16/09/17

Module - 3

Pumping Lemma for Regular Language

Applications – 5 marks

1. pumping Lemma for RL can be used to prove that the given language is not regular

Q prove that the below language $L=\{a^n b^n / n \geq 1\}$ is not regular

A Step 1: Assume that the given language L is regular. So, it should follow pumping lemma for regular language

Step 2: Select string $w \in L$ & $|w| \geq m$

Step 3: decompose w into x y z such that $|xy| \geq 0 \leq m$ and $|y| > 0$

Consider $w = a^5 b^5 = aaaaabbbbb$.

$w =$ [aaaa]$_x$ [a]$_y$ [bbbbb]$_z$

$|xy| = 5 \leq m$

$|y| > 0$

Step 4 Pump the substring y in w

Select $i=2$, $xy^2z = \underbrace{aaaa}_{6}\underbrace{aa}_{5}\ bbbbb = a^6b^5$

(contradiction) to our assumption in step 1. So, the language L is not regular

Q prove that $\{L = ww^R / \text{where } w \in \{a,b\}^*$ is not regular

A $w =$ baaaba and $w^R =$ abaaab

$ww^R =$ baaabalabaaab

$|w| = 2n$ $|m| = n$ $|xy| = 6$

$= 2 \times 6$ $= 6$

$= 12$ $|w| \geq n$

$w =$ [baaab]$_x$ [a]$_y$ [abaaab]$_z$

$|xy| = 6 \leq m$

$|y| > 0$

Select $i = 2$

$xy^2z =$ baaaab $\underline{aa}$ $\underline{abaaab}$

x y x

(contradiction) to our assumption in step 1. So, the language L is not regular.

Grammar (G):- 20/9/17.

A grammar (G) can be formally defined by a 4 tuple.

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

where,

V = Set of Variable or Non terminals.

T = Set of Terminal Symbols (Σ)

S = Starting Symbol of the production

P = production rules of the form $(\alpha \to \beta)$ where $\alpha \in \beta$, $\beta \in (V \cup T)^*$

eg: Grammar corresponds to the language $L = \{a^n b^n / n \geq 1\}$ can be represented as

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

$V = \{S, A, B\}$

$T = \{a, b\}$

$S = S$

$P = \begin{cases} S \to AB & -① \\ A \to Aa & -② \\ A \to \lambda & -③ \\ B \to Bb & -④ \\ B \to b & -⑤ \end{cases}$

Sample derivation of string 'ab'

$S \overset{①}{\Rightarrow} AB$ using $S \to AB$

$\overset{②}{\Rightarrow} aB$ $A \to a$

$\overset{⑤}{\Rightarrow} ab$ $B \to b$.

Context free Grammar:-

A grammar $G = (V, T, S, P)$ is said to be context free if all production in P are of the form $\alpha \to \beta$. where $\alpha \in V$ and $\beta \in V \cup T^*$

eg: ~~CFG~~ CFG corresponds to the language ~~L = {ww^R}~~

$L = \{ww^R / w \in \{a,b\}^+\}$

where $V = \{S\}$

$T = \{a, b\}$

$S = S$

$P = \begin{cases} S \to asa & -① \\ S \to bSb & -② \\ S \to \varepsilon & -③ \end{cases}$

derivation of string abbacbba

$S \overset{①}{\Rightarrow} asa$

$\overset{②}{\Rightarrow} abSba$

$\overset{②}{\Rightarrow} abbasabba$

25/9/17.

Q Construct context free grammar corresponding to the language $L = \{w \# w^R / w \in (a,b)^*\}$

A) $G_1 = \{V, T, S, P\}$

$V = \{S\}$

$T = \{a, b, \#\}$

$S = S$

$P = \begin{cases} S \to asa \mid bsb \mid \# & -(1) \\ S \to \# & -(2) \\ S \to asa & -(3) \\ S \to bsa & -(4) \end{cases}$

$S \overset{(1)}{\Rightarrow} asa \overset{(1)}{\Rightarrow} aasaa \overset{(2)}{\Rightarrow} aabsbaa \overset{(2)}{\Rightarrow} aab\#baa.$

Q Construct CFG for the ~~gram~~ language

$L = \{w / n(a) = n(b)\}$

Q Generation of derivation tree / Parsetree

A derivation tree is an ordered rooted tree that graphically represents semantic information of a string derived from a context free grammar.

