UNIT-III: CONTEXT FREE GRAMMAR AND PUSH DOWN AUTOMATA 3.1-3.65

3

CONTEXT FREE GRAMMAR AND LANGUAGES

3.1. GRAMMAR – INTRODUCTION

The most useful and general system for representing languages is based on the formal notion of a grammar.

A **grammar** is a quadruple (Σ, V, S, P), where

1. Σ is a finite non-empty set called the **terminal alphabet**. The elements of Σ are called the **terminals**.
2. V is a finite non-empty set disjoint from Σ. The elements of V are called the **non-terminals** or **variables.**
3. $S \in V$ is a distinguished non-terminal called the **start symbol.**
4. P is a finite set of **productions** (or **rules**) of the form.

$$\alpha \rightarrow \beta$$

where $\alpha \in (\Sigma \cup V)^* V(\Sigma \cup V)^*$ and $\beta \in (\Sigma \cup V)^*$, *i.e.*, α is a string of terminals and non-terminals containing atleast one non-terminal and β is a string of terminals and non-terminals.

Example 2.1 ***Let $G_1 = (\{0, 1\}, \{S, T, O, I\}, S, P)$ where P contains the following productions.***

$$S \rightarrow OT$$
$$S \rightarrow OI$$
$$T \rightarrow SI$$
$$O \rightarrow 0$$
$$I \rightarrow 1$$

Solution:

It generates the languages $L = \{0^n 1^n \mid n \geq 1\}$

3.2. TYPES OF GRAMMAR

Grammars can be divided into four classes by gradually increasing the restrictions on the form of the productions. Such a classification is due to Chomsky [Chomsky, 1956, Chomsky, 1963] and is called the Chomsky Hierarchy.

Definition: Let $G = (\Sigma, V, S, P)$ be a grammar.

1. G is also called a **Type-0** grammar or an **unrestricted** grammar, which is of the form $\alpha \rightarrow \beta$, where α, β are string of variables or terminals.
2. G is a **Type-1** or **context-sensitive** grammar if each production $\alpha \rightarrow \beta$ in P satisfies $| \alpha | \leq | \beta |$. By "special dispensation", we also allow a Type-1 grammar to have the production $S \rightarrow \varepsilon$, provided S does not appear on the right hand side of any production.
3. G is a **Type-2** or **context-free** grammar if each production $\alpha \rightarrow \beta$ in P satisfies $| \alpha | = 1$, *i.e.*, α is a single non-terminal.
4. G is a **Type-3** or **right linear** or **regular** grammar if each production has one of the following three forms:

$$A \rightarrow cB, A \rightarrow c, A \rightarrow \varepsilon$$

where A, B are non-terminals (with B = A allowed) and c is a terminal.

Example 3.2 ***Construct a grammar for generating English sentences.***

☺***Solution:***

<sentence> → <subject> <predicate>
<subject> → <noun>
<predicate> → <verb> <article> <noun>
<noun> → Ram
<noun> → letter
<verb> → writes
<article> → a

For the sentence, 'Ram writes a letter'

<sentence> ⇒ <subject> <predicate>
⇒ <noun> <verb> <article> <noun>
⇒ Ram writes a letter

3.3. CONTEXT FREE GRAMMARS AND LANGUAGES

The context-free languages are very similar to regular sets notably

- ✓ In defining programming languages,
- ✓ In formalizing the notion of parsing,
- ✓ Simplifying translation of programming languages and
- ✓ In other string-processing applications

Definition

There are four components inside a CFG G = (V, T, P, S) where

1. A finite set of variables 'V' also called as non-terminals.
2. A finite set of symbols called terminals T.
3. $S \subseteq V$ is the start symbol or variable.
4. A finite set of productions or rules which is of the form

$$A \rightarrow \alpha$$

where A - variable

α - string of zero or more terminals and strings

Example 3.3 ***A CFG G = (V, T, P, S)***

Where V – variable

T – Terminals

S – Start symbol

Whose productions are given by

$A \rightarrow Ba$

$B \rightarrow b$

which produces the string ba

Example 3.4 ***Construct the context-free grammar representing the set of palindromes, over (0 + 1)*.***

☺***Solution:***

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

(*i*) First possibility is S → 0 | 1 | ε palindrome with length = 1

(*ii*) if length > 1, then

$$S \rightarrow 0S0$$

Or

$$S \rightarrow 1S1$$

∴ The CFG for a palindrome is given by

$$S \rightarrow 0 \mid 1 \mid \varepsilon$$

$$S \rightarrow 0S0$$

$$S \rightarrow 1S1$$

Example 3.5 ***Construct a context free grammar for the given expression (a + b) (a + b + 0 + 1)*.***

☺***Solution:***

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

Operators ⇒ +, *

Identifiers ⇒ a, b, 0, 1

Here we need two variables to represent

(*i*) the start state and the expressions

(*ii*) the identifiers

The possible strings generated by the given expression are

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

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

$$I \rightarrow a$$

$$I \rightarrow b$$

$$I \rightarrow Ia$$

$$I \rightarrow Ib$$

$$I \rightarrow I0$$

$$I \rightarrow I1$$

In order to formulate the expression, we need some more productions like

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

Thus we had ten productions

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

Example 3.6 ***Construct a CFG for the language $L = \{a^n \mid n \text{ is odd}\}$.***

☺***Solution:***

The possible strings are a, a^3, a^5, ...

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

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

∴ The productions are

$$E \rightarrow a$$
$$E \rightarrow aaE$$

∴ CFG = (V, T, P, S) where

V = {E}

T = {a} S = E

P: $E \rightarrow a$

$E \rightarrow aaE$

Example 3.7 ***Construct a CFG for the language***

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

☺***Solution:***

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

The possible productions are aca, abcba, ...

So the productions are

$$E \rightarrow c$$
$$E \rightarrow aEa$$
$$E \rightarrow bEb$$

Thus the CFG is given by

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

Where

$$V = \{E\}$$
$$T = \{a, b, c\}$$
$$S = E$$
$$P : E \rightarrow c$$
$$E \rightarrow aEa$$
$$E \rightarrow bEb$$

3.4. DERIVATIONS AND LANGUAGES

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

- ✓ Using the rules from body to head.
- ✓ Using the rules from head to body.

The first approach is called by the name *recursive inference*.

Here we take strings from each variables, concatenate them in proper order and infer that the resulting string is in the language of the variable in the head.

Another approach is called as *derivation*.

Here we use the productions from head to body *i.e.*, from start symbol expanding till reaches the given string. This use of grammar is called derivation.

Example 3.8 ***For the language L = {wcw^R | w ε (0 + 1)*} check whether the string 01c10 belongs to the language L or not.***

☺***Solution:***

For the language L, we had the productions like

(*i*) E → c

(*ii*) E → 0E0

(*iii*) E → 1E1

(*i*) Recursive inference

	String inferred	For language	Production used	String used
1	C	E	(*i*)	-
2	0E0	E	(*ii*)	
3	01E10	E	(*iii*)	(2)
4	01c10	E	(*iv*)	(1), (3)

Thus identifying the strings sequentially and arrange it in proper order to find the result.

(*ii*) Derivation

$$E \Rightarrow 0E0 \qquad [\because E \to 0E0]$$
$$\Rightarrow 01E10 \qquad [\because E \to 1E1]$$
$$\Rightarrow 01c10 \qquad [\because E \to c]$$

Thus 01c10 belongs to the given CFL L = {wcw^R }

3.4.1. RELATIONS

There are two relations between strings in (VUT)*

(*i*) $\overset{}{\underset{G}{\Rightarrow}}$ Single derivation

(*ii*) $\overset{*}{\underset{G}{\Rightarrow}}$ Multiple derivation

Eg. (*i*) E $\overset{}{\underset{G}{\Rightarrow}}$ 0E0

(*ii*) E $\overset{*}{\underset{G}{\Rightarrow}}$ 01c10

3.4.2. LEFTMOST AND RIGHTMOST DERIVATIONS

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

Eg.

Let G = (V, T, P, S), V = {E}, T = {+, *, id}, S = E

Where P is given by

$$E \rightarrow E + E$$

$$E \rightarrow E * E$$

$$E \rightarrow id$$

Construct the leftmost derivation for id + id * id

$$E \underset{lm}{\Rightarrow} \underline{E} + E$$

$$\underset{lm}{\Rightarrow} id + \underline{E} \qquad [\because E \rightarrow id]$$

$$\underset{lm}{\Rightarrow} id + \underline{E} * E \qquad [\because E \rightarrow E * E]$$

$$\underset{lm}{\Rightarrow} id + id * \underline{E} \qquad [\because E \rightarrow id]$$

$$\underset{lm}{\Rightarrow} id + id * id \qquad [\because E \rightarrow id]$$

Rightmost Derivation

A derivation in which the rightmost variable is replaced at each step is called rightmost derivation.

Eg: For the same string id + id * id, the rightmost derivation is given by

$$E \underset{rm}{\Rightarrow} E * \underline{E}$$

$$\underset{rm}{\Rightarrow} \underline{E} * id \qquad [\because E \rightarrow id]$$

$$\underset{rm}{\Rightarrow} E * \underline{E} * id \qquad [\because E \rightarrow E + E]$$

$$\underset{rm}{\Rightarrow} \underline{E} + id * id \qquad [\because R \rightarrow id]$$

$$\underset{rm}{\Rightarrow} id + id * id \qquad [\because E \rightarrow id]$$

Conventions of CFG Derivations

- ✓ Lower case letters like a, b, … are terminals and also the digits 0-9 other characters like +, –, *, /, (,) are terminals.
- ✓ Upper case letters are non-terminals / variables.
- ✓ Lower-case greek letters such as α, β are strings or grammar symbols consisting of terminals and/or variables.

Language of the CFG

If G = (V, T, P, S) be a CFG, then the language L(G) is the set of terminal strings that have derivations from the start symbol.

$$L(G) = \{w \text{ in } T \mid S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

The language generated by a context-free grammar is called context-free language.

Sentential Form

The strings are produced/derived from the start symbol is called as 'sentential form'.

If G = (V, T, P, S) is a CFG, then a in (VUT)* such that

$$S \overset{*}{\Rightarrow} \alpha \text{ is a sentential form}$$

If $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$, then α is left sentential form

If $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha$, then α is right sentential form

SOLVED PROBLEMS

Example 3.9 ***Consider the CFG defined by the productions $S \rightarrow \varepsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1$. Show that the string w is a palindrome if and only if w is generated by the above CFG.***

☺***Solution:***

If part

Suppose w is a palindrome, we show by induction on | w | that w is in L(G).

Basis

If $|w| = 0$ or $|w| = 1$, then w is ε, 0 or 1. Since we have productions like

$$S \to \varepsilon$$

$$S \to 0$$

$$S \to 1$$

Then we say that $S \overset{*}{\Rightarrow} w$ in one of these cases. So, w is a palindrome for $|w| = 1$.

Induction

If $|w| \geq 0$.

Since $w = w^R$, we must begin and end with the same symbol, that is

$$w = 0x0 \text{ or}$$

$$w = 1x1$$

$\therefore$ $w = w^R$, thus the symbols are same at either end of w.

So w is a palindrome

Only if part

Assume w is in L(G), then $S \overset{*}{\Rightarrow} w$. The proof is an induction on the number of steps in a derivation.

Basis

For $|w| = 0$ and $|w| = 1$

$S \to \varepsilon$, $S \to 0$, $S \to 1$

All are directly derived from S in a single step, $S \Rightarrow w$

Induction

By induction, we can write

$$S \overset{*}{\underset{G}{\Rightarrow}} w$$

Eg: $S \Rightarrow 0S0 \overset{*}{\Rightarrow} 0x0$ where x is a palindrome with $|x| < w$

$\therefore$ $S \Rightarrow 0S0 \overset{*}{\underset{G}{\Rightarrow}} 0x0 \overset{*}{\underset{G}{\Rightarrow}} w$

Similarly

$$S \Rightarrow 1S1 \overset{*}{\Rightarrow} 1x1 \Rightarrow w \text{ } i.e., \text{ } S \overset{*}{\Rightarrow} w$$

Thus w is in the language generated by the given grammar.

∴ By the induction method, it is proved that all the strings generated by CFG are palindromes.

Example 3.10 ***Show that the G generates the languages L consisting equal number of a's and b's.***

$S \rightarrow aB \mid bA \mid \varepsilon$

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

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

☺***Solution:***

From the above productions, it is clear that

(*i*) S derives the string consisting of equal number of a's and b's

(*ii*) A derives the string consisting of one more a's than b's

(*iii*) B derives the string consisting of one more b's than a's

Basis

Let w be the string

(*i*) if $|w| = 1$.

For A, $A \rightarrow a$

For B, $B \rightarrow b$

(*ii*) if $|w| = 2$

$S \rightarrow ab$

∴ S derives equal number of a's and b's

Induction

(*iii*) if $|w| = n + 1$

S produces productions are of the forms

$$S \rightarrow aB \quad \ldots (1)$$

$$S \rightarrow bA \quad \ldots (2)$$

If we take the (1) production

$$S \Rightarrow aB \overset{*}{\underset{G}{\Rightarrow}} w = aw' \text{ where } B \overset{*}{\underset{G}{\Rightarrow}} w'$$

If we take the (2) production

$$S \Rightarrow bA \overset{*}{\underset{G}{\Rightarrow}} w = bw'', \text{ where } A \overset{*}{\underset{G}{\Rightarrow}} w''$$

provided the length of w' and w'' are same and w' consists of one more b's than a's and w'' consists of one more a's than b's.

From this we can conclude that S derives the string w consisting of equal number of a's and b's.

Example 3.11 ***Design context-free grammar for the following languages.***

(i) The set $\{0^n 1^n \mid n \geq 1\}$ that is, the set of all strings of one or more 0's followed by an equal number of 1's.

☺***Solution:***

The possible strings generated for the language $0^n 1^n$ are

01, 0011, 000111, ...

i.e., equal number of 0's and 1's

The possible CFG for this language is

$$S \rightarrow 0S1$$

$$S \rightarrow 01$$

The CFG G = (V, T, P, S)

where

$$V = \{S\}$$

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

$$S = S$$

$$P : S \rightarrow 0S1$$

$$S \rightarrow 01$$

3.4.3. APPLICATIONS OF CFG

1. Parsers
2. YACC parser – generator
3. Markup languages design
4. XML and DTD construction

3.5. AMBIGUITY IN GRAMMARS AND LANGUAGES

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

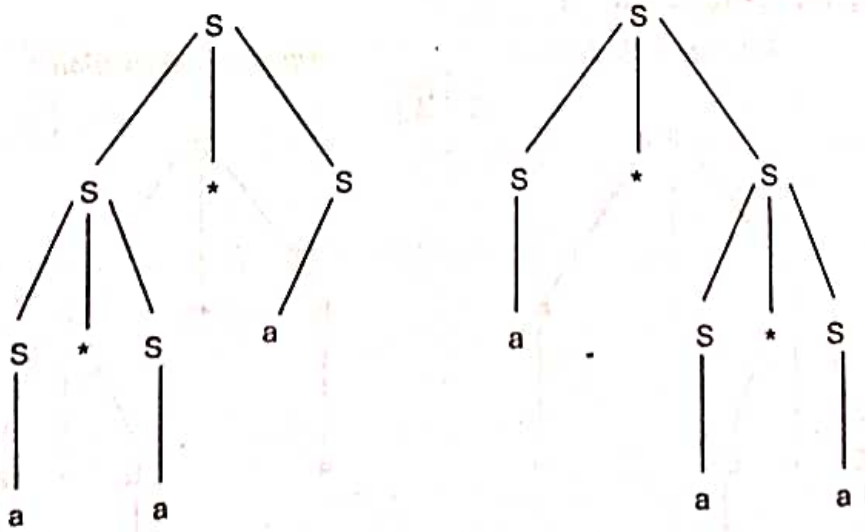
Ambiguous Grammar

A CFG G = (V, T, P, S) is ambiguous if there is atleast one string w in T* is having two different leftmost derivations or rightmost derivations, each with the same root S and same yield w.

