NPR College of Engineering & Technology
NPR Nagar, Natham, Dindigul - 624401, Tamil Nadu, India.
Phone No: 04544- 246 500, 246501 246502
Website : www.nprcolleges.org E-mail: nprgi@nprcolleges.org

TITLE: CONTEXT FREE GRAMMAR (CFG)

OBJECTIVE: Students can able to learn Structure of CFG and its derivations

APPLICATIONS:
* EARLY APPLICATION - Structure of Programming Languages.
* NEW APPLICATION - Extensible Markup Language (Document Type Definition).

UNIVERSITY QUESTIONS:

8 MARKS

1) Derive a string "aababa" for the following Context free Grammar (CFG). $S \to aSX/b$, $X \to Xb/a$ [NOV/DEC '17]

2) Consider the Context free Grammar (CFG) given below. Give the left most derivation for the string bbaa using the grammar. $S \to bS/aT/\epsilon$, $T \to aT/bU/\epsilon$, $U \to aT/\epsilon$ [NOV/DEC '18]

3) Generate CFG for $(011+1)^*$ [APR/MAY '15]

4) i) Construct a CFG to generate even and odd set of palindromes over alphabet $\{a, b\}$.

ii) Generate CFG for the language $L = \{0^i 1^j 0^k \mid j > i+k\}$ [NOV/DEC '17]

NOTES

CONTEXT FREE GRAMMAR

DEFINITION :

Context free Grammar is a formal grammar, which helps to generate all possible strings in a given language.

Context free Grammar (CFG) has four tuples.

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

Where,

G - Grammar

V - Finite set of Variables (NON-TERMINALS)

T - Finite set of Symbols (TERMINALS)

P - Production/Rules

S - Start Symbol

CONVENTIONS (OR) NOTATIONS OF CFG

Ex:

- NON-TERMINALS $\Rightarrow (A, B \ldots Z)$
- TERMINALS $\Rightarrow (a, b \ldots z), (0, 1 \ldots 9), (+, -, *, /, \alpha, \beta, \gamma, (,), \{, \}, \ldots)$
- PRODUCTIONS $\Rightarrow A \rightarrow \alpha$

Where A - Variable

α - zero, Terminals, Non-Terminals.

NOTE :

It has no mechanism to restrict the usage of production rule $A \rightarrow \alpha$ within some specific types Context unlike other types of grammar. That is why it is called as Context free grammar.

DERIVATIONS OF CFG

* Process of defining a String based on the rules which begins from the starting Symbol.
* There are two ways to derive a Strings formed by the grammar.

1) Recursive Inference
2) Derivation.

1) RECURSIVE INFERENCE

* It takes the rules from body to head.

Ex:
1) E → I - Head
2) E → E+E
3) I → a
4) I → b

(3) and (4) : Body

Recursive Inference.

TIPS

String Infered - Based on the input (Ex: a+b)

Language - Non-Terminals (productions question)
(E → E+E)
⇓
Language.

Production - Reference No/.

String Used
⇓
Combine
Each string to get result

Reference Number ← { 1) E+E, 2) E-E }

Ex:

1. E → I
2. E → E+E
3. I → a
4. I → b

Output ⇒ a+b

STEPS	STRING INFERED	LANGUAGE	PRODUCTION	STRINGS USED
i)	a	I	3	-
ii)	b	I	4	-
iii)	a+b	E	2	(i)(ii)

DERIVATION ($\Rightarrow$)

* It derives the strings from head to Body. Ex:

$E \to E+E$ Head

$E \to 0$ Derivation

$E \to 1$ Body

* Two Types.

1) Left Most Derivation (LMD)
2) Right Most Derivation (RMD)

RELATIONS

1) $\underset{G}{\Longrightarrow}$ Single Derivation Ex: $E \underset{G}{\Rightarrow} aEb$
2) $\underset{G}{\overset{*}{\Longrightarrow}}$ Multiple Derivation Ex: $E \underset{G}{\overset{*}{\Rightarrow}} abab$

DERIVATION TYPES

1) LEFT MOST DERIVATION (LMD)

At each step in the derivation, the Production rule is applied to the leftmost Variable.

Ex: $E \to E+E$
$E \to id$
String (Input) $\Rightarrow id+id$

Solution

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

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

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

2) RIGHT MOST DERIVATION (RMD).

At each step in the derivation, Production is applied to the rightmost variable.

Ex: $E \to E+E$

$E \to id$

String $\Rightarrow id+id$

Solution

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

LANGUAGE OF CFG

* Consider the Grammar $G=(V,T,P,S)$, the language of the grammar is denoted as $L(G)$.

* It has the set of terminals strings that are derived from the start symbol.

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

* The language generated by Context free grammar is called Context free Language (CFL).

SENTENTIAL FORM

The Strings are produced (or) derived from the start symbol is called Sentential Form.

If $G=(V,T,P,S)$ is a CFG, then α in $(V \cup T)^*$ such that,

$S \overset{*}{\Rightarrow} \alpha$ Sentential Form

If $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$ Left Sentential Form

$S \overset{*}{\underset{rm}{\Rightarrow}} \alpha$ Right Sentential Form

EXAMPLES

1) Derive the string (a+(a*a)) using Left most and Right most derivation for the following Productions.

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

Solution

Given Input ⇒ (a+(a*a))

Left Most Derivation

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

$\therefore E \overset{*}{\underset{lm}{\Rightarrow}} (a+(a*a))$ //

Since * ⇒ Multiple Derivation

Right Most Derivation

$$E \underset{rm}{\Rightarrow} (E)$$
$$\underset{rm}{\Rightarrow} (E+E)$$
$$\underset{rm}{\Rightarrow} (E+(E))$$
$$\underset{rm}{\Rightarrow} (E+(E*E))$$
$$\underset{rm}{\Rightarrow} (E+(E*a))$$
$$\underset{rm}{\Rightarrow} (E+(a*a))$$
$$\underset{rm}{\Rightarrow} (a+(a*a))$$

$\therefore E \overset{*}{\underset{rm}{\Rightarrow}} (a+(a*a))$ //

Thus Proved.

2) Derive the string b+(b+all) using LMD & RMD for the following production

$E \to I / E+E / E*E / (E)$

$I \to a / b / Ia / Ib / I0 / I1$

SOLUTION

String = b*(b+all)

LMD

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

$\underset{lm}{\Rightarrow} I*E$

$\underset{lm}{\Rightarrow} b*E$

$\underset{lm}{\Rightarrow} b*(E)$

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

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

$\underset{lm}{\Rightarrow} b*(b+I)$

$\underset{lm}{\Rightarrow} b*(b+I1)$

$\underset{lm}{\Rightarrow} b*(b+I11)$

$\underset{lm}{\Rightarrow} b*(b+all)$

$\therefore E \overset{+}{\underset{lm}{\Rightarrow}} b*(b+all)$

RMD

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

$\underset{rm}{\Rightarrow} E*(E)$

$\underset{rm}{\Rightarrow} E*(E+E)$

$\underset{rm}{\Rightarrow} E*(E+I)$

$\underset{rm}{\Rightarrow} E*(E+I1)$

$\underset{rm}{\Rightarrow} E*(E+I11)$

$\underset{rm}{\Rightarrow} E*(E+all)$

$\underset{rm}{\Rightarrow} E*(b+all)$

$\underset{rm}{\Rightarrow} b*(b+all)$

$E \overset{*}{\underset{rm}{\Rightarrow}} b*(b+all)$

PROBLEMS FOR PRACTICE

RECURSIVE INFERENCE

1) Derive the string (a+a)+a using Recursive Inference for the following CFG.

$E \rightarrow E+E / E*E / (E) / a$

String $\Rightarrow$ (a+a)+a

2) Derive the string "00101" using Recursive Inference for the following CFG.

1) $S \rightarrow A1B$
2) $A \rightarrow 0A$
3) $A \rightarrow \epsilon$
4) $B \rightarrow 0B$
5) $B \rightarrow 1B$
6) $B \rightarrow \epsilon$

String = 00101

LMD & RMD

3) The following grammar generate the language for the expression $0^*1(0+1)^*$

$S \rightarrow A1B$

$A \rightarrow 0A / \epsilon$

$B \rightarrow 0B / 1B / \epsilon$

Derive the strings i) 00101 ii) 1001

iii) 00011 use LMD & RMD.

4) Consider the CFG and derive the string aabbab use LMD & RMD.

$S \rightarrow aSbS / bSaS / \epsilon$

5) RECURSIVE INFERENCE

for the language $L = \{wcw^R \mid w \in (0+1)^+\}$ check whether the string 01c10 belongs to L or not.

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

KEY WORDS

CFG, G= V, T, P, S

V- Variable (Capital letters)

T- Terminals (Number, Digits, Special Characters)

P- Production Symbol

S- Start Symbol.

TITLE: PARSE TREES

OBJECTIVE: To learn how to construct parse tree for the given CFG

APPLICATIONS:
* High Speed Networks
* Multifunctional Radar Constructions
* Neural Networks

UNIVERSITY QUESTIONS:

1) Construct a parse tree of $(a+b)^*c$ for the grammar $E \rightarrow E+E / E*E / (E) / id$ [APR/MAY '15]

2) Construct G whose productions are $S \rightarrow aAS / a$, $A \rightarrow Sba / AA / ba$. Show that $S \Rightarrow aabbaa$ and Construct a derivation tree whose yield is aabbaa [NOV/DEC'9]

NOTES

PARSE TREES

DEFINITION:

The Strings that are derived from the Context free Grammar (CFG) are represented in a tree format is called Parsetree.

PARSE TREES

* Graphical representation of symbol
* Symbol / Terminal (or) Non-Terminal } derives ⟹ from Start Symbol
* Start Symbol ⇒ Root of the parse tree.

RULES

i) Leaf nodes → terminals

ii) Interior nodes → Non-terminals

YIELD OF A PARSE TREE

The String can be formed by Concatenating the leaves of a parse tree from the left is called as yield of a parse tree.

Yield ⟹ root
derived from

Leaves ⇒ Terminals (or) ∈

∴ Yield should be Terminal String.

