4.1 Definition

AU : May-10, Marks 2

The pushdown automata will have **input tape, finite control and stack.**

The input tape is divided in many cells. At each cell only one input symbol is placed thus certain input string is placed on tape. The finite control has some pointer which point the current symbol which is to be read. At the end of input $ or Δ (blank) b (blank) symbol is placed which indicates end of input.

The stack is such a structure in which you can push and remove the items from one end only. For example if you place some coins one above the other, then a stack of coins is formed. While removing a coin we can only remove the coin which is at the top. Similarly while adding more coins to the stack we can place a new coin only on the topmost coin. Thus the stack of coin shows clear cut use of only one end of the stack.

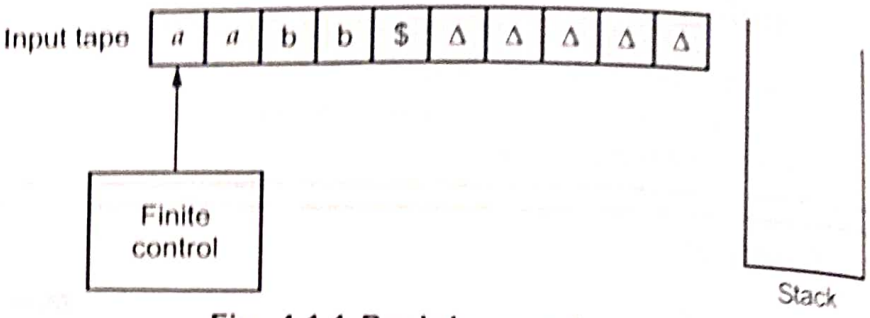


Fig. 4.1.1 Pushdown automata

In the pushdown automata, we are using the stack for storing the items temporarily. Inserting the symbol onto the stack is called push operation and removing a symbol from stack is called pop operation.

Let us have a formal definition of pushdown automata (PDA).

The PDA can be defined as a collection of seven components.

1. The finite set of states Q.
2. The input set Σ.
3. Γ is a stack alphabet.
4. q_0 is initial stage, $q_0 \in Q$.
5. Z_0 is a start symbol which is in Γ.
6. Set of final states $F \subseteq Q$.
7. δ is mapping function used for moving from current state to next state.

Following symbols are used while drawing the pushdown automata.

As we have discussed earlier PDA is more powerful than FA. Any language which can be accepted by FA can also be accepted by PDA. Not even this, PDA accepts a class of languages which even can not be accepted by FA. Thus PDA is much more superior to FA. Let us see how it works.

