Unit: 3

Context free grammars & Languages

09/01/23

Components of Grammar

→ Variables → non terminals which are used to identify grammar in the lang

→ Terminals

→ Productions → Production will be shown with →

→ Start symbol

L.H.S R.H.S

$\alpha \longrightarrow \beta (VUT)^*$

non-terminals → non-terminal / denoted using S' terminals

Grammar: A grammar G is four tuple or quadrpule $G = (V, T, P, S)$

where, V is the set of variables / non-terminals

T set of terminals

P set of productions

Each production is of the form $\alpha \longrightarrow \beta$

where, α is a string in $(VUT)^+$

and hence, α cannot be ϵ and epsilon cannot occur on the R.H.T of any production.

β is a string in $(VUT)^*$ & hence, it includes ϵ also.

S is the start symbol.

Start

__/__/__

Terminals

→ Can have keyword
→ Any value ranging from 0 to 9.
→ Symbols +, -, /, % etc
→ Lower case letters at the beginning. a, b, c, ...
→ Bold letters **id**

Non-terminals

→ Capital letters
→ Letter S has to used as the start symbol.
→ Lower case can be used for statements.

if ∈ then $L(R) = a^*$

$S \rightarrow aS \mid \epsilon$ $\quad L(R) = a^+$

S: non-terminal; a: terminal; S: non-terminal

Chomsky hierarchy

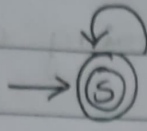
Type 0 (Phase structured grammar)
Type 1 (Context sensitive grammar)
Type 2 (Context free)
Type 3 (Regular)

Context free G eg:

$S \rightarrow aB \mid \epsilon \mid aA$
$A \rightarrow aS$
$B \rightarrow aB$

__/__/____

⇒



$\delta(S, a) = S$

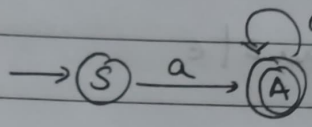
$S \to aS$

accepts a^*.
S can also have
$S \to \epsilon$

$S \to aS \mid \epsilon$

* Can be used to even create ϵ state as aS state.

⇒



$\delta(S, a) = A$

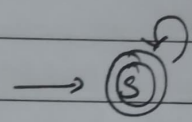
$\delta(A, a) = A$

$S \to aA$

$A \to aA \mid \epsilon$ (OR)

$A \to \epsilon$

⇒



$\delta(S, a) \Rightarrow S$

$\delta(S, b) = S$

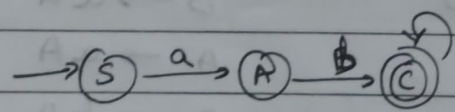
$S \to aS \mid bS \mid \epsilon$

$S \to aS$

$S \to bS$

$S \to \epsilon$

=>



$\delta(S, a) = A$

$\delta(A, B) = C$

$\delta(C, a) = C$

$\delta(C, b) = C$

$S \to aA$

$A \to bC$

$C \to aC$

$C \to bC$

$C \to \epsilon$

__/__/____

-> Obtain a grammar to generate string consisting of even no of a's.

aa

$L = \{w \mid |w| \bmod 2 = 0,\ w \in \Sigma(a)\}$

$L = \{\epsilon, aa, aaaa, aaaaaa \cdots\}$

$S \to \epsilon$
$S \to aaS$ } $S \to aaS \mid \epsilon$

I aaS II aaaaS

=> What is the lang. accepted by → (S) with self-loop a, b are a^* & b^*.

=> Obtain a grammer to generate a string consisting of any no of a's & b's with atleast one a.