Example 3.12 ***If G is the grammar $S \rightarrow S + S \mid S * S \mid a$, show that G is ambiguous.***

☺***Solution:*** Consider w = a * a * a

Leftmost derivation ***Rightmost derivation***

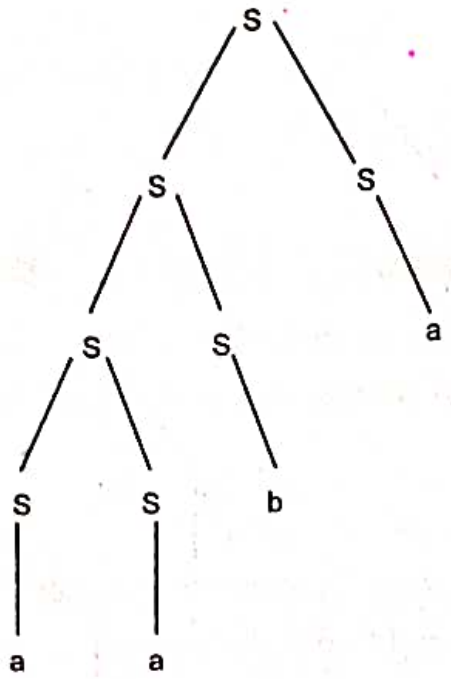


Thus we have two leftmost derivation trees for w, thus G is ambiguous.

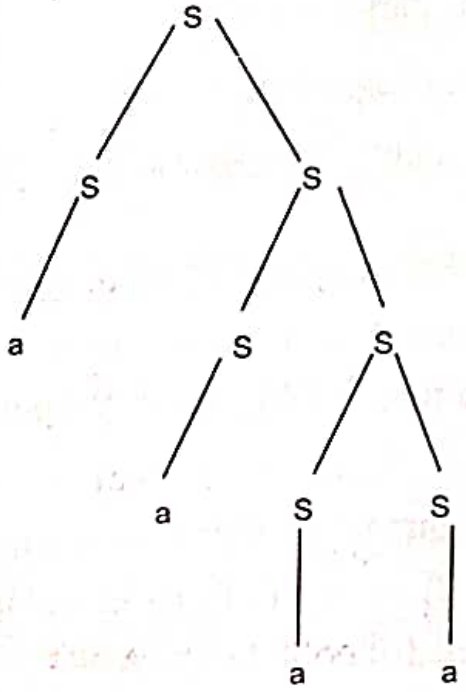
Example 3.13 ***Show that the grammar defined by the productions $S \rightarrow SS \mid a \mid b$ is ambiguous.***

☺***Solution:*** Considering the string aaba

Leftmost derivation



Leftmost derivation



Since aaba has two leftmost derivation trees, it is ambiguous

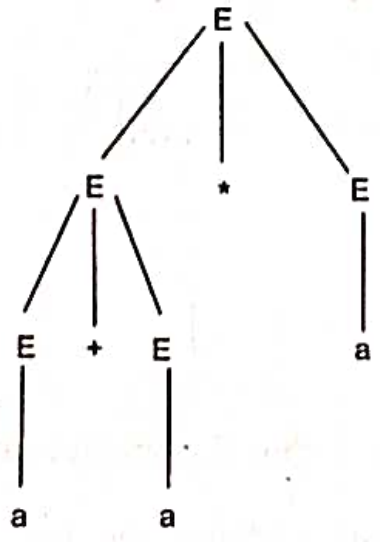
Example 3.14 ***Show that the grammar defined by the productions***

$E \rightarrow E + E \mid E * E \mid a$ ***(Apr/May – 2005)***

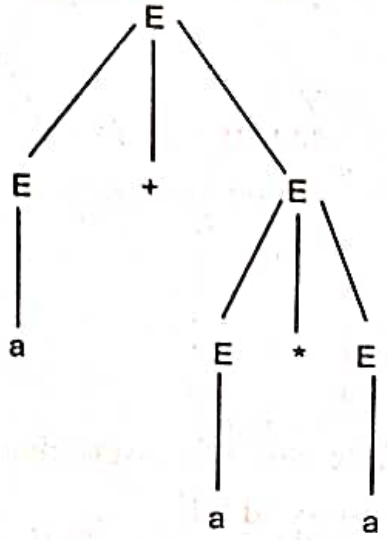
☺***Solution:***

Considering the string a + a * a

Rightmost derivation



Rightmost derivation

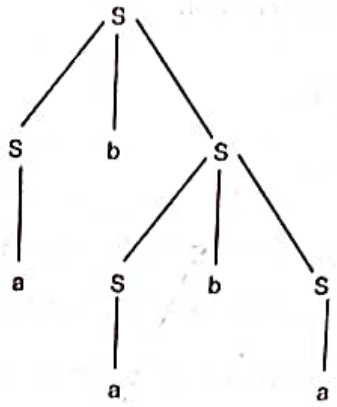


Since a + a * a has two rightmost derivation trees, then it is ambiguous.

Example 3.15 ***If G is the grammar S → SbS | a, show that G is ambiguous.***

☺***Solution:***

Let w = ababa



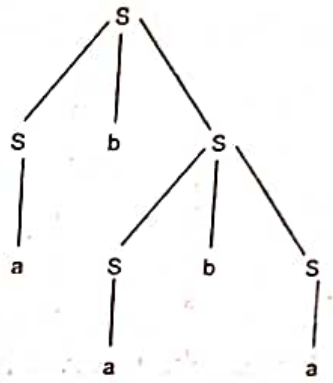
(*i*) $S \underset{lm}{\Rightarrow} SbS$

$\underset{lm}{\Rightarrow}$ abS [$\because S \to a$]

$\underset{lm}{\Rightarrow}$ abSbS [$\because S \to SbS$]

$\underset{lm}{\Rightarrow}$ ababS [$\because S \to a$]

$\underset{lm}{\Rightarrow}$ ababa [$\because S \to a$]



(*ii*) $S \overset{rm}{\Rightarrow} SbS$

$\overset{rm}{\Rightarrow}$ SbSbS [$\because S \to SbS$]

$\overset{rm}{\Rightarrow}$ SbSba [$\because S \to a$]

$\overset{rm}{\Rightarrow}$ Sbaba [$\because S \to a$]

$\overset{rm}{\Rightarrow}$ ababa [$\because S \to a$]

For the string w = ababa, we have two derivations

$\therefore$ G is ambiguous

Example 3.16 ***Show that E → E + E | E * E | (E) | id is ambiguous.***

☺***Solution:***

When a CFG has two or more derivations for a single word or yield, then the grammar G is said to be ambiguous.

(*i*) $E \underset{lm}{\Rightarrow} E + E$

$\underset{lm}{\Rightarrow}$ id + E

$\underset{lm}{\Rightarrow}$ id + E * E

$\underset{lm}{\Rightarrow}$ id + id * E

$\underset{lm}{\Rightarrow}$ id + id * id

(*ii*) $E \overset{rm}{\Rightarrow} E * E$

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

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

$\overset{rm}{\Rightarrow} id + id * id$

Since id + id * id has a two parse trees, the grammar G is said to be ambiguous.

3.5.1. REMOVING AMBIGUITY FROM GRAMMARS

The following are the causes of ambiguity in the grammar.

(*i*) Precedence of operators is not followed.

Eg. E + E * E group the * and + in two ways.

(*ii*) A sequence of identical operators can group either from the left or from the right.

The solution to the problem of enforcing precedence is to introduce several variables, each of which represents those expressions that share a level of "binding strength".

1. A factor is an expression that cannot be broken apart by any adjacent operator either * or +.
   - (*a*) Identifiers: It is not possible to separate the letters of an identifier by attaching an operator.
   - (*b*) Parenthesized expression: It is the purpose of parentheses to prevent what is inside from becoming the operand of any operator outside the parentheses.
2. A term is a product of one or more factors. This expression is not broken by the + operator. For eg. The proper grouping of a * b + c is (a * b) + c.
3. An expression can be broken by either an adjacent * or an adjacent +. An expression is a sum of one or more terms.

For example:

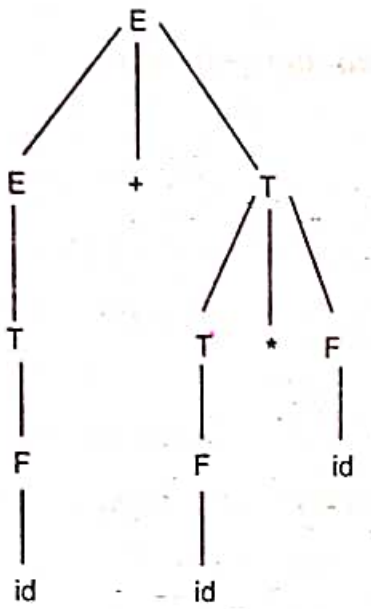
An unambiguous expression grammar is given by

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow id$

Find the parse tree for id + id * id.



∴ The yield is id + id * id.

3.5.2. INHERENT AMBIGUITY

A context free language L is said to be inherently ambiguous if all its grammar are ambiguous. For an ambiguous grammar, it is possible to construct an unambiguous grammar.

Example 3.17 ***An inherently ambiguous grammar is given by***

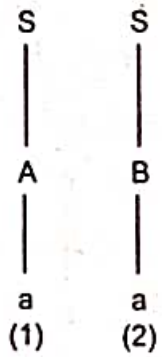
$S \rightarrow A \mid B$

$A \rightarrow a$

$B \rightarrow a$

For the input sting a

☺Solution:



There are two parse trees for this grammar.

SOLVED PROBLEMS

Example 3.18 ***Prove that the grammar***

$S \rightarrow aB \mid ab$

$A \rightarrow aAB \mid a$

$B \rightarrow ABb \mid ab$

Is ambiguous

☺***Solution:***

For the input string ab

(1) $S \Rightarrow aB \Rightarrow ab$

(2) $S \Rightarrow ab$

$\therefore$ It is ambiguous.

Example 3.19 ***Show that the grammar***

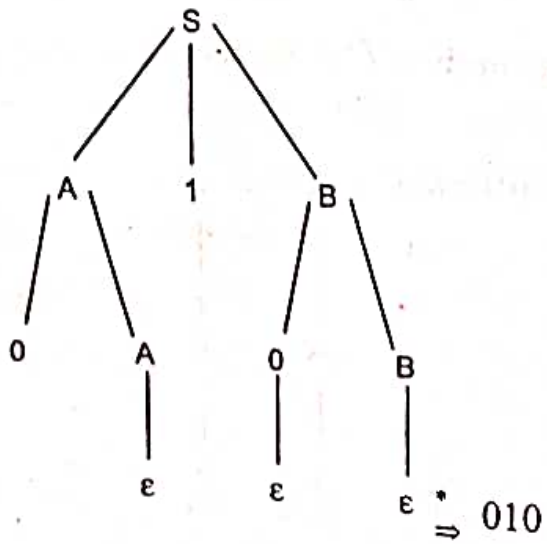
$S \rightarrow A1B$

$A \rightarrow 0A \mid \varepsilon$

$B \rightarrow 0B \mid 1B \mid \varepsilon$ ***is unambiguous***

☺***Solution:***

Take the input string 010



$\overset{*}{\Rightarrow}$ 010

which is unique

$$S \underset{lm}{\Rightarrow} A1B$$

$$\underset{lm}{\Rightarrow} 0^i\ 1B \qquad [\because A \to 0A \mid \varepsilon]$$

$$\Rightarrow 0^i\ 1\ (0+1)^i \qquad [\because B \to 0B \mid 1B \mid \varepsilon]$$

is unique for w.

By applying induction hypothesis, every string has unique leftmost derivation.

So the grammar is unambiguous.

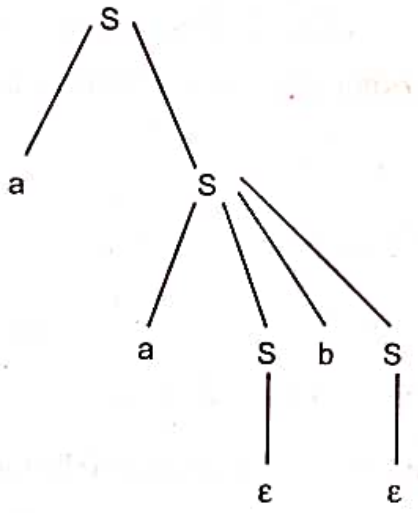
Example 3.20 ***Show that the grammar***

$S \to aS \mid aSbS \mid \varepsilon$ is ambiguous and find the unambiguous grammar.

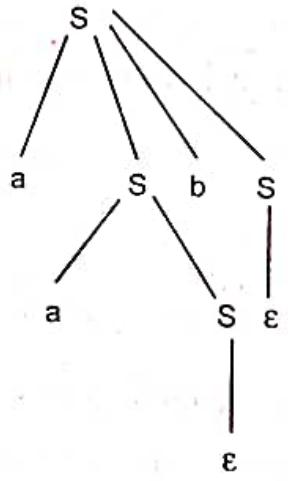
☺***Solution:***

Let us take an input string aab.

Leftmost derivation



Leftmost derivation



Since the input string aab has two leftmost derivations, the grammar is ambiguous.

Ultimately S produces either a or ab or ε.

So $E \to a \mid ab \mid \varepsilon$

Then rewrite the expression for unambiguous grammar as

$S \to T$

$T \to F \mid TF$

$F \to E$

$E \to a \mid ab \mid \varepsilon$

∴ The parse tree for the input string aab.

3.6. PARSE TREES

The derivations can be represented by trees using 'Parse trees'. But in compilers parse tree is used as a data structure to represent the source program.

Let G = (V, T, P, S) be a grammar.

The parse tree for G is a tree with the following conditions.

1. Each interior node is labeled by a variable in V.
2. Each leaf is labeled by either a variable, a terminal or ε.
3. If an interior node is labeled A, and its children are labeled X_1, X_2, ... X_k respectively from the left, then

   $A \rightarrow X_1, X_2, \ldots X_k$ is a production in P
4. If $A \rightarrow \varepsilon$, then A is considered to be the label.

Example 3.21 ***Construct a parse tee for the grammar***

$E \rightarrow E + E$

$E \rightarrow id$

☺***Solution:***

The parse tree is constructed as follows

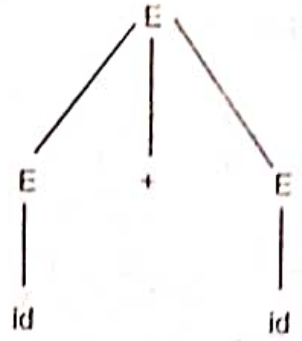


Fig. 3.1. Parse tree showing the derivation $E \to E + E \overset{*}{\Rightarrow} id + id$

The Yield of a Parse Tree

- ✓ The string obtained by concatenating the leaves of a parse tree from the left is called the *yield* of a parse tree.
- ✓ The yield is always derived from the root. Here root is the start symbol.
- ✓ All leaves are labeled either with a terminal or with ε, thus the yield is a terminal string.