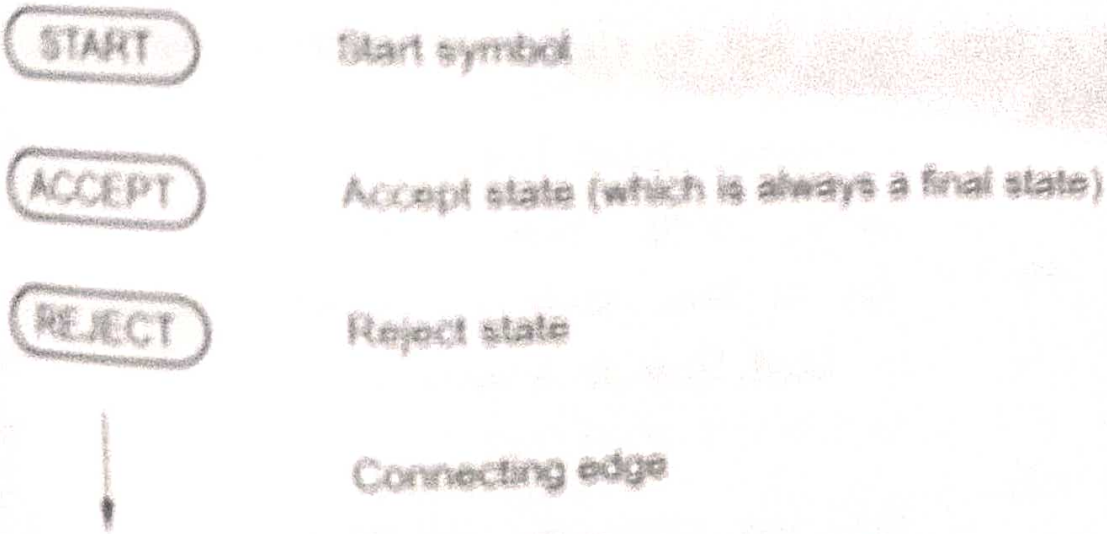


Fig. 4.1.2 Symbols used in PDA

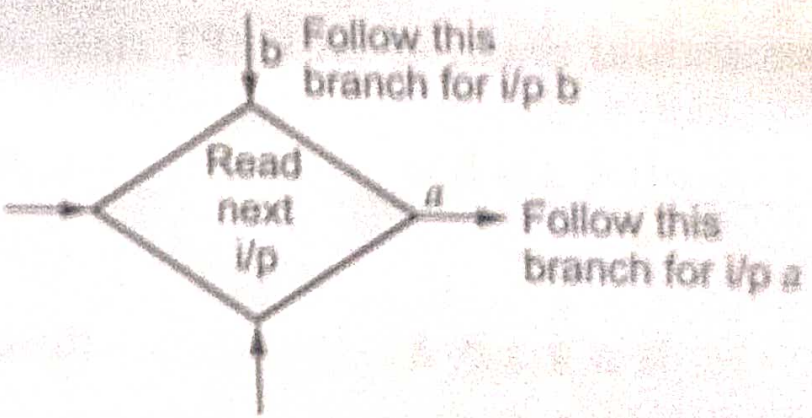


Fig. 4.1.3 Read symbol

Example 4.2.3 *For a given PDA*

AU : Dec.-09, Marks 8; Dec.-10, Marks 7

$M = (\{q_0, q_1, q_2\}, \{a, b, c\}, \{a, b, Z_0\}, \delta, q_0, Z_0, \{q_2\})$ *, the mapping function* δ *is given as*

$R_1 : \delta(q_0, a, Z_0) = \delta(q_0, a Z_0)$

$R_2 : \delta(q_0, b, Z_0) = \delta(q_0, b Z_0)$

$R_3 : \delta(q_0, a, a) = \delta(q_0, a a)$

$R_4 : \delta(q_0, b, a) = \delta(q_0, b a)$.

$R_5 : \delta(q_0, a, b) = \delta(q_0, a b)$

$R_6 : \delta(q_0, b, b) = \delta(q_0, b b)$

$R_7 : \delta(q_0, c, Z_0) = \delta(q_1, Z_0)$

$R_8 : \delta(q_0, c, a) = \delta(q_1, a)$

$R_9 : \delta(q_0, c, b) = \delta(q_1, b) R_{10} : \delta(q_1, a, a) = \delta(q_1, \varepsilon)$

$R_{11} : \delta(q_1, b, b) = \delta(q_1, \varepsilon)$

$R_{12} : \delta(q_1, \varepsilon, Z_0) = \delta(q_2, Z_0)$

Simulate for the input i) bbacabb

Solution : We will first draw the transition graph for given δ transitions.

Now we will simulate this PDA for the input bbacabb.

From initial ID, we will try to match rules R_1 to R_{12}.

$(q_0, \underline{b}bacabb, \underline{Z}_0) \vdash (q_0, \underline{b}acabb, \underline{b}Z_0)$

$\vdash (q_0, \underline{a}cabb, \underline{b}bZ_0)$

$\vdash (q_0, cabb, abbZ_0)$

$\vdash (q_1, \underline{a}bb, \underline{a}bbZ_0)$

$\vdash (q_1, \underline{b}b, \underline{b}bZ_0)$

$\vdash (q_1, b, bZ_0)$

$\vdash (q_1, \varepsilon, Z_0)$

$\vdash (q_2, Z_0)$

which is ACCEPT state.