EXAMPLES

1) For the grammar G defined by the productions

$S \to A1B$, $A \to 0A / \varepsilon$, $B \to 0B / 1B / \varepsilon$

Find the parse tree for the yields.

i) 1001 ii) 00101 iii) 00011

i) 1001

SOLUTION (USE LMD (OR) RMD)

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

$\underset{lm}{\Rightarrow} 1B$ ($\because A \to \varepsilon$)

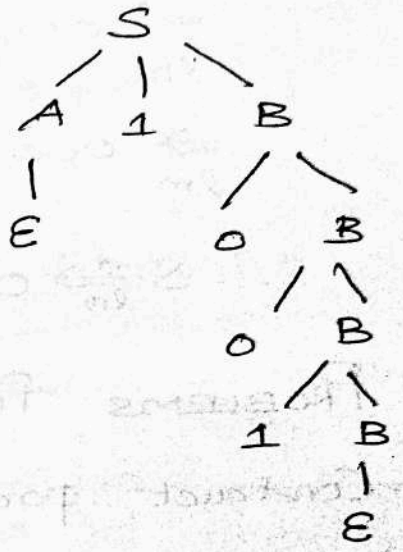
$\underset{lm}{\Rightarrow} 10B$ ($\because B \to 0B$)

$\underset{lm}{\Rightarrow} 100B$ ($\because B \to 0B$)

$\underset{lm}{\Rightarrow} 1001B$ ($\because B \to 1B$)

$\underset{lm}{\Rightarrow} 1001$ ($\because B \to \varepsilon$)

PARSE TREE



ii) 00101

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

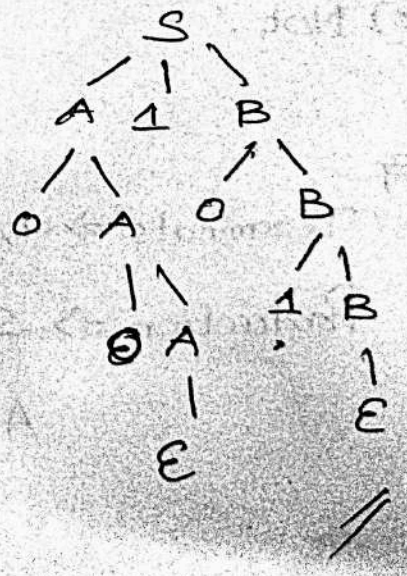
$\underset{lm}{\Rightarrow} 0A1B$ ($\because A \to 0A$)

$\underset{lm}{\Rightarrow} 00A1B$ ($\because A \to \varepsilon$)

$\underset{lm}{\Rightarrow} 0010B$ ($\because B \to 0B$)

$\underset{lm}{\Rightarrow} 00101B$ ($\because B \to 1B$)

$\underset{lm}{\Rightarrow} 00101$ ($\because B \to \varepsilon$)



iii) 00011

PARSE TREE

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

$\underset{lm}{\Rightarrow} 0A1B$ [$\because A \to 0A$]

$\underset{lm}{\Rightarrow} 00A1B$ [$\because A \to 0A$]

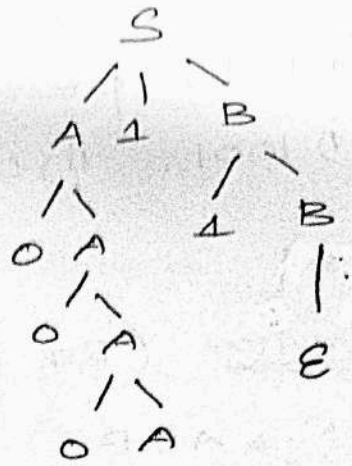
$\underset{lm}{\Rightarrow} 000A1B$ [$\because A \to 0A$]

$\underset{lm}{\Rightarrow} 000\,1B$ [$\because A \to \varepsilon$]

$\underset{lm}{\Rightarrow} 00011B$ [$B \to 1B$]

$\underset{lm}{\Rightarrow} 00011$ [$B \to \varepsilon$]

$\therefore S \overset{*}{\underset{lm}{\Rightarrow}} 00011$



PROBLEMS FOR PRACTICE

1) Construct parse tree for the following grammar

$E \to E+E$

$E \to id$

Check whether yield id+id+id is accepted (or) Not.

2) CFG

Terminals => a, b Non-Terminals => S, A

Production => $S \to AAA \mid AA$

$A \to AA \mid aA \mid a \mid b$

3) Construct parse tree for the string a*(a+boo) for the following CFG.

$E \to I$

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

$I \to a$

$I \to b$

$I \to Ia \mid Ib \mid IO \mid I1$

INFERENCE, DERIVATION & PARSE TREES

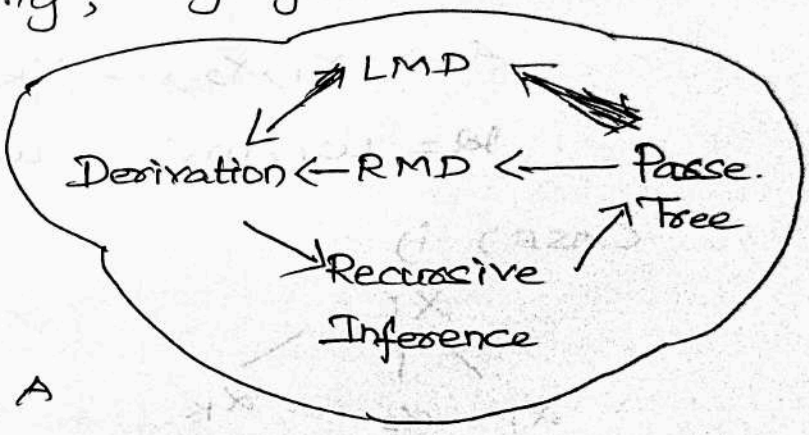
Given a Grammar, $G = (V, T, P, S)$

ω = Terminal String, Language Variable = A

1) $A \overset{*}{\Rightarrow} \omega$

2) $A \underset{lm}{\overset{*}{\Rightarrow}} \omega$

3) $A \underset{rm}{\overset{*}{\Rightarrow}} \omega$



4) Parse Tree Root, A

Yield $\Rightarrow \omega$

THEOREM

FROM INFERENCE TO PARSE TREES

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

String $= \omega$ Language $= A$

Root $\Rightarrow$ A Yield $\Rightarrow \omega$

SOLUTION

To PROOVE

$A \overset{+}{\Rightarrow} \omega$ i, Parsetree with Root A

PROOF

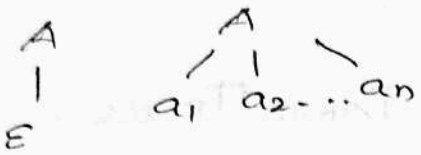
By Induction Hypothesis

BASIS

$A \to a_1, a_2, \ldots a_n$

(or)

$A \to \varepsilon$



INDUCTION

Steps $\Rightarrow n+1 \Rightarrow K$

$A \to X_1, X_2 \ldots X_K$

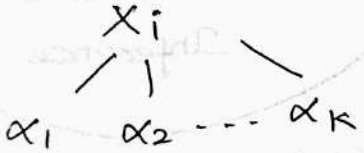
$\omega = \omega_1, \omega_2 \ldots \omega_K$

2 CASES

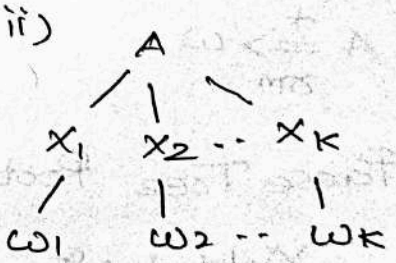
i) $\omega_i = x_i$ (terminals) (Strings)

ii) $A \overset{+}{\Rightarrow} \omega_1, \omega_2 \ldots \omega_K \overset{+}{\Rightarrow} \omega$

CASE) i)



ii)



∴ Yield can be processed from left to right. //

FROM TREES TO DERIVATIONS

THEOREM Given

Let $G = (V, T, P, S)$ be a CFG

Parse Tree Root A

Yield $\Rightarrow w$ $\quad A \underset{lm}{\overset{+}{\Rightarrow}} w$ in CFG.

PROOF :

To Prove $\quad A \underset{lm}{\overset{+}{\Rightarrow}} w$

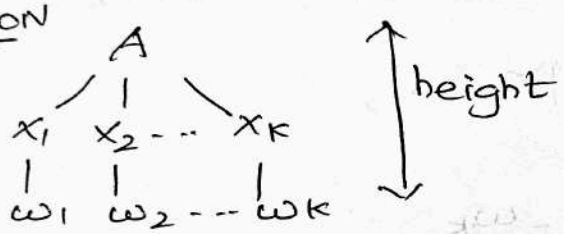
Based on the Height of ~~tree~~ tree.

BASIS

$A \to w$

$A \underset{lm}{\Rightarrow} w$ (1 Step LMD)

INDUCTION



LMD $\Rightarrow A \underset{lm}{\overset{*}{\Rightarrow}} x_1, x_2 \ldots x_n$

$x_i \Rightarrow$ Terminals (or) Variables

i) If x_i is a terminal then $x_i = w_i$

ii) $x_i \Rightarrow$ Variable (Root Node)

By applying Induction Hypothesis,

$$x_i \underset{lm}{\overset{+}{\Rightarrow}} w_i$$

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

Construct LMD of w,

$$A \underset{lm}{\Rightarrow} X_1, X_2 \dots X_k$$

$$A \underset{lm}{\Rightarrow} \omega_1, \omega_2 \dots \omega_k, X_{i+1}, X_{i+2} \dots X_k$$

i) $X_i \Rightarrow$ Terminal (No change)

$$A \overset{*}{\underset{lm}{\Rightarrow}} \omega_1, \omega_2 \dots \omega_i\ X_{i+1}, X_{i+2} \dots X_k$$

ii) $X_i \Rightarrow$ Variable

$$X_i \underset{lm}{\Rightarrow} \alpha_1 \Rightarrow \alpha_2 \dots \Rightarrow \omega_1$$

$$\therefore A \underset{lm}{\Rightarrow} \omega_1, \omega_2 \dots \omega_{i-1}, X_i, X_{i-1} \dots X_k$$

$$\underset{lm}{\Rightarrow} \omega_1, \omega_2 \dots \omega_{i-1}, \alpha_1, X_{i-1}, X_k$$

$$\underset{lm}{\Rightarrow} \omega_1, \omega_2 \dots \omega_{i-1}, \alpha_2, X_{i-1}, X_k$$

By Repeating the process,

We get,

$$A \Rightarrow \omega_1, \omega_2 \dots \omega_k$$

This theorem is similar to RMD.

THEOREM

FROM DERIVATIONS TO RECURSIVE INFERENCE

Given

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

$A \overset{*}{\underset{\omega}{\Rightarrow}}$, then Recursive Inference procedure is applied to G determines ω is in language of Variable A.

PROOF

Based on the length of the derivation

$A \overset{*}{\Rightarrow} \omega$.