Example 3.22 ***For the grammar G defined by the productions***

$S \to A \mid B$

$A \to 0A \mid \varepsilon$

$B \to 0B \mid 1B \mid \varepsilon$

Find the parse tree for the yields.

(i) 1001 (ii) 00101 (iii) 00011

☺***Solution:***

(*i*) 1001

Start from the root

$S \Rightarrow A1B$

$\Rightarrow 1\,B$ [$\because A \to \varepsilon$]

$\Rightarrow 10B$ [$\because B \to 0B$]

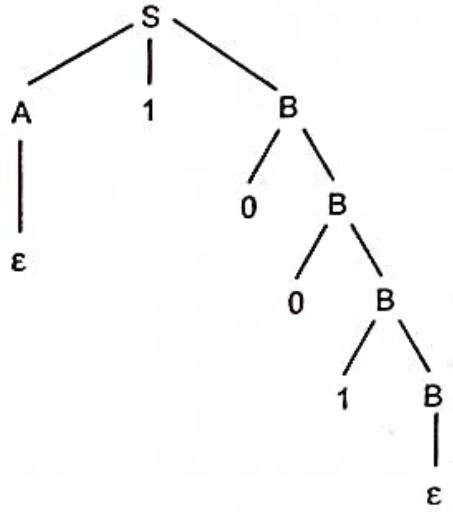
$\Rightarrow 100B$ [$\because B \to 0B$]

$\Rightarrow 1001B$ [$\because B \to 1B$]

$\Rightarrow 1001$ [$\because B \to \varepsilon$]

$\therefore \quad S \overset{*}{\Rightarrow} 1001$

The corresponding parse tree is given by



∴ The yield is 1001.

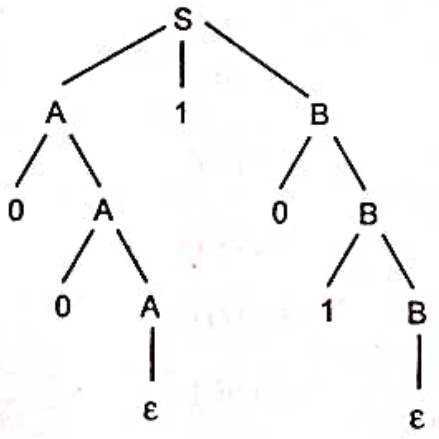
(*ii*) 00101

Start from the root

$$
\begin{aligned}
S &\Rightarrow A1B \\
&\Rightarrow 0A1B && [A \to 0A] \\
&\Rightarrow 00A1B && [A \to 0A] \\
&\Rightarrow 001B && [A \to \varepsilon] \\
&\Rightarrow 0010B && [B \to 0B] \\
&\Rightarrow 00101B && [B \to 1B] \\
&\Rightarrow 00101 && [B \to \varepsilon]
\end{aligned}
$$

$\therefore S \overset{*}{\Rightarrow} 00101$

The parse tree is given by



∴ The yield is 00101

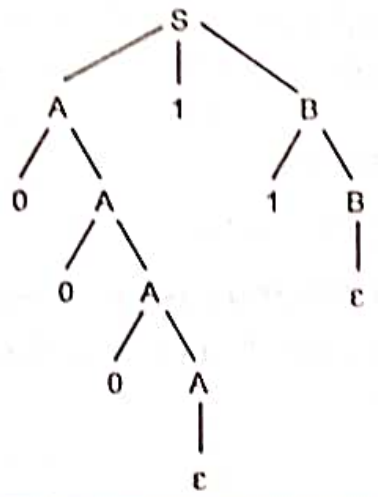
(*iii*) 00011

Start from the root

$$
\begin{aligned}
S &\Rightarrow A1B \\
&\Rightarrow 0A1B && [A \to 0A] \\
&\Rightarrow 00A1B && [A \to 0A] \\
&\Rightarrow 000A1B && [A \to 0A] \\
&\Rightarrow 0001B && [A \to \varepsilon] \\
&\Rightarrow 00011B && [B \to 1B] \\
&\Rightarrow 00011 && [B \to \varepsilon]
\end{aligned}
$$

$\therefore \quad S \overset{*}{\Rightarrow} 00011$

The parse tree is given by



3.7. RELATIONSHIP BETWEEN DERIVATIONS AND DERIVATION TREES

Given a grammar G = (V, T, P, S)

The following are equivalent with derivations and derivation trees or parse trees.

1. The recursive inference procedure determines that terminal string w is in the language of variable A.
2. $A \overset{*}{\Rightarrow} w$
3. $A \overset{*}{\underset{lm}{\Rightarrow}} w$
4. $A \overset{*}{\underset{rm}{\Rightarrow}} w$
5. There is a parse tree with root A and yield w.

The equivalent is shown using the diagram given below

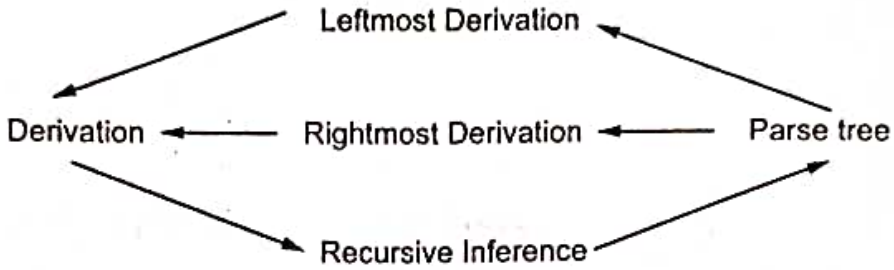


Fig. 3.2. Equivalence diagram

From this diagram, it is easy to prove two arcs, *i.e.,* derivation with leftmost and rightmost derivations.

Proof:

If w has a leftmost derivation from A, then it surely has a derivation from A. since a leftmost derivation is also a derivation. This is same for rightmost derivation.

3.7.1. FROM INFERENCE TO TREES

Theorem:

Let G = (V, T, P, S) be a CFG. If the recursive inference procedure tells that the terminal string w is in the language of variable A, then there is a parse tree with root A and yield w.

☺ ***Solution:***

To prove:

If $A \overset{*}{\Rightarrow} w$, there is a parse tree with root A.

Proof:

By induction hypothesis

Basis

Let the number of steps used be 1. Then the tree has a single leaf.

It may be either

$$A \rightarrow a_1, a_2, \ldots a_n \qquad [a_1, a_2, \ldots a_n \text{ - terminals}]$$

$$\text{(or)} \quad A \rightarrow \varepsilon$$

So it produces the yield $w = \varepsilon$ or $w = a_1, a_2, \ldots a_n$ then there is a parse tree with root A and yield w.

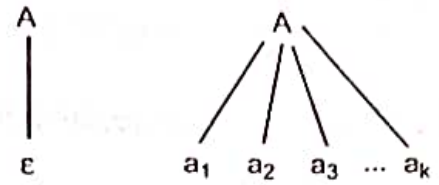


Fig. 3.3. Parse tree of one step derivation

Induction

- ✓ Let the number of steps used be n + 1.
- ✓ Assume that there is a parse tree for all the variables B and strings x inferred from B using n or less steps.
- ✓ Consider the string w inferred from a variable A after n + 1 steps, and also consider the last step of the inference that assures w is in the language of A.
- ✓ This inference uses some productions for A, say

$$A \rightarrow X_1, X_2, \ldots X_k$$

where each X_i is either a variable or a terminal. We have to break up the string w having k components as

$$w = w_1, w_2, \ldots w_k$$

Here we have two cases,

(*i*) If X_i is a terminal, then $w_i = x_i$

(*ii*) If X_i is a variable, then w_i is a string that was previously inferred to be in language L in atmost n steps.

(*a*) $x_i \Rightarrow w_i$ (*b*) $A \overset{*}{\Rightarrow} w_1, w_2, \ldots w_k$

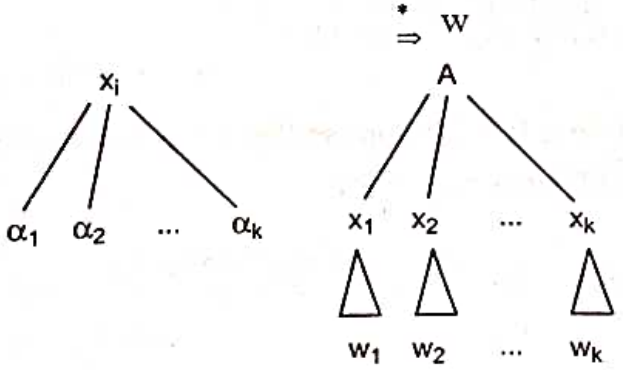


Fig. 3.4. Parse tree

Thus the yield is obtained by concatenating the sub-trees from left to right. Thus the theorem is proved.

Example 3.23 ***Consider the grammar with the productions.***

$E \rightarrow E + E$

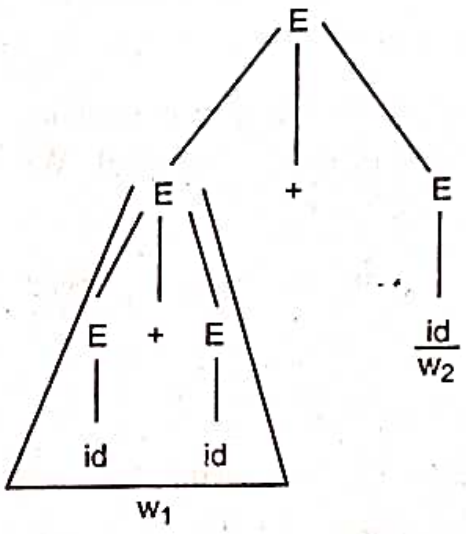
$E \rightarrow id$

Check whether the yield id + id + id is having the parse tree with root E or not.

☺***Solution:***

$$E \Rightarrow E + E$$
$$\Rightarrow E + E + E \qquad [\because E \rightarrow E + E]$$
$$\Rightarrow id + id + id \qquad [\because E \rightarrow id]$$
$$\therefore E \overset{*}{\Rightarrow} id + id + id$$

Parse tree for the yield is id + id + id.



$$E \overset{*}{\Rightarrow} id + id + id$$

3.7.2. FROM TREES TO DERIVATIONS

Theorem

Let G = (V, T, P, S) be a CFG. Suppose there is a parse tree with root A and with yield w, then there is a leftmost derivation.

$$A \overset{*}{\underset{lm}{\Rightarrow}} w \text{ in grammar G}$$

Proof

To prove: $A \overset{*}{\underset{lm}{\Rightarrow}} w$

Let us prove this theorem by induction on the height of the tree.

Basis

If the height of the parse tree is 1, then the tree must be of the form given in the figure below, with root A and yield w.

Fig. 3.5. Parse tree of height 1

This is possible only with the production

$$A \rightarrow w \text{ in } G$$

$$\therefore A \underset{lm}{\Rightarrow} w \text{ is a one step leftmost derivation}$$

Induction

If the height of the parse tree is n, the parse tree must look like Fig.3.6.

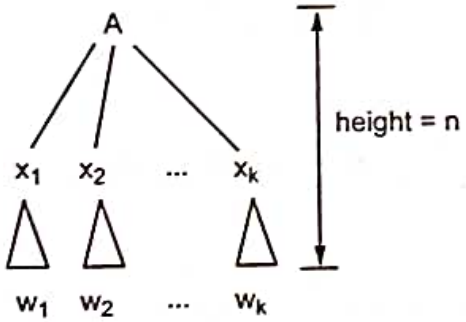


Fig. 3.6. Parse tree with height n

Assume that there exists a leftmost derivation $A \overset{*}{\Rightarrow} w$ for every parse tree of height less than n.

Consider a parse tree of height n.

Let the leftmost derivation be

$$A \underset{lm}{\overset{*}{\Rightarrow}} X_1 X_2 \ldots X_n$$

The x_i's may be either terminals or variables.

(*i*) If X_i is a terminal, then $X_i = w_i$

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

By applying the induction hypothesis, there is a leftmost derivation.

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

Note that $w = w_1 w_2 \ldots w_k$

Construct a leftmost derivation of w as follows

$$A \underset{lm}{\Rightarrow} X_1 X_2 \ldots X_k$$

For each $i = 1, 2, \ldots k$ in order, we show that

$$A \overset{*}{\underset{lm}{\Rightarrow}} w_1 w_2 \ldots w_i X_{i+1} X_{i+2} \ldots X_k$$

(*i*) If X_i is a terminal, then no need to change anything

$\therefore$ $$A \overset{*}{\underset{lm}{\Rightarrow}} w_1 w_2 \ldots w_i X_{i+1} X_{i+2} \ldots X_k$$

(*ii*) If X_i is a variable, then derive the string for each X_i to w_i

By our earlier statement

$$X_i \underset{lm}{\Rightarrow} \alpha_1 \underset{lm}{\Rightarrow} \alpha_2 \Rightarrow \ldots \Rightarrow w_i$$

Therefore

$$A \overset{*}{\underset{lm}{\Rightarrow}} w_1 w_2 \ldots w_{i-1} X_i X_{i+1} \ldots X_k$$
$$\overset{*}{\underset{lm}{\Rightarrow}} w_1 w_2 \ldots w_{i-1} \alpha_1 X_{i+1} \ldots X_k$$
$$\overset{*}{\underset{lm}{\Rightarrow}} w_1 w_2 \ldots w_{i-1} \alpha_2 X_{i+1} \ldots X_k$$
$$\overset{*}{\underset{lm}{\Rightarrow}} w_1 w_2 \ldots w_{i-1} w_i X_{i+1} \ldots X_k$$

By repeating the process, we can get

$$A \Rightarrow w_1 w_2 \ldots w_k$$

Thus the theorem is proved

This proof is similar for the conversion of parse tree to rightmost derivation.

Example 3.24 ***Consider the grammar with the production***

$E \rightarrow E * E$

$E \rightarrow id$

Prove that id * id * id is derived by E.

☺*Solution:*

$$\begin{aligned} E &\Rightarrow E * E \\ &\underset{lm}{\Rightarrow} E * E * E \\ &\underset{lm}{\Rightarrow} id * E * E \\ &\underset{lm}{\Rightarrow} id * id * E \\ &\underset{lm}{\Rightarrow} id * id * id \end{aligned}$$

$$\therefore \quad E \overset{*}{\underset{lm}{\Rightarrow}} id * id * id$$

3.7.3. FROM DERIVATIONS TO RECURSIVE INFERENCE

Theorem

Let G = (V, T, P, S) be a CFG. If there is $A \overset{*}{\Rightarrow} w$, then the recursive inference procedure applied to G determines that w is in the language of variable A.

Proof

The proof is an induction on the length of the derivation $A \overset{*}{\Rightarrow} w$.