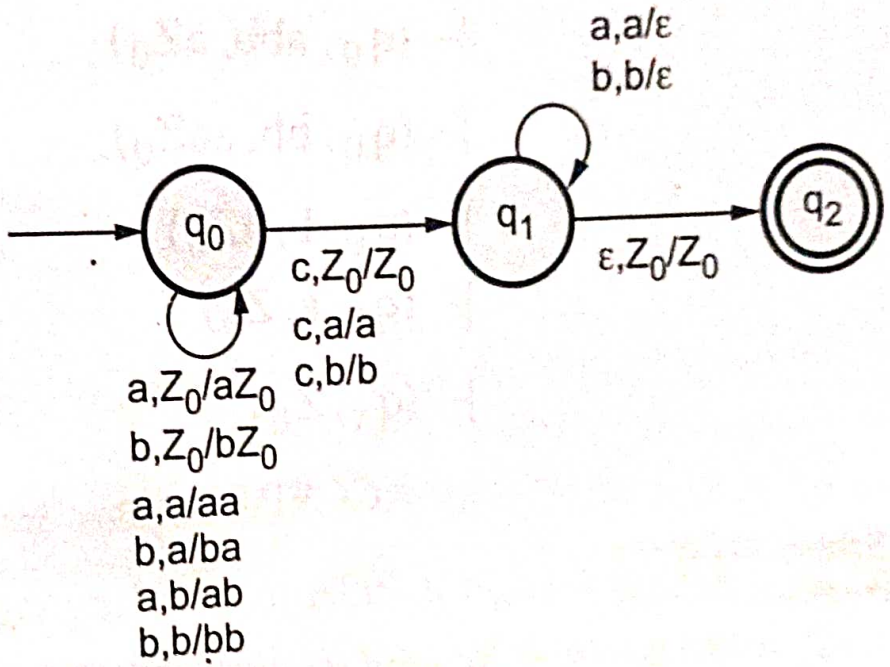


Fig. 4.2.3

Thus the input bbacabb is accepted by given PDA.

This is a language $L(M) = \{wCw^R \mid w \in (a + b)^*\}$ where w^R is reverse of w.

AU : May-07, Marks 8

... the language

Example 4.2.1 *Design a PDA for accepting a language* $\{L = a^n b^n \mid n \geq 1\}$.

AU : May-14, Marks 10

Solution : This is a language in which equal number of a's are followed by equal number of b's. The logic for this PDA can be applied as : first we will push all a's onto the stack. Then on reading every single b each a is popped from the stack. If we read all b and remove all a's and if we get stack empty then that string will be accepted. The instantaneous description can be given as -

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

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

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

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

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

where q_0 is a start state and q_2 is accept state.

We will simulate this PDA for following string -

$(q_0, aaabbb, Z_0) \vdash (q_0, aabbb, aZ_0)$

$\vdash (q_0, abbb, aaZ_0)$

$\vdash (q_0, bbb, aaaZ_0)$

$\vdash (q_1, bb, aaZ_0)$

$\vdash (q_1, b, aZ_0)$

$\vdash (q_1, \varepsilon, Z_0)$

$\vdash (q_2, \varepsilon)$

ACCEPT state.

Example 4.2.2 *Construct PDA for the language* $L = \{a^n b^{2n} \mid n \geq 1\}$ **AU : Dec.-07, Marks 8**

Solution : In this language n number of a's should be followed by 2n number of b's. Hence we will apply a very simple logic and that is if we read single 'a' we will push two a's onto the stack. As soon as we read 'b' then for every single 'b' only one 'a' should get popped from the stack. This basically maintains the $a^n b^{2n}$ count and sequence. The ID can be constructed as follows -

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

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

Now when we read b we will change the state from q_0 to q_1 and start popping corresponding 'a'. Hence ,

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

Thus this process of popping will be repeated unless all the symbols are read. Note that popping action occurs in state q_1 only.

$$\therefore \quad \delta (q_1, b, a) = (q_1, \varepsilon)$$

After reading all b's all the corresponding a's should get popped. Hence when we read ε as input symbol there should be nothing in the stack. Hence the move will be -

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

where the PDA $P = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, Z_0\}, \delta, q_0, Z_0, \{q_2\})$

We can summarize the ID as

$$\delta (q_0, a, Z_0) = (q_0, aaZ_0)$$

$$\delta(q_0, a, a) = (q_0, aaa)$$
$$\delta(q_0, b, a) = (q_1, \varepsilon)$$
$$\delta(q_1, b, a) = (q_1, \varepsilon)$$
$$\delta(q_1, \varepsilon, Z_0) = (q_2, \varepsilon)$$

Let us simulate this PDA for some input string "aaabbbbbb"

$\delta(q_0, aaabbbbbb, Z_0) \vdash (q_0, aabbbbbb, aaZ_0)$
$\vdash (q_0, abbbbbb, aaaaZ_0)$
$\vdash (q_0, bbbbbb, aaaaaaZ_0)$
$\vdash (q_1, bbbbb, aaaaaZ_0)$
$\vdash (q_1, bbbb, aaaaZ_0)$
$\vdash (q_1, bbb, aaaZ_0)$
$\vdash (q_1, bb, aaZ_0)$
$\vdash (q_1, b, aZ_0)$
$\vdash (q_1, \varepsilon, Z_0)$
$\vdash (q_2, \varepsilon)$

Final state or ACCEPT state.

