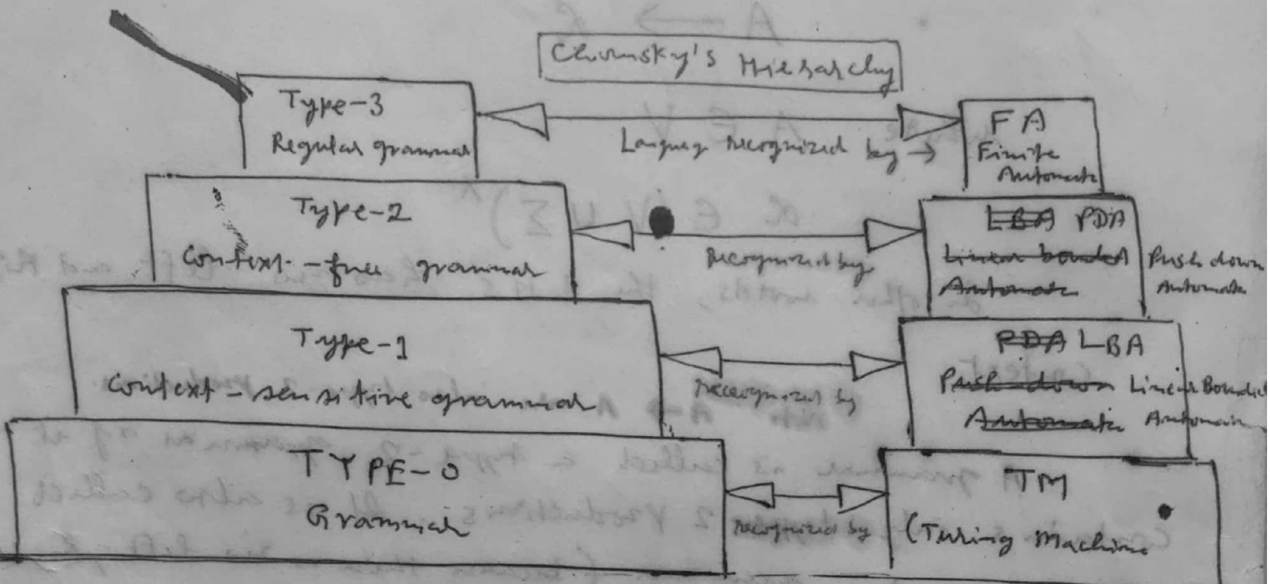
CHOMSKY'S HIERARCHY

Noam Chomsky classified the grammars into four types in terms of productions $G = (V, \Sigma, S, P)$

1. Type-0 grammar
2. Type-1 grammar (context-sensitive grammar)
3. Type-2 grammar (context-free grammar)
4. Type-3 grammar (Regular grammar)



(I) Type-'0' grammar:

A type-'0' grammar is any phrase structure grammar without any restrictions (All the grammar we consider as type-0 grammar) (This language is recognized by Turing Machine

(II) Type-'1' grammar: (context-sensitive grammar)

A production of the form

Left context ← → Right context

$\phi A \psi \rightarrow \underline{\phi \alpha \psi}$ is called type-1 production if $\boxed{\alpha \neq \Lambda}$

replacement string.

$|\phi A \psi| \leq |\phi \alpha \psi|$

[It does not increase the length of working string as $\alpha \neq \Lambda$]

i.e. erasing of 'A' is not permitted

So grammar is called type-1 or context-sensitive if all of its productions are type-'1' productions.

A grammar $G = (V, \Sigma, S, P)$ is said to be context sensitive if all productions are of the form $x \rightarrow y$ where $x, y \in (V \cup \Sigma)^+$ and $|x| \leq |y|$ length of successive sentential form can never decrease

→ These language can be recognized by a linear bounded automaton (Non deterministic Turing machine)

The production $S \rightarrow '\Lambda'$ is also allowed in type-1 grammar, but in this case 'S' does not appear on the RHS of any productions.

