions with identical first components, it is not deterministic. That is, since

$$\delta_1(q, \lambda, S) = \{(q, \lambda), (q, aSb)\}.$$

M_1 is not deterministic. M_1 can be modified to become a deterministic PDA M_2 that accepts $L\$$.

Define $M_2 = (\{p, q, q_a, q_b, q_\$\}, \{a, b, \$\}, \delta_2, p, \{q_\$\}$ where δ_2 is defined as follows.

1. $\delta_2 (p, \lambda, \lambda) = (q, S)$
2. $\delta_2 (q, a, \lambda) = (qa, \lambda)$
3. $\delta_2 (q_a, \lambda, a) = (q, \lambda)$
4. $\delta_2 (q, b, \lambda) = (qb, \lambda)$
5. $\delta_2 (q_b, \lambda, b) = (q, \lambda)$
6. $\delta_2 (q, \$, \lambda) = (q_\$, \lambda)$
7. $\delta_2 (q_a, \lambda, S) = (q_a, aSb)$
8. $\delta_2 (q_b, \lambda, S) = (q_b, \lambda)$

From state q, M_2 reads one input symbol and without changing the stack, enters one of the three new states q_a, q_b or $q_\$$. Then uses the information to differentiate the two transitions

$$\delta_2 (q, \lambda, S) = (q, aSb)$$

$$\delta_2 (q, \lambda, S) = (q, \lambda)$$

The first transition is retained only from state q_a and the second only from state q_b. So M_2 is deterministic. It accepts the input $ab\$$ as follows:

Step	*State*	*Unread Input*	*Stack*	*Transition used*	*Rule of G*
0	p	$ab\$$	λ	–	
1	q	$ab\$$	S	1	
2	q_a	$b\$$	S	2	
3	q_a	$b\$$	aSb	7	$S \to aSb$
4	q	$b\$$	Sb	3	
5	q_b	$\$$	Sb	4	
6	q_b	$\$$	b	8	$S \to \lambda$
7	q	$\$$	λ	5	
8	$q_\$$	λ	λ	6	

δ_0 M_2 is a DPDA accepting $L = \{a^n b^n : n \geq 0\}$.

Devices such as M_2 are called parsers. In particular M_2 is *a* top-down parser because tracing its operation at the steps where nonterminals are replaced on the stack reconstructs a parse tree in a top-down, left-to-right fashion.

Note. By a technique called 'left factoring', we resolve the nondeterminism. It can be summarized as follows.

3.7.2. Theorem

Let G be a consext free grammar having two productions of the from $A \to \alpha\beta$, $A \to \alpha\nu$ with $\alpha \neq \lambda$. Then replace them by the rules $A \to \alpha A^1$, $A^1 \to \beta$, $A^1 \to \nu$, where A^1 is a new non terminal.

Proof. The equivalence can be proved by showing that the effect of applying $A \to \alpha\beta$, $A \to \alpha\nu$ in *a* derivation can be realised by applying $A \to \alpha A^1$, $A^1 \to \beta$, $A^1 \to \nu$ and vice versa.

A variable A is called left recursive if there is production of the form $A \to A\alpha$. Such *a* production can cause a top-down parser into an infinite loop. Techniques for avoiding left recursion are as follows.

3.7.3. Theorem

Let G be a context-free grammar. Let the set of all productions be $\{A \to A\alpha_1, A \to A\alpha_2, \ldots, A \to A\alpha_n, A \to \beta_1, \ldots A \to \beta_m\}$. Then replace these rules by $A \to \beta_1 A^1, \ldots A \to \beta_m A^1, A^1 \to \alpha_1 A^1, A^1 \to \alpha_2 A^1, \ldots A^1 \to \alpha_n A^1$ and $A^1 \to \lambda$, where A^1 is a new nonterminal.

Proof. A grammar G having the property (by looking ahead for k symbols we derive a given input string in $L(G)$ is called an $LL(k)$ grammar.

Construction of Top down Parser.

(*i*) Eliminate left recursion in G by repeatedly applying Theorem 3.7.3 all left recursive variables.

(*ii*) Apply theorem 3.7.2 to get left factoring whenever necessary.

(*iii*) If the resulting grammar is $LL(k)$ for some natural number k apply top down parsing.

3.7.4. Example

Consider the language consisting of all arthmetic expressions involving + and * and variables x_1 and x_2. This language is generated by a grammar $G = (\{T, F, E\}, \Sigma, P, E)$, where $\Sigma = (x, 1, 2, +, *, (,)\}$ and P consists of

$E \to E + T$, $\quad F \to (E)$

$E \to T$ $\quad F \to x_1$

$T \to T * F$ $\quad F \to x_2$

$T \to F$.

We construct a top-down parser for $L(G)$.

Step 1. Since in the rule $E \to E + T$, the nonterminal on the L.H.S is repeated

as the first symbol of the right hand side, so we have a left recursion from the rule $E \to E + T$. We simply replace it by the rules $E \to TE'$, $E' \to TE'$, and $E' \to \lambda$ where E' is a new nonterminal.

Similarly $T \to T*F$ and $T \to F$ are replaced by $T \to F\,T'$, $T' \to *FT'$, $T' \to \lambda$.

The resulting equvalent grammar is

$G' = (\{T, F, E, T', E'\}, \Sigma, P', E)$,

where P' = Consists of

$E \to TE'$ $T' \to \lambda$

$E' \to +TE'$ $F \to (F)$

$E' \to \lambda$ $F \to x_1$

$T \to FT'$ $F \to x_2$

$T' \to *FT'$.

Step 2. Apply theorem 3.7.2 for left factoring $F \to x_1$ and $F \to x_2$ to get new productions $F \to xN$, $N \to 1$ and $N \to 2$.

The resulting equivalent grammar is

$G_2 = (\{T, F, E, T', E'\}, \Sigma, P'', E)$ where P'' consists of

$P_1 : E \to TE'$ $P_6 : T' \to \lambda$

$P_2 : E' \to +TE'$ $P_7 : F \to (E)$

$P_3 : E' \to \lambda$ $P_8 : F \to xN$

$P_4 : T \to FT'$ $P_9 : N \to 1$

$P_5 : \text{T1} \to *FT'$ $P_{10} : N \to 2$.

Step 3. The grammar G_2 obtained in step 2 is an $LL(1)$ grammar. The parsing table is given in Table. The table provides the production to be applied for a given variable with a particular look ahead for one symbol. Let e denote an error. It indicates that the given input string is not in $L(G)$.

	λ	x	1	2	+	*	(	)
E	e	P_1	e	e	e	e	P_1	e
T	e	P_4	e	e	e	e	P_4	e
F	e	P_8	e	e	e	e	P_7	e
T'	P_6	e	e	e	e	P_5	e	P_6
E'	P_3	e	e	e	P_2	e	e	P_3
N	e	e	P_9	P_{10}	e	e	e	e

3.7.5 Bottom up parsing

Reconstruct a parse tree from the leaves to the root, rather than the other way, and this method is called bottom-up.

In bottom-up parsing we have to reverse the production to get S finally. This suggests the following moves for a PDA acting as bottom-up parser.

(*i*) $\delta(p, \lambda, \alpha^R) = \{(p, A)$: there exists a production $A \to \alpha\}$

(ii) $\delta(p, \sigma, \lambda) = \{(p, \sigma)\}$ for all $\sigma \in \Sigma$.

For acceptability we require some moves when the PDS has S or z_0 on the top.

In bottom-up parsing we follow two types of operations, viz., shifting and reducing. By shifting we mean pushing the input symbol on to the stack. By reducing we mean replacing α^R by A when $A \to \alpha$ is a production in G. (moves given by (*i*)). For (*i*) we use a relation P called a precedence relation. If $(a, b) \in P$, a is the top most symbol on PDS and b is the input symbol then we reduce. Otherwise we shift b onto the stack.

3.7.6. Example

Consider again the grammar for arithmetic expressions analysed in example 3.7.4.

The rules of this grammar G are the following. $E \to E + T$, $E \to T$, $T \to T* F$, $T \to F$, $F \to (E)$, $F \to x_1$, $F \to x_2$. Using these productions we can construct the precedence relation P. If $(a, b) \in P$, then we have a tick mark ✓ in the (a, b) cell of the table using the table we can decide the moves. For example, if the stack symbol if F, and the next input symbol is *, then we apply reduction. If the stack symbol is E, then any input symbol is pushed auto the stack.

The Precedence Relation

Stack symbol/ Input symbol	x	(	)	1	2	+	*	$
z_0								
x								
(								
)			✓			✓	✓	✓
1			✓			✓	✓	✓
2			✓			✓	✓	✓
+								
*								
E								
T			✓			✓		✓
F			✓			✓	✓	✓

Using the precedence relation and the moves we have defined, it is easy to construct a DPDA.

The DPDA which acts as a bottom up parser is $M' = (Q, \Sigma', \Gamma, \delta, p, z_0, \phi)$, where

$$\Sigma' = \Sigma \cup \{\$\},\ Q = \{p\} \cup \{p_a : a \in \Sigma'\},$$

$$\Gamma = \{E, T, F\} \cup \Sigma \cup \{z_0, \$\}$$

δ is given by the following rules

R_1: $\delta(p, \sigma, \lambda) = (p\sigma, \lambda)$ for each $\sigma \in \Sigma'$

R_2 : $\delta(p\sigma, \lambda, a) = (p, \sigma a)$ for all $(a, \sigma) \notin P$

R_3 : $\delta(p\sigma, \lambda, T + E) = (p\sigma, E)$ for each $\sigma \in \Sigma'$ such that $(T, \sigma) \in P$

R_4 : $\delta(p\sigma, \lambda, Ta) = (p\sigma, Ea)$ for each $\sigma \in \Sigma'$ such that $(T, \sigma) \in P$ and $a \in G - \{+\}$

R_5 : $\delta(p_a, \lambda, F^* T) = (p_\sigma, T)$ when $(F, \sigma) \in P$

R_6 : $\delta(p_a, \lambda, Fa) = (p_\sigma, T_a)$ when $(F, \sigma) \in P$ and $a \in \Gamma - \{*\}$.

R_7 : $\delta(p_\sigma, \lambda,) E ( ) = (p_\sigma, F)$ for each $\sigma \in \Sigma'$ such that $( ), \sigma) \in P$.

R_8 : $\delta(p_\sigma, \lambda, 1x) = (p_\sigma, F)$ for each $\sigma \in \Sigma'$ such that $(1, \sigma) \in P$

R_9 : $\delta(p_\sigma, \lambda, 2x) = (p_\sigma, F)$ for each $\sigma \in \Sigma'$ such that $(2, \sigma) \in P$

R_{10} : $\delta(p_\$, \lambda, E) = (p_\$, \lambda)$

$\delta(p_\$, \lambda, z_0) = (p_\$, \lambda)$. Here M' is deterministic. Further it can accept $L(G)$ \$, and infact it is a bottom-up parser for $L(G)$ \$. The bottom up parsing for $x_1 + (x_2)$ is given below.

Step	State	Unread Input	Stack	Transition used	Rule of G.
0	p	$x_1 + (x_2)$	z_0	–	
1	p_x	$x_1 + (x_2)$	z_0	–	
2	p_x	$1 + (x_2)$	z_0	R_1	
3	p	$+(x_2)$	xz_0	R_2	
4	p_1	$+(x_2)$	xz_0	R_1	
5	p	(x_2)	$1xz_0$	R_2	
6	p_+	(x_2)	$1x\ z_0$	R_1	
7	p_+	(x_2)	Fz_0	R_8	$F \to x_1$

(Contd.)