Basis

If the derivation is one step, then

$A \rightarrow w$ must be a production

and $\therefore A \Rightarrow w$ where w is in the language of A

Induction

Suppose the derivation takes n + 1 steps, then the derivation should be like this,

$$\begin{aligned} A &\Rightarrow X_1 X_2 \ldots X_k \\ &\overset{*}{\Rightarrow} w \end{aligned}$$

Break w as $w_1 w_2 \ldots w_k$ where

(*i*) if X_i is a terminal, then $w_i = X_i$

(*ii*) if X_i is a variable, then $X_i \overset{*}{\Rightarrow} w_i$

$$\begin{aligned} \therefore \quad A &\Rightarrow X_1 X_2 \ldots X_k \\ &\overset{*}{\Rightarrow} w_1 X_2 \ldots X_k \\ &\overset{*}{\Rightarrow} w_1 w_2 \ldots X_k \\ &\vdots \\ &\overset{*}{\Rightarrow} w \end{aligned}$$

Thus the theorem is proved.

3.8. PUSHDOWN AUTOMATA

The pushdown automaton is essentially a finite automaton with control of both an input tape and a stack on which it can store a string of stack symbols.

With the help of a stack, the pushdown automaton can remember an infinite amount of information.

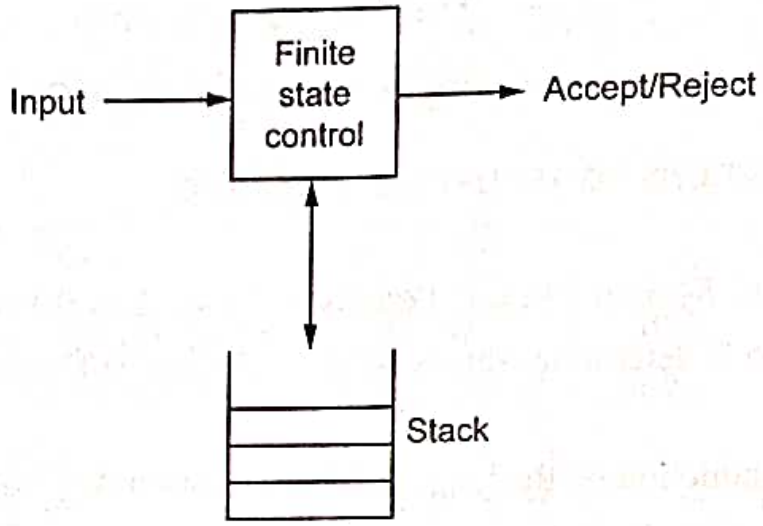


Fig. 3.7 A pushdown automaton

3.9. MODEL OF PDA

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

3.9.1 DEFINITION OF PDA

A pushdown automaton consists of seven tuples,

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

where

Q - a finite non-empty set of states

Σ - a finite set of input symbols

Γ - a finite non-empty set of stack symbols

q_0 - q_0 in Q is the start state

z_0 - initial start symbol of the stack

F - $F \subseteq Q$, set of accepting states or final states

δ - The transition function is given by

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$$

3.9.2 MOVES

The interpretation of

$$\delta (q, a, Z) = (p_1, \gamma_1), (p_2, \gamma_2) \ldots (p_M, \gamma_m)$$

where d, P_i - states

a - input symbol

Z - stack symbol

γ_i - a symbol in Γ^*

is that the PDA enters state P_i, replaces the symbol z by the string γ_i and advances the input head one symbol. The convention used here is that the leftmost symbol of γ_i will be placed highest on the stack and the rightmost symbol lowest on the stack.

The interpretation of

$$\delta (q, \varepsilon, z) = \{(p_1, \gamma_1), (p_2, \gamma_2) \ldots (p_m, \gamma_m)\}$$

is that independent of the input symbol being scanned, the PDA can enter state P_i and replaced Z by γ_i for any i, $1 \leq i \leq m$. Here the input head is not advanced.

3.9.3 INSTANTANEOUS DESCRIPTIONS (ID)

The ID must record the state and stack contents

If $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0)$ be a PDA

Then

$(q, aw, z\alpha) \overset{\vdash}{_M} (p, w, \beta\alpha)$ if $\delta(q, a, z) = (p, \beta)$

3.9.4 DESIGN

Example 3.25 *Construct a PDA that accepts $\{wcw^R \mid w \text{ in } (0+1)^*\}$ by an empty stack.*

☺ *Solution:*

- ✓ For each move, the PDA writes a symbol on the top of the stack.
- ✓ If the tape head reaches the input symbol c, stop pushing onto the stack.
- ✓ Compare the stack symbol with the input symbol, if it matches, pop the stack symbol.
- ✓ Repeat the process till reaches the final state or empty stack.

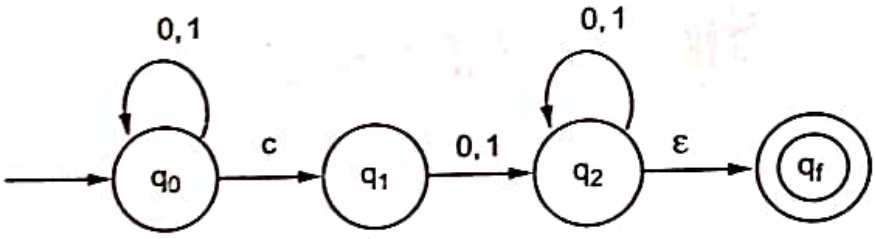


Fig. 3.8 wcw^R

From the diagram Fig.3.8 it is clear that q_0 is the start state, on reading the symbols 0's or 1's the PDA remains in the same state. On reading the input symbol c, the PDA goes to the state q_1. From q_1, the PDA goes to the state q_2 till reads the ε symbol. The productions are given below

1.	$\delta(q_0, 0, z_0) = \{(q_0, 0\, z_0)\}$	
2.	$\delta(q_0, 1, z_0) = \{(q_0, 1\, z_0)\}$	
3.	$\delta(q_0, 0, 0) = \{(q_0, 00)\}$	Push
4.	$\delta(q_0, 1, 0) = \{(q_0, 10)\}$	
5.	$\delta(q_0, 0, 1) = \{(q_0, 01)\}$	
6.	$\delta(q_0, 1, 1) = \{(q_0, 11)\}$	
7.	$\delta(q_0, C, 1) = \{(q_1, 1)\}$	Accept the separator C
8.	$\delta(q_0, C, 0) = \{(q_1, 0)\}$	

9.	$\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$	POP
10.	$\delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$	
11.	$\delta(q_1, \varepsilon, z_0) = \{(q_f, \varepsilon)\}$	

For all others, the input string is rejected.

Let 100C001 be the input string

$w = 100$ $\quad w^T = 001$

$\delta(q_0, 100C001, z_0) \vdash (q_0, 00C001, 1z_0)$

$\vdash (q_0, 0C001, 01\ z_0)$

$\vdash (q_0, C001, 001\ z_0)$

$\vdash (q_1, 001, 001\ z_0)$

$\vdash (q_1, 01, 01\ z_0)$

$\vdash (q_1, 1, 1\ z_0)$

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

$\vdash (q_1, \varepsilon)$ accepted

Let 110C10 be the another input string

$\delta(q_0, 110C10, z_0) \vdash (q_0, 10C10, 1z_0)$

$\vdash (q_0, 0C10, 11z_0)$

$\vdash (q_0, C10, 011z_0)$

$\vdash (q_0, 10, 011z_0)$

Rejected

This design is an example for language acceptance by empty stack.

3.9.5 THE LANGUAGE OF A PDA

A PDA can accept its input by consuming it and entering into an accepting state.

There are two ways of language acceptances

(*i*) Acceptance by Final state

(*ii*) Acceptance by Empty stack

Acceptance by Final state

Let M = (Q, Σ, Γ, δ, q_0, z_0, F) be a PDA, then the language accepted by a final state is,

$$L(M) = \{w \mid (q_0, w, z_0) \overset{*}{\vdash} (p, \varepsilon, \gamma) \text{ for some P in F and } \gamma \text{ in } \Gamma^*\}$$

Acceptance by Empty stack

Let M = (Q, Σ, Γ, δ, q_0, z_0, F) be a PDA, then the language accepted by a empty stack is

$$N(M) = \{w \mid (q_0, w, z_0) \overset{*}{\vdash} (p, \varepsilon, \varepsilon) \text{ for some P in Q}\}$$

Example 3.26 ***Construct a PDA for the language L = {ww^R \| w in (0 + 1)*}***

☺ ***Solution:***

Let P = ({q_1, q_2}, {0, 1}, {0, 1, z_0}, δ, q_0, z_0, {q_2}) be the PDA

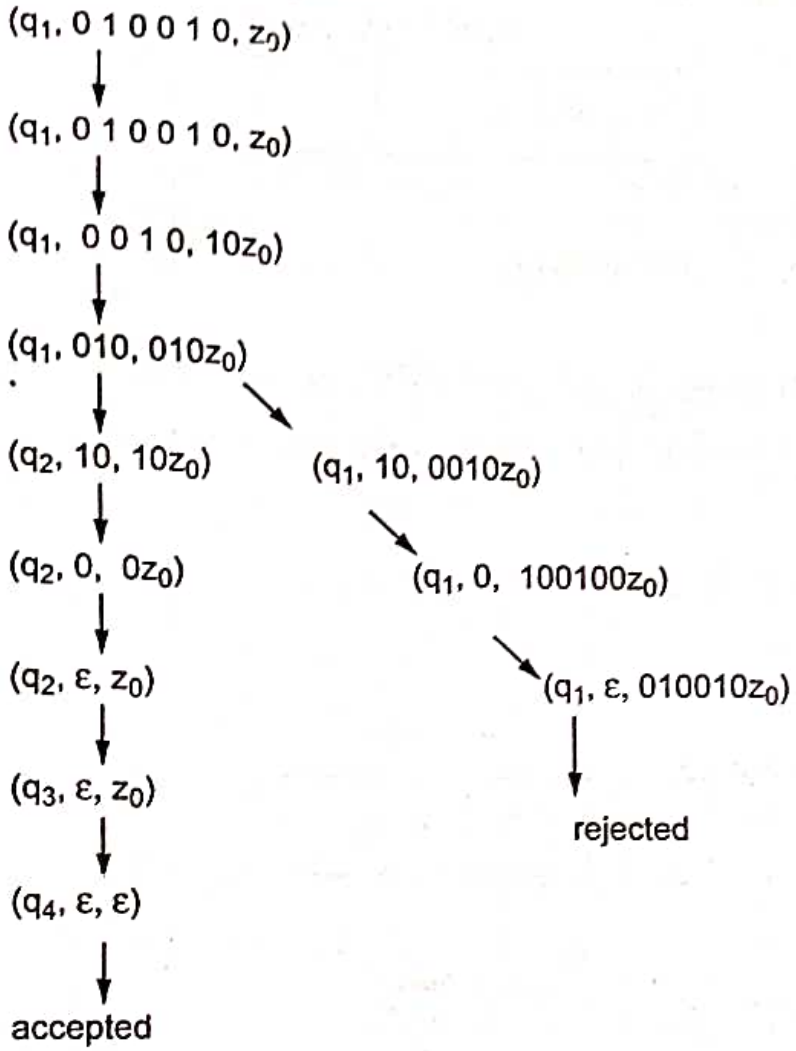
The productions are given below

1. $\delta(q_1, 0, z_0) = (q_1, 0z_0)$
2. $\delta(q_1, 1, z_0) = (q_1, 1z_0)$
3. $\delta(q_1, 0, 0) = \{(q_1, 00), (q_2, \varepsilon)\}$
4. $\delta(q_1, 0, 1) = (q_1, 01)$
5. $\delta(q_1, 1, 0) = (q_1, 10)$
6. $\delta(q_1, 1, 1) = \{(q_1, 11), (q_2, \varepsilon)\}$
7. $\delta(q_2, 0, 0) = (q_2, \varepsilon)$
8. $\delta(q_2, 1, 1) = (q_2, \varepsilon)$
9. $\delta(q_2, \varepsilon, z_0) = (q_3, z_0)$ [∵ Acceptance by final state]
10. $\delta(q_3, \varepsilon, z_0) = (q_4, \varepsilon)$ [∵ Acceptance by empty stack]

The rules (1) to (6) allow P to store the input on to the stack. In rules (3) and (6) there are two choices one to push the stack symbol and another to POP the stack symbol. P may decide that the middle of the input string has been reached and make the second choice. P goes to state q_2 and tries to match the remaining input symbols with the contents of the stack.

If P guessed right the input is of the form WW^R, then the inputs will match, M will empty its stack and then accept the input string. Assume that there will be no wrong guesses.

For example, take the input string 010010



The transition diagram for ww^R is given below

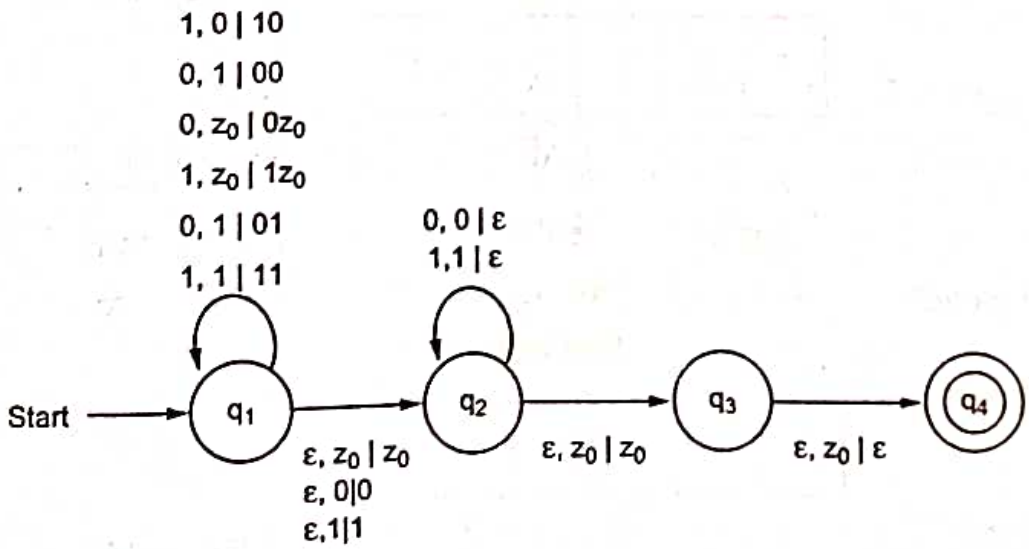
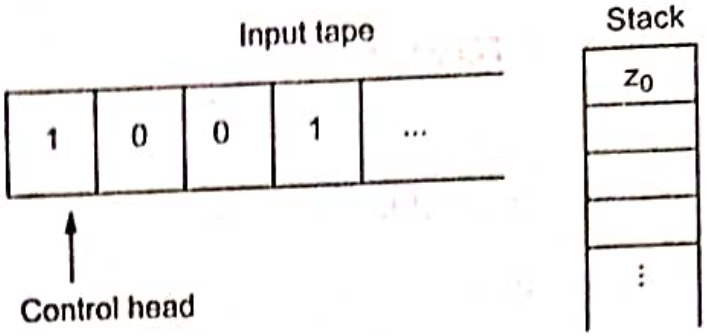


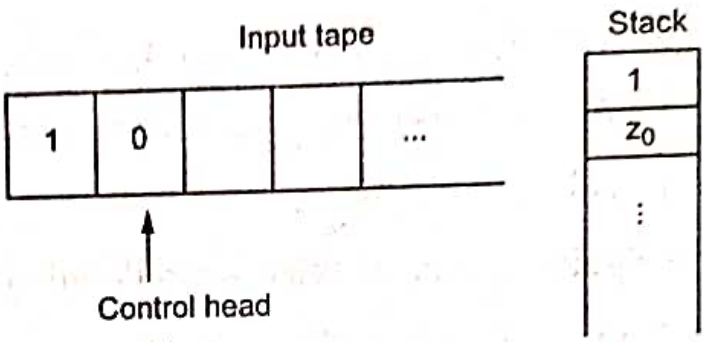
Fig.3.9. Transition diagram for ww^R

Let the input string be 1001.

