CONTEXT FREE GRAMMAR (CFG)

Grammar →

$G = (V, T, P, S)$ P = set of production

V_1 = Variable symbols

N = Non-terminal symbols.

T = Terminal symbols.

* Any symbol which can be expanded or be replaced by any other symbol / string is a <u>variable</u>.

P = Rules governing the rules of variable replacement.

S = Starting state.

<u>Notations :</u>

i) Lowercase English alphabet (a, b, c) → Terminal symbols.

ii) Uppercase alphabet (A, B, C) → Variable (S is reserved)

iii) x, y, z → String of terminal symbols.

iv) X, Y, Z → $\underbrace{\text{Single symbol}}_{\text{when we do not know what will be the next variable.}}$ (Variable / Terminal).

v) α, β, γ → Strings of grammar symbols.

α is in $(V \cup T)^*$ (may contain variable or terminal)

vi) Γ, Δ → String of Variables only.

(upper case Greek letter)

$\rightarrow$ = Derives

$$A \rightarrow \alpha \quad (\text{production})$$

read as "Variable A derives the string α"

$$S \rightarrow \underbrace{\cdots}_{\text{Sentential form}} \rightarrow \underbrace{\cdots}_{\text{Sentential form}} \rightarrow \underbrace{\cdots}_{\text{Sentential form}} \rightarrow \alpha$$

General form of production $\rightarrow$

$$\alpha \rightarrow \beta.$$

(String derives string)

If all α's are single variables i.e. all production are of the form $A \rightarrow \beta$, then

$\boxed{\text{G is either regular or CFG.}}$

- G is regular if β consists of ~~single or zero terminal symbols i.e. β is of the form w or wB.~~ a string of only terminals or a string of terminal followed by a variable

β = w or wB (right linear)

β = w or Bw (left linear)

$\rightarrow$ Linear grammar.

* CFG can be anything. * *

If $\alpha \to \beta$, length of α in every production < length of β is called as context free grammar.

$$G = (\{S\}, \{0,1\}, P, S)$$

$$P: \quad S \to 0S1$$
$$S \to \epsilon$$

$\therefore$ The language accepted by this grammar $= 0^n 1^n$.

~~and~~ instead of writing; $A \to \beta_1$; $A \to \beta_2$; $A \to \beta_3$

we can write; $A \to \beta_1 | \beta_2 | \beta_3$.

$$S \to aSa \,|\, bSb \,|\, a \,|\, b \,|\, \epsilon$$

Language accepted by it, $L = \{ w \,|\, w = w^R \}$

// all palindromes.

(C- is not nested; we can't define a function inside another function).

$$E \to E+E \,|\, E*E \,|\, (E) \,|\, a$$

(Generates all binary expressions with '+' and '*').

Derivation sequence of $id + id * id$

$$E \to E+E$$
$$\to E + E*E$$
$$\to id + E*E$$
$$\to id + id*E$$
$$\to id + id * id$$

* If always the leftmost variable is replaced then it leftmost derivation. Otherwise, rightmost derivation. Always maintain the consistency.

Leftmost derivation → $E \rightarrow E+E$ (id+id)*id

(* is given precedence) $\rightarrow id+E$

$\rightarrow id + E*E$

$\rightarrow id + id * E$

$\rightarrow id + id * id$

Another leftmost derivation → $E \rightarrow E*E$

(+ is given precedence) $\rightarrow E+E*E$

$\rightarrow id+E*E$

$\rightarrow id+id*E$

$\rightarrow id+id*id$

> If G has two or more leftmost derivation or two or more rightmost derivations, then the Grammar is ambiguous.

Grammar ⇒

$$\left.\begin{aligned} E &\rightarrow E+T \mid T \\ T &\rightarrow T*F \mid F \\ F &\rightarrow (E) \mid id \end{aligned}\right\}$$

In this case, multiplication will always have higher priority than addition

2 + 3 + 5 + 9 } '+' is left associative

* exponentiation is right associative.