Step	State	Unread Input	Stack	Transition used	Rule of G.
8	p_+	(x_2)	Tz_0	R_6	$T \to F$
9	p_+	(x_2)	Ez_0	R_4	$E \to T$
10	p	(x_2)	$+Ez_0$	R_2	
11	p_c	$x_2)$	$+\ Ez_0$	R_1	
12	p	$x_2)$	$(+Ez_0$	R_2	
13	p_x	$2)$	$(+Ez_0$	R_1	
14	p_2	$2)$	$x(+Ez_0$	R_2	
15	p_2	$)$	$x(\ +Ez_0$	R_1	
16	$p)$	$)$	$2x(\ +Ez_0$	R_2	
17	$p)$	$\$$	$2x(\ +Ez_0)$	R_1	
18	$p)$	$\$$	$F(\ +Ez_0)$	R_9	$F \to x_2$
19	$p_)$	$\$$	$T(\ +E\ z_0)$	R_6	$T \to F$
20	$p)$	$\$$	$E(\ +E\ z_0)$	R_4	$E \to T$
21	$p_\$$	$\$$	$)\ E(\ +E\ z_0$	R_2	
22	$p_\$$	λ	$)\ E(\ +E\ z_0$	R_1	
23	$p_\$$	λ	$F + E\ z_0$	R_7	$F \to (E)$
24	$p_\$$	λ	$T + E\ z_0$	R_6	$T \to F$
25	$p_\$$	λ	$E\ z_0$	R_3	$E \to E + T$
26	$p_\$$	λ	z_0	R_{10}	
27	$p_\$$	λ	λ	R_{11}	

By backtracking we get a right most derivation

$$E \Rightarrow E + T \Rightarrow E + F \Rightarrow E + (E) \Rightarrow E + (T)$$

$$\Rightarrow E + (F) \Rightarrow E + (x_2) \Rightarrow T + (x_2) \Rightarrow F + (x_2) \quad x_1 + x_2).$$

Note. Use of precedence relation to decide whether to shift or reduce, and in some cases reducing the longest possible string do not work in all situations. The grammars for which they do work are called weak precedence grammars; in practice, many grammars can be converted into weak precedure grammars.

3.8. SOLVED PROBLEMS

1. Construct a PDA to accept the language

$$L = \{a^n b^m c^m d^n / m, n \geq 1\}$$ by final state.

Solution. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where

$$Q = \{q_0, q_1, q_2\}, \Sigma = \{a, b, c, d\}, F = \{q_2\}$$

$$\Gamma = \{z_0, z_1, z_2, z_3, z_4, z_5, z_6\}$$

and δ is defined as follows:

$$\delta(q_0, \lambda, z_0) = \{(q_1, z_0 z_5)\}$$

$$\delta(q_1, \lambda, z_5) = \{(q_1, z_4 z_5, z_1), (q_1, z_4 z_6 z_1)\}$$

$$\delta(q_1, \lambda, z_6) = \{(q_1, z_3 z_6 z_2), (q_1, z_3 z_2)\}$$

$$\delta(q_1, a, z_1) = \{(q_1, \lambda)\}$$

$$\delta(q_1, b, z_2) = \{(q_1, \lambda)\}$$

$$\delta(q_1, c, z_3) = \{(q_1, \lambda)\}$$

$$\delta(q_1, d, z_4) = \{(q_1, \lambda)\}$$

$$\delta(q_1, \lambda, z_0) = \{(q_2, z_0)\}.$$

It is easily seen that $T(M) = L$.

2. Construct a PDA to accept the language

$$L = \{a^n b^m a^n / n, m \geq 1\}$$ by empty store.

Solution. Let $M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$

where

$$Q' = \{q_0'\}$$

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

$$G' = \{z_0', z_1', z_2', z_3'\},$$

$$F = \phi$$

and δ' is defiend as follows

$$\delta'(q_0', \lambda, z_0') = \{(q_0', z_2' z_0' z_2'), (q_0', z_2' z_1' z_2')\}$$

$$\delta'(q_0', \lambda, z_1') = \{(q_0', z_1' z_3'), (q_0', z_3')\}$$

$$\delta'(q_0', a, z_2') = \{(q_0', \lambda)\}$$

$$\delta'(q_0', b, z_3') = \{(q_0', \lambda)\}.$$

We easily see that $N(M') = L$.

3. Construct a PDA to accept the language

$$L = \{ww^R / w \text{ in } (0 + 1)^*$$ by empty store.

Soltuion. Let $M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$

where

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

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

$$\Gamma' = \{z_0', z_1', z_2'\}$$

$$F' = \phi.$$

and δ' is deinfed as follows:

$$\delta'(q_0', 0, z_0') = \{(q_0', z_1' z_0')\}$$

$$\delta'(q_0', 1, z_0') = \{(q_0', z_2' z_0')\}$$

$$\delta'(q_0', 0, z_1') = \{(q_0', z_1' z_1'), (q_1', \lambda)\}$$

$$\delta'(q_0', 0, z_2') = \{(q_0', z_1' z_2')\}$$

$$\delta'(q_0', 1, z_1') = \{(q_0', z_2' z_1')\}$$

$$\delta'(q_0', 1, z_2') = \{(q_0', z_2' z_2'), (q_1', \lambda)\}$$

$$\delta'(q_1', 0, z_1') = \{(q_1', \lambda)\}$$

$$\delta'(q_1', 1, z_2') = \{(q_1', \lambda)\}$$

$$\delta'(q_0', \lambda, z_0') = \{(q_1', \lambda)\}$$

$$\delta'(q_1', \lambda, z_0') = \{(q_1', \lambda)\}.$$

It is easily seen that $L = N(M')$.

4. Construct a PDA to accept the language of palindromes by final state

(or)

Construct a PDA to accept the language $L = \{ww^R / w \text{ in } (a + b)^*\}$ by final state.

Solution. Let $M^* = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where $Q = \{q_0, q_1, q_2\}$,

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

$$\Gamma = \{z_0, z_1, z_2\}$$

$$F = \{q_2\}$$

and δ is defined as follows :

$$\delta(q_0, a, z_0) = \{(q_0, z_1 z_0'), (q_1, z_0)\}$$

$$\delta(q_0, b, z_0) = \{(q_0, z_2 z_0), (q_1, z_0)\}$$

$$\delta(q_0, a, z_1) = \{(q_0, z_1z_1), (q_1, z_1)\}$$
$$\delta(q_0, b, z_1) = \{(q_0, z_2z_1), (q_1, z_1)\}$$
$$\delta(q_0, a, z_2) = \{(q_0, z_1z_2), (q_1, z_2)\}$$
$$\delta(q_0, b, z_2) = \{(q_0, z_2z_2), (q_1, z_2)\}$$
$$\delta(q_0, \lambda, z_0) = \{(q_1, z_0)\}$$
$$\delta(q_0, \lambda, z_1) = \{(q_1, z_1)\}$$
$$\delta(q_0, \lambda, z_2) = \{(q_1, z_2)\}$$
$$\delta(q_1, a, z_1) = \{(q_1, \lambda)\}$$
$$\delta(q_1, b, z_2) = \{(q_1, \lambda)\}$$
$$\delta(q_2, \lambda, z_0) = \{(q_2, z_0)\}.$$

It is easily seen that $L = T(M)$.

5. Construct DPDA to accept the language $L = \{x \in \{a, b\}^* / n_a(x) > n_b(x)\}$ by final state.

Solution. Construct a PDA,

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F) \text{ where}$$
$$Q = \{q_0, q_1\},$$
$$\Sigma = \{a, b\},$$
$$\Gamma = \{z_0, z_1, z_2\}$$
$$F = \{q_1\}$$

and δ is defined as follows :

$$\delta(q_0, a, z_0) = \{(q_1, z_0)\}$$
$$\delta(q_0, b, z_0) = \{(q_0, z_2 z_0)\}$$
$$\delta(q_0, a, z_2) = \{(q_0, \lambda)\}$$
$$\delta(q_0, b, z_2) = \{(q_0, z_2z_2)\}$$
$$\delta(q_1, a, z_0) = \{(q_1, z_1z_0)\}$$
$$\delta(q_1, b, z_0) = \{(q_0, z_0)\}$$
$$\delta(q_1, a, z_1) = \{(q_1, z_1z_1)\}$$
$$\delta(q_1, b, z_1) = \{(q_1, \lambda)\}$$

For example for the string *abbabaa* we have

$$(q_0, abbabba, z_0) \vdash (q_1, bbabaa, z_0)$$

$$\vdash (q_0, babaa, z_0)$$
$$\vdash (q_0, abaa, z_2z_0)$$
$$\vdash (q_0, baa, z_0)$$
$$\vdash (q_0, aa, z_2z_0)$$
$$\vdash (q_0, a, z_0)$$
$$\vdash (q, \lambda, z_0).$$

It is easily seen that $L = T(M)$.

6. Construct a PDA accepting the set of all strings over $\{a, b\}$ with equal number of *a's* and *b's* by empty store.

Solution. L = {set of all strings over $\{a, b\}$ with equal number of *a's* and *b's*}.

Construct $M' = (q', \Sigma', \Gamma', \delta', q_0', z_0', F')$

where
$$Q' = \{q_0'\}$$
$$\Sigma' = \{a, b\}$$
$$\Gamma' = \{z_0', z_1', z_2'\}$$
$$F' = \phi$$

and δ' is defined as follows :

$$\delta'(q_0', a, z_0') = \{(q_0', z_0' z_1')\}$$
$$\delta'(q_0', b, z_0') = \{(q_0', z_0' z_2')\}$$
$$\delta'(q_0', a, z_1') = \{(q_0', z_1' z_1')\}$$
$$\delta'(q_0', b, z_2') = \{(q_0', z_2' z_2')\}$$
$$\delta'(q_0', a, z_2') = (q_0', \lambda)\}$$
$$\delta'(q_0', b, z_1') = \{(q_0', \lambda)\}$$
$$\delta'(q_0', \lambda, z_0') = \{(q_1', \lambda)\}$$

It is easily seen that $L(G) = N(M')$. If w has equal number of *a's* and *b's*, then

$$(q_0', w, z_0') \vdash^* (q_0', \lambda, z_0') \vdash (q, \lambda, \lambda).$$

7. Construct a PDA accepting the set of all strings over $\{a, b\}$ with equal number of *a's* and *b's* by final state.

Solution. L = {set of all strings over $\{a, b\}$] with equal number of *a's* and *b's*}. Consider the PDA M' given in problem 6. We have seen that M' accepts L by empty store. Now we use Theorem 3.2.3 to construct the required PDA.

Construct $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

where $Q = Q' \cup \{q_0, q_j\}\ j = \{q_0', q_0, q_j\}$

$\Gamma = \Gamma' \cup \{z_0\}$ where $z_0 \notin \Gamma'$

$= \{z_0', z_1', z_2', z_0\}$

$F = \{q_j\}$ and δ is defined as follows:

$\delta(q_0, \lambda, z_0) = \{(q_0', z_0\ z_0')\}$

$\delta(q_0', a, z_0') = \{(q_0', z_0'\ z_1')\}$

$\delta(q_0', b, z_0') = \{(q_0', z_1'\ z_2')\}$

$\delta(q_0', a, z_1') = \{(q_0', z_1'\ z_1')\}$

$\delta(q_0', b, z_2') = \{(q_0', z_2'\ z_2')\}$

$\delta(q_0', a, z_2') = (q_0', \lambda)\}$

$\delta(q_0', b, z_1') = \{(q_0', \lambda)\}$

$\delta(q_0', \lambda, z_0') = \{(q_0', \lambda)\}$

$\delta(q_0', \lambda, z_0') = \{(q_j, z_0)\}$.

It is easily seen that $L = T(M)$.

8. Construct a PDA accepting $\{a^n b^m a^n / m, n \geq 1\}$ by empty store.

Solution. [For solution 1 see problem 2.]