→ Representation technique.

1) Root vertex must be labelled by starting symbol

2) vertex: intermediate vertex labelled by variables

3) Leaves: labelled by terminal symbol or ε

→ Derivation of yield of a tree

The derivation yield of a parse tree is the final string obtained by concatinating leaves of a tree from left to right.

eg: Let a CFG, $G_1 = \{V, T, S, P\}$

$V = \{S\}$

$T = \{a, b\}$

$S = \{S\}$

$P = \begin{cases} S \to SS & -(1) \\ S \to asb & -(2) \\ S \to \varepsilon & -(3) \end{cases}$

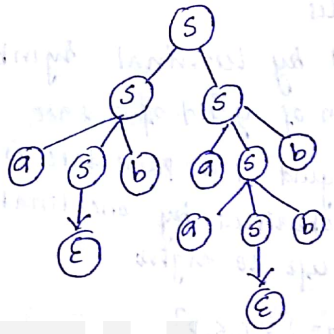
? Find its derivation tree for the string abaabb.

Derivation of abaabb.

$S \overset{(2)}{\Rightarrow} asb \overset{(1)}{\Rightarrow} absbb \rightarrow$

$S \xRightarrow{1} SS \xRightarrow{2} aSbS \xRightarrow{3} abS \xRightarrow{4} abaSb \xRightarrow{2} abaaSbb \xRightarrow{3} abaabb$.

Derivation Tree

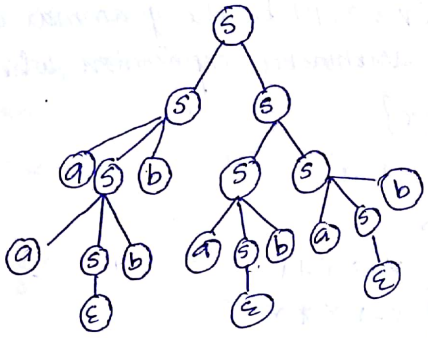


abaabb

(Q:2) aabbabab

~~$S \xRightarrow{4} aSb \xRightarrow{2} aaSbb \xRightarrow{1} aaSSbb$~~

$S \xRightarrow{1} SS \xRightarrow{2} aSbS \xRightarrow{2} aaSbbS \xRightarrow{3} aabbS \xRightarrow{3} aabbSS \xRightarrow{2} aabbaSbS \xRightarrow{3} aabbabS \xRightarrow{2} aabbabaSb \xRightarrow{3} aabbabab$



3/10/17

Leftmost & Rightmost derivation.

Leftmost derivation.

It is obtained by applying production to the leftmost variable in each step. Leftmost derivation it is denoted by using the symbol $\xRightarrow{lm}$

Rightmost derivation.

A rightmost derivation is obtained by applying production to each variable. Rightmost derivation is denoted by using the symbol $\xRightarrow{rm}$.

Let $G = (V, S, T, P)$ be the grammar corresponding to arithmetic expression where

$V = \{X\}$

$T = \{+, *, a\}$

$S = X$

$P = \begin{cases} X \to X + X \\ X \to X * X \\ X \to a \end{cases}$

Leftmost derivation for the string

$w = a * a + a$

Rightmost derivation for the string

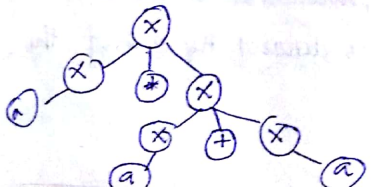
$w = a * a + a$

A) Leftmost

$X \underset{lm}{\overset{②}{\Longrightarrow}} X * X \underset{lm}{\overset{③}{\Longrightarrow}} a * X \underset{lm}{\overset{①}{\Longrightarrow}} a * X + X$

$\underset{lm}{\overset{③}{\Longrightarrow}} a * a + X \underset{lm}{\overset{③}{\Longrightarrow}} a * a + a.$

Leftmost derivation tree.