$S \rightarrow aA \mid \Lambda$ is allow

$S \rightarrow aAS \mid \Lambda$ is not allow

(III) **Type-'2' grammar** (context-~~sensitive~~ free grammar)

A type-'2' production is a production of the form

$$A \longrightarrow \alpha$$

where $A \in V$

$$\alpha \in (V \cup \Sigma)^*$$

In other words, the LHS has no left and Right content.

Note $A \rightarrow \Lambda$ are also type-2 productions.

A grammar is called a type-2 grammar if it contains only type-2 productions. It is also called the context-free grammar (because there is no left, Right content) These languages can be recognized by pushdown automaton. These languages are useful in checking the syntax of most programming languages

(IV) **Type-'3' grammar** (Regular grammar)

In production have a form

$$A \longrightarrow a$$

or

$$A \longrightarrow aB$$

(These language can be recognized by ~~regular expression~~ Finite state automaton. Regular languages are commonly used to define search patterns and the lexical structure of programming languages.)

where $A, B \in V$,

$$a \in \Sigma^*$$

called type-3 production

A grammar is called type-3 or regular grammar if all its production are type-3 productions, A production

$$S \rightarrow \Lambda$$

is allowed in type-3 grammar but in this case 'S' does not appear on the RHS of any productions. ~~[The languages which are generated by type-3 grammar are recognised by FSM [illegible] identical to Regular [illegible]]~~

Type-3 Grammar (Regular grammar)

OR

Right & Left linear Grammars

A grammar $G = (V, \Sigma, S, P)$ is said to be right-linear if all productions are of the form

$$A \rightarrow xB \quad \text{or} \quad A \rightarrow x$$

where $A, B \in V$, $x \in \Sigma^*$

A grammar is said to be left-linear if all productions are of the form

$$A \rightarrow Bx \quad \text{or} \quad A \rightarrow x$$

A regular grammar is one that is either right-linear or left-linear.

(Note: In regular grammar, <u>at most</u> one variable appears on the right side of any production) Goto page-16 (Type-3 Grammars)

Type-2 Grammar (Context Free Grammar)

The productions in a regular grammar are restricted in two ways ① The left side must be a single variable ② while the right side has a special form. To create grammars that are more powerful we must <u>relax</u> some of these restrictions. By retaining the restriction on the left side, but permitting anything on the right, we get context free grammars.

(18)

A grammar $G = (V, \Sigma, S, P)$ is said to be context-free if all productions in P have the form

$$A \rightarrow x$$

where $A \in V$

$x \in (V \cup \Sigma)^*$ Goto page No (16) Type-2

A language L is said to be context free if and only if there is a context free grammar G such that $L = L(G)$

~~eva~~ $R.G \subseteq CFG$

Type-1 Grammar (Context Sensitive Grammars)

A grammar $G = (V, \Sigma, S, P)$ is said to be context sensitive if all productions are of the form

$$x \rightarrow y$$

where $x, y \in (V \cup \Sigma)^+$

and $|x| \leq |y|$

All such can be Rewritten in a normal form

$$\underbrace{x}_{\text{Left context}} A \underbrace{y}_{\text{Right context}} \rightarrow x\, v\, y$$

Goto Page No (15)

This is equivalent to saying that

$A \rightarrow v$ ~~ca~~

$A \rightarrow v$ can be applied only in the situation where A in a context of the string x on the left and the string y on the Right.

(12)

Ex-10 Let $G = (\{S, A_1\}, \{0, 1, 2\}, P, S)$

where P consists of

$$P = \{S \to 0SA_1 2, \; S \to 012, \; 2A_1 \to A_1 2, \; 1A_1 \to 11\}$$

determine the Language L(G) for Grammar G.