Construct a PDA $M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$

where $Q' = \{q_0', q_1'\}$

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

$\Gamma' = \{z_0', z_1'\}$

$F' = \phi$ and δ' is defined as follows:

1. $\delta'(q_0', a, z_0') = \{(q_0', z_0'\ z_1')\}$
2. $\delta'(q_0', a, z_1') = \{(q_0', z_1'\ z_1')\}$
3. $\delta'(q_0', b, z_1') = \{(q_1', z_1')\}$
4. $\delta'(q_1', b, z_1') = \{(q_1', z_1')\}$
5. $\delta'(q_1', a, z_1') = \{(q_1', \lambda)\}$
6. $\delta'(q_1', \lambda, z_0') = (q_1', \lambda)\}$.

It is easily seen that $L = N(M')$ [For, we start with the symbol a's until a 'b' occurs (moves 1 and 2). when the current input symbol is b, the state changes, but no change in PDS occurs (move 3). Once all the b's in the input string are

exhausted (using move 4), the reaining a's are erased (move 5). Using move 6. z_0 is erased. So

$(q_0', a^n b^m a^n, z_0') \vdash^* (q_1', \lambda, z_0)$

$\vdash (q_1, \lambda, \lambda)]$.

9. Construct a PDA, accepting the language $L = \{a^n b^m a^n / m, n \geq 1\}$ by final state.

Solution. Consider a PDA M' given in problem 8. We have proved that $L = N(M')$. Now we construct a PDA M accepting L by final state using Theorem 3.2.3.

Construct $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where

$Q = Q' \cup \{q_0, q_j\} = \{q_0', q_1', z_0, q_j\}$

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

$\Gamma = \Gamma \cup \{z_0\} = \{z_0', z_1', z_0\}$.

$F = \{q_j\}$ and δ is defined as follows :

$\delta(q_0, \lambda, z_0) = \{(q_0', z_0\ z_0')\}$

$\delta(q_0', a, z_0') = \{(q_0', z_0'\ z_1')\}$

$\delta(q_0', a, z_1') = \{(q_0', z_1'\ z_1')\}$

$\delta(q_0', b, z_1') = \{(q_0', z_1')\}$

$\delta(q_1', b, z_1') = \{(q_1', z_1')\}$

$\delta(q_1', a, z_1') = \{(q_1', \lambda)\}$

$\delta(q_1', \lambda, z_0') = \{(q_1', \lambda)\}$

$\delta(q_0', \lambda, z_0) = \{(q_j, z_0)\}$

$\delta(q_1', \lambda, z_0) = \{(q_j, z_0)\}$

It is easily seen that $L = T(M)$.

10. Construct a PDA to accept the language

$L = \{a^n b^{2n} / n \geq 1\}$ by empty store.

Solution. Construct a PDA

$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$

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

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

$\Gamma' = \{z_0', z_1'\}$

$$F = \phi$$

and δ' is defined as follows:

$$\delta'(q_0', a, z_0') = \{(q_1', z_0' z_1')\}$$
$$\delta'(q_1', a, z_1') = \{(q_1', z_1' z_1')\}$$
$$\delta'(q_1', b, z_1') = \{(q_2', z_1')\}$$
$$\delta'(q_2', b, z_1') = \{(q_1', \lambda)\}$$
$$\delta'(q_1', \lambda, z_0') = \{(q_1', \lambda)\}.$$

It is easily seen that $L = N(M')$.

11. Construct a PDA to accept the language $L = \{a^n b^{2n} / n \geq 1\}$ by final state.

Solution. Consider the PDA M' given in problem 10. We have proved that $L = N(M')$. Now we use 3.2.3. to construct the required PDA.

Construct a PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

where
$$Q = Q' \cup \{q_0, q_j\}$$
$$= \{q_0', q_1', q_2', q_0, q_j\}$$
$$\Sigma = \Sigma' = \{a, b\}$$
$$F = \{q_j\}$$
$$\Gamma = \Gamma' \cup \{z_0\}$$
$$= \{z_0, z_0', z_1'\}$$

and δ is defined as follows :

$$\delta(q_0', \lambda, z_0) = \{(q_1', z_0 z_0')\}$$
$$\delta(q_0', a, z_0') = \{(q_0', z_0' z_1')\}$$
$$\delta(q_1', a, z_1') = \{(q_1', z_1' z_1')\}$$
$$\delta(q_1', b, z_1') = \{(q_2', z_1')\}$$
$$\delta(q_2', b, z_1') = \{(q_1', \lambda)\}$$
$$\delta(q_1', \lambda, z_0') = \{(q_1, \lambda)\}$$
$$\delta(q_0', \lambda, z_0) = \{(q_j, z_0)\}$$
$$\delta(q_1', \lambda, z_0) = \{(q_j, z_0)\}$$
$$\delta(q_2', \lambda, z_0) = \{(q_j, z_0)\}$$

It is easily seen that $L = N(M)$.

12. Construct a PDA to accept the language $L = \{a^m b^n c^n / m, n \geq 1\}$ by empty store.

Solution. Construct a PDA $M' = (Q, \Sigma, \Gamma, \delta', q_0', z_0', F')$

where
$$Q = \{q_0', q_1'\}$$
$$\Sigma = \{a, b, c\}$$
$$\Gamma = \{z_0', z_1'\}$$
$$F = \phi$$

and δ' is defined as follows :

$$\delta'(q_0', a, z_0') = \{(q_0', z_0' z_1')\}$$
$$\delta'(q_0', a, z_1') = \{(q_0', z_1' z_1')\}$$
$$\delta'(q_0', b, z_1') = \{(q_1', \lambda)\}$$
$$\delta'(q_1', b, z_1') = \{(q_1', \lambda)\}$$
$$\delta'(q_1', c, z_0') = \{(q_1', z_0')\}$$
$$\delta'(q_1', \lambda, z_0') = \{(q_1', \lambda)\}.$$

It is easily seen that $L = N(M')$.

[For, on reading a, we add z_1' ; on reading b we remove z_1' and the state is changed. If the input is completely read and stack symbol is z_0', then it is removed by a λ-move.]

13. Construct a PDA to accept the language $L = \{a^m b^m c^n / m, n \geq 1\}$ by final state.

Solution. Consider a PDA M' given in problem 12. We have proved that $L = N(M')$. We now construct the required PDA by using M' and Theorem 3.2.3.

Construct $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

where
$$Q = Q' \cup \{q_0, q_j\}$$
$$= \{q_0', q_1', q_0, q_j\}$$
$$\Sigma = \Sigma' = \{a, b, c\}$$
$$\Gamma = \Gamma' \cup \{z_0\} = \{z_0', z_1', z_0\}$$
$$F = \{q_j\}$$ and δ is defined as follows.

$$\delta(q_0', \lambda, z_0) = \{(q_0', z_0 z_0')\}$$
$$\delta(q_0', a, z_0') = \{(q_0', z_0' z_1')\}$$

$$\delta(q_0', a, z_1') = \{(q_0', z_1'z_1')\}$$
$$\delta(q_0', b, z_1') = \{(q_1', \lambda)\}$$
$$\delta(q_1', b, z_1') = \{(q_1', \lambda)\}$$
$$\delta(q_1', c, z_0') = \{(q_1', \lambda)\}$$
$$\delta(q_0', \lambda, z_0) = \{(q_f, z_0)\}$$
$$\delta(q_1', \lambda, z_0) = \{(q_f, z_0)\}$$

It is easily seen that $L = T(M)$.

14. Consider a PDA $M = (\{q_0, q_1, q_f\}, \{a, b, c\}, \{a, b, z_0\}, \delta, q_0, z_0, \{q_f\})$ where δ is defined as

$$\delta(q_0, a, z_0) = \{(q_0, z_0a)\}$$
$$\delta(q_0, b, z_0) = \{(q_0, z_0b)\}$$
$$\delta(q_0, a, a) = \{(q_0, aa)\}$$
$$\delta(q_0, b, a) = \{(q_0, ab)\}$$
$$\delta(q_0, a, b) = \{(q_0, ba)\}$$
$$\delta(q_0, b, b) = \{(q_0, bb)\}$$
$$\delta(q_0, c, a) = \{(q_1, a)\}$$
$$\delta(q_0, c, b) = \{(q_1, b)\}$$
$$\delta(q_0, c, z_0) = \{(q_1, z_0)\}$$
$$\delta(q_1, a, a) = \{(q_1, \lambda)\}$$
$$\delta(q_1, b, b) = \{(q_1, \lambda)\}$$
$$\delta(q_1, \lambda, z_0) = \{(q_f, z_0)\}$$

If an initial ID of the PDA M is $(q_0, bacab, z_0)$ what is the ID after processing of *bacab* ? If the string is (*i*) *abcba* (*ii*) *abcb* (*iii*) *acba* (*iv*) *abaa* (*v*) *abab*, will M process the entire string. If so what will be the final ID.

Solution. Given that $(q_0, bacab, z_0)$ is the initial ID.

$$(q_0, bacab, z_0) \vdash (q_0, acab, z_0b)$$
$$\vdash (q_0, cab, z_0ba)$$
$$\vdash (q_1, ab, z_0ba)$$
$$\vdash (q_1, b, z_0b)$$
$$\vdash (q_1, \lambda, z_0)$$

$$\vdash (q_f, \lambda, z_0)$$

(*i*) $(q_0, abcba, z_0) \vdash (q_0, bcba, z_0a)$
$$\vdash (q_0, cba, z_0ab)$$
$$\vdash (q_1, ba, z_0ab)$$
$$\vdash (q_1, a, z_0a)$$
$$\vdash (q_1, \lambda, z_0)$$
$$\vdash (q_f, \lambda, z_0).$$

Yes. The final ID is (q_f, λ, z_0).

(*ii*) $(q_0, abcb, z_0) \vdash (q_0, bcb, z_0a)$
$$\vdash (q_0, cb, z_0ab)$$
$$\vdash (q_1, b, z_0ab)$$
$$\vdash (q_1, \lambda, z_0a).$$

Yes the final ID is (q_1, λ, z_0a)

(*iii*) $(q_0, acba, z_0) \vdash (q_0, cba, z_0a)$
$$\vdash (q_1, ba, z_0a).$$

No, the PDA halts at (q_1, ba, z_0a).

(*iv*) $(q_0, abac, z_0) \vdash (q_0, bac, z_0a)$
$$\vdash (q_0, ac, z_0ab)$$
$$\vdash (q_0, c, z_0aba)$$
$$(q_1, \lambda, z_0aba).$$

Yes the final ID is $(q_1, \lambda\ z_0aba)$.

(*v*) $(q_0, abab, z_0) \vdash (q_0, bba, z_0a)$
$$\vdash (q_0, ba, z_0ab)$$
$$\vdash (q_0, a\ z_0abbb)$$
$$\vdash (q_0, \lambda, z_0abbba).$$

Yes the final ID is (q_0, λ, z_0abba).

15. Let M be a PDA,

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F) \text{ where}$$
$$Q = \{q_0, q_1, q_f\}$$

$$\Sigma = \{a, b\}$$
$$\Gamma = \{a, z_0\}$$
$$F = \{q_f\}$$

and δ is given by

$$\delta(q_0, a, z_0) = \{(q_0, z_0a)\}$$
$$\delta(q_1, b, a) = \{(q_1, \lambda)\}$$
$$\delta(q_0, a, a) = \{(q_0, aa)\}$$
$$\delta(q_1, \lambda, z_0) = \{(q_1, \lambda)\}$$
$$\delta(q_0, b, a) = \{(q_1, \lambda)\}$$

What is the ID that the PDA M reaches after processing (*i*) a^3b^2 (*ii*) a^2b^3 (*iii*) a^5 (*iv*) b^5 (*v*) b^3a^2 (*vi*) *ababab* if starts with the initial ID.