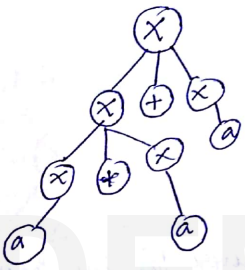


Rightmost

$X \underset{rm}{\overset{①}{\Longrightarrow}} X + X \underset{rm}{\overset{③}{\Longrightarrow}} X + a \underset{rm}{\overset{②}{\Longrightarrow}} X * X + a \underset{rm}{\overset{③}{\Longrightarrow}}$

$X * a + a \underset{rm}{\overset{③}{\Longrightarrow}} a * a + a$

Rightmost derivation tree



Since 5 marks

Ambiguity in CFG

A grammar $G = \{V, S, T, P\}$ is said to be ambiguous if some string w element of $L(G)$ has more than one derivation tree.

A grammar $G = (V, S, T, P)$ is said to be ambiguous for some string w element of $L(G)$ has more the one left most

derivation tree
OR
Morethan on ~~te~~ rightmost derivation tree

eg Let $G = (V, S, T, P)$ be the grammar corresponding to arithmethic expression where

$V = \{X\}$

$T = \{+, *, a\}$

$S = X$

$P = \begin{cases} X \\ X \\ X \end{cases}$

It is ambiguous because for the same string $W = a*a+a$ has more than one rightmost derivation

Rightmost derivation-1

$X \underset{rm}{\overset{①}{\Rightarrow}} X+X \underset{rm}{\overset{③}{\Rightarrow}} X+a \underset{rm}{\overset{②}{\Rightarrow}} X*X+a \overset{③}{\Rightarrow}$

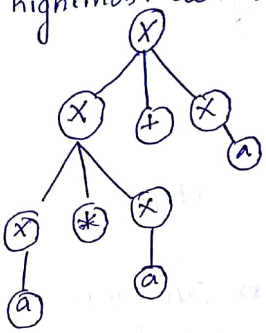
$X*a+a \underset{rm}{\overset{①}{\Rightarrow}} a*a+a.$

Rightmost derivation-2

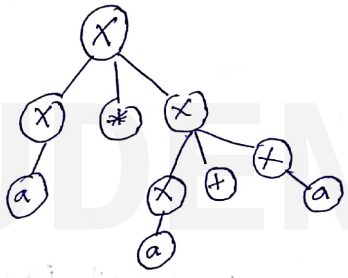
$X \underset{rm}{\overset{①}{\Rightarrow}} X*X \overset{①}{\Rightarrow} X*X+X \underset{rm}{\overset{②}{\Rightarrow}} X*X+a \underset{rm}{\overset{③}{\Rightarrow}}$

$X*a+a \Rightarrow a*a+a.$

Rightmost derivation tree-



Rightmost derivation tree - 2



Inherently Ambiguous ~~Grammar~~:

If every grammar ~~inherently ambigou~~ that ~~Grammar~~ generates L is ambigous then the language L is called Inherently Ambiguous Language

eg:- The language $L = \{a^n b^n c^m\} \cup \{a^n b^m c^m\}$ with n and m non-negative is an inherently

ree language
lemma for

ing

3. Simplification of CLF

Removal of useless production

A variable is useful if and only if it occurs in atleast one derivation. A variable ie is not useful is called useless.

Let $G_1 = \{V, T, S, P\}$ be a context free Grammar a variable $A \in V$ is said to be useful if & only if there is atleast one $\omega \in L(G_1)$ such that

$$S \overset{*}{\Longrightarrow} xAy \Rightarrow \omega \text{ with } x, y \text{ in } (V \cup T)^*$$

eg Eliminate useless symbol & production from &

$G_1 = \{V, T, S, P\}$ where $V = \{S, A, B, C\}$ and $T = \{a, b\}$

and $P = \begin{cases} S \to aS \mid A \mid C \\ A \to a \\ B \to bb \\ C \to acb \end{cases}$

: ✗ Identify variables that directly derives variables or string

✗) $A \to a$

$B \to bb$

Step 2: check whether they are derived from starting symbol of the production.