Sol $\because$ $S \to 012$ is a production rule (terminal)

$S \Rightarrow 012$ $\therefore$ $012 \in L(G)$

$S \Rightarrow 0SA_1 2$ by applying $S \to 0SA_1 2$

$\Rightarrow 0(0SA_1 2)A_1 2$ "

$\Rightarrow 0(0(0SA_1 2)A_1 2)A_1 2$ "

$\Rightarrow$ ------

$\vdots$

$\Rightarrow 0^{n-1} S (A_1 2)^{n-1}$ [up to n-1 times]

$\Rightarrow 0^{n-1}(012)(A_1 2)^{n-1}$ by $S \to 012$

$\Rightarrow 0^{n-1}.0.1.\underbrace{2(A_1 2)}\underbrace{(A_1 2)}\underbrace{(A_1 2)}\ldots\underbrace{2)(A_1 2)}$ upto n-1 times

~~$\Rightarrow 0^{n-1}.0.1 (A_1 2)(A_1 2)(A_1 2) \ldots (A_1 2).2$ upto n-1 times [By apply $2A_1 \to A_1 2$ upto n-1 times]~~

~~$\Rightarrow 0^n.1.$~~

$\Rightarrow 0^n.1.\underbrace{2A_1}\underbrace{2A_1}\underbrace{2A_1}\ldots\underbrace{2A_1}\underbrace{2A_1}.2$

$\Rightarrow 0^n 1 . \underbrace{A_1 2}\underbrace{A_1 2}\underbrace{A_1 2}\ldots\underbrace{A_1 2}\underbrace{A_1 2}\; 2$

$\Rightarrow 0^n 1 . A_1 A_1 \underbrace{2A_1}\underbrace{2A_1}2 \ldots A_1 2 \; A_1 2 . 2.2$

$\Rightarrow 0^n.1. \underline{A_1 A_1 A_1} 2 \; A_1 2 \; A_1 2 \; A_1 2 \; A_1 2 . \underline{2.2.2}$

apply $2A_1 \Rightarrow A_1 2$ Several times.

$\vdots$

$\overset{*}{\Rightarrow} 0^n . 1. A_1^{n-1} 2.2^{n-1}$

$\Rightarrow 0^n \; 1(A_1 A_1^{n-2}.2^n$

$$S \Rightarrow 0^n . 11 . A_1^{n-2} . 2^n \quad \text{by } 1A_1 \to 11$$

$$\Rightarrow 0^n . 1 . \underbrace{1A_1}_{\downarrow} . A_1^{n-3} . 2^n$$

$$\Rightarrow 0^n . 1 . 11 . A_1^{n-3} . 2^n \quad \text{by } 1A_1 \to 11$$

$$\Rightarrow 0^n . 1 . 1 . \underbrace{1A_1}_{\downarrow} A_1^{n-4} . 2^n$$

$$\Rightarrow 0^n . 1 . 1 . 1 . \underline{1 A_1} A_1^{n-5} 2^n \quad \text{by } 1A_1 \to 11$$

$\vdots$ up to $n-1$ times

$$\overset{*}{\Rightarrow} 0^n . 1^n . 2^n \quad \text{for all } n \geq 1$$

$$\therefore L(G) = \{ 0^n 1^n . 2^n \mid n \geq 1 \}$$

Ex-11 Construct a grammar G generating $\{a^n b^n c^n : n \geq 1\}$

Solution: We already known how to construct $a^n b^n$ recursively

Therefore in our problem we will do it in two part

(i) First we construct $a^n \alpha^n$

(ii) Then we convert α^n into $b^n c^n$

For stage (i) Production will be

$$S \to aS\alpha$$
$$S \to a\alpha$$

In stage (ii) we can't take $\alpha = bc$ $\because (bc)^n \neq b^n c^n$

$\therefore$ Take $\alpha = B.C.$ B & C are variables

To bring B's together we introduce new rule

$$C.B \to B.C$$

For converting B's into b and C's into c

$aB \to ab$ $\quad bC \to bc$

$bB \to bb$ $\quad cC \to cc$