(*i*) $(q_0, a^3b^2, z_0) \vdash (q_0, a^2b^2, z_0a)$
$\vdash (q_0, ab^2, z_0aa)$
$\vdash (q_0, b^2, z_0aaa)$
$\vdash (q_1, z_0aa)$
$\vdash (q_1, \lambda, z_0a)$,

Yes the final ID is (q_1, λ, z_0a).

(*ii*) $(q_0, a^2b^3, z_0) \vdash (q_0, ab^3, z_0a)$
$\vdash (q_0, b^3, z_0a)$
$\vdash (q_0, b, z_0)$

No, PDA halts at (q_0, b, z_0).

(*iii*) $(q_0, a^5, z_0) \vdash (q_0, a^4, z_0a)$
$\vdash (q_0, a^3, z_0aa)$
$\vdash (q_0, a^2, z_0aaa)$
$\vdash (q_0, a, z_0aaaa)$
$\vdash (q_0, \lambda, z_0aaaaa)$.

Yes the final ID is (q_0, λ, z_0a^5).

(*iv*) (q_0, b^5, z_0).

No PDA does not move

(*v*) (q_0, b^3a^2, z_0).

No PDA does not move.

(*vi*) $(q_0, ababab, z_0) \vdash (q_0, babab, z_0a)$
$\vdash (q_1, abab, b)$.

No PDA halts at $(q_1, abab, z_0)$.

16. Construct a PDA for the following language. The set of all strings over alphabet $\{a, b\}$ that contains exactly twice as many *a's* as *b's*.

Solution. Let L = {The set of all strings over alphabet $\{a, b\}$ that contains exactly twice as many *a*'s as *b*'s}. We know that

$$G = (N, T, S, P) \text{ where}$$
$$N = \{S\}$$
$$T = \{a, b\}$$
$$P = \{S \to SaSbSaS, S \to SaSaSbS, S \to SbSaSaS, S \to \lambda\}$$

generates L. Using 3.3.1, Construct a PDA

$$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$$

where

$$Q_0' = \{q_0'\}$$
$$\Sigma' = T = \{a, b\}$$
$$\Gamma' = N \cup T = \{S, a, b\}$$
$$Z_0' = S,$$
$$F' = \phi$$

and δ' is defined as follows:

$\delta'(q_0', \lambda, S) = \{(q_0', SaSbSaS), (q_0', SbSaSaS), (q_0', SaSaSbS), (q_0', \lambda)\}$

$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$

$\delta'(q_0', b, b) = \{(q_0', \lambda)\}$

It is easily seen that $L = N(M')$.

17. Construct a PDA to accept the language L = {The set of all strings over alphabet $\{a, b\}$ that contains exactly twice as many *a's* as *b's*} by final state.

Solution. Consider a PDA M' of problem 16. We have proved that $L = N(M')$. Now we construct a required PDA using Theorem 3.2.3.

Construct $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where

$$Q = Q' \cup \{q_0, q_f\}$$
$$= \{q_0', q_0, q_f\}$$

$$\Sigma = \{a, b\}$$
$$F = \{q_j\}$$
$$\Gamma = \Gamma' \cup \{z_0\}$$
$$= \{S, a, b, z_0\}$$

and δ is defined as follows:

$$\delta(q_0, \lambda, z_0) = \{(q_0', z_0S)\}$$
$$\delta(q_0', \lambda, S) = \{(q_0', SaSbSaS), (q_0', SbSaSaS), (q_0', SaSaSbS), (q_0', \lambda)\}$$
$$\delta(q_0', a, a) = \{(q_0', \lambda)\}$$
$$\delta(q_0', b, b) = \{(q_0', \lambda)\}$$
$$\delta(q_0', \lambda, z_0) = \{(q_j, z_0)\}.$$

It is easily seen that $L = T(M)$.

18. Let $L = \{a^m b^n / n < m\}$. construct (*i*) a context free grammar generating L (*ii*) a PDA accepting L by empty store.

Solution. (*i*) Construct a CFG $G = (N, T, S, P)$

where
$$N = \{S\}$$
$$T = \{a, b\}$$
$$P = \{S \to aSb, S \to aS, S \to a\}.$$

Then
$$L(G) = \{a^m b^n / n < m\}$$

(*ii*) Construct a PDA

$$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$$

where
$$Q' = \{q_0'\}$$
$$\Sigma' = \{a, b\}$$
$$\Gamma' = N \cup T = \{S, a, b\}$$
$$Z_0' = S$$
$$F' = \phi$$

and δ' is defined as follows :

$$\delta'(q_0', \lambda, S) = \{(q_0, bSa), (q_0', Sa), (q_0', a)\}$$
$$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$$

$$\delta'(q_0', b, b) = \{(q_0', \lambda)\}$$

It can be easily seen that $L = N(M')$.

19. Let $L = \{a^m b^n / n < m\}$. Construct a PDA to accept L by final state.

Solution. Consider a PDA M' given in problem 18. we have proved that $L = N(M')$. We construct the required PDA using Theorem 3.2.3.

Construct a PDA

$$M = \{Q, \Sigma, \Gamma, \delta, z_0, F\} \text{ where}$$
$$Q = Q' \cup \{q_0, q_j\}$$
$$= \{q_0', q_0, q_j\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \Gamma' \cup \{z_0\}$$
$$= \{S, a, b, z_0\}$$
$$F = \{q_j\}$$

and δ is defined as follows :

$$\delta(q_0, \lambda, z_0) = \{(q_0', z_0 z_0')\}$$
$$\delta(q_0', \lambda, S) = \{(q_0', bSa), (q_0', Sa), (q_0', a)\}$$
$$\delta(q_0', a, a) = \{(q_0', \lambda)\}$$
$$\delta(q'_0, b, b) = \{(q_0', \lambda)\}$$
$$\delta(q_0', \lambda, z_0) = \{(q_j, \lambda)\}.$$

It is easily seen that $L = T(M)$.

20. Construct a context free grammar generating the following language $L = \{a^n b^n / n \geq 1\} \cup \{a^m b^{2m} / m \geq 1\}$ and hence a PDA accepting L by empty store.

Solution. Construct $G = (N, T, S, P)$ where $N = \{S, A, B\}$, $T = \{a, b\}$, $P = \{S \to A, S \to B, A \to aAb, A \to ab, B \to aBbb, B \to abb\}$.

It is easily seen that $L = L(G)$. Using Theorem 3.3.1 construct a PDA

$$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$$

where
$$Q' = \{q_0'\}$$
$$\Sigma' = \{a, b\}$$
$$Z_0' = S$$

$$\Gamma' = N \cup T$$
$$= \{S, A, B, a, b\},$$
$$F = \phi$$

and δ' is defined as follows:

$$\delta'(q_0', \lambda, S) = \{(q_0', A), (q_0', B)\}$$
$$\delta'(q_0', \lambda, A) = \{(q_0', bAa), (q_0', ba)\}$$
$$\delta'(q_0', \lambda, B) = \{(q_0', bbBa), (q_0', bba)\}$$
$$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$$
$$\delta'(q_0', b, b) = \{(q_0', \lambda)\}.$$

It is easily seen that $L = N(M')$.

21. Construct a PDA to accept the language $L = \{a^n b^n / n \geq 1\} \cup \{a^m b^{2m} / m \geq 1\}$ by final state.

Solution. Consider a PDA M' given in problem 20. We have proved that $L = N(M')$. Now we construct the required PDA by using Theorem 3.2.3.

Construct a PDA

$$M = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$$

where

$$Q = Q' \cup \{q_0, q_j\}$$
$$= \{q_0', q_0, q_j\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \Gamma' \cup \{z_0\}$$
$$= \{S, A, B, a, b, z_0\}$$
$$F = \{q_j\}$$

and δ is defined as follows.

$$\delta(q_0, \lambda, z_0) = \{(q_0', z_0 S)\}$$
$$\delta(q_0', \lambda, S) = \{(q_0', A), (q_0', B)\}$$
$$\delta(q_0, \lambda, A) = \{(q_0', bAA), (q_0', ba)\}$$
$$\delta(q_0', \lambda, B) = \{(q_0', bbBa), (q_0', bba)\}$$
$$\delta(q_0', a, a) = \{q_0', \lambda)\}$$
$$\delta(q_0', b, b) = \{(q_0', \lambda)\}$$
$$\delta(q_0', \lambda, z_0) = \{(q_j, z_0)\}$$

It is easily seen that $L = T(M)$.

22. Construct a PDA to accept the language $L = \{a^n b^m c^m d^n / m, n \geq 1\}$ by empty store.

Solution. Consider the PDA M, given in problem 1. We have proved that $L = T(M)$. Now we construct a required PDA using Theorem 3.2.3.

Construct a PDA

$$M' = (Q', \Sigma', \Gamma, \delta', q_0', z_0', F')$$

where

$$Q' = Q \cup \{q_e, q_0'\}$$
$$= \{q_0, q_1, q_2\}$$
$$\Gamma' = \Gamma \cup \{z_0'\}$$
$$= \{z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_0'\}$$
$$F' = \phi$$

and δ' is defined as follows.

$$\delta'(q_0', \lambda, z_0') = \{(q_0', z_0' z_0)\}$$
$$\delta'(q_0, \lambda, z_0) = \{(q_1, z_0 z_5)\}$$
$$\delta'(q_1, \lambda, z_5) = \{(q_1, z_4 z_5 z_1), (q_1, z_4, z_6, z_1)\}$$
$$\delta'(q_1, \lambda, z_6) = \{(q_1, z_3 z_6 z_2), (q_1, z_3 z_2)\}$$
$$\delta'(q_1, a, z_1) = \{(q_1, \lambda)\}$$
$$\delta'(q_1, b, z_2) = \{(q_1, \lambda)\}$$
$$\delta'(q_1, c, z_3) = \{(q_1, \lambda)\}$$
$$\delta'(q_1, d, z_4) = \{(q_2, \lambda)\}$$
$$\delta'(q_1, \lambda, z_0) = \{(q_2, z_0)\}$$
$$\delta'(q_2, \lambda, z_1) = \{(q_e, \lambda)\}$$
$$\delta'(q_2, \lambda, z_2) = \{(q_e, \lambda)\}$$
$$\delta'(q_2, \lambda, z_3) = \{(q_e, \lambda)\}$$
$$\delta'(q_2, \lambda, z_4) = \{(q_e, \lambda)\}$$
$$\delta'(q_2, \lambda, z_5) = \{(q_e, \lambda)\}$$
$$\delta'(q_2, \lambda, z_6) = \{(q_e, \lambda)\}$$
$$\delta'(q_2, \lambda, z_0') = \{(q_e, \lambda)\}$$
$$\delta'(q_2, \lambda, z_0) = \{(q_e, \lambda)\}$$

$$\delta'(q_e, \lambda, z_1) = \{(q_e, \lambda)\}$$
$$\delta'(q_e, \lambda, z_2) = \{(q_e, \lambda)\}$$
$$\delta'(q_e, \lambda, z_3) = \{(q_e, \lambda)\}$$
$$\delta'(q_e, \lambda, z_4) = \{(q_e, \lambda)\}$$
$$\delta'(q_e, \lambda, z_5) = \{(q_e, \lambda)\}$$
$$\delta'(q_e, \lambda, z_6) = \{(q_e, \lambda)\}$$
$$\delta'(q_e, \lambda, z_0') = \{(q_e, \lambda)\}.$$

It is easily seen that $L = N(M')$

23. Construct a CFG which accepts $N(M)$, where $M = (\{q_0, q_1\}, \{a, b\}, \{q$ $z_0\}, \delta, q_0, z_0, \phi)$ where δ is defined by

1. $\delta(q_0, a, z_0) = \{(q_0, a\, z_0)\}$
2. $\delta(q_0, a, a) = \{(q_0', aa)$
3. $\delta(q_0, b, a) = \{(q_1, a)\}$
4. $\delta(q_1, b, a) = \{(q_1, a)\}$
5. $\delta(q_1, a, a) = \{(q_1, \lambda)\}$
6. $\delta'(q_1, \lambda, z_0) = (q_1, \lambda)\}$