Thus input gets accepted by using the constructed PDA..

Example 4.2.2 For a given PDA

Example 4.2.10 Construct the PDA for $L = \{ww^R \mid w \text{ is in } (a + b)^*\}$

AU : May-12, Marks 10; May-13, Marks 8

Solution : Let, the PDA $P = (Q, \Sigma, \Gamma, q_0, F, \delta)$

The mapping function δ is as given below -

$$\left.\begin{aligned} \delta(q_0, a, \varepsilon) &= (q_0, a) \\ \delta(q_0, b, \varepsilon) &= (q_0, b) \end{aligned}\right\} \text{ Pushing the elements onto stack}$$

$$\delta(q_0, \varepsilon, \varepsilon) = (q_f, \varepsilon)$$

$$\left.\begin{aligned} \delta\ (q_f,\ a,\ a) &= (q_f,\ \varepsilon) \\ \delta\ (q_f,\ b,\ b) &= (q_f,\ \varepsilon) \end{aligned}\right\}$$ Popping the elements.

$$\delta\ (q_f,\ \varepsilon,\ \varepsilon) = (q_f,\ \varepsilon) \rightarrow \text{ACCEPT}$$

Simulation of abba

$$\delta\ (q_0,\ (\underline{a}bba,\ \underline{\varepsilon}) \vdash \delta\ (q_0,\ \underline{b}ba,\ \underline{\varepsilon}a)$$

$$\vdash \delta\ (q_0,\ \underline{\varepsilon}ba,\ \underline{\varepsilon}ba)$$

$$\vdash \delta\ (q_f,\ \underline{b}a,\ \underline{b}a)$$

$$\vdash \delta\ (q_f,\ \underline{a},\ \underline{a})$$

$$\vdash \delta\ (q_f,\ \varepsilon,\ \varepsilon)$$

ACCEPT

ACCEPT

Example 4.2.9 *Build a PDA for the language* $L=\{0^n\ 1^m\ 0^n \mid m,n \geq 1\}$ *by empty stack.*

Solution : In this PDA n number of 0's are followed by any number of 1's followed by n number of 0's. Hnece the logic for design of such PDA will be as follows. Push all 0's onto the stack on encountering first zeros. Then if we read 1, just do nothing. Then read 0, and on each read of 0, pop one 0 from the stack. For instance :