BASIS

Derivation → one step then,

$A \to \omega$ (production)

$A \Rightarrow \omega$ (ω in A)

INDUCTION

Derivation → n+1 steps

$A \Rightarrow X_1 X_2 \cdots X_k \overset{*}{\Rightarrow} \omega$

Break $\omega = \omega_1, \omega_2 \cdots \omega_k$

i) X_i → terminal then $\omega_i = X_i$

ii) X_i → Variable then $X_i \overset{*}{\Rightarrow} \omega_i$

$A \Rightarrow X_1 X_2 \cdots X_k$

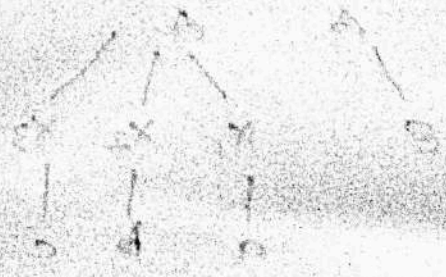
$\overset{*}{\Rightarrow} \omega_1 X_2 \cdots X_k$

$\overset{*}{\Rightarrow} \omega_1, \omega_2, \cdots X_k$

$\vdots$

$\overset{*}{\Rightarrow} \omega$

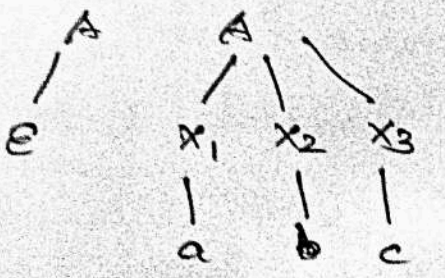
// Proved



KEYWORDS

* Start Symbol ⇒ Root Node
* Symbols / Terminals ⇒ Leaf Nodes

Ex:



ε - Empty String.

TITLE: AMBIGUITY IN GRAMMARS & LANGUAGES

OBJECTIVE: To learn how to Construct two different Parse trees.

APPLICATIONS:
* Data Processing
* Natural Language Processing
* Human Activities Recognization

UNIVERSITY QUESTIONS:

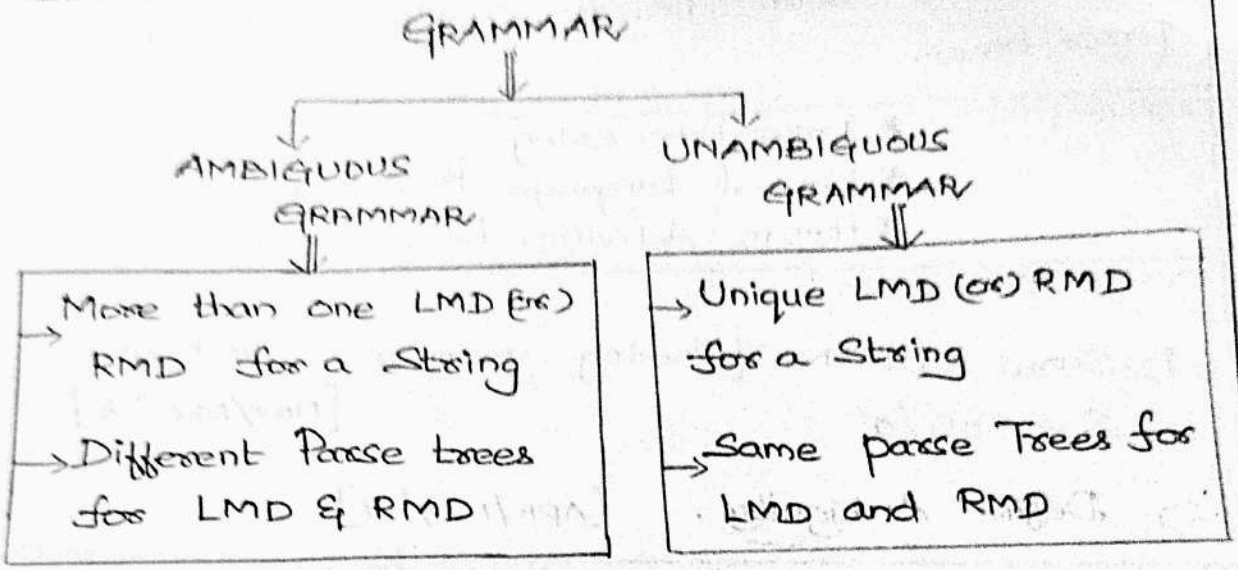
1) Show that the following grammar is ambiguous. [NOV/DEC '18]
$S \rightarrow SbS/a$

2) Define Ambiguity. [APR/MAY '18]

NOTES

AMBIGUOUS GRAMMAR

A CFG $G = (V, T, P, S)$ is ambiguous if there exists a string that has more than one Parse tree.



EXAMPLES

1) If G is the grammar $S \rightarrow S+S \mid S*S \mid a$, Show that G is Ambiguous. String $\Rightarrow w = a*a*a$

Solution

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

$\underset{lm}{\Rightarrow} a*S$

$\underset{lm}{\Rightarrow} a*S*S$

$\underset{lm}{\Rightarrow} a*a*a$

ii) $S \underset{rm}{\Rightarrow} S*S$

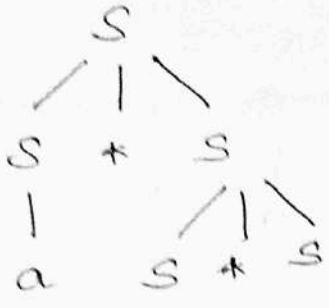
$\underset{rm}{\Rightarrow} S*a$

$\underset{rm}{\Rightarrow} S*S*a$

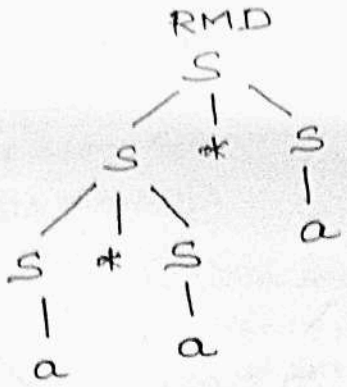
$\underset{rm}{\Rightarrow} a*a*a$

PARSE TREE

LMD



RMD



The String a*a*a has 2 parse trees which are different.

∴ The Given grammar G is Ambiguous Grammar.

2) Show that the grammar defined by the productions $S \to SS/a/b$ is ambiguous.

$w = aaba$.

Solution

LMD

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

$\underset{lm}{\Rightarrow} aS$

$\underset{lm}{\Rightarrow} aSS$

$\underset{lm}{\Rightarrow} aaS$

$\underset{lm}{\Rightarrow} aaSS$

$\underset{lm}{\Rightarrow} aabS$

$\underset{lm}{\Rightarrow} aaba$

RMD

$S \underset{rm}{\Rightarrow} SS$

$\underset{rm}{\Rightarrow} Sa$

$\underset{rm}{\Rightarrow} SSa$

$\underset{rm}{\Rightarrow} Sba$

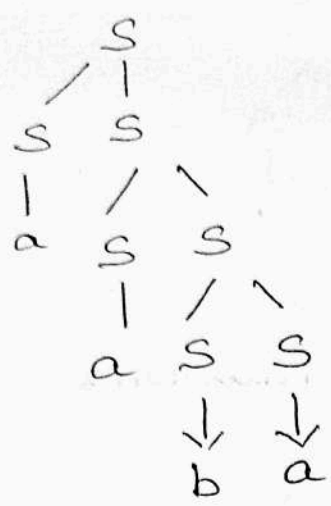
$\underset{rm}{\Rightarrow} SSba$

$\underset{rm}{\Rightarrow} Saba$

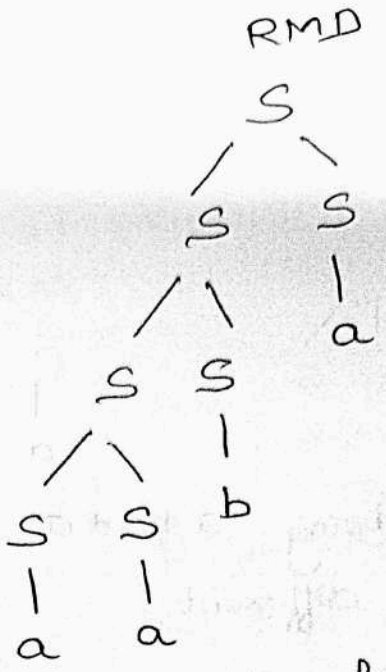
$\underset{rm}{\Rightarrow} aaba$

PARSE TREE

LMD



RMD



Since both LMD and RMD has different Parse trees. ∴ The given grammar G is Ambiguous.

PROBLEMS FOR PRACTICE

1) Show that the grammar defined by the Productions.

$E \rightarrow E+E \mid E*E \mid a$

Hint - a+a*a , a*a+a , a+a+a , a*a*a

Note : Input (or) String (w) - Based on your own choice

2) Consider the following grammar, $S \rightarrow aS \mid aSbS \mid \epsilon$

Show that this grammar is ambiguous for the string 'aab' that has two parse trees, two left most derivations and two rightmost derivations.

3) Check whether the following grammar is ambiguous

$E \rightarrow E+E/E-E/E*E/a \mid (E)$

Hint - Suggested String - a+a*a (or) (a+a-a) (or) (a*a-a)

4) Show that $E \rightarrow E+E / E*E / (E) / id$ is ambiguous

Hint - (id+id), (id+id*id), id+id*id

REMOVING AMBIGUITY FROM GRAMMARS

REASONS

i) Operators precedence is not followed.

Ex: $E+E*E$

We can group * & + in two ways.

ii) Sequence of identical operators can group either from left (or) right.

SOLUTIONS ⇒ Binding Strength [Share the level of the expressions].

i) FACTOR - An Expression which cannot broken by adjacent operator, either * or +.

Factor

Identifiers

Soln ⇒ (Not possible to Seperate letters → attaching by an operator).

Parenthesized Expression

(Prevent from becoming an operand) ◆ Any operator outside the parenthesis

ii) TERM - Product of one (or) more factors.

iii) EXPRESSION - Possible expressions which can be broken by either an adjacent * (or) +.

EXAMPLES

1) An unambiguous expressions grammar is given by,

$E \rightarrow E+T / T \quad T \rightarrow T*F, \quad F \rightarrow id$

Find the parse tree for id+id*id

Solution

LMD

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

$\underset{lm}{\Rightarrow} T+T$

$\underset{lm}{\Rightarrow} F+T$

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

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

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

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

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

RMD

$E \underset{rm}{\Rightarrow} E+T$

$\underset{rm}{\Rightarrow} E+T*F$

$\underset{rm}{\Rightarrow} E+T*id$

$\underset{rm}{\Rightarrow} E+F*id$

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

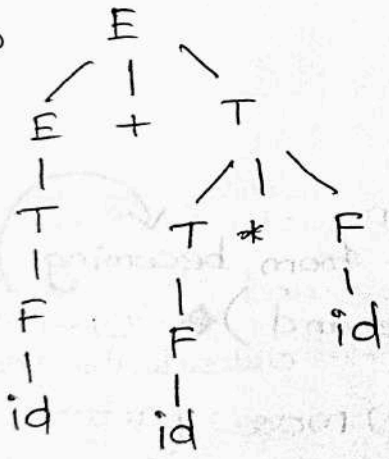
$\underset{rm}{\Rightarrow} T+id*id$

$\underset{rm}{\Rightarrow} F+id*id$

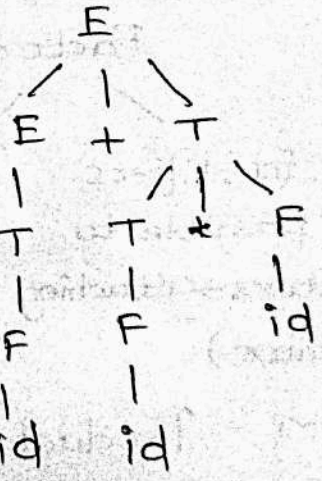
$\underset{rm}{\Rightarrow} id+id*id$

PARSE TREES

LMD



RMD



∴ The yield id+id*id has same parsetrees.