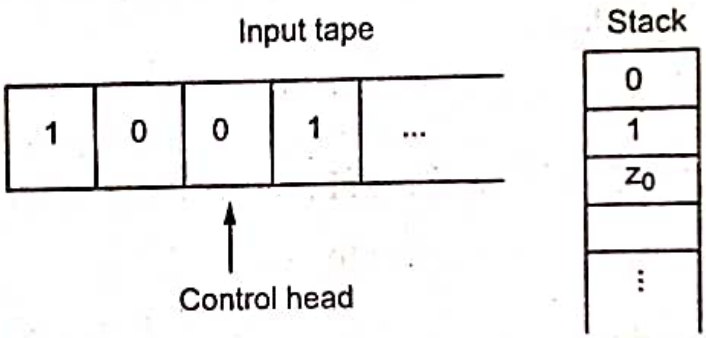
At start position



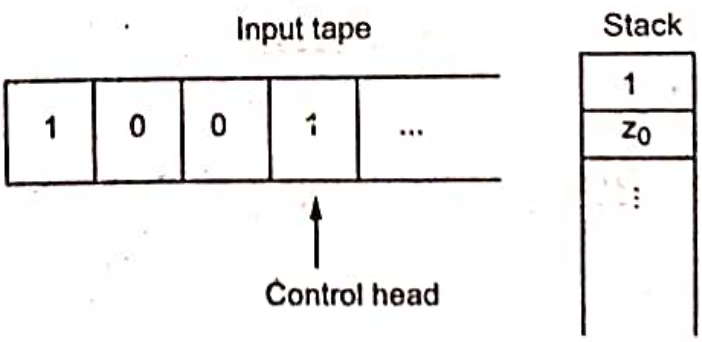
First move



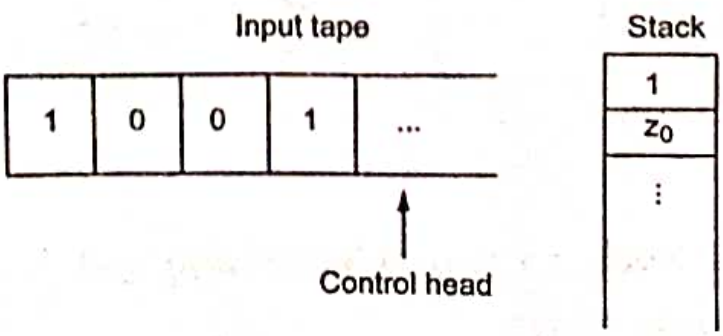
Second move



Third move



Fourth move



3.10 EQUIVALENCES OF ACCEPTANCE

3.10.1 FROM EMPTY STACK TO FINAL STATE

Theorem:3.1

If $L = N(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta, q_0 z_0, F)$ then there is a PDA P_F such that $L = L(P_F)$.

Proof:

The theorem states that if there is a PDA which has acceptance by empty stack, then there should be a PDA which also has acceptance by final state.

$P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, z_0, F)$ be a PDA

It is planned to have P_F simulate P_N and find when P_F empties its stack.

Let $P_F = (Q, \cup, \{q_0^1, q_f\}, \Sigma, \Gamma \cup \{x_0\}, \delta^1, q_0^1, x_0, \{q_f\})$

where δ^1 is given as follows

1. $\delta^1 (q_0^1, \varepsilon, x_0) = \{(q_0, z_0 X_0\}$
2. For all q in Q, a in $\Sigma \cup \{\varepsilon\}$ and Z in Γ, $\delta^1 (q, a, z) = \delta (q, a, z)$
3. For all q in Q, $\delta^1 (q, \varepsilon, x_0)$ contains (q_f, ε)

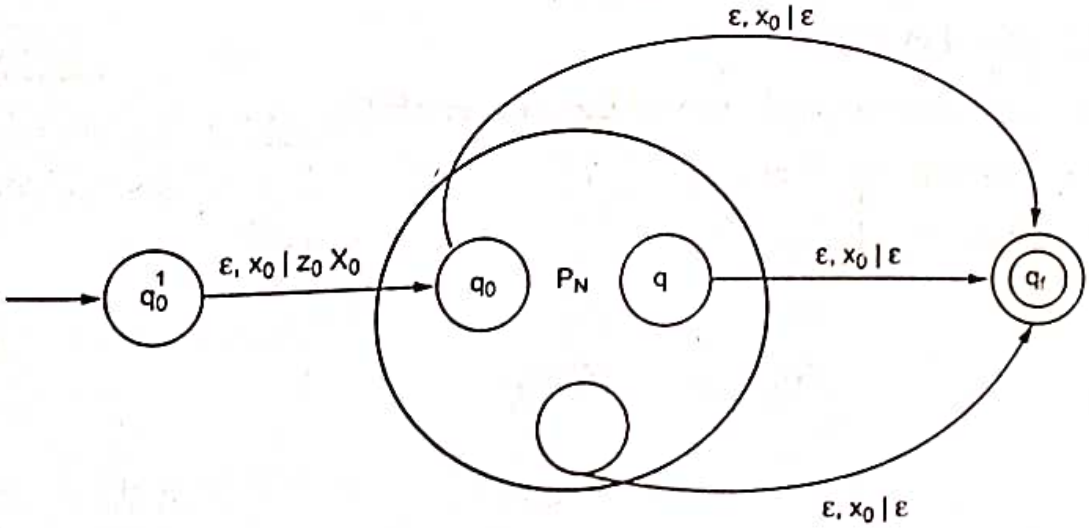


Fig 3.10. P_F simulates P_N and accepts if P_N empties its stack

To prove

w is in $L(P_F)$ if and only if w is in $N(P_N)$

If part

$$(q_0^1, w, x_0) \overset{}{\vdash}_{P_F} (q_0, w, z_0 x_0) \quad \text{[Rule 1]}$$

$$\overset{*}{\vdash} (q, \varepsilon, x_0) \quad \text{[Rule 2]}$$

$$\overset{}{\vdash}_{P_F} (q_f, \varepsilon, \varepsilon) \quad \text{[Rule 3]}$$

Since x_0 is at the bottom of the stack, conclude

$$(q_0^1, w, x_0) \overset{*}{\vdash} (q, \varepsilon, x_0) \quad \text{[Rule 2]}$$

This P_F accepts w by final state.

Only-if-Part

Rule (1) causes P_N to enter the initial ID of P_F except that P_F will have its own bottom of stack marker z_0, which is the symbol of P_F's stack. It is to prove that

$$(q_0, w, z_0) \overset{*}{\vdash}_{P_N} (q, \varepsilon, \varepsilon)$$

That is w is in $N(P_N)$.

Example:3.27 ***Design a PDA that process the if...else loops using the acceptance equivalence of empty stack to final state.***

☺ ***Solution:***

In a programming language the program structure should be having equal number of 'if' and 'else' keywords.

z is the stack symbol used to count the number of if's.

'if' is represented by 'i' and else by 'e'.

Let $P_N = (\{q\}, \{i, e\}, \{z\}, \delta, q, z)$

δ is given by $\quad \delta(q, i, z) = \{(q, zz)\}$

$$\delta(q, e, z) = \{(q, \varepsilon)\}$$

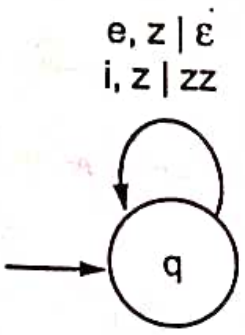


Fig.3.11. PDA accepts if-else by empty stack

If i is encountered, push it on to the stack, whenever e is encountered, POP, the 'i' on the stack till read the initial stack symbol z.

Stimulate $P_F = (\{(q^1, q_f, q), \{i, e\}, \{z, x_0\}, \delta^1, q^1, q_f\}$

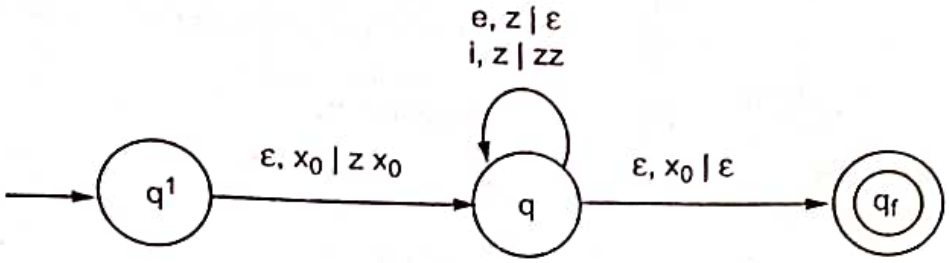


Fig.3.12. PDA accepting from empty stack to final stqte

The transition function δ^1 is given by

1. $\delta^1 (q^1, \varepsilon, x_0) = \{(q, z\, x_0)\}$
   Here x_0 is the bottom of stack marker
2. $\delta^1 (q, i, z) = \{(q, zz)\}$ P_F pushes z
3. $\delta^1 (q, e, z) = \{(q, \varepsilon)\}$ P_F POPs z
4. $\delta^1 (q^1, \varepsilon, x_0) = \{(q_f, \varepsilon, \varepsilon)\}$

P_F accepts the string by empting the stack.

3.10.2 FROM FINAL STATE TO EMPTY STACK

Theorem:3.2

Let L be the $L(P_F)$ for some PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, z_0, F)$ then there is a PDA P_N such that $L = N(P_N)$

Proof:

Let us simulate P_N from P_F as follows:

$$P_N = (Q, \cup, \{q_e, q_0^1\}, \Sigma, \Gamma \cup \{x_0\}, \delta^1, q_0^1, x_0)$$

where δ^1 is given as follows

(*i*) $\delta^1 (q_0^1, \varepsilon, x_0) = \{(q_0, z_0\, x_0)\}$

(*ii*) $\delta^1(q, a, z) = \delta (q, a, z)$ for all q in Q, a in Σ and z in Γ

(*iii*) $\delta^1(q, \varepsilon, z) = (q_e, \varepsilon)$ for all q in F and z in $\Gamma \cup \{x_0\}$

(*iv*) $\delta^1(q, \varepsilon, z) = (q_e, \varepsilon)$ for all z in $\Gamma \cup \{x_0\}$.

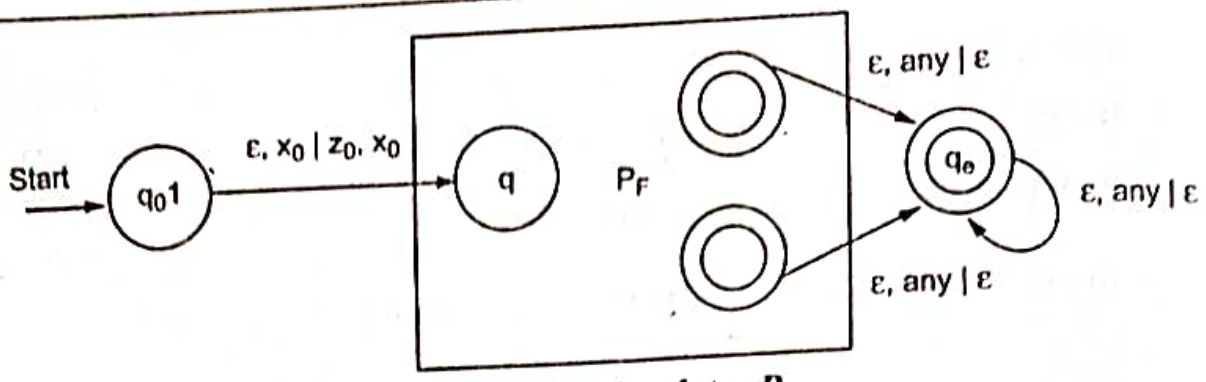


Fig.3.13. P_N simulates P_F

Let x ε $L(P_F)$. Then

$$(q_0, x, x_0) \overset{*}{\vdash}_{P_F} (q_0, x, z_0 x_0)$$

By rule (2) every move of P_N is a legal move of P_F.

$$(q_0, x, z_0) \overset{*}{\vdash}_{P_N} (q, \varepsilon, \gamma)$$

Since P_N simulates P_F and emptying its stack

$$\begin{aligned}(q_0, x, x_0) &\overset{*}{\vdash} (q_0, x, z_0 x_0)\\ &\overset{*}{\vdash} (q_i, \varepsilon, \gamma x_0)\\ &\overset{*}{\vdash} (q_e, \varepsilon) \qquad \text{[By rule (3) and (4)]}\\ \therefore \quad x \ \varepsilon \ &L(P_N)\end{aligned}$$

Next let us show that if x ε $L(P_F)$, then the derivation of P_N must be

$$\begin{aligned}(q_0^1, x, x_0) &\vdash (q_0, x, z_0 x_0)\\ &\overset{*}{\vdash}_{P_N} (q, \varepsilon, \gamma x_0)\\ &\overset{*}{\vdash}_{P_N} (q_e, \varepsilon)\end{aligned}$$

Since we are using x_0 at the bottom most of the stack, we get the derivation by removing x_0.

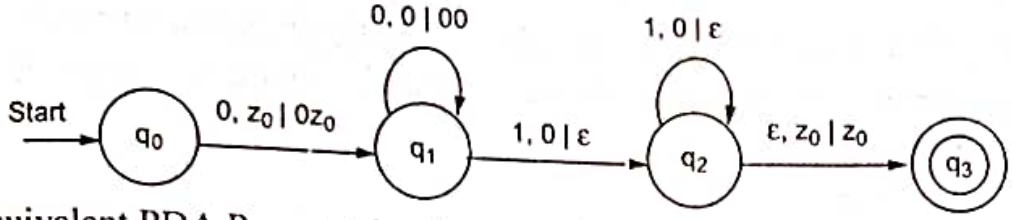
$$(q_0, x, z_0) \vdash (q_i, \varepsilon, \gamma)$$

and q_i is the final state *i.e.*, x ε $L(P_F)$.