This scenario can be written in the ID form as

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

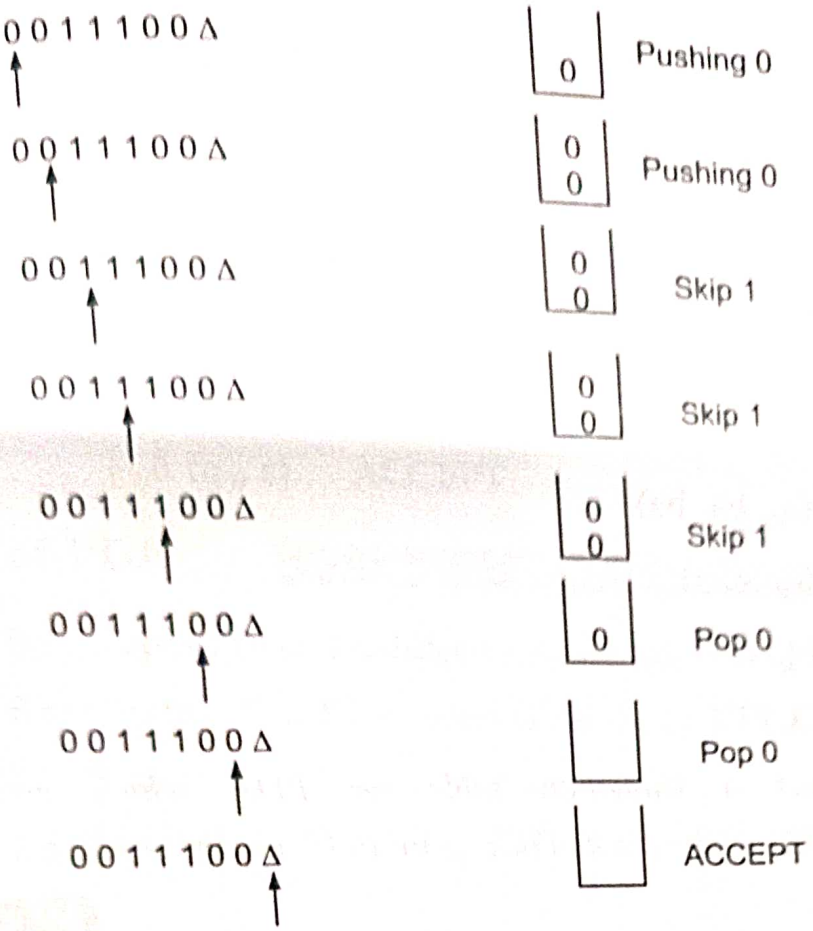
$$\delta(q_0, 0, 0) = \delta(q_0, 00)$$

$$\delta(q_0, 1, 0) = \delta(q_1, 0)$$

$$\delta(q_0, 1, 0) = \delta(q_1, 0)$$

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

$$\delta(q_0, \varepsilon, Z_0) = \delta(q_2, Z_0) \leftarrow \text{ACCEPT state}$$



For example :

$$\delta(q_0, 0011100, Z_0) \vdash \delta(q_0, 011100, 0Z_0)$$
$$\vdash \delta(q_0, 11100, 00Z_0)$$
$$\vdash \delta(q_0, 1100, 00Z_0)$$
$$\vdash \delta(q_1, 100, 00Z_0)$$
$$\vdash \delta(q_1, 00, 00Z_0)$$
$$\vdash \delta(q_1, 0, 0Z_0)$$
$$\vdash \delta(q_1, \varepsilon, Z_0)$$
$$\vdash \delta(q_2, Z_0)$$

ACCEPT

ACCEPT state.

Example 4.2.8 *Construct PDA for the language* $\{L = a^m b^m c^n \mid m, n \geq 1\}$

AU : May-07, Marks 8

Solution : This is the language in which all the a's are followed by equal number of b's followed by any number of c's. The simple logic that we can apply is : as we read as go on pushing each a onto the stack. As soon as we read b, pop one a from the stack. Repeat this process while reading all b's. Now when we read c, simply read c and do

nothing, because number of c's are arbitrary in given language. The Instantaneous Description (ID) for this logic can be as follows -

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

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

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

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

$$\delta(q_1, c, Z_0) = \delta(q_1, Z_0)$$

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

Let us derive this PDA for some example.

$$\begin{aligned}
\delta(q_0, aabbccc, Z_0) &\vdash \delta(q_0, abbccc, aZ_0) \\
&\vdash \delta(q_0, bbccc, aaZ_0) \\
&\vdash \delta(q_1, bccc, aZ_0) \\
&\vdash \delta(q_1, ccc, Z_0) \\
&\vdash \delta(q_1, cc, Z_0) \\
&\vdash \delta(q_1, c, Z_0) \\
&\vdash \delta(q_1, \varepsilon, Z_0) \\
&\vdash \delta(q_2, Z_0) \\
&\quad \text{ACCEPT}
\end{aligned}$$

Example 4.2.5 *Design a PDA for the language*

$L = \{w \mid w \in (a + b)^* \text{ and } n_a(w) > n_b(w)\}$

Solution : $n_a(w)$ means total number of a's in input string and $n_b(w)$ means total number of b's in input string. The problem states that total number of a's are more than total number of b's in input string. The logic for this PDA will be-

If we read a or b we will simply push it onto the stack. If the stack top has a symbol a and we read a, then also push it onto the stack. Same is true for b. But if we read 'a' and stack top symbol is 'b' then pop. Also if we read 'b' and stack top symbol is 'a' then pop the stack. Finally if we read ε (i.e. Complete string is read) then stack should contain a on the top. This means that total number of a's are more than total number of b's. The ID can be

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

$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$

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

$$\delta(q_0, b, b) = (q_0, bb)$$

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

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

$$\delta(q_0, \varepsilon, a) = (q_f, a)$$

where $P = (\{q_0, q_f\}, \{a, b\}, \{a, b, Z_0\}, \delta, q_0, Z_0, \{q_f\})$

Simulation : We will take some string to simulate this PDA for some input string.