Define $G = (N, T, S, P)$ where N consists of $[q_0, z_0, q_1]$, $[q_1, z_0, q_0]$, $[q_0, a, q_0]$, $[q, a, q_0]$, $[q_0, z_0, q_1]$, $[q_1, z_0, q_1]$, $[q_0, a, q_1]$, $[q_1, a, q_1]$

The productions in P are constructed as follows :

The S — productions are

$$P_1 : S \to [q_0, z_0, q_0]$$
$$P_2 : S \to [q_0, z_0, q_1]$$

$\delta(q_0, a, z_0) = \{(q_0, az_o)$ induces

$$P_3 : [q_0, z_0, q_0] \to a\,[q_0, a, q_0]\,[q_0, z_0, q_0]$$
$$P_4 : [q_0, z_0, q_0] \to a\,[q_0, a, q_1]\,[q_1, z_0, q_0]$$
$$P_5 : [q_0, z_0, q_1] \to a\,[q_0, a, q_0]\,[q_0, z_0, q_1]$$
$$P_6 : [q_0, z_0, q_1] \to a\,[q_0, a, q_1]\,[q_1, z_0, q_1]$$

$\delta(q_0, a, a) = \{(q_0, aa)\}$ yields

$$P_7 : [q_0, a, q_0] \to a\,[q_0, a, q_0]\,[q_0, a, q_0]$$

$$P_8 : [q_0, a, q_0] \to a\,[q_0, a, q_1]\,[q_1, a, q_0]$$
$$P_9 : [q_0, a, q_1] \to a\,[q_0, a, q_0]\,[q_0, a, q_1]$$
$$P_{10} : [q_0, a, q_1] \to a\,[q_0, a, q_1]\,[q_1, a, q_1]$$

$\delta(q_0, b, a) = \{(q_1, a)\}$ gives

$$P_{11} : [q_0, a, q_0] \to b\,[q_1, a, q_0]$$
$$P_{12} : [q_0, a, q_1] \to b\,[q_1, a, q_1]$$

$\delta(q_1, b, a) = \{(q_1, a)\}$ yields

$$P_{13} : [q_1, a, q_0] \to b\,[q_1, a, q_0]$$
$$P_{14} : [q_1, a, q_1] \to b\,[q_1, a, q_1]$$

$\delta(q_1, a, a) = \{(q_1, \lambda)\}$ gives

$$P_{15} : [q_1, a, q_1] \to a$$

$\delta(q_1, \lambda, z_0) = \{(q_1, \lambda)\}$ yields

$$P_{16} : [q_1, z_0, q_1] \to \lambda.$$

24. Construct a PDA A equivalent to the following context free grammar $S \to OBB$, $B \to OS/1S/0$. Test whether 010^4 is in $N(A)$.

Solution. Using 3.3.1. define PDA

$$M' = (\{q_0'\}, \{0, 1\}, \{S, B, 0, 1\}, \delta', q_0', S, \phi)$$

where δ' is defined as follows:

1. $\delta(q_0', \lambda, S) = \{(q_0', BB0)\}$
2. $\delta(q_0', \lambda, B) = \{(q_0', S0), (q_0', S1), (q_0' 0)\}$
3. $\delta(q_0', 0, 0) = \{(q_0', \lambda)\}$
4. $\delta(q_0', 1, 1) = \{(q_0', \lambda)\}$

$(q_0', 010^4, S) \vdash (q_0', 010^4, BB0)$ by 1.

$\vdash (q_0', 10^4, BB)$ by 3

$\vdash (q_0', 10^4, BS1)$ by 2

$\vdash (q_0', 0^4, BS)$ by 4

$\vdash (q_0', 0^4, BBB0)$ by 1

$\vdash (q_0', 0^3, BBB)$

$\vdash^* (q_0', 0^3, 000)$

$\vdash^* (q_0', \lambda, \lambda)$

Thus $010^4 \in N(A)$

25. Show that the set of all strings over $\{a, b, c\}$ in which the number of occurrences of a, b, c is the same is not context free.

Solution. $L = \{x / x \in \{a, b, c\}^* \text{ such that } n_a(x) = n_b(x) = n_c(x)\}$

Take $z = a^n b^n c^n$ in $L(G)$, where $z = uvwxy$, $1 \le |vx| \le n$. So vx cannot contain all the three symbols a, b and c. So uv^2wx^2y contain additional occurrences of the third symbol two symbols (found in vx) and the number of occurrences of the three symbols remains the same. This means the number of occurrences of the three symbols in uv^2wx^2y are not the same and so $uv^2wxy \notin L$. This is a contradiction. Hence the language is not context free.

26. Show that the language

$L = \{a^m b^m c^n / m \le n \le 2m\}$ is not context free.

Solution. As usual, n is the integer obtained from pumping lemma. Let $z = a^n b^n c^{2n}$. Then $z = uvwxy$, where $1 \le |vx| \le n$. So vx cannot contain all the three symbols a, b and c. If vx contains only a's and b's then we can choose i such that $uv^i wx^i y$ has more than $2n$ occurrences of a (or b) and exactly $2n$ occurrences of e. This means $uv^i wx^i y \notin L$, a contradiction. In other two cases also we can get a contradiction by proper choice of i. Thus the given language is not context free.

27. Prove that the language

$L = \{x \in \{a, b, c\}^* / n_a(x) < n_b(x) \text{ and } n_a(x) < n_c(x)\}$ is not context free.

Solution. Suppose L is a CFL and let n be the integer obtained from pumping lemma. Let $z = a^n b^{n+1} c^{n+1}$. Then $z = uvwxy$, where $1 \le |vx| \le n$. The string vwx can contain atmost two distinct symbols. This time, two cases are sufficient.

If v or x contains atleast one a, the vx contain any c's. Therefore vu^2wx^2y contains atleast as many a's and c's and cannot be L. If neither v nor x contains an a, then uv^0wx^0y still contains n a's; since vx contains one of the other two symbols, uv^0wx^0y contains fewer occurrences of that symbol then x does and therefore is not in L. We have obtained a contradiction and may conclude that L is not context free.

28. Construct a PDA to accept the language $L = \{a^{2n} bc / n \ge 0\}$ by empty store.

Solution. We know that $G = (N, T, S, P)$

where $N = \{S\}$, $T = \{a, b\}$,

$P = \{S \to a^2 S, S \to bc\}$ is the CFL generating L.

Usinbg Theorem 3.3.1 construct a PDA

$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$

where

$Q' = \{q_0'\}$

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

$\Gamma = N \cup T = \{S, a, b\}$

$F' = \phi$

$z_0' = S$

and δ' is defined as follows :

$\delta'(q_0', \lambda, S) = \{(q_0', Sa^2), (q_0', cb)\}$

$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$

$\delta'(q_0', b, b) = \{(q_0', b)\}$

It can be easily seen that $L = N(M')$.

29. Construct a PDA to accept the language $L = \{a^{2n} bc / n \ge 0\}$ by final state.

Solution. Consider the PDA M' given in problem 28. We have proved that $L = N(M')$. We use Theorem 3.2.3. to construct the required PDA. Construct a PDA

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

where

$Q = Q' \cup \{q_0, q_j\}$

$= \{q_0', q_0, q_j\}$

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

$\Gamma = \Gamma' \cup \{z_0\} = \{S, a, b, z_0\}$

$\Gamma = \{q_j\}$ and δ is defined as follows :

$\delta(q_0, \lambda, z_0) = \{(q_0', z_0 S)\}$

$\delta(q_0', \lambda, S) = \{(q_0', Sa^2), (q_0', cb)\}$

$\delta(q_0', a, a) = \{(q_0', \lambda)\}$

$\delta(q_0', b, b) = \{(q_0', \lambda)\}$

$\delta(q_0', \lambda, z_0) = \{(q_j, z_0)\}$

It is easily seen that $L = T(M)$.

30. Construct a PDA to accept the language $L = \{(ab)^n / n \ge 1\}$ by empty store.

Solution. $L = \{(ab)^n / n \geq 1\}$. We know that $G = (N, T, S, P)$ where $N = \{S, A, B\}$, $T = \{a, b\}$ and $P = \{S \to aB, B \to b, B \to bA, A \to aB\}$ generates L.

Using Theorem 3.3.1. construct a PDA

$$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$$

where

$$Q' = \{q_0'\}$$
$$\Sigma' = \{a, b\}$$
$$\Gamma' = N \cup T = \{S, A, B, a, b\}$$
$$Z_0' = S$$
$$F' = \phi$$

and δ' is defined as follows.

$$\delta'(q_0', \lambda, S) = \{(q_0', Ba)\}$$
$$\delta'(q_0', \lambda, A) = \{(q_0', Ba)\}$$
$$\delta'(q_0', \lambda, B) = \{(q_0', b), (q_0', Ab)\}$$
$$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$$
$$\delta'(q_0', b, b) = \{(q_0', \lambda)\}$$

It is easily seen that $L = N(M')$.

31. Construct a PDA to accept the language $L = \{(ab)^n / n \geq 1\}$ by final state.

Solution. Consider the PDA M' given in probelm 30. We have proved that $L = N(M')$. Using Theorem 3.2.3. we construct the required PDA as :

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F) \text{ where}$$
$$Q = Q' \cup \{q_0, q_j\}$$
$$= \{q_0', q_0, q_j\}$$
$$\Sigma = \Sigma' = \{a, b\}$$
$$\Gamma = \Gamma' \cup \{z_0\} = \{S, A, B, a, b, z_0\}$$
$$F = \{q_j\}$$

and δ is defined as follows:

$$\delta(q_0, \lambda, z_0) = \{(q_0', z_0 z_0')\}$$
$$\delta(q_0', \lambda, S) = \{(q_0', Ba)\}$$

$$\delta(q_0', \lambda, A) = \{(q_0', Ba)\}$$
$$\delta(q_0', \lambda, B) = \{(q_0', b), (q_0', Ab)\}$$
$$\delta(q_0', \lambda, z_0) = \{(q_f, z_0)\}.$$

It is easily seen that $L = T(M)$.

32. Construct a PDA to accept the language $L = \{$set of all words in a and b with an even number of a's$\}$ by empty store.

Solution. $L = \{$set of all words in a and b with an even number of a's$\}$. We know that $G = (N, T, P, S)$ where $N = \{S, A, B\}$, $T = \{a, b\}$

$P = \{S \to aA, S \to bB, B \to bB, B \to aA, A \to aB, A \to bA, A \to a, B \to b\}$

generates L. We use Theorem 3.3.1. to construct the required PDA. Construct a PDA

$$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$$

where

$$Q' = \{q_0'\}$$
$$\Sigma' = \{a, b\}$$
$$\Gamma' = N \cup T = \{S, A, B, a, b\}$$
$$Z_0' = S$$

and δ' is defined as follows :

$$\delta'(q_0', \lambda, S) = \{(q_0', Aa)\}$$
$$\delta'(q_0', \lambda, A) = \{(q_0', Ba), (q_0', Ab), (q_0', a)\}$$
$$\delta'(q_0', \lambda, B) = \{(q_0', Bb), (q_0', Aa), (q_0', b)\}$$
$$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$$
$$\delta'(q_0', b, b) = \{(q_0', \lambda)\}.$$

It is easily seen that $L = N(M')$.

33. Construct a PDA to accept the language $L = \{$set of all words in a and b with an even number of a's$\}$ by final state.