∴ The Given grammar G is Unambiguous.

PROBLEMS FOR PRACTICE

1) The following grammar generates prefix expressions with operands x and y and binary operators $+, -$ and $*$ and the grammar is as follows.

$$E \rightarrow +EE \mid *EE \mid x \mid y$$

a) Find the leftmost and right most derivation & derivation tree for the string "$+*-xyxy$".

b) Proove that this grammar is unambiguous.

2) Consider the following grammar

$$S \rightarrow A1B \quad A \rightarrow 0A \mid \epsilon \quad B \rightarrow 0B \mid 1B \mid \epsilon$$

a) Show that this grammar is unambiguous.

b) find a grammar for the same language that is ambiguous and demonstrate its ambiguity.

3) Show that the grammar (G_1)

$S \rightarrow aS \mid aSbS \mid \epsilon$ is ambiguous and find the unambiguous grammar.

Hint ⇒ Find Ambiguous & the rewrite Productions and Show G_1 is unambiguous.

4) Check whether the following grammar is ambiguous for $w = aab$

$$S \rightarrow AB \mid aaB \quad A \rightarrow a \mid Aa \quad B \rightarrow b$$

5) The grammar 'G' for palindrome is

$S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$.

Check if G is ambiguous.

6) Check whether the given grammar is ambiguous (or) not.

$S \to iCtS \mid iCtSeS \mid a$

$C \to b$

7) Check whether the following grammar is ambiguous for $\omega = ab$.

$S \to aB \mid ab$, $A \to aAB \mid a$, $B \to ABb \mid b$

8) Check whether the following grammar is ambiguous for $\omega = aabc$

$S \to AB \mid aaB$, $A \to a \mid Aa$, $B \to bC$ $C \to c$

9) Check whether the following grammar is ambiguous for $\omega = abababa$

$S \to SbS \mid a$

10) Check whether the following grammar is ambiguous

$S \to 0S1 \mid SS \mid \epsilon$

11) Consider the grammar 'G' with

Terminals : a, b

Non Terminals : S, X

Productions : $S \to aS \mid aSb \mid X$

$X \to Xa \mid a$

Show that G is ambiguous

13) Consider the grammar 'G' with

Productions : $S \to aS \mid Sa \mid a$

Show that G is ambiguous.

TITLE: PUSH DOWN AUTOMATA (PDA)

OBJECTIVE:
It helps to recognize Context-free Languages

APPLICATIONS:
Implementation of stack applications
Evaluating Arithmetic Expressions

UNIVERSITY QUESTIONS:
1) What is an instantaneous description (ID) of a Push down automaton (PDA)? [NOV/DEC'18]
2) Construct PDA for the language $a^n b^m a^{n+m}$. [NOV/DEC'17]

NOTES

PUSH DOWN AUTOMATA (PDA)

INTRODUCTION

* In chomsky hierarchy, the requirement is needed for a grammar is to generate a language and it is clubbed with a suitable machine that can be used to accept the same language.

When,

* Grammar → simple ⇒ Language (More Complex)

* Hence we need more powerful machine to understand the language and accept it.

EXAMPLE

Type 0 Grammar - Simple - Generate Recursive Enumeral Language (REL)

↓ Need

Turing Machine → To accept all Strings.

Type 2 Grammar - Generates CFL accepted by PDA.

DEFINITION

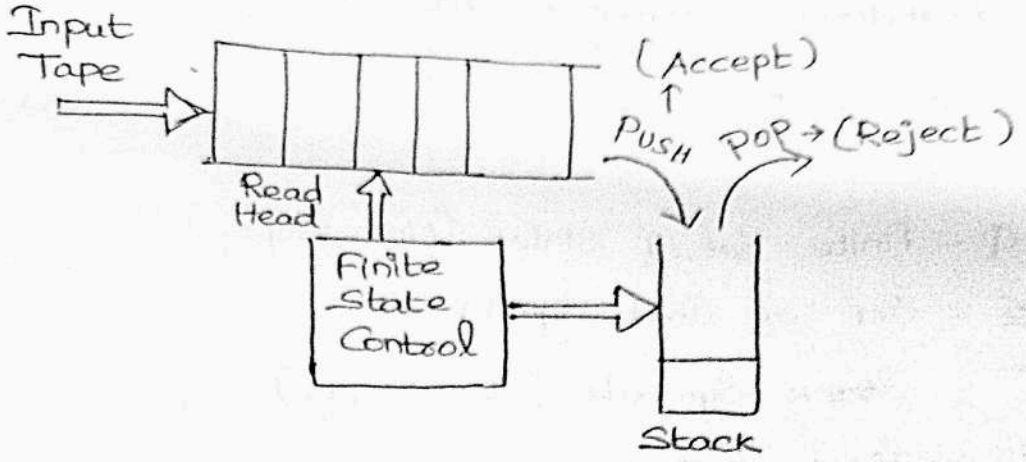
* Push Down Automaton (PDA) is an extension of NFA with ∈ transitions.

* PDA are finite automatons which Consists of three Components.

1) Input tape
2) Finite Control
3) Stack Structure

* It can accessed only in Last-In-First-Out fashion (LIFO).

REPRESENTATION OF PDA
(MODEL)



* The Input tape has several cells which Stores character of an input alphabet.

* It moves only one cell at a time from the left.

Finite Control State - points the Current Symbol to be read.

* Stack - which stores arbitary numbers of Symbols but it can accessed only by Last-In-first Out (LIFO) fashion.

PROBLEMS

* Push Operation - Check Overflow Condition

* Pop Operation - Check Underflow Condition.

ASSUMPTION

* To overcome Underflow Condition - Keep Stack empty

* Initial Stack Symbol is used to declare the String Processing.

FORMAL DEFINITION OF PDA

A Pushdown automaton has Seven tuples

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

Where,

Q - finite Set of States (Non-Empty)

Σ - Set of Input Symbols

Γ - Stack Symbols (Non-Empty)

q_0 - Start state

Z_0 - Initial Start Symbol of Stack.

f - Accepting State (or) Final State

δ - Transition function

PDA - 2 Alphabets

Σ (Input Alphabet) Γ (Stack Alphabet)

MOVES

i) Element - Added to the Stack

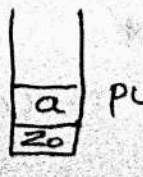
ii) Element - Deleted to the Stack

$(q, a, Z_0) = (q, \epsilon)$

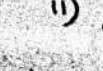
iii) No changes in the State.

Ex:

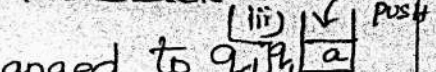
i) $(q, a, Z_0) = (q, aZ_0)$ — PUSH in to the Stack



ii) $(q, a, Z_0) = (q, \epsilon)$ ⇒ Top Symbol Z_0 pop from the stack



iii) $(q, a, Z_0) = (q_1, aZ_0)$ ⇒ a push in to Stack and State is changed to q_1



TRANSITION FUNCTION OF PDA

Based on Current State, Input symbol and top of the Stack.

$$\delta(q,a,x) = (p,\gamma)$$

δ - has triple argument

Output - pair of arguments

Where,

q - State in Q

a - Input Symbol (Empty Input) ϵ (or) $a = \epsilon$

x - Stack Symbol (Top)

P - New State (After reading a)

γ - Symbol pushed on to the stack

& Replaces the top symbol x on the Stack.

Suppose

$\gamma = \epsilon \Rightarrow$ Stack is popped

$\gamma = x \Rightarrow$ Stack is unchanged

$\gamma = yz \Rightarrow x$ replaced by z

y pushed in to Stack.

GRAPHICAL REPRESENTATION OF PDA

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA

Where $Q = \{P, q\}$, $\Sigma = \{a, x, \alpha\}$, $q_0 = P$, $F = \{q\}$

We Know that,

$\longrightarrow$ = Starting State - Accepting State

Ex: $\delta(P, a, x) = (q, \alpha)$

(a, x) goes to $\alpha \Rightarrow$ from State P to q

Where,

a - input symbol

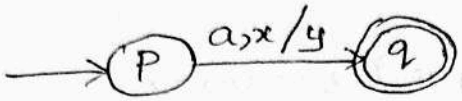
x - old stack symbol (top)

α - New stack symbol (top)

Ex: 1

1) $\delta(p, a, x) = (q, y)$ Read a

Then stack symbol x is replaced by y



State is, p(a, x) move to the state q with stack symbol (α)

2) $\delta(p, b, y) = (q, \epsilon)$

Here, it reads the input b then the stack symbol y is popped.

EXAMPLE

Draw the transition diagram for the following

$\delta(q_0, 0, 1) = (q_0, 01)$

$\delta(q_0, \epsilon, z) = (q_1, \epsilon)$

$\delta(q_1, 1, 0) = (q_1, 10)$

$\delta(q_1, \epsilon, 1) = (q_2, z)$

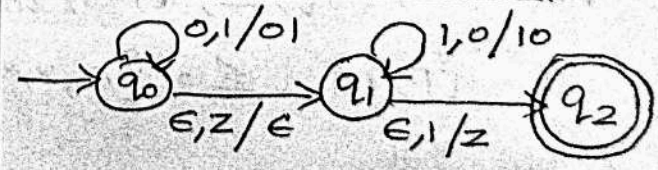
SOLUTION

Here,

q_0 - Initial state

q_2 - Final state

TRANSITION DIAGRAM

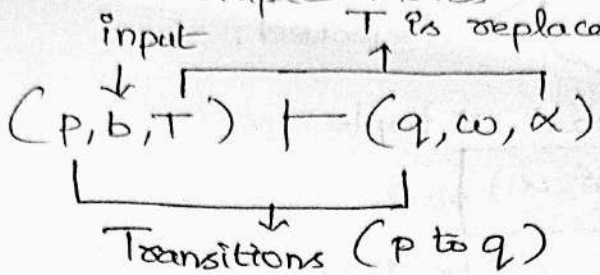


TURNSTILE NOTATION

$\vdash$ - Turnstile Notation
(Means one Move)

$\vdash^*$ => Multiple Moves in Sequence.

Ex:



EXAMPLES

1) Suppose the PDA $P=(\{p,q\},(0,1),\{z_0,x\},\delta,q,z_0,\{p\})$ has the following transition functions.