Now, if '^' is to be given higher expontiations →

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow G\text{^}F \mid G$ // This will make sure

$G \rightarrow (E) \mid id$ // that exponentiation

// is right associative

Context Free Grammar $\longrightarrow$ 12/9/18

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

Every production is of the form ; $A \rightarrow \alpha$

* In case of sentencial form, ~~the~~ the general form is = $\alpha A \beta$

The derivation process =

$$S \longrightarrow \underset{A}{\ldots} \longrightarrow \underset{\gamma}{\ldots} \longrightarrow \ldots$$

For indicating a set of productions to get ω

$$S \Rightarrow \omega$$

for denoting the Grammar $\rightarrow$

$$S \underset{G}{\Rightarrow} \omega$$

for one or more times $\rightarrow$ $S \overset{*}{\underset{G}{\Rightarrow}} \omega$

$\{x \mid n_a(x) = n_b(x)\}$ // number of a = number of b.

$S \rightarrow aSb \mid bSa \mid \cancel{ab} \mid \cancel{ba} \mid \epsilon$ $\rightarrow$ limited

abba
aabb
abab

$\cancel{S \rightarrow SS \mid ab \mid ba}$

Correct one $\rightarrow$

$$\boxed{S \rightarrow aSbS \mid bSaS \mid \epsilon}$$

* In some languages, all the grammars considered are ambiguous. These are called inherently ambiguous languages.

* The above grammar is ambiguous.

$$\left[S \to bSaS \mid aSbS \mid \epsilon \right]$$

for the string abab ; there are two left linear derivations ⇒

$S \to aSbS \longrightarrow abSaSbS \to abaSbS \to ababS \to abab$

$S \to aSbS \to abS \to aba$

For making an unambiguous grammar ⟶

$S \to aB \mid bA$ [$B \to$ string with 1 b more than a's; $A \to$]

$x = \{ a^i b^j c^k \mid i, j, k \geq 0 \}$

~~$S \to \{ bSaS \mid aSbS \mid c \mid \epsilon \}$~~

$$\boxed{\begin{aligned} S &\to \cancel{aDb\epsilon}\ DC \\ D &\to aDb \mid \epsilon \\ C &\to cC \mid \epsilon \end{aligned}}$$

Modified version ⟶

$x = \{ a^i b^j c^k \mid i, j, k \geq 0 \}$ and $i \neq j \}$

$S \to AB$

$A \to aAb \mid aC_1 \mid bC_2.$

$C_1 \to aC_1 \mid \epsilon$

$C_2 \to bC_2 \mid \epsilon$

$B \to cB \mid \epsilon$

Scanned by CamScanner

Definition of identifier →

$$\boxed{alpha\ (alpha \mid digit)^*}$$

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

$I \rightarrow Ia \mid Ib \mid I0 \mid I1 \mid a \mid b$

Last one!! →

$L_1 : \{ a^n b^n c^m d^m \ ; \ n, m \geq 1 \}$

$$\left[\begin{array}{l} S \rightarrow ABCD \\ A \rightarrow aA \mid \epsilon \\ B \rightarrow bB \mid \epsilon \\ C \rightarrow cC \mid \epsilon \\ D \rightarrow d \end{array}\right]$$

This is not correct

$S_1 \rightarrow A_1 B_1$

$A_1 \rightarrow a A_1 b \mid ab$

$B_1 \rightarrow c B_1 d \mid cd$

This is correct

Consider another language ⇒ $L_2 : \{ a^n b^m c^m d^n \mid n, m \geq 1 \}$

$S_2 \rightarrow a S_2 d \mid a C_1 d$

$C_1 \rightarrow b C_1 c \mid bc$

$L_1 \cup L_2 = a^+ b^+ c^+ d^+$

$L_1 \cap L_2 = a^n b^n c^n d^n$

* Context free languages are closed under '$\cup$' (union)

In all the grammars in $L_1 \cup L_2$ are ambiguous.

$\therefore$ $L_1 \cup L_2$ is inherently ambiguous.

a b c