$S \Rightarrow A \Rightarrow a$

So, the productions

$S \to A$
$A \to a$ } are useful

So, $B \to bb$ is a useless product^{n}

3: check whether any other product^{n} is useless

- Set of product^{n} rules after removing useless production.

→ Removing epsilon productions :-

Any production of CFG of the form $A \to \varepsilon$ is called ε production

eg: Remove ε production from below grammar.

$P = \{ S \to aS_1b$
$S_1 \to aS_1b \mid \varepsilon$

Now Set of productions contains ε production.

Substitute ε in RHS of all production that contains S_1

New rules

$S \to ab$
$S_1 \to ab$

Production rules after removing ε production

$S \to aS_1b \mid ab$
$S_1 \to aS_1b \mid ab$

→ Removing Unit Production

Any Production of a CFG of the form $A \to B$ where A, B element of E is called unit production

eg: Remove unit production from $S \to Aa \mid B$
$B \to A \mid bb$
$A \to a \mid bc$

Identify unit production and Remove it

$S \to B$
$B \to A$
$B \to a \mid bc \mid bb$
$S \to Aa \mid a \mid bc \mid bb$
$A \to a \mid bc$.

Simplified CFG

A simplified CFG doesn't contains

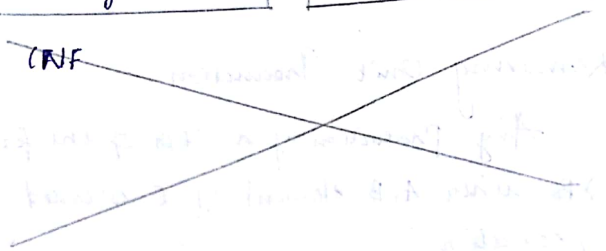
1. Useless symbol
2. Epsilon production
3. Unit production.

NORMAL FORMS

CNF
Chomsky Normal form

GNF
Greibach Normal Form.

CNF



Chomsky

A CFG is in CNF is all productⁿs are of the form.

$A \to BC$ ~~where~~ or $A \to a$

where $A, B, C \in V$ and $a \in T$.

- Convert below grammar into chomsky normal form.

$S \to ABa$ — (1)
$A \to aab$ — (2)
$B \to Ac$ — (3)

Steps to convert CFG → CNF

1: Remove useless production
2: Remove Epsilon production.
3. Remove Unit production.
4. Then convert simplified grammar into CNF

- Check useless production.
  Here all productⁿ are useful
- Remove Epsilon production
  no " "
- Remove Unit production
  no Unit production

∴ Here given production is simplified

(4) Convert simplified CFG to CNF.

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

introduce new productions below.

$X_a \to a$ — (1)
$X_b \to b$
$X_c \to c$

Check the production rule (1) is in CNF
$S \to ABa$ (a is not in CNF

$S \to ABX_a$ $[X_a \to a]$

$S \to AD_1$
$D_1 \to BX_a$ $[D_1 \to BX_a]$

$A \to aab$
$A \to X_a X_a X_b$.

$A \rightarrow X_1 D_2$

$D_2 \rightarrow X_1 X_6$

$B \rightarrow AC$

$B \rightarrow A X_c$

Convert the below grammar

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid a$

$B \rightarrow aBB \mid b$

Greibach Normal Form

A CFG is said to be in Greibach normal form if all productions are of the form $A \rightarrow a\alpha$ where $a \in T$, $A \in V$ and $\alpha \in V^*$

Removal of left Recursion.

A grammar of the form $A \rightarrow A\alpha \mid \beta$ where $A \in V$ and α & $\beta \in V^*$ is called left recursive Grammar.

To remove left recursion in grammar $A \rightarrow A\alpha \mid A\alpha_2 \mid \ldots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m$

we introduce following Rule

1) $A \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n \mid \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_m A'$
2) $A' \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n \mid \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_n A'$