1, $\delta(q,0,z_0)=\{(q,xz_0)\}$
2, $\delta(q,0,x)=\{(q,xx)\}$
3, $\delta(q,1,x)=\{(q,x)\}$
4, $\delta(q,\epsilon,x)=\{(p,\epsilon)\}$
5, $\delta(p,\epsilon,x)=\{(p,\epsilon)\}$
6, $\delta(p,1,x)=\{(p,xx)\}$
7, $\delta(p,1,z_0)=\{(p,\epsilon)\}$

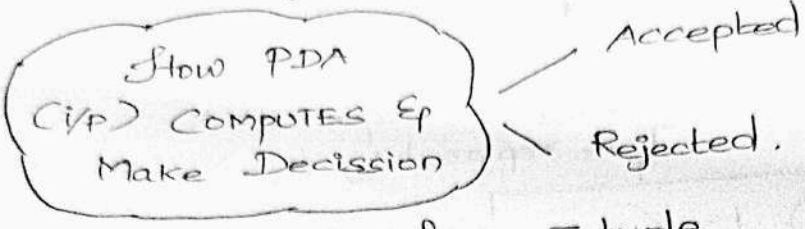
Input (ω) = 01, 0011, 010

Solution

i) $\omega=01$, Initial id => (q,ω,z_0)

$(q,01,z_0) \vdash (q,1,xz_0)$ See Rule 1 $\delta(q,0,z_0)=(q,xz_0)$
$\vdash (q,\epsilon,x)$ Rule 3 $\delta(q,1,x)=(q,x)$
$\vdash (p,\epsilon)$ Rule 4 $\delta(q,\epsilon,x)=(p,\epsilon)$

INSTANTANEOUS DESCRIPTIONS OF PDA (ID)



The ID has 3 tuple

$$ID = (q, \omega, \alpha)$$

Where,

q - Current State

ω - Remaining Input

α - Stack Contents, top at the left.

Ex:

Write instantaneous description for the string ababcbcb.

Solution

$q(ababcbab, \varepsilon) \Rightarrow (q, ababcbab, a)$

$\Rightarrow (q, abcbab, ba)$

$\Rightarrow (q, bcbab, aba)$

$\Rightarrow (q, cbab, baba)$ (When c is pushed, the upcomming input b is popped)

$\Rightarrow (q, bab, cbaba)$

$\Rightarrow (p, ab, aba)$ $\Rightarrow$ (Here a is popped)

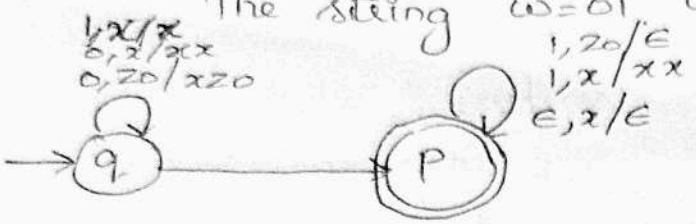
$\Rightarrow (p, b, ba)$ $\Rightarrow$ b is popped

$\Rightarrow (p, \varepsilon, a)$

P (final state) is Reached.

$(q,01,z_0) \vdash (q,1,xz_0) \vdash (q,\epsilon,x) \vdash (p,\epsilon)$

∴ The string $w=01$ is accepted.



ii) $w=0011$

$(q,0011,z_0) \overset{\text{Rule 1}}{\vdash} (q,011,xz_0) \overset{\text{Rule 2}}{\vdash} (q,11,xxz_0)$

$\overset{\text{Rule 3}}{\vdash} (q,1,xxz_0) \overset{\text{Rule 3}}{\vdash} (q,\epsilon,xxz_0) \overset{\text{Rule 4}}{\vdash} (p,\epsilon,xz_0)$

$\overset{\text{Rule 5}}{\vdash} (p,\epsilon)$

∴ The string 0011 was accepted

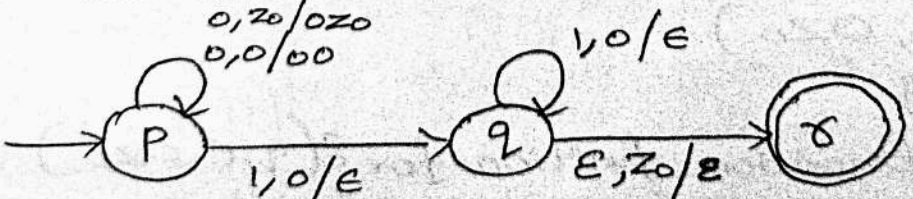
iii) $w=010$

$(q,010,z_0) \vdash (q,10,xz_0)$

$\overset{\text{Rule 1}}{\vdash} (q,0,xz_0) \overset{\text{Rule 3}}{\vdash} (q,0,xxz_0) \overset{\text{Rule 3}}{\vdash} (q,\epsilon,xxz_0)$

$\vdash (p,\epsilon,xz_0) \vdash (p,\epsilon)$

2) Consider the following PDA and find whether the strings 000111 and 00011 is accepted.



SOLUTION

TRANSITION FUNCTION

$\delta(p,0,z_0) = (p,0z_0)$

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

$\delta(p,1,0) = (q,\epsilon)$

$\delta(p,\epsilon,z_0) = (r,\epsilon)$

$\delta(q,1,0) = (q,\epsilon)$

$\delta(q,\epsilon,z_0) = (r,\epsilon)$

Here,

Starting State = P

Accepting State (or) Final State } = r

i) String $\omega = 000111$

$(p,000111,z_0) \vdash (p,00111,0z_0) \vdash (p,0111,00z_0)$

$\vdash (p,111,000z_0) \vdash (q,11,00z_0) \vdash (q,1,0z_0)$

$\vdash (q,\epsilon,z_0) \vdash (r,\epsilon)$

The given input $(\omega) = 000111$ is accepted.

ii) String, $\omega = 00011$

$(p,00011,z_0) \vdash (p,0011,0z_0) \vdash (p,011,00z_0)$

$\vdash (p,11,000z_0) \vdash (q,1,00z_0)$

$\vdash (q,\epsilon,0z_0)$

⇓

No transition function for $\delta(q,\epsilon,0z_0)$

So PDA is not reaching the final state r

∴ String 00011 is not accepted.

PROBLEMS FOR PRACTICE

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

Hint- Draw Transition diagram, function, check input Process. Take input as 100C001 (Accepting)
Note 'C' means Separator. (110C10) - (Rejected)

2) Construct a PDA for the language

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

Hint- Draw Transition diagram, function, check input process. Take input as 010010, 1001
w - Input, w^R - Reverse Input.

3) Design the PDA to accept the language $L = \{0^n 1^n / n \geq 1\}$ accepting by final state.

Hint ⇒ Input 0011 (Accepting State)
011 (Rejection)

4) Design PDA to accept the language $L = \{a^n b^n / n \geq 0\}$ accepting by final state.

Hint ⇒ Input $w = aabb, abb$

5) Design PDA with the set of strings with twice as many as b's than a's with 'a' as starting string. (or)
Design PDA for the set of all strings $\{a, b\}$ with two occurences of b's for each a's.

(or) Design PDA for the language $L=\{a^n b^{2n} / n \geq 0\}$

Hint => Input starts with a
b is twice of a $[a^n \Rightarrow b^{2n}]$
Ex: Input(w)=aabbbb, aaabbbbbb

6) Construct the PDA to accept the set of all strings over the alphabets $\{0,1\}$, such that the total number of 0's are more than total number of 1's.

Hint => Input (w) = 01100 (Accepted)
w = 1010 (Not Accepted)

7) Design a PDA that accepts the language $L=\{a^{n+1} b^n, n \geq 0\}$.

8) Design a PDA for the language $L=\{a^{3n} b^n / n \geq 0\}$

9) Design a PDA to accept the language
$L=\{a^n b^m c^n / m, n \geq 1\}$

10) Design a PDA to accept the language
$L=\{a^{2n} b^{n+1} / n \geq 1\}$ by reading final state and Convert this PDA reaching final state into PDA by empty stack.

LANGUAGE OF PDA

⇒ Accepts the input and enter into the final state.

2 WAYS

1) Acceptance by Final State

2) Acceptance by Empty Stack.

1) ACCEPTANCE BY FINAL STATE

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA, then the language accepted by final state is,

$L(M) = \{ w / (q_0, w, Z_0) \vdash^* (P, \epsilon, \alpha) \text{ for some } P \text{ in } F \text{ and } (\alpha) \text{ in } \Gamma^* \}$

Where,

q_0 - Starting State

P - Final State

α - Stack input

w - Input

Z_0 - Stack top symbol.

2) ACCEPTANCE BY EMPTY STACK

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA, then the language accepted by empty stack is,

$N(M) = \{ w | (q_0, w, Z_0) \vdash^* (P, \epsilon, \epsilon) \text{ for some } P \text{ in } Q \}$

Here,

ϵ - Empty Stack.

THEOREM

FROM EMPTY STACK TO FINAL STATE

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 $L = L(P_F)$

[NOV/DEC '03/04/06]

PROOF

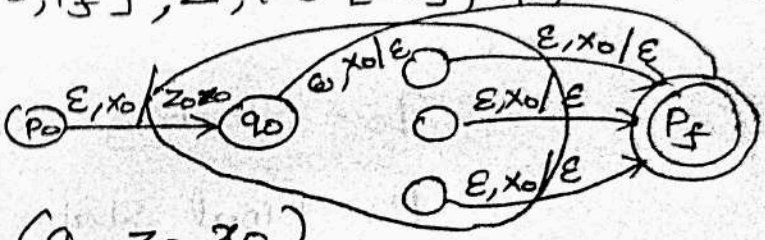
It states that if there is a PDA which has acceptance by empty stack, then there should be a PDA also has acceptance by Final state.

Let $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0, F)$ be a PDA.

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

Where,

δ_F is,



1) $\delta_F(P_0, \in, X_0) = (q_0, Z_0, X_0)$

P_F pushes X_0 on the bottom of the stack.

2) States q in Q

Input a in Σ (or) $a = \in$

Stack symbol y in Γ

$\delta_F(q, a, y)$ contains all pairs in $\delta_N(q, a, y)$

3) $\delta_F(q, \in, X_0) = (P_f, \in)$ ⇒ Accepted & Reach the final state.

∴ We proove that w in $L(P_F) = w$ in $N(P_N)$

The ID of P_F after simulating P_N is given by,

$(P_0, \omega, x_0) \vdash (P_0, \omega, z_0, x_0) \vdash (q, \epsilon, x_0)$

$\vdash (P_f, \epsilon, \epsilon)$

Thus PDA P_f reaches final state and accepts the string ω.

THEOREM

LANGUAGE ACCEPTED BY FINAL STATE TO EMPTY STACK.

If L is a PDA $L(P_F)$ i.e, $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, z_0, F)$, there exists a PDA P_N such that $L = N(P_N)$.