Solution. Consider a PDA M' given in problem 32. We have proved that $L = N(M')$. We use Theorem 3.2.3, to construct the required PDA which is :

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

where

$$Q = Q' \cup \{q_0, q_j\} = \{q_0', q_0, q_j\}$$
$$\Sigma = \Sigma' = \{a, b\}$$
$$\Gamma = \Gamma' \cup \{z_0\}$$

$= \{S, A, B, a, b, z_0\}$

$F = \{q_f\}$

and δ is defined as follows

$\delta(q_0, \lambda, z_0) = \{(q_0', z_0 z_0')\}$

$\delta(q_0', \lambda, S) = \{(q_0', Aa)\}$

$\delta(q_0', \lambda, A) = \{(q_0', Ba), (q_0', Ab\}, (q_0', a)\}$

$\delta'(q_0', \lambda, B) = \{(q_0', Bb), (q_0', Aa), (q_0', b)\}$

$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$

$\delta'(q_0', b, b) = \{(q_0', \lambda)\}$

$\delta(q_0', \lambda, z_0) = \{(q_f, z_0)\}$.

It is easily seen that $L = T(M)$.

34. Construct a PDA to accept the language

$L = \{ab^{2n}c/n \geq 0\}$ by empty store.

Solution. We know that $G = (N, T, S, P)$ where

$N = \{S, A\}$

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

$P = \{S \to aA, A \to bbA, A \to c\}$ generates L.

We use Theorem 3.3.1 to construct the required PDA.

Construct a PDA

$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$

where

$Q' = \{q_0'\}$

$\Sigma' = T = \{a, b, c\}$

$\Gamma' = N \cup T = \{S, A, a, b, c\}$

$Z_0' = S$

$F = \phi$

and δ' is defined as follows :

$\delta'(q_0, \lambda, S) = \{(q_0', Aa)\}$

$\delta(q_0', \lambda, A) = \{(q_0', Abb), (q_0', C)\}$

$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$

$\delta'(q_0', b, b) = \{(q_0', \lambda)\}$

$\delta'(q_0', c, c) = \{(q_0', \lambda)\}$.

It is easily seen that $L = N(M')$.

35. Construct a PDA to accept the lanaguage $L = \{ab^{2n}c/n \geq 0\}$ by final state.

Solution. $L = \{ab^{2n}c/n \geq 0\}$. We use 3.2.3. to construct the required PDA.

Construct a PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

where

$Q = Q' \cup \{q_0, q_j\}$

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

$\Gamma = \Gamma' \cup \{z_0\} = \{S, A, a, b, c, z_0\}$

$F = \{q_j\}$

and δ is defined as follows :

$\delta(q_0, \lambda, z_0) = \{(q_0', z_0 z_0')\}$

$\delta(q_0', \lambda, S) = \{(q_0', Aa)\}$

$\delta(q_0', \lambda, A) = \{(q_0', Abb), (q_0', c)\}$

$\delta(q_0', a, a) = \{(q_0', \lambda)\}$

$\delta(q_0', b, b) = \{(q_0', \lambda)\}$

$\delta(q_0', c, c) = \{(q_0', \lambda)\}$

$\delta(q_0', \lambda, z_0) = \{(q_j', z_0)\}$

It is easily seen that $L = T(M)$.

36. Construct a PDA for the language

$L = \{a^n b^m a^m b^n / m, n \geq 1\}$ by empty store.

Solution. $L = \{a^n b^m a^m b^n / m, n \geq 1\}$. We know that $G = (N, T, S, P)$ where

$N = \{S, A\}$, $T = \{a, b\}$ and

$P = \{S \to aSb, S \to aAb, A \to bAa, A \to ba\}$

generates L. Now we use Theorem 3.3.1 to construct the required PDA.

Construct a PDA

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

where

$Q' = \{q_0'\}$

$$\Sigma' = T = \{a, b\}$$
$$\Gamma' = Z \cup T = \{S, A, a, b\}$$
$$z_0' = S$$

and δ' is defined as follows :

$$\delta'(q_0', \lambda, S) = \{(q_0', bSa), (q_0', bAa)\}$$
$$\delta'(q_0', \lambda, A) = \{(q_0', aAb), (q_0', ab)\}$$
$$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$$
$$\delta'(q_0', b, b) = \{(q_0', \lambda)\}.$$

It is easily seen that $L = N(M')$

37. Construct a PDA to accept the language $L = \{a^n b^m a^m b^n / m, n \geq 1\}$ by final state.

Solution. Consider the PDA M' given in problem 36 we have proved that $L = N(M')$. We use Theorem 3.2.3 to construct the required PDA.

Construct a PDA

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

where

$$Q = Q' \cup \{q_0, q_j\}$$
$$= \{q_0', q_0, q_j\}$$
$$\Sigma = \Sigma' = \{a, b\}$$
$$\Gamma = \Gamma' \cup \{z_0\}$$
$$= \{S, A, a, b, z_0\}$$
$$F = \{q_j\}$$

and δ is defined as follows :

$$\delta(q_0, \lambda, z_0) = \{(q_0', z_0 z_0')\}$$
$$\delta(q_0', \lambda, S) = \{(q_0', bSa), (q_0', bAa)\}$$
$$\delta(q_0', \lambda, A) = \{(q_0', aAb), (q_0', ab)\}$$
$$\delta(q_0', a, a) = \{(q_0', \lambda)\}$$
$$\delta(q_0', b, b) = \{(q_0', \lambda)\}$$
$$\delta(q_0', \lambda, z_0) = \{(q_j, z_0)\}$$

It is easily seen that $L = T(M)$.

38. Construct a PDA to accept the language $L = \{x \in \{0, 1\}^* / n_0(x) = n_1(x)\}$ by empty store.

Solution. We know that $G = (N, T, S, P)$ where $N = \{S\}$, $T = \{0, 1\}$ and $P = \{S \to \lambda/0S1/1S0/SS\}$ generates L. We use Theorem 3.3.1 to construct a PDA

$$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$$

where

$$Q' = \{q_0'\}$$
$$\Sigma' = \{0, 1\}$$
$$\Gamma' = N \cup T = \{S, 0, 1\}$$
$$Z_0' = S$$

and δ' is defined as follows.

$$\delta'(q_0', \lambda, S) = \{(q_0', \lambda), (q_0', 1S0), (q_0', 0S1), (q_0', SS)\}$$
$$\delta'(q_0', 0, 0) = (q_0', \lambda)$$
$$\delta'(q_0', 1, 1) = (q_0', \lambda).$$

It is easily seen that $L = N(M')$.

39. Construct a PDA to accept the language $L = \{x \in \{0, 1\}^* / n_0(x) = n_1(x)\}$ by final state.

Solution. Consider the PDA M' given in problem 38. We have proved that $L = N(M')$. We use Theorem 3.2.3. to construct a PDA

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

where

$$Q = Q' \cup \{q_0, q_j\}$$
$$= \{q_0', q_0, q_j\}$$
$$\Sigma = \{0, 1\}$$
$$\Gamma = \Gamma' \cup \{z_0\}$$
$$= \{S, 0, 1, Z_0\}$$
$$F = \{q_j\}$$

and δ is defnied as follows :

$$\delta(q_0, \lambda, z_0) = (q_0', z_0 S)$$
$$\delta(q_0', \lambda, S) = \{(q_0', \lambda), (q_0', 1S0), (q_0', 0S1), (q_0', SS)\}$$
$$\delta(q_0', \lambda, z_0) = (q_j, z_0).$$

It is easily seen that $L = T(M)$.

40. Construct a PDA to accept the language $L(G) = \{x /$ each prefix of x has atleast as many $a's$ as $b's\}$ by empty store.

Solution. We know that $G = (N, T, S, P)$ where $N = \{S\}$, $T = \{a, b\}$, $P = \{S \to aS,\ S \to aSbS,\ S \to \lambda\}$ generates L. We use Theorem 3.3.1. to construct a PDA

$$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$$

where

$$Q' = \{q_0'\}$$
$$\Sigma' = \{a, b\} = T$$
$$\Gamma' = N \cup T = \{S, a, b\}$$
$$z_0' = S$$

and δ' is defined as follows :

$$\delta'(q_0', \lambda, S) = \{(q_0', Sa), (q_0', SbSa), (q_0', \lambda)\}$$
$$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$$
$$\delta'(q_0', b, b) = \{(q_0', \lambda)\}$$

It is easily seen that $L(G) = N(M')$.

41. Construct a PDA to accept the language

$L(G) = \{x /$each prefix of x has atleast as many $a's$ as $b's\}$

by fianl state.

Solution. Consider the PDA M' given in problem in 40. We have proved that $L = N(M')$. By using Theorem 3.2.3 we construct a PDA.

$$M' = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$$

where

$$Q = Q' \cup \{q_0, q_j\}$$
$$= \{q_0', q_0, q_j\}$$
$$\Sigma = \Sigma' = \{a, b\}$$
$$\Gamma = \Gamma' \cup \{z_0\} = \{S, a, b, z_0\}$$
$$F = \{q_j\}$$

and δ is defined as follows :

$$\delta(q_0, \lambda, z_0) = \{(q_0', z_0S)\}$$
$$\delta(q_0, \lambda, S) = \{(q_0', Sa), (q_0', SbSa), (q_0', \lambda)\}$$
$$\delta(q_0', \lambda\ z_0) = \{(q_j, z_0\}.$$

It is easily seen that $L = T(M)$.

42. Construct a PDA which accepts the language

$$L = \{x \in \{a, b, c\}^* / n_a(x) = n_b(x) = n_c(x)\}.$$

Solution. We know that $G = (N, T, S, P)$ where

$$N = \{S\}$$
$$T = \{a, b, c\}$$
$$P = \{S \to aSbScS/aScSbS/bSaScS/bScSaS/cSaSbS/cSbSaS/\lambda\}$$

We use Theorem 3.3.1 to construct a PDA

$$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$$

where

$$Q' = \{q_0'\}$$
$$\Sigma' = T = \{a, b, c\}$$
$$\Gamma' = N \cup T = \{S, a, b, c\}$$
$$Z_0' = S$$
$$F' = \phi$$

and δ' as defined as follows :

$$\delta'(q'_0, \lambda, S) = \{(q_0', ScSbSa), (q_0', SbScSa), (q_0', ScSaSb), (q_0', SaScSb), (q_0, SbSaSc), (q_0', SaSbSc), (q_0', \lambda)\}$$
$$\delta'(q_0', a, a) = (q_0', \lambda)$$
$$\delta'(q_0', b, b) = (q_0', \lambda)$$
$$\delta'(q_0', c, c) = (q_0', \lambda).$$

It is easily seen that $L = N(M')$.

43. Construct a PDA which accepts the language

$L = \{x \in \{a, b, c\}^* / n_a^{(x)} = n_b^{(x)} = n_c^{(x)}\}$ by final state.

Solution. Consider the PDA M' given problem 42.We have proved that $L = N(M')$. We use Theorem 3.2.3. to construct the PDA

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

where

$$Q = Q' \cup \{q_0, q_j\} = \{q_0', q_0, q_j\}$$
$$\Gamma = \Gamma' \cup \{z_0\} = \{S, a, b, c, z_0\}$$
$$\Sigma = \Sigma' = \{a, b, c\}$$
$$F = \{q_j\}$$

and δ is defined as follows :

$$\delta(q_0, \lambda, z_0) = (q_0', z_0S)$$

$$\delta(q_0', \lambda, S) = \{(q_0', ScSbSa), (q_0', SbScSa), (q_0', ScSaSb), (q_0', SaScSb), (q_0', SbSaSa), (q_0', SaSbSc), (q_0', \lambda)\}$$

$$\delta(q_0', \lambda, z_0) = \{(q_f, z_0)\}$$

$$\delta(q_0', \lambda, z_0) = \{(q_f, z_0)\}.$$

It is easily seen that $L = T(M)$.