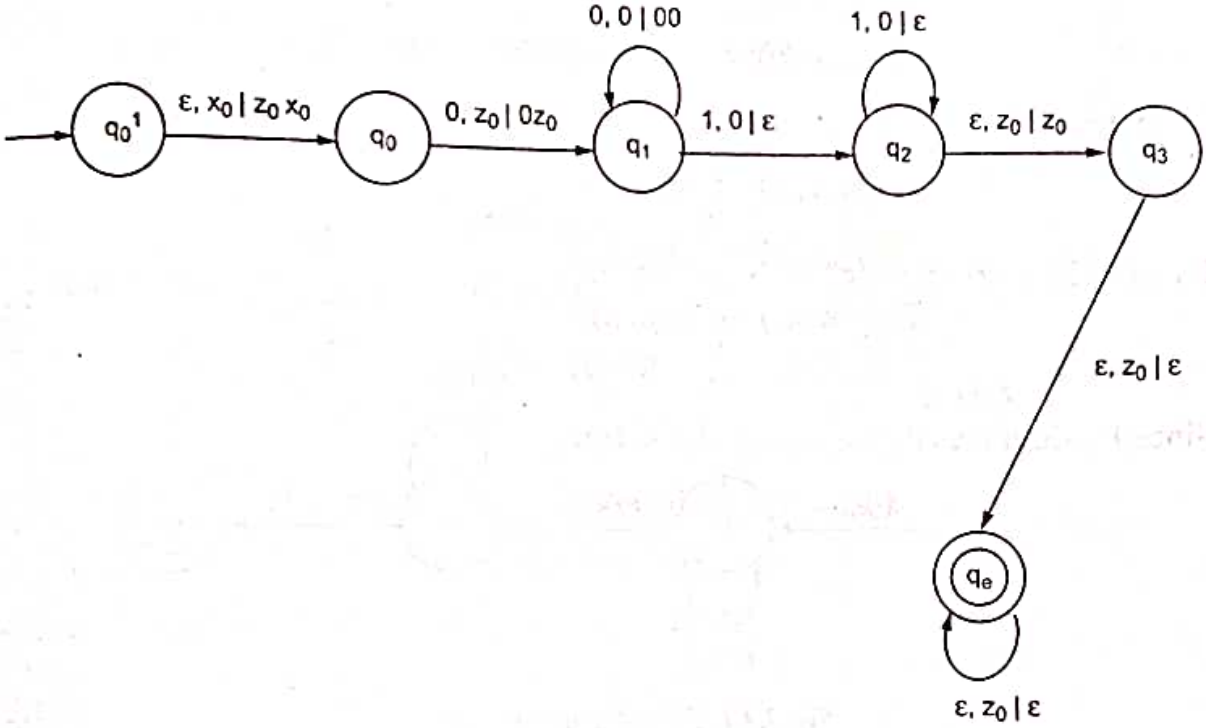
Example: 3.28 ***Construct a PDA P_N that accepts the language $L = \{0^n 1^n, n \geq 1\}$ by empty stack, convert it into P_F accepting by final state.***

☺ ***Solution:***

The PDA P_N can be constructed as follows



The equivalent PDA P_F accepting the given language L is



SOLVED PROBLEMS

Example:3.29 ***Construct a PDA accepting $\{a^n b^m a^n \mid m, n \geq 1\}$ by empty stack.***

☺***Solution:***

Let P be the PDA

$$P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, \{q_f\})$$

where

$$Q = (q_0, q_1, q_2, q_f)$$

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

$$\Gamma = \{a, z_0\}$$

and δ is given by

(*i*) Push the input symbols a^n onto the stack till reads the symbol b.

(*ii*) If b encounters, skip the steps till reads a.

(*iii*) If it reads a, read it and goto next state. From this, if the input symbol and the stack symbol matches, POP the stack symbol. Repeat the steps, till reading the initial stack symbol z_0.

$$\delta : (q_0, a, z_0) = \{(q_0, az_0)\}$$
$$\delta (q_0, a, a) = \{(q_0, aa)\}$$
$$\delta(q_0, b, a) = \{(q_1, a)\}$$
$$\delta(q_1, b, a) = \{(q_1, a)\}$$
$$\delta(q_1, a, a) = \{(q_2, \varepsilon)\}$$
$$\delta(q_2, a, a) = \{(q_2, \varepsilon)\}$$
$$\delta(q_2, \varepsilon, z_0) = \{(q_f, \varepsilon)\}$$

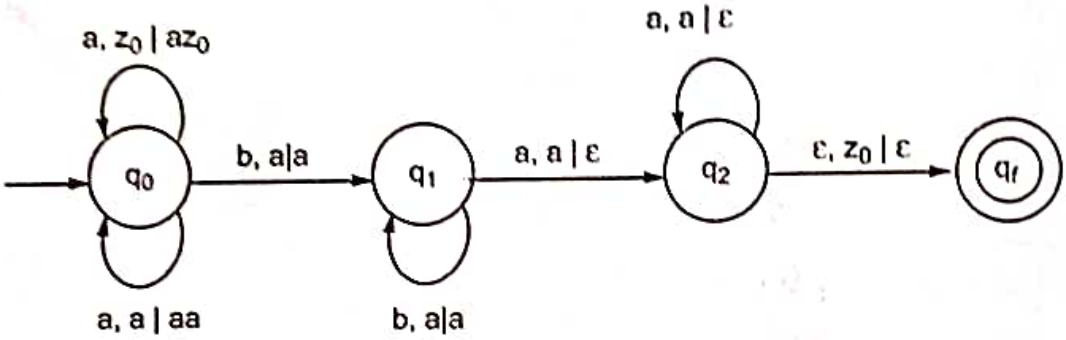


Fig.3.14. PDA for $a^n\ b^m\ a^n$

Example:3.30 ***Construct a PDA to accept the language $L = \{a^n\ b^n \mid n \geq 1\}$ by empty stack***

☺ ***Solution:***

PDA P = $(Q, \Sigma, \Gamma, \delta, q_0, z_0, \{q_f\})$

where

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

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

$\Gamma = \{a, z_0\}$

δ: $\delta(q_0, a, z_0) = \{(q_0, az_0)\}$

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

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

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

$\delta(q_1, \varepsilon, z_0) = \{(q_f, \varepsilon)\}$

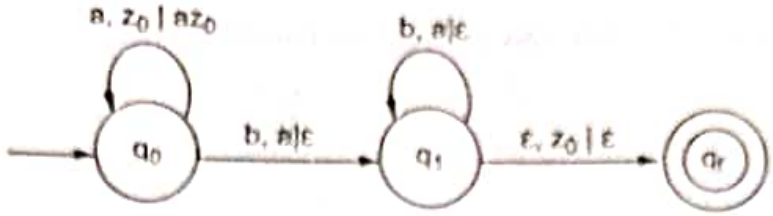


Fig.3.15. PDA for $a^n\ b^n$

Example:3.31 ***Construct a PDA to accept the language $L = \{a^n\ b^{2n} \mid n \geq 1\}$ by final state.***

☺ ***Solution:***

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

where

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

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

$$\Gamma = \{a, z_0\}$$

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

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

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

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

$$\delta(q_1, \varepsilon, z_0) = \{(q_f, z_0)\}$$

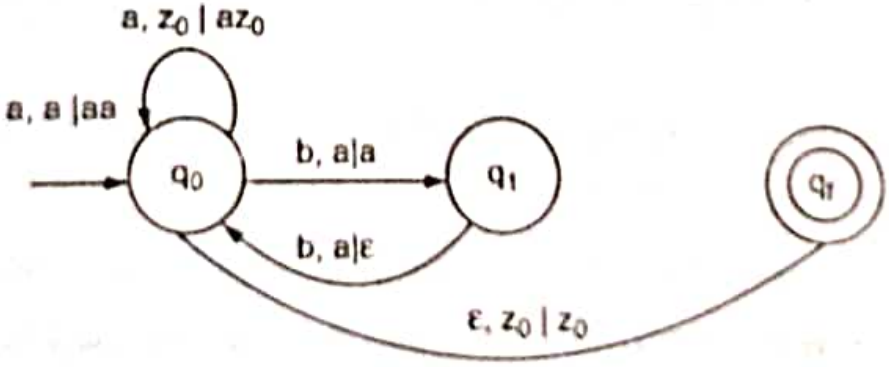


Fig.3.16. PDA for $a^n\ b^{2n}$

Example:3.32 ***Construct a PDA that accepts the language $L = \{a^n\ b^m\ c^m\ d^n \mid m, n \geq 1\}$ by empty stack.***

☺ ***Solution:***

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

where

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_f\}$$

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

$$\Gamma = \{a, b, z_0\}$$

$$\delta: \delta(q_0, a, z_0) = \{(q_1, az_0)\}$$

$$\delta(q_1, a, a) = \{(q_1, aa)\}$$

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

$$\delta(q_2, b, b) = \{(q_2, bb)\}$$

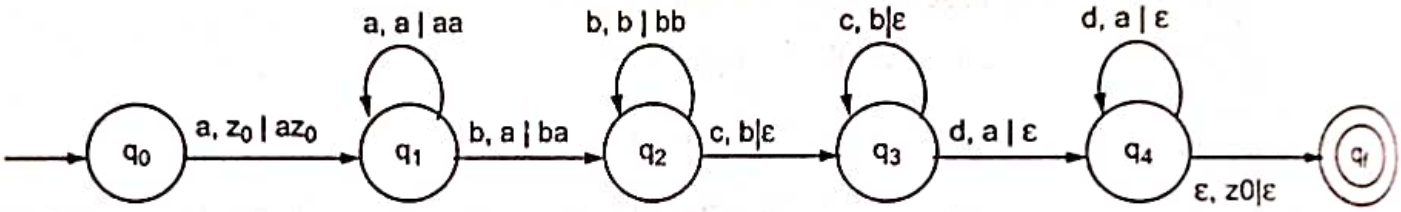
$$\delta(q_2, c, b) = \{(q_3, \varepsilon)\}$$

$$\delta(q_3, c, b) = \{(q_3, \varepsilon)\}$$

$$\delta(q_3, d, a) = \{(q_4, \varepsilon)\}$$

$$\delta(q_4, d, a) = \{(q_4, \varepsilon)\}$$

$$\delta(q_4, \varepsilon, z_0) = \{(q_f, \varepsilon)\}$$



3.11 DETERMINISTIC PUSHDOWN AUTOMATA (DPDA)

Definition

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ is deterministic if and only if it satisfies the following conditions.

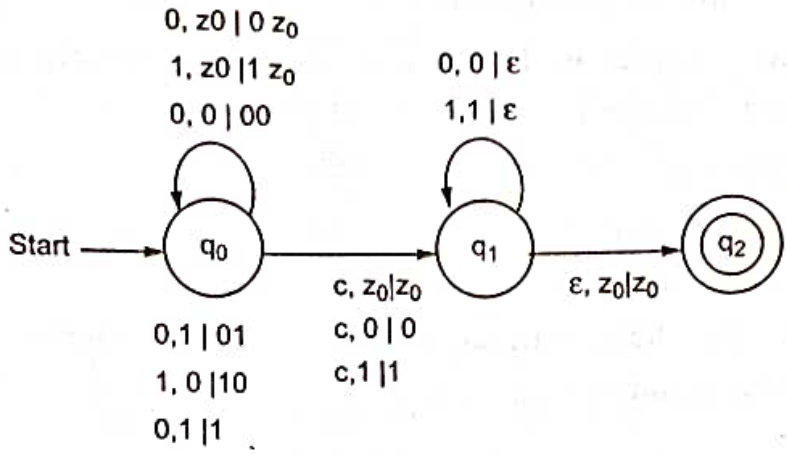
1. $\delta(q, a, x)$ has only one member for any q in Q, a in Σ or $a = \varepsilon$ and x in Γ.
2. If $\delta(q, a, x)$ is nonempty, for some a in Σ, then $\delta(q, \varepsilon, x)$ must be empty.

Example

Let $L = \{wcw^R \mid w \text{ is in } (0 + 1)^*\}$

The PDA for this is designed in such a way that DPDA is to store 0's and 1's on it stack until it sees the middle end marker. After this, it goes to another state in which it matches input symbols against stack symbols and pops the stack if they match, or else rejected. Thus the PDA is strictly deterministic.

It does not have a choice of move in the start state using the same input and stack symbol.



3.11.1 REGULAR LANGUAGES AND DETERMINISTIC PDA'S

Theorem:3.3

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

Proof:

- ✓ A DPDA can simulate a deterministic finite automaton.
- ✓ The PDA keeps some symbols z_0 on its stack, because a PDA has to have a stack.

Let A = (Q, Σ, δ_A, q_0, F) be a DFA

Construct DPDA P = {Q, Σ, {z_0}, δ_p, q_0, z_0, F}

where δ_p (q, a, z_0) = (p, z_0) for all states p and q in Q

such that δ_A (q, a) = p

We can claim that

$$(q_0, w, z_0) \overset{*}{\vdash}_p (p, \varepsilon, z_0)$$

if and only if $\hat{\delta}$ (q_0, w) = P

That is P simulates A using its state.

Since both A and P accept by entering one of the states of F, conclude that their languages are the same.

3.11.2 DPDA'S AND CONTEXT-FREE LANGUAGES

DPDA can accept context free languages L(P). The stack can store the necessary input symbols and use the state to remember the conditions.

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

DPDA and Ambiguity

- ✓ The languages that DPDA's accept by reaching its final state have unambiguous grammar.
- ✓ And also for those languages that accept by empty stack have also unambiguous grammar.

PROBLEMS

Example: 3.33 ***Construct a deterministic PDA that accepts*** $L = \{0^n 1^m;$ $n < m$ ***and*** $n, m \geq 1\}$

☺ ***Solution:***

The PDA has to be designed in such a way that

- ✓ It consumes all 0's at the initial stage q_0 and store the value in the stack.
- ✓ At q_0 if it encounters 1, pop the corresponding 0 on the stack and goto q_1 state. Repeat the process at q_1 till no more 0's inside the stack.
- ✓ At q_1, if the stack has only the initial symbol z_0, it goes to the state q_2.
- ✓ At q_2, it processes the remaining 1's without doing anything on the stack.
- ✓ At q_2, if it reaches the blank symbol B, then the PDA must goto the final state q_3.

The PDA P is given by

$$P = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$$

where

$Q = \{q_0, q_1, q_2, q_3\}$

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

$\Gamma = \{z_0, 0\}$

q_0 - initial state

q_2 - final state

δ is give n by

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

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

$$\delta(q_0, 1, 0) = \{(q_1, \varepsilon)\}$$

$$\delta(q_1, 1, 0) = \{(q_1, \varepsilon)\}$$

$$\delta(q_1, 1, z_0) = \{(q_2, z_0)\}$$

$$\delta(q_2, 1, z_0) = \{(q_2, z_0)\}$$

$$\delta(q_2, \varepsilon, z_0) = \{(q_3, \varepsilon)\}$$

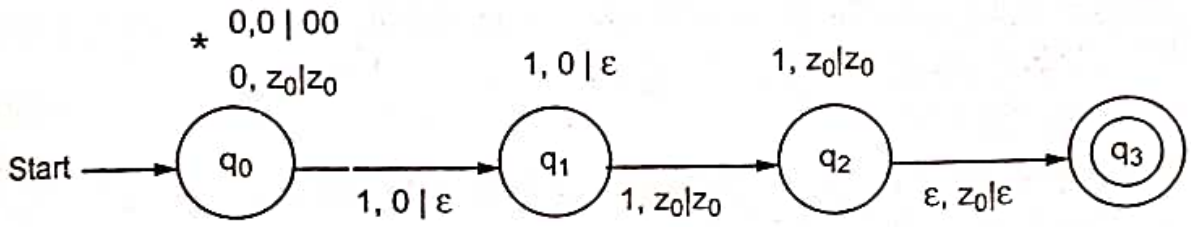


Fig.3.17. DPDA of $0^n\ 1^m$

3.12 EQUIVALENCE OF PDA'S AND CFL

The three classes of languages

1. Context-free languages
2. Languages accepted by final state of PDA
3. Languages accepted by empty stack of PDA

are under the same class.