TO PROOVE

The PDA P_N if and only if there exists PDA P_F.

PROOF

To proove this theorem,

Steps

1) We use x_0 (Stack symbol) on the bottom of the Stack.

2) Construct P_N from P_F

3) q_0 - Initial State of P_F &

We have set of final state for P_F.

4) PDA $P_F \rightarrow$ Enter the final state after accepting the string ω with leaving any symbols on the Stack.

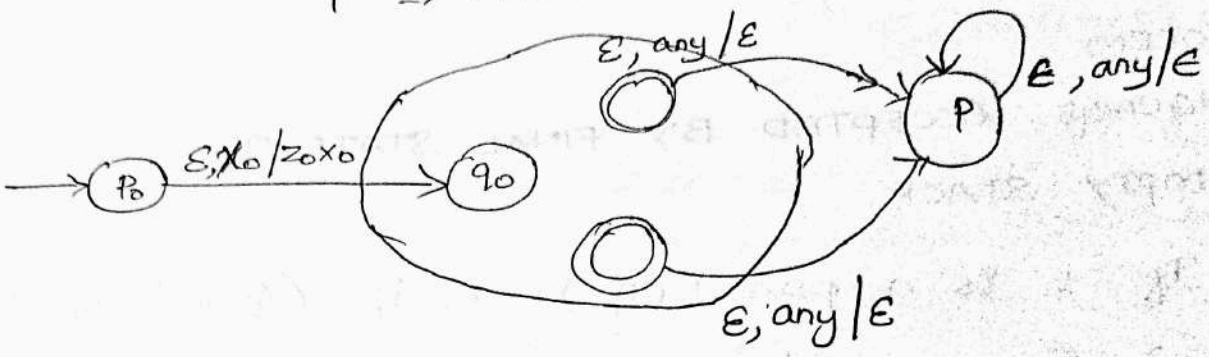
TARGET

Delete all the Stack Symbol of PDA P_F

after entering the final state in order to Construct P_N by empty stack.

Construct, $P_N \Rightarrow P_0$ (New Start state)

$P \Rightarrow$ New Final State.



PDA $P_N = (Q \cup \{P_0, P\}, \Sigma, \Gamma \cup \{x_0\}, \delta_N, P_0, x_0)$

Where,

P_0 - Initial State

x_0 - Starting Stack Symbol

Transition function δ_N,

1) $\delta_N(P_0, \epsilon, x_0) = \{(q_0, z_0 x_0)\}$

Push the start symbol z_0 & enter into the State q_0

2) States q in Q

Input symbol a in Σ (or) $a = \epsilon$

Y in Γ

$\delta_N(q, a, y)$ has pair of $\delta_F(q, a, y)$

3) final states q in F

Stack Symbol y in Γ (or) $y = x_0$

$\delta_N(q, \epsilon, y)$ contains (p, ϵ)

4) Stack symbol y in $\vdash$ (or) $y = x_0$

$$\delta_N(p, \epsilon, y) = \{(p, \epsilon)\}$$

∴ We conclude that,

$$w \text{ in } N(P_N) = L(P_F)$$

The ID can be written as,

$$(p_0, w, x_0) \vdash_{P_N} (q_0, w, z_0 x_0) \vdash^* (q, \epsilon, \alpha x_0)$$

$$\vdash^* (p, \epsilon, \epsilon)$$

Thus PDA P_N accepts w by Empty Stack.

KEYWORDS

P.DA

↳ Input Tape

↳ Finite Control

↳ Stack Structure

↳ Accessed In LIFO order.

TITLE: EQUIVALENCE OF PDA & CFG

OBJECTIVE:
To know about the Conversion of PDA to CFG and Vice-Versa.

APPLICATIONS:
* Evaluating Arithmetic Expressions
* Syntax Analysis

UNIVERSITY QUESTIONS:

1) Convert the following CFG to Pushdown automaton [NOV/DEC '18]
$S \rightarrow aS/bS/a/b$

2) Convert the following PDA to CFG
$M = (\{q_0, q_1\}, \{0,1\}, \{X, Z_0\}, \delta, q_0, Z_0, \phi)$ and δ is given by, [NOV/DEC '18]

NOTES

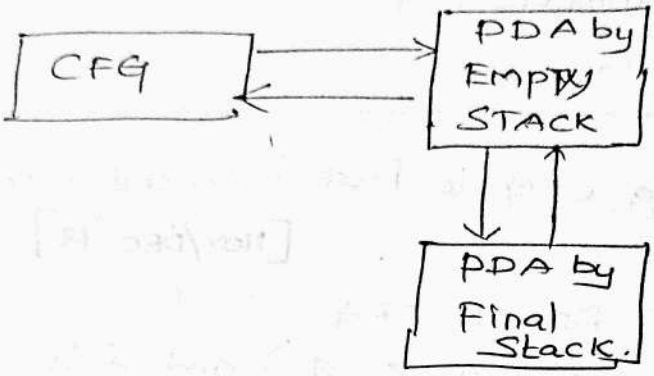
$\delta(q_0, 0Z_0) = \{(q_0, XZ_0)\}$, $\delta(q_1, 1, X) = \{(q_1, \epsilon)\}$
$\delta(q_0, 0, X) = \{(q_0, XX)\}$, $\delta(q_1, \epsilon, X) = \{(q_1, \epsilon)\}$
$\delta(q_0, 1, X) = \{(q_1, \epsilon)\}$, $\delta(q_1, \epsilon, Z_0) = \{(q_1, \epsilon)\}$.

3) Find PDA that accept the given CFG
$S \rightarrow XaaX$
$X \rightarrow aX/bX/\epsilon$ [NOV/DEC '17]

EQUIVALENCE OF PDA AND CFG

3 CLASSES OF LANGUAGE

1) Context free Language (CFL) ⇒ ie, Language defined by CFG
2) Languages accepted by final state of PDA
3) Languages accepted by Empty stack of PDA



So,
Convert
CFG to PDA
PDA to CFG

FROM CFG TO PDA (THEOREM)

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

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

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

Where δ is defined by,

Rule 1) for each Variable A,

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

Rule 2) for every terminal symbol a,

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

EXAMPLES

1. Convert the grammar CFG to a PDA

$E \rightarrow E+E$

$E \rightarrow id.$

Solution

The equivalent PDA is,

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

Where δ is defined by,

$\delta(q, \varepsilon, E) = \{(q, E+E), (q, id)\}$ ⇒ Rule 1

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

$\delta(q, +, +) = \{(q, \varepsilon)\}$ } Rule 2

Now, Process the input id+id+id in N(P)

$\omega = id+id+id$

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

$\vdash (q, \underline{id}+id+id, \underline{id}+E)$

$\vdash (q, \underline{+}id+id, \underline{+}E)$

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

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

$\vdash (q, \underline{id}+id, \underline{id}+id)$

$\vdash (q, \underline{+}id, \underline{+}id)$

$\vdash (q, id, id)$

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

$\therefore id+id+id \in N(P)$

2) Convert the grammar

$S \to 0S1 / A$

$A \to 1A0 / S / \varepsilon$ into a PDA that accepts the same language by empty stack. Check whether 0101 belongs to N(P).

Solution: state ($\Downarrow$), Σ ($\Downarrow$), Γ ($\Uparrow$), starting state ($\Downarrow$)

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

$\Gamma \hookrightarrow$ stack

Where δ is defined by,

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

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

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

$\delta(q, 1, 1) = \{(q, \varepsilon)\}$ } from Rule 2

W = 0101

$\delta(q, 0101, S)$

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

$\vdash (q, 101, S1)$ $[\because S = 0S1]$

$\vdash (q, 101, A1)$ $\Leftarrow [\because S \to A)$

$\vdash (q, 101, 1A01)$ [$\because A \to 1A0$]

$\vdash (q, 01, A01)$ [$\because A \to \varepsilon$]

$\vdash (q, 01, 01)$

$\vdash (q, 1, 1)$

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

$\therefore$ The input 0101 $\in N(P)$.

3) Construct a PDA equivalent to CFG

$S \to 0BB$

$B \to 0S / 1S / 0$

Solution

The PDA can be written as,

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

Where δ is given by

$\delta(q, \varepsilon, S) = \{(q, 0BB)\}$ [$\because S \to 0BB$]

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

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

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

Let us take input $w = 001000$

$\delta(q, 001000, S) \vdash (q, 001000, S)$

$\vdash (q, 001000, 0BB)$

$\vdash (q, 01000, BB)$ [$\because B \Rightarrow 0$]

$\vdash (q, 01000, 0B)$

$\vdash (q, 1000, B)$

$\vdash (q, 1000, 1S)$ [$\because B \to 1S$]

$\vdash (q, 000, S)$

$\vdash (q, 000, 0BB)$ [$\because S \to 0BB$]

$\vdash (q, 00, 0B]$ [$\because B \to 0$]

$\vdash (q, 0, B)$ [$\because B \to 0$]

$\vdash (q, 0, 0)$

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

$\therefore$ The input 001000 is accepted.

$\therefore 001000 \in N(P)$

$\Downarrow$

(Pronounce $\Rightarrow$ belongs to)

PROBLEMS FOR PRACTICE

(CFG to PDA)

Note $\Rightarrow$ Take Input based on your choice (But it depends on the grammar)

1) Construct the PDA for the following grammar

$E \to E+E \mid E*E \mid a$

Hint $\Rightarrow$ input = E+E*a, a*a+a

2) Generate the PDA for the following grammar

$E \to E+E \mid E*E \mid E-E \mid E/E \mid I$

$I \rightarrow a/b/Ia/Ib/Io/I1$

Hint $\Rightarrow$ Input = a * a + boo

3) Convert the grammar

$S \rightarrow 0S1/A$

$A \rightarrow 1A0/S/\epsilon$

To a PDA that accepts the same language by empty stack.

Hint $\Rightarrow$ Input $\Rightarrow$ 0101

4) Convert the grammar to PDA by empty stack.

$S \rightarrow aAA$

$A \rightarrow aS/bS/a$

Hint $\Rightarrow$ Input $\Rightarrow$ aabaaa

5) Construct the PDA for the grammar

$S \rightarrow aB/bA$

$B \rightarrow b/bS/aBB$

$A \rightarrow a/aS/bAA$

Hint $\Rightarrow$ Input $\Rightarrow$ bbaaba

6) Convert the grammar to PDA by empty stack.

$S \rightarrow AA/0$

$A \rightarrow SS/1$

Hint $\Rightarrow$ Input $\Rightarrow$ 110011

7) Construct the context free grammar for the language. $\{a^n b^m c^{2(n+m)} / n \geq 0, m \geq 0\}$ and use the grammar to construct PDA.

Hint $\Rightarrow$ Input $\Rightarrow$ abccCC

PAGE No. 70/16

8) Construct the CFG for $L = \{0^n 1 0^n \mid n \geq 0\}$ and use it to construct PDA.