44. Construct a PDA to accept the language $L\ \{a^n/n \geq 1\}$ by empty store.

Solution. We know that $G = (N, T, S, P)$ where $N = \{S\}$, $T = \{a\}$, $P = \{S \to as/a\}$ generagtes L. We use Theorem 3.3.1. to construct the PDA.

$$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$$

where

$$Q' = \{q_0'\}$$

$$\Sigma' = \{a\}$$

$$\Gamma' = N \cup T = \{S, a\}$$

$$z_0 = S$$

$$F = \phi \text{ and } \delta' \text{ is defined as follows:}$$

$$\delta'(q_0', \lambda, S) = \{(q_0', Sa), (q_0', a)\}$$

$$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$$

$$(q_0', a^n, S) \vdash (q_0', a^n, Sa)$$
$$\vdash (q_0', a^{n-1}, S)$$
$$\vdash (q_0', a^{n-1}, Sa)$$
$$\vdash (q_0', a^{n-2}, S)$$
$$\vdash^* (q_0', a, a)$$
$$\vdash (q_0', \lambda, \lambda)$$

$$a^n \in N(M')$$

Therefore,

That is, $L = N(M')$.

45. Construct a PDA to accept the language $L = \{a^n/n \geq 1\}$ by final state.

Solution. Consider the PDA M' given in problem 44. We have proved that $L = N(M')$. Using Theorem 3.2.3. construct a PDA

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F) \text{ where}$$

$$Q = Q' \cup (q_0, q_f) = \{q_0', q_0, q_f\}$$

$$\Sigma = \Sigma' = \{a\}$$

$$\Gamma = \Gamma' \cup \{z_0\}$$

$$= \{S, a, z_0\}$$

$$F = \{q_f\}$$

and δ is defined as follows:

$$\delta(q_0, \lambda, z_0) = \{(q_0', z_0S)\}$$

$$\delta(q_0', \lambda, S) = \{(q_0', Sa), (q_0', a)\}$$

$$\delta(q_0', a, a) = \{(q_0', \lambda)\}$$

$$\delta(q_0, \lambda, z_0) = \{(q_f, z_0)\}$$

$$(q_0, q_n, z_0) \vdash (q_0', a_n, z_0S)$$
$$\vdash^* (q_0', \lambda, z_0)$$
$$\vdash (q_f, \lambda, z_0) \quad \text{[See prob. 44]}$$

Therefore $a^n \in T(M)$

Thus $L = T(M)$.

46. Construct a PDA to accept the language $L = \{ww^R/w \in (a, b)^+\}$ by empty store.

Solution. We know that $G = (N, T, S, P)$

where $N = \{S\}, T = \{a, b\}$

$$P = \{S \to aSa, S \to bSb, S \to aa, S \to bb\}$$

generates L. Now we use Theorem 3.3.1. to construct a PDA $M' = (Q', \Sigma', \Gamma', \delta', z_0', q_0', F')$

where

$$Q' = \{q_0'\}$$

$$\Sigma' = \{a, b\} = T$$

$$\Gamma' = N \cup T = \{S, a, b\}$$

$$z_0' = S$$

and δ' is defined as follows :

$$\delta'(q_0', \lambda, S) = \{(q_0', aSa), (q_0', bSb), (q_0', aa), (q_0', bb)$$

$$\delta'(q_0', a, a) = \{(q_0', \lambda)\}$$

$$\delta'(q_0', b, b) = \{(q_0', \lambda)\}$$

It is easily seen that $L = N(M')$.

47. Construct a PDA to accept the language $L = \{ww^R / w \in (a, b)^*\}$ by final state.

Solution. Consider the PDA M' given in problem 46. We have proved that $L = N(M')$. By using Theorem 3.2.3. construct a PDA

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

where

$$Q = Q' \cup (q_0, q_j\}$$
$$= \{q_0', q_0, q_j\}$$
$$\Sigma = \{a, b\} = T$$
$$\Gamma = N \cup T = \{S, a, b\}$$
$$F = \{q_j\}$$

and δ is defined as follows :

$$\delta(q_0, \lambda, z_0) = \{(q_0', z_0S)\}$$
$$\delta(q_0', \lambda, S) = \{(q_0', aSa), (q_0', bSb), (q_0', aa)\ (q_0', bb)\}$$
$$\delta(q_0', a, a) = \{(q_0', \lambda)\}$$
$$\delta(q_0', b, b) = \{(q_0', \lambda)\}$$
$$\delta(q_0', \lambda, z_0) = \{(q_j, z_0)\}.$$

It is easily seen that $L = T(M)$.

48. Prove that the language $\{a^{n^2} / n \geq 1\}$ is not context free.

Solution. Let n be the natural number got by applying pumping lemma.

Let $z = a^{n^2}$. Write $z = uvwxy$ where $1 \leq |vx| \leq n$ (This is possible since $|vwx| \leq n$ by (ii) of pumping lemma). Let $|vx| = m$, $m \leq n$. By pumping lemma, uv^2wx^2y is in L. As $|uv^2wx^2y| > n^2$, $|uv^2wx^2y| = k^2$, where $k \geq n + 1$. But $|uv^2wx^2y| = n^2 + m < n^2 + 2n + 1$. So $|uv^2wx^2y|$ strictly lies between n^2 and $(n + 1)^2$ which means $uv^2wx^2y \notin L$, a contradiction. Hence $\{a^{n^2} / n \geq 1\}$ is not context free.

49. Construct a PDA to accept the language

$L = \{a^ib^j\} \mid i \leq j \leq 2i\}$ by empty store.

Solution. We know that $G = (N, T, S, P)$

where $N = \{S\}$, $T = \{a, b\}$ and

$P = \{S \to aSb, S \to ab, S \to SbSa\ SbS, S \to SbSbSaS, S \to SbSaSaS, S \to \lambda)\}$

generates L. We use 3.3.1. to construct a PDA.

$$M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$$

where

$$Q' = \{q_0'\}$$
$$\Sigma' = T = \{a, b\}$$
$$\Gamma' = N \cup T = \{S, a, b\}$$
$$F' = \phi$$
$$Z_0' = S$$

and δ' is defined as follows :

$$\delta'(q_0', \lambda, S) = \{(q_0', bSa), (q_0', ba), (q_0', SbSaSbS), (q_0', SaSbSbS), (q_0', SaSaSbS), (q_0', \lambda)\}$$
$$\delta', (q_0', a, a) = \{(q_0', \lambda)\}$$
$$\delta'(q_0', b, b) = \{(q_0', \lambda)\}.$$

It is easily seen that $L = N(M')$.

50. Construct a PDA to accept the language $L = \{a^ib^j / i \leq j \leq 2i\}$ by final state.

Solution. Considet the PDA M' given in problem 49. We have proved that $L = N(M')$. We use Theorem 3.2.3. to construct a PDA

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

where

$$Q = Q' \cup \{q_0, q_j\}$$
$$= \{q_0', q_0, q_j\}$$
$$\Sigma = \Sigma' = \{a, b\}$$
$$\Gamma = \Gamma' \cup \{z_0\}$$
$$= \{S, a, b, z_0\}$$
$$F = \{q_j\}$$

and δ is defined as follows :

$$\delta(q_0, \lambda, z_0) = \{(q_0', z_0S)\}$$
$$\delta(q_0', \lambda, S) = \{(q_0', bSa), (q_0', ba), (q_0', SbSaSbS), (q_0', SaSbSbS), (q_0', SaSaSbS), (q_0', \lambda)\}$$
$$\delta(q_0', a, a) = \{(q_0', \lambda)\}$$

$$\delta(q_0', b, b) = \{(q_0', \lambda)\}$$
$$\delta(q_0', \lambda, z_0) = \{(q_f, z_0)\}.$$

It is easily seen that $L = T(M)$.

51. Construct a Deterministic PDA to accept the language $L = \{a^n b^n : n \geq 0\} \cup \{a\}$.

Solution. Consider an NDPA

$$M = (Q, \Sigma, \Gamma, D, q_0, z_0, F)$$

where

$$Q = \{q_0, q_1, q_2, q_3\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \{0, 1\}$$
$$F = \{q_3\}$$

and δ is defined as follows:

$$\delta(q_0, a, 0) = \{(q_1, 10), (q_3, \in)\}$$
$$\delta(q_0, \lambda, 0) = \{(q_3, \varepsilon)\}$$
$$\delta(q_1, a, 1 = \{q_1, 11)\}$$
$$\delta(q_1, b, 1) = \{(q_2, \varepsilon)\}$$
$$\delta(q_2, b, 1) = \{(q_2, \varepsilon)\}$$
$$\delta(q_2, \varepsilon, 0) = \{(q_3, \varepsilon)\}$$

It is easily seen that $L = T(M)$.

52. Construct a PDA to accept the language $L = \{a^n b^n : n \geq 0\} \cup \{a\}$ by empty store.

Solution. Consider a PDA M in problem 51. We have proved that $L = T(M)$. We use 3.2.3. to construct the required PDA.

Construct $M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$

where $Q' = Q \cup \{q_0', q_e\}$ where $q_0', q_e \notin Q$.

$$L = \{q_0, q_1, q_2, q_3, q_0', q_e\}$$
$$\Sigma' = \Sigma = \{a, b\}$$
$$\Gamma' = \Gamma \cup \{z'_0\} = \{0, 1, z'_0\}, z_0' \notin \Gamma$$

and δ' is defined as follows:

$$\delta'(q_0', \varepsilon, z_0') = (q_0, z_0' z_0)$$
$$\delta'(q_0, a, 0) = \{(q, 10, (q_3, \varepsilon)\}$$

$$\delta'(q_0, \varepsilon, 0) = \{(q_3, \varepsilon)\}$$
$$\delta'(q_1, a, 1) = \{(q, 11)\}$$
$$\delta'(q_1, b, 1) = \{(q_2, \varepsilon)\}$$
$$\delta'(q_2, b, 1) = \{(q_2, \varepsilon)\}$$
$$\delta'(q_2, \varepsilon, 1) = \{(q_3, \varepsilon)\}$$
$$\delta'(q_3, \varepsilon, 0) = \{(q_e, \varepsilon)\}$$
$$\delta'(q_3, \varepsilon, 1) = \{(q_e, \varepsilon)\}$$
$$\delta'(q_3, \varepsilon, z_0') = \{(q_e, \varepsilon)\}$$
$$\delta'(q_e, \varepsilon, 0) = \{(q_e, \varepsilon)\}$$
$$\delta'(q_e, \varepsilon, 1) = \{(q_e, \varepsilon)\}$$
$$\delta'(q_e, \varepsilon, z_0') = \{(q_e, \varepsilon)\}$$

It is easily seen that $L = N(M')$.

53. Construct a PDA for the CFG with productions $S \to aSbb/a$,

$A \to bB$,

$B \to b$.

Solution. We use 3.3.1 to construct the required PDA.

Construct $M' = (Q', \Sigma', \Gamma', q_0', \delta', z_0', F')$

where

$Q' = \{q_0'\}$

$\Sigma' = T = \{a, b\}$

$\Gamma' = N \cup T = \{a, b, S, A, B\}$

$F' = \phi$ and

δ' is defined as follows:

$$\delta'(q_0', \varepsilon, S) = \{(q_0', bbSa), (q_0', a), (q_0', aSA), (q_0', a)\}$$
$$\delta'(q_0', \varepsilon, A) = \{(q_0', Bb)\}$$
$$\delta'(q_0', \varepsilon, B) = \{(q_0', b)\}$$
$$\delta'(q_0', a, a) = \{(q_0', \varepsilon)\}$$
$$\delta'(q_0', b, b) = \{(q_0', \varepsilon)\}$$

It is easily seen that $L = N(M')$.