Consider the input aababab

$$\delta(q_0, aababab, Z_0) \vdash (q_0, ababab, aZ_0)$$

$$\vdash (q_0, babab, aZ_0)$$

$$\vdash (q_0, abab, aZ_0)$$

$$\vdash (q_0, bab, aaZ_0)$$

$$\vdash (q_0, ab, aZ_0)$$

$\vdash (q_0, b, aaZ_0)$

$\vdash (q_0, \varepsilon, aZ_0)$

$\vdash (q_f, a)$

Accept state.

Thus input gets accepted.

...unmatched statements.

Example 3.6.3 *If G is the grammar* $S \rightarrow SbS \mid a$. *Show that G is ambiguous.*

AU : May-04, 14, Dec.-14, Marks 6

...ution : For the derivation of same string if there exists more than one parse tree then ... grammar is supposed to be an ambiguous grammar.

Let w = *ababa* be the string such that w ∈ G then, for production rules

$S \rightarrow SbS$

$S \rightarrow a$

The derivation trees are

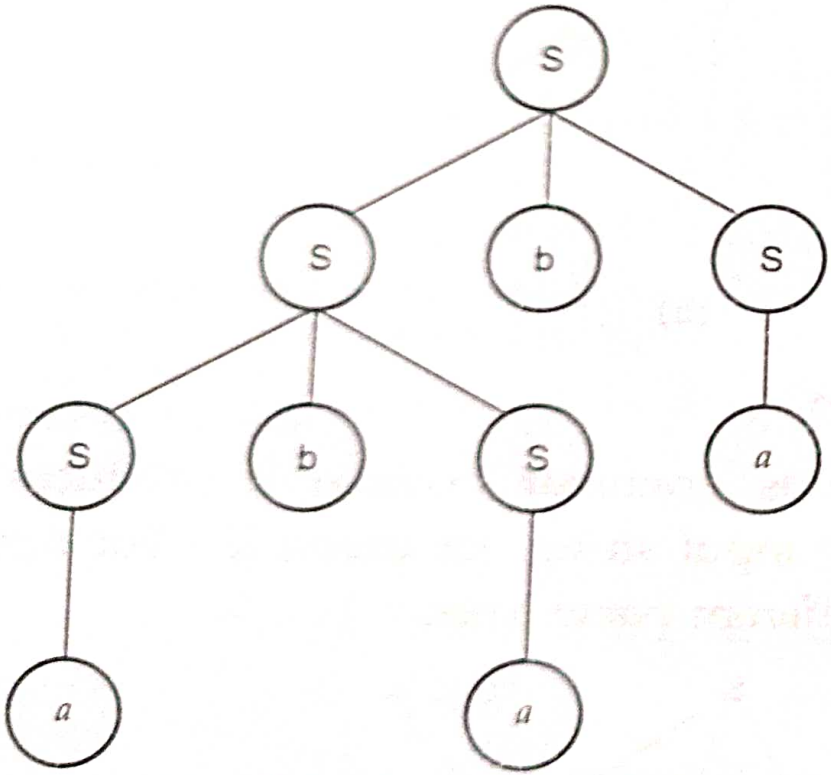


Fig. 3.6.2 Parse tree 1

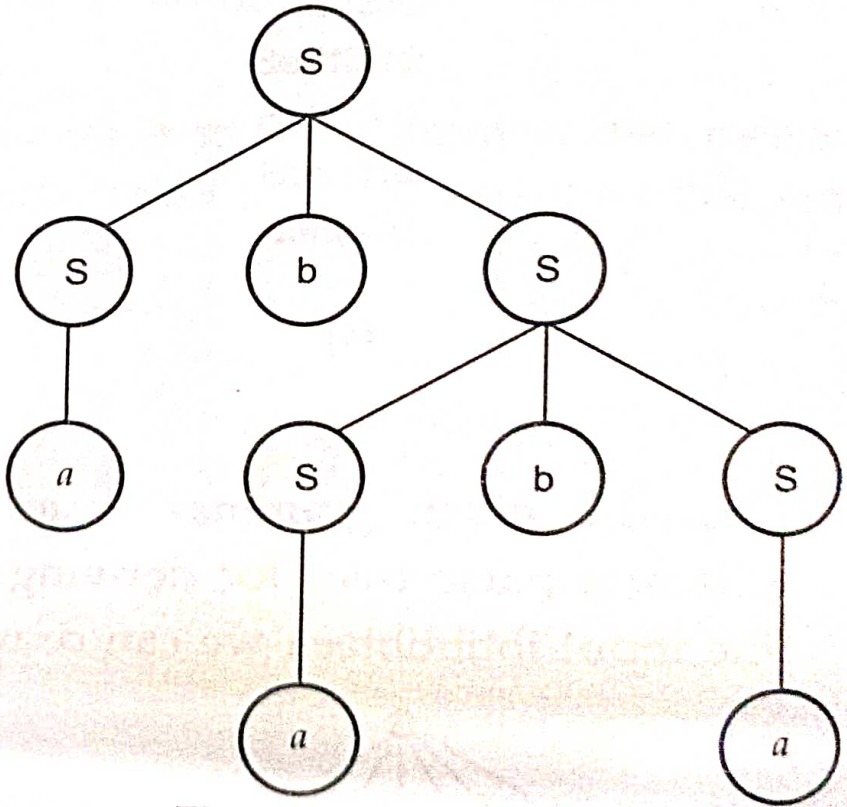


Fig. 3.6.3 Parse tree 2

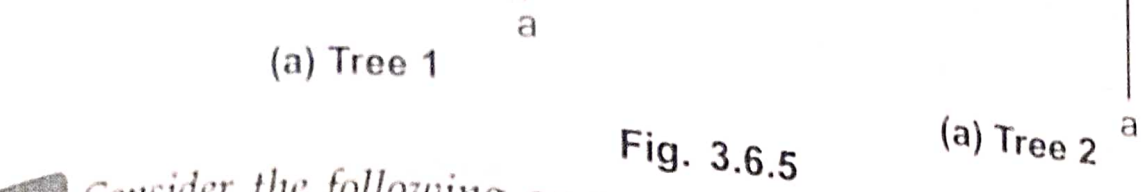


Fig. 3.6.5

Example 3.6.6 *Consider the following grammar G with productions*

$S \rightarrow 0B/1A \quad A \rightarrow 0/0S/1AA \quad B \rightarrow 1/1S/0BB.$

for the sentence $\omega = 00110101$

i) Construct a leftmost derivation and right most derivation.

ii) Show the corresponding parse tree for the above sentence.

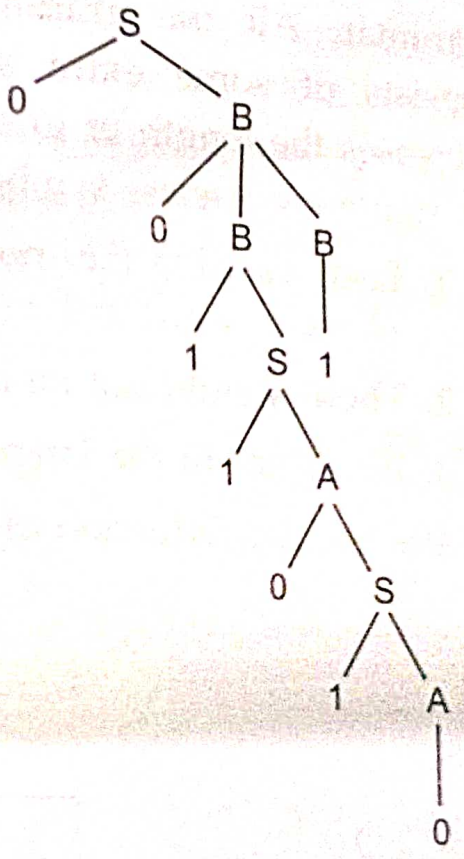
iii) Is the above grammar ambiguous ? If so, prove it.