~~$L = \{w\}$~~

(S) —a→ (A), self-loop b on S, self-loop a, b on A

$S \to bS$
$S \to aA$
$A \to aA$
$A \to bA$
$A \to \epsilon$

~~$L = \{w \mid w \in \Sigma a$~~

$\delta(S, a) = A$
$\delta(S, b) = S$
$\delta(A, a) = A$
$\delta(A, b) = A$

16/01/23

Q. Obtain the grammar to generate strings of a's and b's containing substring ab.

$(a+b)^*\ ab\ (a+b)^*$ → only ab → n number of a's & b's

ab can be at before the string or in the middle of the string or at the end. Hence, this eq. is considered

consider this as 3 diff equation

$S \rightarrow A\ ab\ A$

A can be any no of strings of a & b

$S \rightarrow AabA$

$A \rightarrow \epsilon \mid aA \mid bA$ (Grammar to generate any no of strings)

last class → (5) a,b

Grammar to accept $(a+b)^*$

Q. Obtain a grammar to generate a string of a's & b's ending with ab

$(a+b)^*\ ab$ → ending with ab

→ a's & b's string

$S \rightarrow A\ ab$

~~$S \rightarrow \epsilon$~~ $S \rightarrow Aab$

$A \rightarrow \epsilon \mid aA \mid bA$

Show an example if asked for 6-7 marks.

__/__/__

Derivation

$A \to \alpha\beta\gamma$ (A produces α, β & γ) & ($B \to \beta$, B produces β)

⇒ Let A produces α, β & γ & (B produces β) are productions in grammar G, where, α, β, γ are strings of terminal and/or non-terminals. A & B are not-terminal.

⇒ The non-terminal A produces the string $\alpha\beta\gamma$ by replacing the non-terminal B in $\alpha\beta\gamma$ By the string β by applying the production B produces β & can be written as $A \Rightarrow \alpha\beta\gamma$.

⇒ This process of obtaining string of terminal & / or non-terminal from the state symbol by applying some / all productions are called derivation.

Q ⇒

$E \to E + E$

$E \to E - E$

$E \to E * E$

$E \to E / E$

$E \to id$

Obtain string $id + id * id$

first we have + so considering this equation

Sol" ⟶

$E \to E + E$

$E \to id + E$ → substituting one E by id (∵ given equation)

$E \to id + (E * E)$ → taking $E \to E * E$

$E \to id + id * E$ & considering one E as id

$E \to id + id * id$

__/__/____

7-8 marks

Q ⇒ $S \to aB \mid bA$ obtain string aaabbabbba

$A \to as \mid bAA \mid a$

$B \to bs \mid aBB \mid b$

Sol ⇒

$S \to aB$

$S \to aaBB$ (Replaced with aBB)

$S \to aaaBBB$ (with aBB)

$S \to aaabbB$ (Both b's with b)

$S \to aaabbaBB$

$S \to aaabbabB$

$S \to aaabbabbS$

$S \to aaabbabbbA$

$S \to aaabbabbbba$

write all the transitions.

aBB
↓
abB

* Left most derivation eg:
* Right most derivation eg: aBB
aBb

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

id+id * id

$E \overset{LM}{\Rightarrow} E+E$

$E \Rightarrow id+E$

$E \Rightarrow id+E*E$

$E \Rightarrow id+id*E$

$E \Rightarrow id+id*id$

Left most

$E \overset{RM}{\Rightarrow} E*E$

$E \Rightarrow E*id$

$E \Rightarrow E+E*id$

$E \Rightarrow E+id*id$

$E \Rightarrow id+id*id$

right most

__/__/____

Parse Trees (Derivation Tree)

(context free)

Let $G = (V, T, P, S)$ be a CFG. The tree is parse tree with the following properties:

-> The root has the label S.

-> Every vertex has a label which is in $(V \cup T \cup \epsilon)$

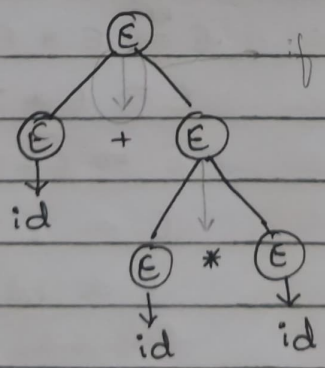
-> Every leaf node has a label from T & an interior vertex has a label from V.