54. Construct a PDA for the CFG with productions

$S \rightarrow aA$

$A \rightarrow aABC/bB/a$

$B \rightarrow b$

$C \rightarrow c$

Solution. We use 3.3.1. to construct the required PDA

Construct a PDA $M' = (Q', \Sigma', \Gamma', \delta', q_0', z_0', F')$

where
$Q' = \{q_0'\}$
$\Sigma' = \{a, b, c\}$
$\Gamma' = N \cup T = \{S, A, B, C, a, b, c\}$
$F' = \phi$ and

δ' is defined as follows

$\delta'(q_0', \varepsilon, S) = \{(q_0', Aa)\}$
$\delta'(q_0', \varepsilon, A) = \{(q_0', CBAa), (q_0', Bb), (q_0', a)\}$
$\delta'(q_0', \varepsilon, B) = \{(q_0', b)\}$
$\delta'(q_0', \varepsilon, C) = \{(q_0', c)\}$
$\delta'(q_0', a, a) = \{(q_0', \varepsilon)\}$
$\delta'(q_0', b, b) = \{(q_0', \varepsilon)\}$
$\delta'(q_0', c, c) = \{(q_0', \varepsilon)\}$

It is easily seen that $L = N(M')$.

55. Prove that the language $L = \{a^n b^n : n \geq 0\}$ is *a* deterministic context free language.

Solution. Consider the PDA

$M = (Q, \Sigma, \Gamma, \delta, q_0, O, F)$

where
$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b\}$
$\Gamma = \{0, 1\}$
$F = \{10\}$ and

δ is defined as follows:

$\delta(q_0, a, 0) = \{(q_1, 10)\}$

$\delta(q_1, a, 1) = \{(q_1, 11)\}$

$\delta(q_1, b, 1) = \{(q_2, \varepsilon)\}$

$\delta(q_2, b, 1) = \{(q_2, \varepsilon)\}$

$\delta(q_2, \varepsilon, 0) = \{(q_0, \varepsilon)\}$

It is easily seen that $L = L(M)$

Since M is deterministic, L is *a* DCFL.

56. Give an example of *a* CFL which is not *a* DCFL.

Solution. Let $L_1 = \{a^n b^n : n \geq 0\}$

$L_2 = \{a^n b^{2n} : n \geq 0\}$

and

We know that L_1 and L_2 are CFL's. Hence $L_1 \cup L_2$ is also a CFL.

Since the PDA has either to match one b or two against each a, and so has to make an initial choice whether the input is in L_1 or L_2. There is no information available at the beginning of the string by which the choice can be made deterministically. Hence L is a DCFL.

Note. It is observed that {set of all DCFL} $\subset$ {set of all CFL}

57. Prove that the language

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

Solution. Let m be the number in the pumping Lemma. Let $z = a^m b^m c^m$. Then $Z \in L$. If V_{xy} to contain only a's then the pumped string will obviously not be in L. If we choose a string containing an equal number of a's and b's then, the pumped string $a^m b^m c^m$ with $k \neq m$ can be generated and gain we have generated a string not in L. If V_y has the same number of a's and b's and c's then $| V_{xy} | \neq m$. Therefore, L is not context free.

58. Prove that the language $L = \{ww : w \in \{a, b\}^*\}$.

Solution. Consider the string $a^m b^m a^m b^m$. There are many choices for the string V_{xy}. If we take the choice as shown in Fig. 3.5.

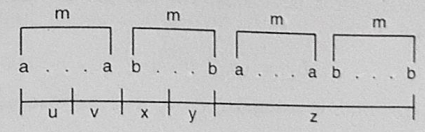

Fig. 3.5.

Take $i = 0$ to get a string of the form

$a^k b^j a^m b^m$, $k < m$ or $j < m$ which is not in L.

Similar arguments can be made for other choices also. We conclude that L is not context free.

59. Show that the language

$$L = \{a^n b^j : n = j^2\} \text{ is not context free.}$$

Solution. Let m be the number in the pumping Lemma. Consider the string $a^{m^2} b^m$. Now we have several choices. The only one that requires much thought is the one given in the following Fig. 3.6.

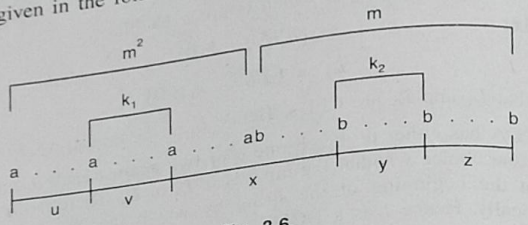

Fig. 3.6.

Pumping i times will yield a new string with $m^2 + (i-1)\,k$, a's and $m + (i-1)\,k^2$ b's. If we take $k_1 \neq 0$, $k_2 \neq 0$. We can pick $i = 0$. Since

$$\begin{aligned}(m - k_2)^2 &\leq (m-1)^2 \\ &= m^2 - 2m + 1 \\ &< m^2 - k_1,\end{aligned}$$

the result is not in L. If we take $k_1 = 0$, $k_2 \neq 0$ or $k_1 \neq 0$, $k_2 = 0$, then again with $i = 0$, the pumped string is not in L. Hence we conclude that L is not a context free language.

60. Show that the language

$$L = \{a^n b^n : n \geq 0,\ n \neq 100\} \text{ is context free.}$$

Solution. It is possible to prove this claim by constructing a PDA or a CFG for the language, but the proces is tedious. We can get a much neater argument with Theorem 3.4.6.

Let $L_1 = \{a^{100}\ b^{100}\}$

Then, because L_1 is finite, it is regular. Also, it is easy to see that

$$L = \{a^n b^n : n \geq 0\} \cap \overline{L}_1$$

Since the complement of a regular language is regular, $\overline{L}_1$ is also regular. Also $\{a^n b^n : n \geq 0\}$ is a CFL. By Theorem 3.4.6 it follows that L is a CFL.

EXERCISES

1. Give context free grammars generating the following sets:
   (*a*) The set of all strings over alphabet $\{a, b\}$ not of the form ww for some string w.
   (*b*) $\{a^i\ b^j\ c^k \mid i \neq j \text{ or } j \neq k\}$.
   (*c*) The set of even length strings in $\{a, b\}^*$ with the two middle symbols equal.
   (*d*) The set of all odd length strings in $\{a, b\}^*$ whose first, middle and the last symbols are the same.
   (*e*) $\{a^i\ b^j\ c^k \mid i = j + k\}$
   (*f*) $\{a^i\ b^j\ c^k \mid j = i + k\}$
   (*g*) $\{a^i\ b^j \mid i < 2j\}$
   (*h*) $\{a^i\ b^j\ c^k \mid j = i \text{ or } j = k\}$
   (*i*) $\{0^n\ 1^n \mid n \geq 0$
   [**Hint:** See problem 30 (*j*) $\{0^n\ 1\ 0^n \mid n \geq 1\}$]
   [**Hint:** See problem 32]
2. Construct a PDA to accept the language $L = \{a^i b^j \mid j > i\}$ by empty store. [See problem 18.]
3. Construct a PDA to accept the language $L = \{a^i b^j \mid j > i\}$ by final state [See problem 19.]
4. construct a PDA to accept the language $L = \{a^i b^j \mid i \leq 2j\}$ by empty store.
5. Construct a PDA to accept the language $L = \{a^i b^j \mid i \leq 2j\}$ by final state.
6. Construct a PDA equivalent to the following CFG
   (*a*) $S \to 0S0/1S1/A$, $A \to 2B3$, $B \to 2B3/3$.
   (*b*) $S \to aB/bA$, $A \to a/as/bAA$, $B \to b/bs/aBB$
   (*c*) $S \to AB$, $B \to a$, $A \to Aa$, $A \to bB$, $B \to Sb$
   (*d*) $S \to OB/1A$, $A \to 0/0S/1AA$, $B \to 1/1S/0BB$.
   (*e*) $S \to SbS/a$.
   (*f*) $S \to aSbb/aab$
7. Show that the following languages are not context free.
   (*a*) $L = \{a^m b^n / n = m^2\}$
   (*b*) $L = \{ww / w \in \{a, b\}^*\}$

(c) $L = \{a^i b^i a^i b^i / i \geq 0\}$

(d) $L = \{a^i b^i a^j b^i / i, j \geq 0\}$

(e) $L = \{wcw / w \in \{a, b\}^*\}$

(f) $L = \{a^n : n \geq 0\}$

(g) $L = \{a^n b^n : n \geq 0$, n is not a multiple of 5$\}$

(h) $L = \{w \in \{a, b\}^* : n^a(w) = n^b(w)$, w does not contain a substring $aab\}$.

8. Construct a context free grammar generating each of the following languages and hence find a PDA accepting them by final state.

(a) $\{a^n b^m a^n / m, n \geq 1\} \cup \}\{a^n c^n / n \geq 1\}$

(b) $\{a^n b^m c^m d^n / m, n \geq 1\}$

9. Construct a PDA accepting by empty store each of the following languages.

(a) $L = \{a^m b^n / m > n \geq 1\}$

(b) $L = \{x \in \{a, b\}^* / n_a(x) < 2n_b(x)\}$

(c) $L = \{a^n b^{n+m} a^m / n, m \geq 0\}$

(d) $L = \{0^n 1^n / n \geq 0\}$

(e) $L = \{0^n 1^n / n \geq 1\}$

(f) $L = \{a, b\}^+$

(g) $L = \{a\}^+$

(h) $L = \{0^m 1^n / m \neq n$ and atleast one of m and $n \geq 1\}$

(i) $L = \{01^m 2^n{}_3 / m, n \geq 1\}$

(j) $L = \{a^n b^n c^i / n \geq 1, i \geq 0\}$

(k) $L = \{a^n b a^m / n, m \geq 1\}$

(l) $L = \{a^j b^n c^n / n \geq 1, j \geq 0\}$

(m) $L = \{0^n 1^n / n \geq 1\} \cup \{1^m 0^m / m \geq 1\}$

(n) $L = \{a^l b^m c^n /$ one of λ, m, n equals 1 and the remaining two are equal.$\}$

(o) $L = \{a^l b^m c^n / l + m = n\}$

(p) $L = \{(ab)^n / n \geq 1\} \cup \{(ba)^n / n \geq 1\}$

(q) $L = \{a^n b^{2n} : n \geq 0\}$

(r) $L = \{a^n b^m c^{n+m} : n \geq 0, m \geq 0\}$

(s) $L = \{a^3 b^n c^n : n \geq 0\}$

(t) $L = \{a^n b^m : n \leq m \leq 3n\}$

(u) $L = \{w : n_a(w) = n_b(w) + 1\}$

(v) $L = \{w : n_a(w) + n_b(w) = n_c(w)\}$

(w) $L = \{w : 2n_a(w) \leq n_b(w) \leq 3n_a(w)\}$

[**Hint:** Construct CFG generating the language then use 3.3.1. to construct the required PDA.]

10. Construct a PDA accepting by final state each of the languages given in problem 8.

[**Hint :** Consider the PDA of problem 8. Then use 3.2.3 to construct the required PDA.].

11. Find an NPDA with atmost two states for the language $L = \{a^n b^{n+1} : n \geq 0\}$.

12. Give reasons why one might conjecture that the following language is not deterministic.

$L = \{a^n b^m c^k : n = m$ or $m = k\}$

13. Is the language $\{wcw^R : w \in \{a, b\}^*\}$ deterministic.

14. Show that $L = \{w \in \{a, b\}^* : n_a(w) \neq n_b(w)\}$ is a deterministic context free language.

15. Show that every regular language is a deterministic context free language.

16. Show that if L_1 is deterministic context free and L_2 is regular, then the language $L_1 \cup L_2$ is deterministic context free.

17. Show that under the conditions of problem 15, $L_1 \cap L_2$ is a deterministic context free language.

18. Is the following language context free ?

$L = \{a^{nm} : n$ and m are prime numbers$\}$

19. Prove that the complement of the language

$L = \{a^{n^2} : n \geq 0\}$ is not context free.

20. Give an example of a CFL whose complement is not a CFL.