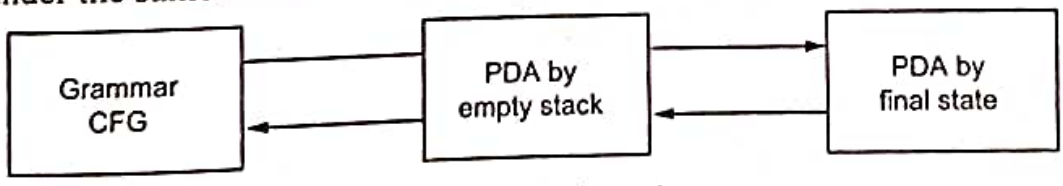


Fig.3.18. Equivalent classes

3.12.1 FROM CFG TO PDA

Let G = (V, T, Q, S) be a CFG.

Construct the PDA P that accepts L(G) by empty stack as follows.

$$P = (\{q\}, T, V \cup T, \delta, q, S)$$

where δ is defined by

1. For each variable A,

$$\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \to \beta \text{ is a production in G}\}$$

2. For every terminal symbol a,

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

Theorem:3.4 ***[CFG to PDA]***

If PDA P is constructed from CFG G, then N(P) = L(G)

☺ ***Solution:***

If part

Suppose w Σ L(G). Then w has the leftmost derivation

$$S = \gamma_1 \underset{lm}{\Rightarrow} \gamma_2 \underset{lm}{\Rightarrow} \gamma_3 \underset{lm}{\Rightarrow} \cdots \underset{lm}{\Rightarrow} \gamma_n = w$$

where $\gamma_i = x_i a_i$

where x_i is a terminal, α_i is a string of variables and terminals.

Prove this by induction on i that

$$(q, w, S) \overset{*}{\underset{P}{\vdash}} (q, y_i, \alpha_i)$$

Basis:

$$\text{if } i = 1, \gamma_1 = S, \text{ then}$$

$$x_1 = \varepsilon$$

$$y_1 = w$$

$$(q, w, S) \overset{*}{\underset{0_{move}}{\vdash}} (q, w, s)$$

Thus basis is proved.

Induction:

If $i > n + 1$

Assume

$$(q, w, S) \vdash^* (q, y_i, \alpha_i)$$

$y_i \Rightarrow y_{i+1}$ involves replacing A by one of its production bodies say β.

From the rule (1) it is clear that the PDA POPs, the terminal symbols and replaces the variables by one of its productions in the stack. And rule (2) allows us to match any terminals on the top of the stack with the next input symbols

i.e., $\delta(q, a, a) = \{(q, \varepsilon\}$.

$\therefore$ We get the ID $(q, y_{i+1}, \alpha_{i+1})$ which gives the value for γ_{i+1}.

Only-if part

Let $w \varepsilon N(P)$. This implies

If $(q, x, A) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon)$ then $\overset{*}{\underset{G}{\Rightarrow}} x$

This is proved by induction on the number of moves of P.

Basis:

If it happens in one move, then there should be $A \rightarrow \varepsilon$ and $w = \varepsilon$, so we can conclude $A \overset{*}{\underset{G}{\Rightarrow}} w$.

Induction:

Assume P takes n moves.

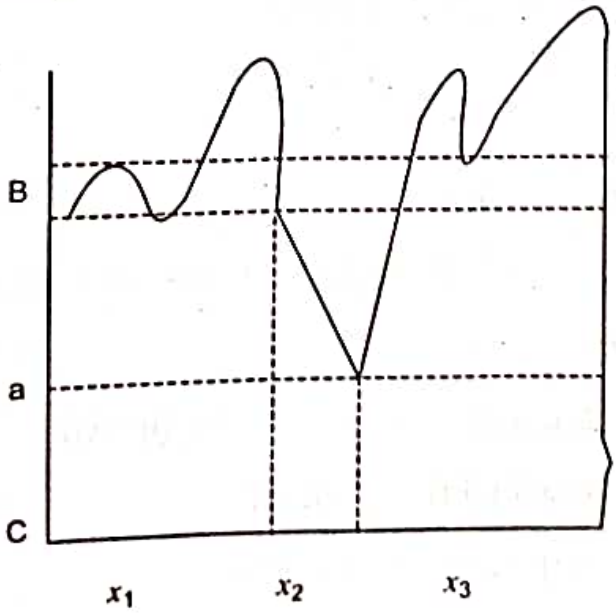


Fig.3.19. PDA P consumes x and POPs BaC from its stack

In the first move, the PDA replaces the variable A by one of its productions $A \to y_1 y_2 \dots y_k$ in G.

In the next n – 1 moves, the PDA consumes x by popping each of $y_1 y_2 \dots$ from the stack one by one.

Let $x = x_1 x_2 \dots x_k$ such that x_i is the portion of the string consumed by P while popping y_i.

$$(q, x_i x_{i+1}, \dots x_k, y_i) \overset{*}{\vdash} (q, x_{i+1}, \dots x_k, \varepsilon)$$

Since the number of moves required to do it is less than n, we have

$$y_i \overset{*}{\underset{G}{\Rightarrow}} x_i$$

If y_i is a terminal, then $y_i = x_i$

$$A = y_1 y_2 \dots y_k \overset{*}{\Rightarrow} x_1 y_2 \dots y_k \overset{*}{\Rightarrow} x_1 x_2 \dots x_k$$

$$\therefore \quad A \overset{*}{\underset{G}{\Rightarrow}} w$$

$$\therefore \quad N(P) \subseteq L(G)$$

Thus the theorem is proved.

Example:3.34 ***Convert the grammar CFG to a PDA***

$E \to E + E$

$E \to id$

☺ ***Solution:***

The equivalent PDA is given by

$$P = (\{q\}, \{+, id\}, \{E + id\}, \delta, q, E)$$

where δ is defined by

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

$$\delta(q, id, id) = \{(q, \varepsilon)\}$$

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

Test whether the input id + id + id is in N(P)

$$w = id + id + id$$

$(q, id + id + id, E) \vdash (q, id + id + id, E + E)$

$\vdash (q, id + id + id, id + E)$

$\vdash (q, + id + id, + E)$

$\vdash (q, id + id, E)$

$\vdash (q, id + id, E + E)$

$\vdash (q, id + id, id + id)$

$\vdash (q, + id, + id)$

$\vdash (q, id, id)$

$\vdash (q, \varepsilon, \varepsilon)$

$\therefore$ id + id + id ε N(P).

Example:3.35 ***Convert the grammar***

$S \rightarrow 0S1 \mid A$

$A \rightarrow 1A0 \mid S \mid \varepsilon$

into a PDA that accepts the same language by empty stack. Check whether 0101 belongs to N(P).

☺ ***Solution:***

$P = (\{q\}, \{0, 1\}, \{S, A, 0, 1\}, \delta, q, S)$

where δ is defined by

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

$\delta(q, \varepsilon, A) = \{(q, 1A0), (q, S), (q, \varepsilon)\}$

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

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

$w = 0101$

$(q, 0101, S) \vdash (q, 0101, 0S1)$

$\vdash (q, 101, S1)$

$\vdash (q, 101, 1A01)$

$\vdash (q, 01, A01)$

$\vdash (q, 01, 01)$

$\vdash (q, 1, 1)$

$\vdash (q, \varepsilon, \varepsilon)$

$\therefore 0101 \varepsilon N(P)$

Example:3.36 ***Construct a PDA equivalent to the CFG***

$S \rightarrow 0BB$

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

☺ ***Solution:***

The PDA is given by

$$P = (\{q\}, \{0, 1\}, \{S, B, 0, 1\}\ \delta, q, S)$$

where δ is defined by

$$\delta(q, \varepsilon, S) = \{(q, 0BB)\}$$

$$\delta(q, \varepsilon, B) = \{(q, 0S), (q, 1S), (q, 0)\}$$

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

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

3.12.2 FROM PDA'S TO GRAMMARS

Theorem:3.5

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0)$ be a PDA. Then there is a context free grammar G such that $L(G) = N(M)$.

Proof:

Let M be the PDA and $G = (V, \Sigma, P, S)$ be a context free grammar, where

(*i*) S is the start symbol

(*ii*) V is the set of objects of the form [q, A, p]

where q and p in Q,

A in Γ

p is the set of productions which are defined as follows:

(1) $S \rightarrow [q_0, z_0, q]$ for each q in Q

(2) $[q, A, q_{M+1}] \rightarrow a[q_1, B_1, q_2]\,[q_2, B_2, q_3] \ldots [q_M, B_M, q_{M+1}]$ for each a in $\Sigma \cup \varepsilon$

$q_1, q_2, \ldots q_{M+1}$ in Q

$A, B_1, B_2 \ldots B_M$ in Γ

such that

$$\delta(q, a, A) = (q_1, B_1\, B_2 \ldots B_M)$$

(2.1) if M = 0, *i.e.*, $\delta(q, a, A) = (q_1, \varepsilon)$ then

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

The variables and productions of G have been defined in such a way that a leftmost derivation in G of a sentence x is a simulation of the PDA M when fed the input x.

The variables that appear in any step of a leftmost derivation in G correspond to the symbol on the stack of M.

The intention is that [q, A, p] derive x if and only if x causes M to erase an A from its stack by some sequence of moves beginning in state q and ending in state p.

To show that L(G) = N(M)

By induction on the number of steps in a derivation that

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

if and only if $(q, x, A) \overset{*}{\underset{M}{\vdash}} (p, \varepsilon, \varepsilon)$ ... (1)

If part

Basis

If i = 1, then

$$\delta(q, x, A) = (p, \varepsilon) \text{ where } x = \varepsilon \text{ or single input symbol}$$

which produces

$$[q, A, p] \rightarrow x \text{ is a production of G.}$$

Induction

If $i > 1$ and $x = ay$, then

$$(q, ay, A) \vdash (q_1, y_1, B_1 B_2 \ldots B_n)$$
$$\vdash (q_1, y_1, y_2, \ldots y_n, B_1 B_2 \ldots B_n)$$
$$\vdash (q_2, y_2, \ldots y_n, B_1 B_2 \ldots B_n)$$
$$\vdash$$
$$\vdots$$
$$\overset{i-1}{\vdash} (p, \varepsilon, \varepsilon)$$

Here $y = y_1 y_2 \ldots y_n$, $A \rightarrow B_1 B_2 \ldots B_n$

The input symbol y_j has the effect of popping B_j from the stack after a long sequence of moves. In general B_j remains on the stack unchanged while $y_1 y_2 \ldots y_{j-1}$ is read.

And also $Q = \{q_1, q_1, \ldots q_{n+1}\}$ where $q_{n+1} = P$

$$\therefore (q_j, y_j, B_j) \overset{*}{\vdash} (q_{j+1}, \varepsilon, \varepsilon)$$

Thus it produces the production

$$(q_j, B_j, q_{j+1}) \overset{*}{\Rightarrow} y_j \text{ for } 1 \leq j \leq n$$

The popping of B_j is shown in the Fig. given below

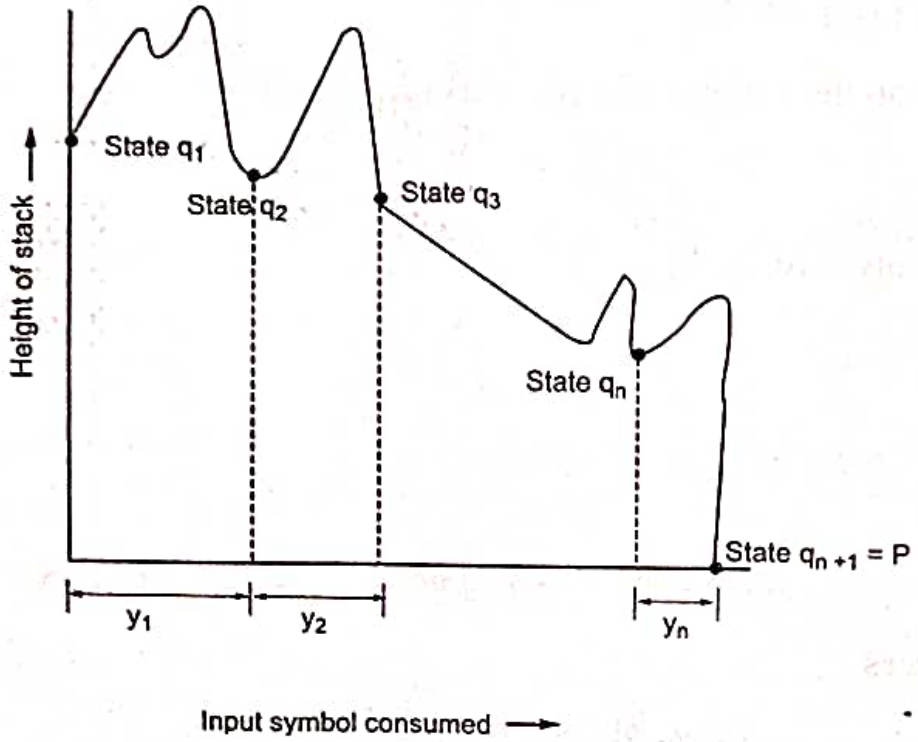


Fig.3.20. Height of stack as a function of input

If $(q, ay, A) \vdash (q_1, y, B_1 B_2 \ldots B_n)$ then

$[q, A, p] \quad a\,[q_1, B, q_2]\,[q_2, B_2, q_3] \ldots [q_M, B_n, q_{n+1}]$

$\therefore \quad [q, A, p] \Rightarrow a\, y_1\, y_2 \ldots y_n = x$

Only-if part

Suppose $[q, A, p] \overset{i}{\Rightarrow} x$

This is proved by induction on i, the number of steps.

Basis:

If i = 1, then

$$[q, x, A] \overset{*}{\vdash} (p, \varepsilon, \varepsilon)$$

Because $[q, A, p] \rightarrow x$ must be a production of G.

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

Induction:

If i > n, then

$$(q_j, x_j, B_j) \overset{*}{\vdash} (q_{j+1}, \varepsilon, \varepsilon) \text{ for } 1 \le j \le n$$

If we insert $B_{j+1} \ldots B_n$ at the bottom of each stack in the above sequence of ID's

$$(q_j, x_j, B_j B_{j+1} \ldots B_n) \overset{*}{\vdash} (q_{j+1}, \varepsilon, B_{j+1}, \ldots B_n) \qquad \ldots (2)$$

From [q, x, A] derivation, we know that

$$(q, x, A) \overset{*}{\vdash} (q_1, x_1 x_2 \ldots x_n, B_1 B_2 \ldots B_n)$$

is a legal move of M,

$\therefore \qquad (q, x, A) \overset{*}{\vdash} (p, \varepsilon, \varepsilon)$ for j = 1, 2, ... n

The proof concludes with the observation that

$$[q, A, p] \overset{*}{\underset{G}{\Rightarrow}} x \text{ with } q = q_0 \text{ and } A = z_0$$

$$[q_0, z_0, p] \overset{*}{\Rightarrow} x \text{ if and only if}$$

$$(q_0, x, z_0) \overset{*}{\vdash} (p, \varepsilon, \varepsilon)$$

$\therefore S \overset{*}{\Rightarrow} x$ if and only if

$$(q_0, x, z_0) \overset{*}{\vdash} (p, \varepsilon, \varepsilon) \text{ for some state in P}$$

That is x is in L(G) if and only if x is in N(M).