-> If a vertex is labeled as A & if $x_1, x_2 \dots x_n$ are all children from A from the left then $A \to x_1, x_2 \dots x_n$ must be a production in P.

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

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

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



if we want to!

id + id * id → yield of the tree

Partial Parse Trees (Partial Derivation Tree)

Let $G = \{A, T, P, S\}$ be a CFG. The tree is PPT with the following properties:

- The root has the label S
- Every vertex has a label which is in $V \cup T \cup \epsilon$
- Every leaf node has a label from $V \cup T \cup \epsilon$
- If a vertex is labeled A & if $x_1, x_2, \ldots x_n$ are all children of A from left then A produces $x_1, x_2, \ldots x_n$ must be in a production in P.

Ambiguous grammar

which has one or more parse trees

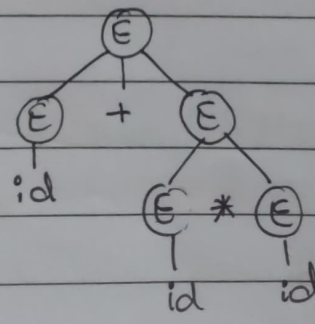
Left most derivation
Right most derivation
] 1+ parse trees

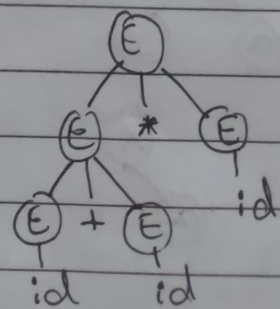
Eg: $E \to E+E \mid E-E \mid E*E \mid E/E \mid id$

$id + id * id$

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

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





more than one parse tree

__/__/____

Q ⇒ $S \to aS \mid X$ aaaa

$X \to aX \mid a$

$S \overset{lm}{\Rightarrow} aS$

$\Rightarrow aaX$

$\Rightarrow aaaX$

$\Rightarrow aaaa$

$X \overset{lm}{\Rightarrow} aX$

$\Rightarrow aaX$

$\Rightarrow aaaX$

$\Rightarrow aaaa$

$S \overset{lm}{\Rightarrow} X$

$\Rightarrow aX$

$\Rightarrow aaX$

$\Rightarrow aaaX$

$\Rightarrow aaaa$

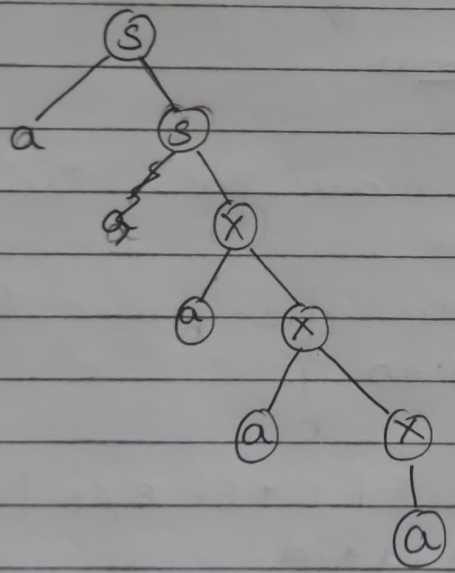
$S \overset{lm}{\Rightarrow} aS$

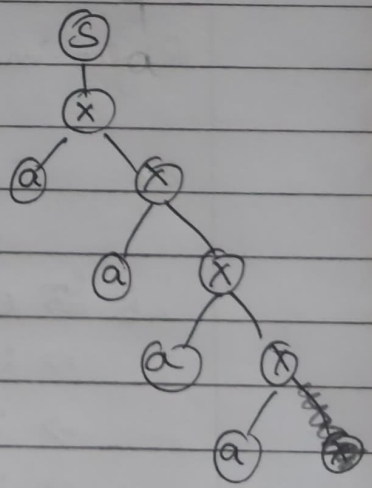
$\Rightarrow aaS$

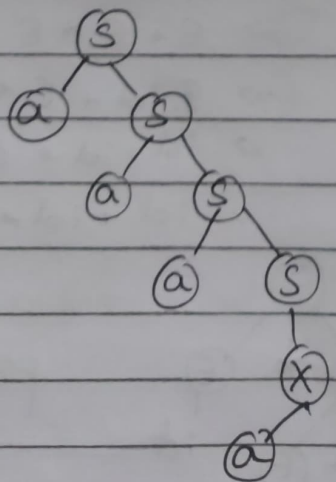
$\Rightarrow aaaS$

$\Rightarrow aaaX$

$\Rightarrow aaaa$







__/__/____

Q ⇒ $S \to AB \mid aaB$ aab.