Solution :

i)

Leftmost derivation	Rightmost derivation
S → 0B	S → 0B
00BB	00BB
001SB	00B1
0011AB	001S1
00110SB	0011A1
001101AB	00110S1
0011010B	001101A1
00110101	00110101

ii)



iii) The given grammar

Example for Understanding

Example 3.6.1 *The CFG given by* $G = (V, T, P, S)$

where $V = \{E\}$

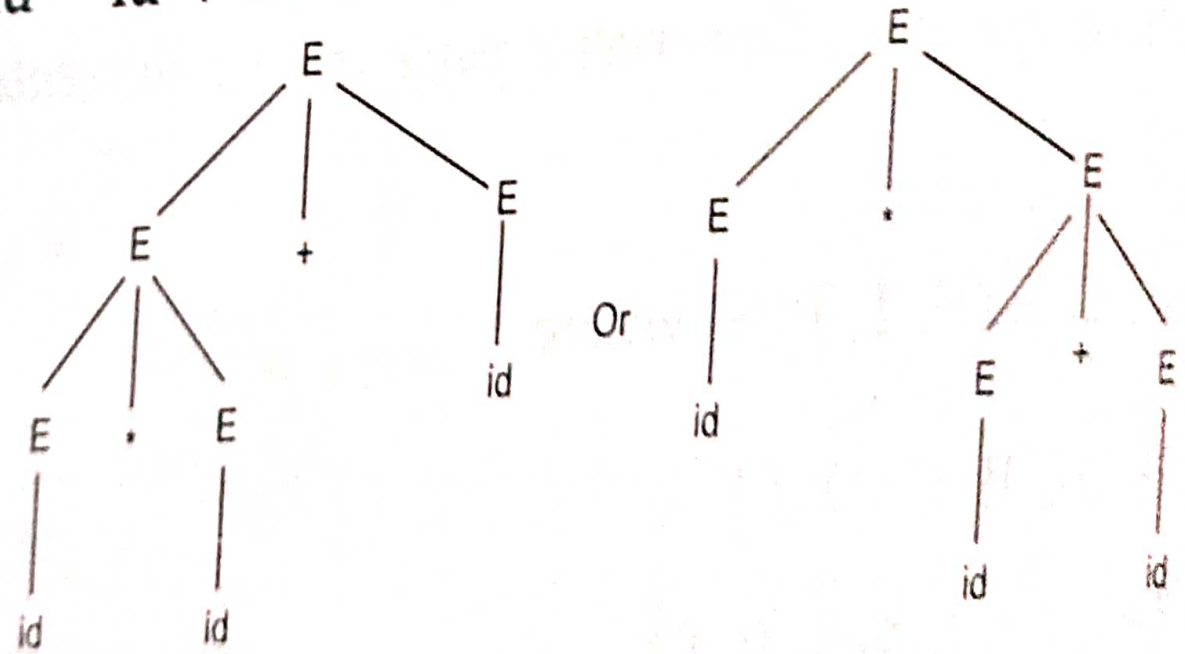
$T = \{id\}$

$P = \{E \rightarrow E + E, E \rightarrow E * E$

$E \rightarrow id\ \}$

$S = \{E\}$. *Is the grammar ambiguous ?* **AU : Dec.-10, Marks 2, May-12, Marks 6**

Solution : Now if the string is **id * id + id** then we can draw the two different parse trees indicating our id * id + id.



Thus the above grammar is an ambiguous grammar.

Solved Examples

3.2 Context Free Grammars and Languages

Definition :

The context free grammar can be formally defined as a set denoted by G = (V, T, P, S) where V and T are set of non-terminals and terminals respectively. P is set of production rules, where each production rule is in the form of

non-terminal → non-terminals

or non-terminal → terminals

S is a start symbol.

For example,

$$P = \{ S \rightarrow S + S$$
$$S \rightarrow S * S$$
$$S \rightarrow (S)$$
$$S \rightarrow 4 \}$$

If the language is 4 + 4 * 4 then we can use the production rules given by P. The start symbol is S. The number of non-terminals in the rules P is one and the only non terminal i.e. S. The terminals are +, *, (,) and 4.

We are using following conventions.

1. The capital letters are used to denote the non-terminals.
2. The lower case letters are used to denote the terminals.

Example : The formation of production rules for checking syntax of any English statement is

SENTENCE → NOUN VERB

NOUN → Rama / Seeta / Gopal

VERB → goes / writes / sings

Thus if want to derive a string "Rama sings" then we can follow the above rules.

Derivations and Languages

AU : May-10,11,12, Dec.-11,14, Marks 16