Example:3.37 *Let M = ({q}, {i, e}, {z}, δ_N, q, z) Where δ_N s given by δ_N (q, i, z) = {(q, zz)} δN (q, e, z) (q, ε)*

Solution:

Construct N(M)

where

$$V = \{S, [q, z, q]\}$$

$$T = \{i, e\}$$

The production for S

$$S \rightarrow [q, z, q]$$

δ_N (q, i, z) = {(q, zz)}

$$[q, z, q] \rightarrow i[q, z, q][q, z, q]$$

δ_N (q, e, z) = {q, ε}

$$[q, z, q] \rightarrow e$$

All the productions for all the variables have been found out. Eliminate the variables which do not have any derivations:

$\therefore$

$$S \rightarrow [q, z, q]$$

$$[q, z, q] \rightarrow i\,[q, z, q]\,[q, z, q] \mid e$$

Replace [q, z, q] by a symbol E

$\therefore$

$$S \rightarrow E$$

$$E \rightarrow iEE \mid e$$

The CFG is given by

$$G = (\{S, E\}, \{i, e\}, \{S \rightarrow iEE|e, E)$$

Example:3.38 *Let $M = (\{q_0, q_1\}, \{0, 1\}, \{x, z_0\}, \delta, q_0, z_0, \phi)$ where δ is given by*

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

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

$\delta(q_0, 0, x) = \{(q_0, xx)\}$

$\delta(q_1, \varepsilon, x) = \{(q_1, \varepsilon)\}$

$\delta(q_0, 1, x) = \{(q_1, \varepsilon)\}$

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

Construct a CFG for the PDA M.

☺ ***Solution:***

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

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

Production for S

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

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

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

Here in the (i) function, q0=q0, a=0, A=z0 in the LHS and q1=q0,B_1 =x, B_2=z0 in the RHS.

(*i*) $\delta(q_0, 0, z_0) = (q_0, x\ z_0)$

For q_0

$$[q_0, z_0, q_0] \rightarrow 0\ [q_0, x, q_0]\ [q_0, z_0, q_0]$$

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

For q_1

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

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

(*ii*) $\delta(q_0, 0, x) = \{(q_0, xx)\}$

$$[q_0, x, q_0] \rightarrow 0\ [q_0, x, q_0]\ [q_0, x, q_0] \mid 0\ [q_0, x, q_1]\ [q_1, x, q_0]$$

$$[q_0, x, q_1] \rightarrow 0\ [q_0, x, q_0]\ [q_0, x, q_1] \mid 0\ [q_0, x, q_1]\ [q_1, x, q_1]$$

(*iii*) $\delta(q_0, 1, x) = \{(q_1, \varepsilon)\}$

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

(*iv*) $\delta(q_1, 1, x) = \{(q_1, \varepsilon)\}$

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

(*v*) $\delta(q_1, \varepsilon, x) = \{(q_1, \varepsilon)\}$

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

(*vi*) $\delta(q_1, \varepsilon, z_0) = \{(q_1, \varepsilon)\}$

$$(q_1, z_0, q_1) \rightarrow \varepsilon$$

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

$$[q_1, x, q_0], \quad [q_1, z_0, q_0]$$

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

Deleting all the productions which involve these variables, the following productions were left.

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

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

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

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

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

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

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

The resultant productions are as follows

$\Rightarrow$

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

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

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

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

$$[q_1, x, q_1] \rightarrow \{\varepsilon, 1\}$$

Example:3.39 *Convert the PDA $P = \{(p, q), \{0, 1\}, \{x, z_0\}, \delta, q, z_0\}$ to a CFG, if δ is given by*

$\delta(q, 1, z_0) = \{(q, x\, z_0)\}$ $\qquad$ $\delta(q, \varepsilon, z_0) = \{(q, \varepsilon)\}$

$\delta(q, 1, x) = \{(q, xx)\}$ $\qquad$ $\delta(p, 1, x) = \{(p, \varepsilon)\}$

$\delta(q, 0, x) = \{(p, x)\}$ $\qquad$ $\delta(p, 0, z_0) = \{(q, z_0)\}$

☺ ***Solution:***

The variables involved in this PDA are

$$V = \{S, [q, x, q], [q, x, p], [p, x, q], [p, x, p] \; [q, z_0, q], [q, z_0, p], [p, z_0, q] \; [p, z_0, p]\}$$

The productions are

$$S \rightarrow [q, z_0, q]$$

$$S \rightarrow [q, z_0, p]$$

(*i*) $\delta(q, 1, z_0) = \{(q, x\, z_0)\}$

$$[q, z_0, q] \rightarrow 1\,[q, x, q]\,[q, z_0, q] \mid 1\,[q, x, p]\,[q, z_0, q]$$

$$[q, z_0, p] \rightarrow 1\,[q, x, q]\,[q, z_0, p] \mid 1\,[q, x, p]\,[q, z_0, p]$$

(*ii*) $\delta(q, 1, x) = \{(q, xx)\}$

$$[q, x, q] \rightarrow 1\,[q, x, q]\,[q, x, q] \mid 1\,[q, x, p]\,[p, x, q]$$

$$[q, x, p] \rightarrow 1\,[q, x, q]\,[q, x, p] \mid 1\,[q, x, p]\,[p, x, p]$$

(*iii*) $\delta(q, 0, x) = \{(p, x)\}$

$$[q, x, q] \rightarrow 0\,[p, x, q]$$

$$[q, x, p] \rightarrow 0\,[p, x, p]$$

(*iv*) $\delta(q, \varepsilon, z_0) = \{(q, \varepsilon)\}$

$$[q, z_0, q] \rightarrow \varepsilon$$

(*v*) $\delta(p, 1, x) = \{(p, \varepsilon)\}$

$$[p, x, p] \rightarrow 1$$

(*vi*) $\delta(p, 0, z_0) = \{(q, z_0)\}$

$$[p, z_0, q] \rightarrow 0\,[q, z_0, q]$$

$$[p, z_0, p] \rightarrow 0\,[q, z_0, p]$$

After eliminating the unwanted variables,

The CFG is given by

$$S \rightarrow [q, z_0, q]$$
$$[q, z_0, q] \rightarrow \{1\ [q, x, p]\ [p, z_0, q], \varepsilon\}$$
$$[q, x, p] \rightarrow \{1\ [q, x, p]\ [p, x, p], 0\ [p, x, p]\}$$
$$[p, x, p] \rightarrow 1$$
$$[p, z_0, q] \rightarrow 0\ [q, z_0, q]$$

$\Rightarrow$

$$S \rightarrow [q, z_0, q]$$
$$[q, z_0, q] \rightarrow 1\ [q, x, p]\ [p, z_0, q] \mid \varepsilon$$
$$[q, x, p] \rightarrow 1\ [q, x, p]\ [p, x, p] \mid 0\ [p, x, p]\}$$
$$[p, z_0, q] \rightarrow 0\ [q, z_0, q]$$
$$[p, x, p] \rightarrow 1$$

Example:3.40 ***Construct a context free grammar which accepts N(A) where*** $A = (\{q_0, q_1\}, \{a, b\}, \{z_0, z_1\}, \delta, q_0, z_0, \phi)$ ***δ is given by***

$\delta(q_0, b, z_0) = \{(q_0, zz_0)\}$

$\delta(q_0, \varepsilon, z_0) = \{(q_0, \varepsilon)\}$

$\delta(q_0, b, z) = \{(q_0, zz)\}$

$\delta(q_0, a, z) = \{(q_1, z)\}$

$\delta(q_1, b, z) = \{(q_1, \varepsilon)\}$

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

☺ ***Solution:***

The variables involved in this grammar are

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

The productions are

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

(i) $\delta(q_0, b, z_0) = \{(q_0, zz_0)\}$

$$[q_0, z_0, q_0] \rightarrow b\,[q_0, z, q_0]\,[q_0, z_0, q_0] \mid b\,[q_0, z, q_1]\,[q_1, z_0, q_0]$$

$$[q_0, z_0, q_1] \rightarrow b\,[q_0, z, q_0]\,[q_0, z_0, q_1] \mid b\,[q_0, z, q_1]\,[q_1, z_0, q_1]$$

(ii) $\delta(q_0, \varepsilon, z_0) = \{(q_0, \varepsilon)\}$

$$[q_0, z_0, q_0] \rightarrow \varepsilon$$

(iii) $\delta(q_0, b, z) = \{(q_0, zz)\}$

$$[q_0, z, q_0] \rightarrow b\,[q_0, z, q_0]\,[q_0, z, q_0] \mid b\,[q_0, z, q_1]\,[q_1, z, q_0]$$

$$[q_0, z, q_1] \rightarrow b\,[q_0, z, q_0]\,[q_0, z, q_1] \mid b\,[q_0, z, q_1]\,[q_1, z, q_1]$$

(iv) $\delta(q_0, a, z) = \{(q_1, z)\}$

$$[q_0, z, q_0] \rightarrow a\,[q_1, z, q_0]$$

$$[q_0, z, q_1] \rightarrow a\,[q_1, z, q_1]$$

(v) $\delta(q_1, b, z) = \{(q_1, \varepsilon)\}$

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

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

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

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

After eliminating the unwanted variables, the CGF is given by

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

$$[q_0, z_0, q_0] \rightarrow b\,[q_0, z, q_1]\,[q_1, z_0, q_0] \mid \varepsilon$$

$$[q_0, z_0, q_1] \rightarrow b\,[q_0, z, q_1]\,[q_1, z_1, q_1] \mid a\,[q_1, z, q_1]$$

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

$$[q_1, z_1, q_1] \rightarrow b$$

TWO MARKS QUESTIONS AND ANSWERS

1. *Define pushdown automaton.*

A pushdown automaton is a finite automaton with extra resource called stack.

It contains 7 tuples

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

where

- Q - Finite non-empty set of states
- Σ - Finite set of input symbols
- Γ - Finite non-empty set of stack symbols
- δ – q_0 in Q is the start state
- z_0 - initial start symbol of the stack
- F - $F \subseteq Q$, set of accepting states
- δ - transition function:
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$

2. *What are the different ways of language acceptances by a PDA and define them?*

There are 2 ways of language acceptances

(*i*) Acceptance by final state

$$L(M) = \{w \mid (q_0, w, z_0) \overset{*}{\underset{\Gamma}{}} (p, \varepsilon, \gamma) \text{ for some P in F and } \gamma \text{ in } \overset{*}{\underset{\Gamma}{}}\}$$

(*ii*) Acceptance by empty stack

$$N(M) = \{w \mid (q_0, w, z_0) \overset{*}{\underset{\Gamma}{}} (p, \varepsilon, \varepsilon) \text{ for some P in Q}\}$$

3. *Construct a PDA that accepts the language generated by the grammar.*

S → aSbb; S → aab

☺ ***Solution:***

The PDA A = ({q},{a, b}, {S, a, b}, δ, q, S)

where δ:

(*i*) $\delta(q, z_0, S) = \{(q, aSbb), (q, aab)\}$

(*ii*) $\delta(q, a, a) = \{(q, \varepsilon)\}$

(*iii*) $\delta(q, b, b) = \{(q, \varepsilon)\}$

4. ***Construct a PDA that accepts the language generated by the grammar.***

$S \rightarrow aABB$

$A \rightarrow aB \mid a$

$B \rightarrow bA \mid b$

☺ ***Solution:***

The PDA is given by

$$A = (\{q\}, \{a, b\}, \{S, A, B, Z, a, b\}, \delta, q, S)$$

where δ is given by

$\delta(q, z, S) = \{(q, aABB)\}$

$\delta(q, z, A) = \{(a, aB), (q, a)\}$

$\delta(q, z, B) = \{(a, bA), (q, b)\}$

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

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

5. ***Define the language accepted by final state in a PDA.***

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA. Then L(P), the language accepted by P by final state is $L(P) = \{w \mid (q_0, w, z_0) \overset{*}{\vdash} (q, \varepsilon, \alpha)\}$ for some state q in F and any stack string α.

6. ***How do you convert CFG to a PDA?***

Let G = (V, T, P, S) be a CFG. Then construct a PDA P that accepts L(G) by empty stack as follows:

$$P = (\{q\}, T, V, \cup, T, \delta, q, S)$$

Where δ is given by

1. For each variable A.

$\delta(q, \varepsilon, A) = \{(q,\beta) \mid A \rightarrow B$ is a production of P$\}$

2. For each terminal a,

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

7. ***Define deterministic PDA.***

A PDA P = (Q, Σ, Γ, δ, q_0, z_0, F) to be deterministic if and only if

(*i*) δ(q, a, X) has at most one member for any q in Q, a in Σ or a = ε and X in Γ.

(*ii*) If δ(q, a, ẋ) is not empty, for some a in Σ, then δ(q, ε, X) must be empty.

8. ***Is it true that non-deterministic PDA is more powerful than that of deterministic PDA? Justify your answer?***

No, NPDA is not more powerful than DPDA. Because NPDA may produce ambiguous grammar by reaching its final state or by emptying its stack. But DPDA produces only unambiguous grammar.

9. ***Define Pumping Lemma for CFL.***

Let L be any CFL. Then there exists a constant n, depending only on L such that if z is in L and | z | ≥ n, then we can write z = uvwxy such that

(*i*) | vx | ≥ 1

(*ii*) | vwx | ≤ n

(*iii*) for all i ≥ 0, $uv^i wx^i y$ Σ L

10. ***What is the additional feature PDA has when compared with NFA? Is PDA superior over NFA in the sense of language acceptance? Justify your answer.***

PDA is superior to NFA by having the following additional features.

Stack which is used to store the necessary tape symbols and use the state to remember the conditions.

Two ways of language acceptances, one by reaching its final state and another by emptying its stack.

REVIEW QUESTIONS

1. If L is $N(M_1)$ (the language accepted by empty stack) for some PDA M_1, then L is $L(M_2)$ (the language accepted by final state) for some PDA M_2.
2. Let $M = (\{q_0, q_1\}, \{0, 1\}, \{x_1, z_0\}, \delta, q_0, z_0, \phi)$

   where δ is given by

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

   $\delta(q_0, 0, x) = \{(q_0, xx)\}$

   $\delta(q_0, 1, x) = \{(q_1, \varepsilon)\}$

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

   $\delta(q_1, \varepsilon, x) = \{(q_1, \varepsilon)\}$

   $\delta(q_1, \varepsilon, z_0) = \{(q_1, \varepsilon)\}$
3. If L is context-free language then prove that there exists a PDA M such that L = N(M) (language accepted by empty stack)
4. If L is $L(M_2)$ for some PDA M_2, then show that L is $N(M_1)$ for some PDA M_1.
5. Construct a context free grammar G which accepts N(M) where

   $M = (\{q_0, q_1\}, \{a, b\}, (z_0, z\}, \delta, q_0, z_0, \phi)$ where δ is given by

   $\delta(q_0, b, z_0) = \{(q_0, zz_0)\}$

   $\delta(q_0, \varepsilon, z_0) = \{(q_0, \varepsilon)\}$

   $\delta(q_0, b, z) = \{(q_1, zz)\}$

   $\delta(q_0, a, zx) = \{(q_1, z)\}$

   $\delta(q_1, \beta, z) = \{(q_1, \varepsilon)\}$

   $\delta(q_1, \alpha, z_0) = \{(q_0, z_0)\}$
6. Construct a PDA accepting $\{a^n\ b_m a^n\}$ m, n ≥ 1} by empty stack. Also construct the corresponding context free grammar accepting the same set.
7. Construct a PDA for the given grammar $L = \{wcw^T \mid w \text{ is in } (0 + 1)^*\}$
8. Construct a PDA for the given grammar $L = \{ww^T \mid w \text{ is in } (0 + 1)^*\}$.