$A \to a \mid Aa$

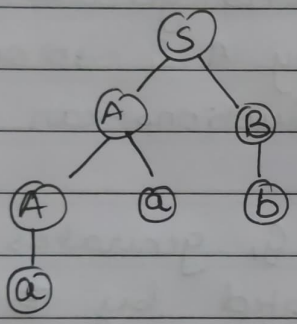
$B \to b$

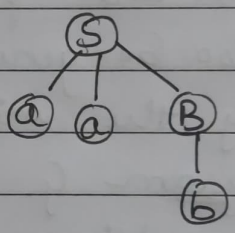
$S \overset{lm}{\Rightarrow} aaB$
$\Rightarrow aab$

$S \overset{lm}{\Rightarrow} AB$
$\Rightarrow AaB \to aaB$
$\Rightarrow aaB$

$S \overset{rm}{\Rightarrow} AB$
$\Rightarrow Ab$
$\Rightarrow Aab$
$\Rightarrow aab$

$S \overset{rm}{\Rightarrow} aaB$
$\to aab$





Theorem **Elimination of unit production**

Let $G = \{V, T, P, S\}$ be a CFG & has unit productions, and no ϵ production. An equivalent grammar G_1 without unit productions can be obtained such that $L(G) = L(G_1)$, but the grammar G_1 has no unit production.

A unit production can be deleted using following steps:

1 Remove all the productions of $A \to A$.

__/__/____

② Add all non unit productions P_1.

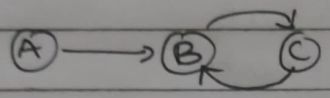
③ For each variable A find all variables B such that $A \overset{*}{\Rightarrow} B$ with 0 or > productions.

④ Obtain a dependency graph

Eg: for $A \to B$

$B \to C$

$C \to B$



⑤ Note from the dependancy grapy that productions from $A \to B$ i.e., B can be obtained by A so all non unit productions genrated by B can also be genrated by A & so on.

⑥ Finally the unit productions can be deleted by gram G.

⑦ The resulting gram G_1 genrates the same Lang as accpted by G.

8 < 10 marks similar

Q ⇒

$S \to AB$ non-unit

$A \to a$

$B \to c \mid b$ $B \to C$ unit $B \to b$ non-unit

$C \to D$ unit

$D \to \epsilon \mid bC$

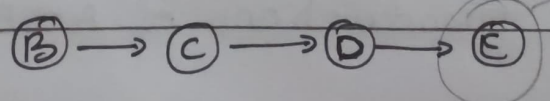
$E \to d \mid Ab$

unit productions: $B \to C$

$C \to D$

$D \to E$

Graph:



→ independent grammar generator

Replacing the value of E in D.

$D \rightarrow d \mid Ab \mid bC$

We with D in eq. C

$C \rightarrow d \mid Ab \mid b\,C$

$B \rightarrow d \mid Ab \mid bC \mid b$

All the unit productions are eliminated

~~$S \rightarrow a \mid dt \mid Ab \mid b$~~

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow d \mid Ab \mid bC \mid b$

$C \rightarrow d \mid Ab \mid bC$

$D \rightarrow d \mid Ab \mid bC$

$E \rightarrow d \mid Ab$.

} have no unit production

$A \rightarrow B$

$B \rightarrow aB$

$C \rightarrow A \mid Bd$ ~~$A \rightarrow B$~~

$BA \quad B \rightarrow aB \quad B \rightarrow C$

$C \rightarrow A$