Hint ⇒ Input ⇒ 0001000

FROM PDA TO CFG

THEOREM

Suppose M is a PDA. Then there is a grammar G such that $L(G) = L(M)$ is context free.

PROOF

Given PDA $M = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, \phi\}$

Then Construct grammar G such that $L(G) = L(M)$

Rules for convert PDA to CFG

Rule 1

The Productions for the start symbol S are given by,

$S \rightarrow [q_0, z_0, q]$ for each state q in Q

Rule 2

for each move, it pops a symbol from the stack with transition.

$\delta(q, a, z_0) = (q_1, \varepsilon)$

$[q, z_0, q_1] \rightarrow a$ [for q in Q]

Rule 3

for each move that doesnot pop a symbol from the stack

Then the transition function is,

$$\delta(q, a, z_0) = (q_1, z_1 z_2 z_3 z_4 \dots)$$

Then Productions are,

$$[q, z_0, q_m] \rightarrow a[q_1, z_1, q_2]\,[q_2, z_2, q_3]\,[q_3, z_3, q_4] \dots\dots [q_{m-1}, z_m, q_m]$$

for each state q_m in Q.

After applying all rules, apply simplification of grammar to get the reduced grammar.

$\therefore L(G) = L(M)$

EXAMPLES

1) Give the equivalent CFG for the following PDA

$M = \{(q_0, q_1), (a, b), (z, z_0), \delta, q_0, z_0, \phi\}$

where δ is defined 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

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

STEP 1 Productions

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

STEP 2

$T \Rightarrow \{0, 1\}$ $\quad \vdash \Rightarrow \{z_0, z\}$

$S = S$

States $= \{q_0, q_1\}$

STEP 4 PRODUCTIONS OF CFG

Starting ~~Symbol~~ State

Rule 1

$P_1 : S \rightarrow [q_0, z_0, q_0]$

$P_2 : S \rightarrow [q_0, z_0, q_1]$

$P_3 : S \rightarrow [q_0, z, q_0]$

$P_4 : S \rightarrow [q_0, z, q_1]$

Rule 1

S [initial state, Stack symbol, final state]

S [q_0 , ~~z~~ z_0 (q_0, q_1)]

2 symbol $\Rightarrow z, z_0$

$2^n = 2^2 = 4$ states

STEP 5

Statement 1

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

Question $\delta(q_0, b, z_0) = (q_0, zz_0)$

P_3: $\delta[q_0, z_0, q_0] = b[q_0, z, q_0][q_0, z_0, q_0]$

P_4: $\delta[q_0, z_0, q_0] = b[q_0, z, q_1][q_1, z_0, q_0]$

P_5: $\delta[q_0, z_0, q_1] = b[q_0, z, q_0][q_0, z_0, q_1]$

P_6: $\delta[q_0, z_0, q_1] = b[q_0, z, q_1][q_1, z_0, q_1]$

Statement 2

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

P_7: $\delta[q_0, z_0, q_0] \rightarrow \varepsilon$

Rule 2

$\delta(q, a, z_0) = (q_1, \varepsilon)$

Statement 3

$\delta[q_0, b, z) = (q_0, zz)$

Rule 3 (Refer Pg-)

P_7:

$\delta[q_0, z, q_0] \rightarrow b[q_0, z, q_0][q_0, z, q_0]$

P_8: $[q_0, z, q_0] \rightarrow b[q_0, z, q_1][q_1, z, q_0]$

P_9: $[q_0, z, q_1] \rightarrow b[q_0, z, q_0][q_0, z, q_1]$

P_{10}: $[q_0, z, q_1] \rightarrow b[q_0, z, q_1][q_1, z, q_1]$

Statement 4 — Follow Rule 3

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

Since $z = 1$ variable, $2^1 = 2$ states

P_{11}: $\delta(q_0, z, q_0) \rightarrow a\,[q_1, z, q_0]$

P_{12}: $\delta(q_0, z, q_1) \rightarrow a\,[q_1, z, q_1]$

Statement 5 — Follow Rule 2

P_{13}: Given $\delta(q_1, b, z) = (q_1, \epsilon)$

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

Statement 6

P_{14}: $\delta(q_1, a, z_0) = (q_0, z_0)$

Since $z_0 \Rightarrow$ single variable $\therefore 2^1 = 2$ states

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

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

Step 5

$\therefore S = z_0$ //

---

2) Let $A = (\{q_0, q_1\}, \{0, 1\}, \{S, A\}, \delta, q_0, S, \phi)$

Where δ is given by,

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

$\delta(q_0, \epsilon, S) = \{(q_0, \epsilon)\}$

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

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

$\delta(q_0, 1, A) = \{(q_1, \epsilon)\}$

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

Construct CFG G for PDA A.

Solution

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

STEP 1

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

STEP 2

$T = \{0, 1\}$

STEP 3

Production for S

P_1 : $S \rightarrow [q_0, S, q_0]$

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

P_2 : $S \rightarrow [q_0, A, q_0]$

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

Rule 1

δ [initial state, Stack Symbol, final State]

q_0 S, A (q_0, q_1)

STEP 4

Statement No. 1

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

Use Rule 3

$n = 2, 2^2 = 4$ states.

P_3: $\delta[q_0, S, q_0] \rightarrow 1[q_0, A, q_0][q_0, S, q_0]$

P_4: $\delta[q_0, S, q_0] \rightarrow 1[q_0, A, q_1][q_1, S, q_0]$

P_5: $\delta[q_0, S, q_1] \rightarrow 1[q_0, A, q_0][q_0, S, q_1]$

P_6: $\delta[q_0, S, q_1] \rightarrow 1[q_0, A, q_1][q_1, S, q_1]$

STATEMENT NO. - 2

$\delta(q_0, \epsilon, S) = \{(q_0, \epsilon)\}$ $\Rightarrow$ Use Rule 2

P_7: $\delta(q_0, S, q_0) \longrightarrow \epsilon$

STATEMENT NO.-3

$\delta(q_0, 1, A) = \{(q_0, AA)\}$ $\Rightarrow$ Rule 3

P_8: $\delta(q_0, A, q_0) \rightarrow 1[q_0, A, q_0][q_0, A, q_0]$

P_9: $\delta(q_0, A, q_0) \rightarrow 1[q_0, A, q_1][q_1, A, q_0]$

P_{10}: $\delta(q_0, A, q_1) \rightarrow 1[q_0, A, q_0][q_0, A, q_1]$

P_{11}: $\delta(q_0, A, q_1) \rightarrow 1[q_0, A, q_1][q_1, A, q_1]$

STATEMENT NO. 4

$\delta(q_0, 0, A) = \{(q_1, A)\}$ Use Rule 3

P_{12}: $\delta[q_0, A, q_0] = 0[q_0, A, q_0]$

P_{13}: $\delta[q_0, A, q_1] = 0[q_0, A, q_1]$

States $\Rightarrow$ 1

$2^1 = 2$ Productions.

STATEMENT NO: 5

Follow Rule 2

$\delta(q_0, 1, A) = \{(q_1, e)\}$

P_{14} : $\delta[q_0, A, q_1] \rightarrow 1$

STATEMENT NO: 6

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

Follow Rule 3

P_{15} : $\delta[q_1, A, q_0] \rightarrow 0[q_1, S, q_0]$

P_{16} : $\delta[q_1, A, q_1] \rightarrow 0[q_1, S, q_1]$

STEP 5

$S = S$ //

PROBLEMS FOR PRACTICE

1) Construct a CFG G which accept N(A) Where,

$A = (\{q_0, q_1\}, \{a, b\}, \{z_0, z\}, \delta, q_0, z_0, \phi)$,

δ is given by,

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

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

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

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

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

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

2) Construct a CFG 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,

1) $\delta(q_0, b, z_0) = \{(q_0, zz_0)\}$
2) $\delta(q_0, \epsilon, z_0) = \{(q_0, \epsilon)\}$
3) $\delta(q_0, b, z) = \{(q_1, zz)\}$
4) $\delta(q_0, a, z) = \{(q_1, z)\}$
5) $\delta(q_0, b, z) = \{(q_1, \epsilon)\}$
6) $\delta(q_1, a, z_0) = \{(q_0, z_0)\}$

3) Convert PDA $P = (\{p, q\}, \{a, b\}, \{x, z_0\}, \delta, q, z_0)$ to a CFG, if δ is given by,

1) $\delta(q, b, z_0) = \{(q, xz_0)\}$
2) $\delta(q, b, x) = \{(q, xx)\}$
3) $\delta(q, a, x) = \{(p, x)\}$
4) $\delta(q, \epsilon, x) = \{(q, \epsilon)\}$
5) $\delta(p, b, x) = \{(p, \epsilon)\}$
6) $\delta(p, a, z_0) = \{(q, z_0)\}$

4) Construct CFG for the following PDA

Where δ is given by,

$P = (\{q_0, p\}, \{0, 1\}, \{x, z_0\}, \delta, q, z_0, \phi)$

a) $\delta(q_0, 0, z_0) = \{(q_0, xz_0)\}$
b) $\delta(q_0, 0, x) = \{(q_0, xx)\}$
c) $\delta(q_0, 1, x) = \{(q_1, \epsilon)\}$

TITLE: DETERMINISTIC PUSHDOWN AUTOMATA (DPDA)

OBJECTIVE:

To learn DPDA & how the language is accepted by DPDA.

APPLICATIONS:

+ Parsing
+ The Evaluation of Arithmetic Expression

UNIVERSITY QUESTIONS:

1) What is an instantaneous description (ID) of a Push down Automaton (PDA)? [NOV/DEC '18]

2) What is a deterministic pushdown automaton? Comment on the language accepting capabilities of a deterministic push down NOTES automaton. [NOV/DEC '18]

3) Prove that deterministic and non deterministic PDA are not equivalent. [NOV/DEC '17]

DETERMINISTIC PUSH DOWN AUTOMATA

DPDA's & CFL

* DPDA can accept CFL (LCP))
* It can store necessary input symbols and use the State transitions for remembering Conditions.

DEFINITION (DPPA)

The PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is said to be DPDA.

1) $\delta(q, a, X)$ has atmost one member for any q in Q, a in ϵ, X in Γ

2) $\delta(q, a, X)$ - Non-Empty state.

   $\delta(q, \epsilon, X)$ - Empty State

3) DPDA - No choice of Moving in the same State by using same input and Stack Symbol.

i.e., L accepted by DPDA is Deterministic Context Free Language.

DPDA AND AMBIGUITY

* The languages that DPDA's accepts the final state have unambiguous grammar.
* Accepts the Empty stack also have Unambiguous grammar.

THEOREM

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

PROOF

DPDA can Simulate DFA

PDA has Symbols Z_0 on the stack.

Let $A = (Q, \Sigma, \delta_A, q_0, F)$ // DFA

Construct

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

Where,

$\delta_P(q, a, Z_0) = (P, Z_0)$ $\forall$ states p and q in Q.

Such that $\delta_A(q, a) = P$

We Conclude that,

$(q_0, \omega, Z_0) \overset{*}{\underset{P}{\vdash}} (P, \varepsilon, Z_0)$

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

i.e., P can Simulate A.

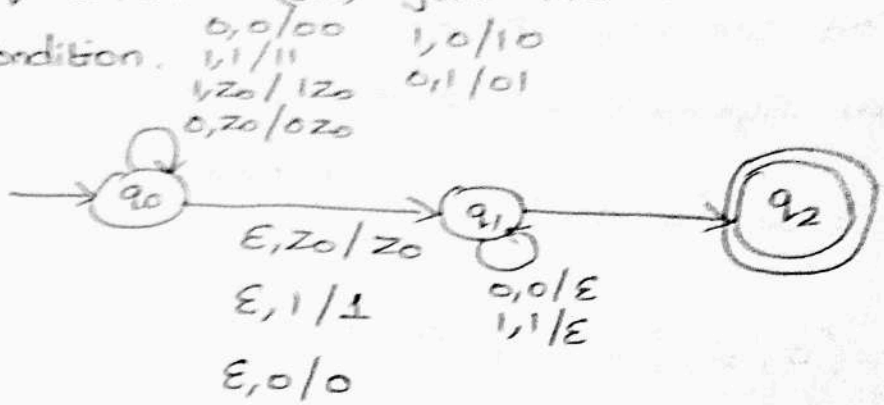
∴ A & P are accepting the final State, F

∴ Their languages are Same.

$$\boxed{L = L(P)}$$

EXAMPLES

1) Check whether the following PDA is deterministic or not. Justify it by using the conditions of DPDA (OR) find the rules that violates for Condition.



SOLUTION

RULES

1) $\delta(q,a,x)$ - has atmost one member for any q.

2) $\delta(q,a,x)$ - Non empty then

$\delta(q,\epsilon,x)$ - Empty.

Rule1 => Transitions of given PDA Satisfies DPDA.

Rule 2 Transitions

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

$\delta(q_0,\epsilon,Z_0) = \{(q_1,Z_0)\}$

2) $\delta(q_0,1,Z_0) = \{(q_0,1Z_0)\}$

$\delta(q_0,\epsilon,Z_0) = \{(q_1,Z_0)\}$

3) $\delta(q_0,1,1) = \{(q_0,11)\}$

$\delta(q_0,0,1) = \{(q_0,01)\}$

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

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

$\delta(q_0,1,0) = \{(q_0,10)\}$

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

∴ In above transitions,

$\delta(q,a,X)$ $\forall q$ in Q & a in Σ, X in Γ is not empty &

$\delta(q,a,X)$ - Not Empty

∴ PDA is not DPDA.

2) Check whether the following PDA is deterministic

i) $\delta(q,0,z_0) = \{(q,Xz_0)\}$

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

iii) $\delta(q,1,X) = \{(q,X)\}$

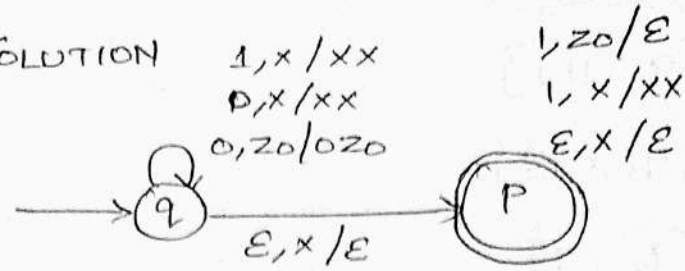
iv) $\delta(q,\epsilon,X) = \{(p,\epsilon)\}$

v) $\delta(p,\epsilon,X) = \{(p,\epsilon)\}$

vi) $\delta(p,1,X) = \{(p,XX)\}$

vii) $\delta(p,1,z_0) = \{(q,\epsilon)\}$

SOLUTION



According to Rule 1, It has only one transition for the input symbol and stack symbol.

Rule 2

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

$\delta(q, \epsilon, x) = \{(p, \epsilon)\}$

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

} ⇒ Non Empty State

2) $\delta(p, 1, x) = \{p, xx\}$

$\delta(p, \epsilon, x) = (p, \epsilon)$

} Non Empty State

∴ The PDA is not DPDA

3) Construct a DPDA that accepts

$L = \{0^n 1^m ; n < m \text{ and } n, m \geq 1\}$

SOLUTION

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

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 given by,

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

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

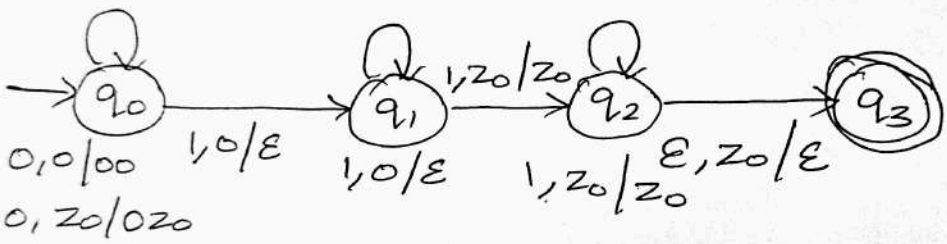
$$\delta(q_0, 1, 0) = \{(q_1, \varepsilon)\}$$

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

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

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

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



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

* $\delta(q, a, x) \rightarrow$ Non Empty State

  $\delta(q, \epsilon, x) \rightarrow$ Empty State.

3.2.2 Inference, Derivations and Parse Trees

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

The following are equivalent with derivations and 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. $A \underset{lm}{\overset{*}{\Rightarrow}} w$
4. $A \underset{rm}{\overset{*}{\Rightarrow}} w$
5. There is a Parse tree with root A and yield w.

The equivalence is shown using the diagram given below.

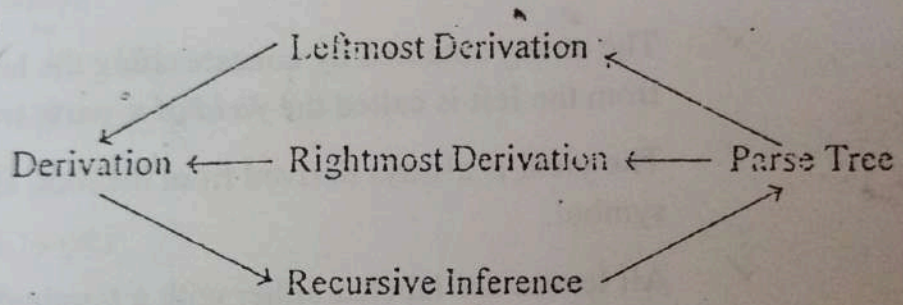


Fig. 3.2 Equivalence Diagram.

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.

3.2.3 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

$A \rightarrow a_1, a_2, \ldots a_n$ [$\because a_1, a_2, \ldots a_n$ – Terminals]

or $A \rightarrow \varepsilon$

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.

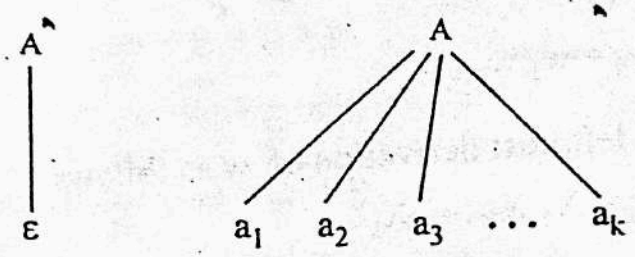


Fig. 3.3 Parse tree of one step derivation.

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 such that

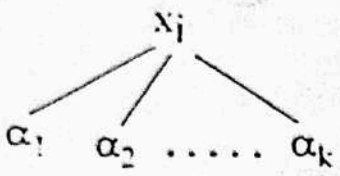
$w = w_1, w_2 \ldots w_k$

Here we have two cases,

(i) If x_i is a terminals, 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) $A \overset{*}{\Rightarrow} w_1, w_2 \ldots w_k$

$\overset{*}{\Rightarrow} w$

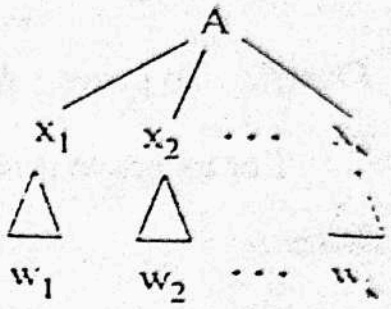


Fig 3.4 Parse Tree

Thus the yield is obtained by concatenating the sub–trees from left to right.

Thus the theorem is proved.

3.2.3 From Trees to Derivations

(Nov/Dec 2003, Nov/Dec 2004 , Apr/May 2005, Nov/Dec 2005)

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

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

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

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 figure below, with root A and yield w.

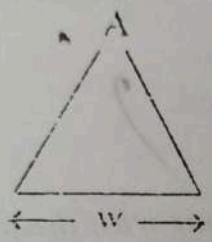


Fig. 3.4 Parse tree of height 1.

This is possible only with the production.

$$A \to w \text{ in G.}$$

$\therefore$ $A \underset{lm}{\Rightarrow} w$ is an one step leftmost derivation.

Induction :

If the height of the parse tree is n. the parse tree must look like in fig. 3.5

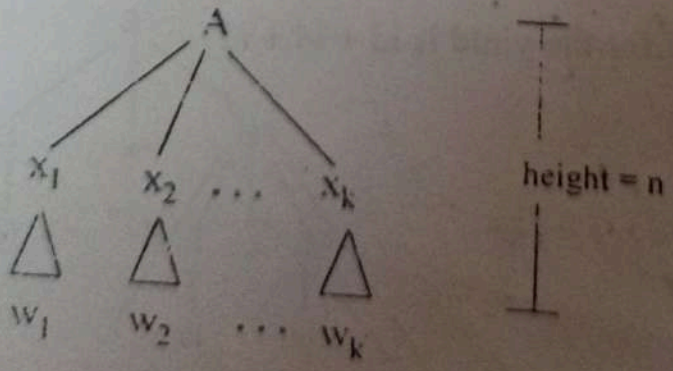


Fig. 3.5 Parse tree with height n.

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.

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

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, . . . k in order, we show that

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

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

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

(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

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

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

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

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

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 parse tree to the rightmost derivation.

Example *Consider the Grammar with the productions*

$$E \rightarrow E * E$$

$$E \rightarrow id$$

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

Solution :

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

3.2.4 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 sho like this,

$$A \Rightarrow X_1 X_2 \ldots X_k$$
$$\overset{*}{\Rightarrow} w$$

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 $X_i \overset{*}{\Rightarrow} w_i$.

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

Thus the theorem is proved.

3.2.5 Applications of CFG

1. Parsers
2. YACC parser – Generator
3. Markup Languages
4. XML and DTD