gradeup

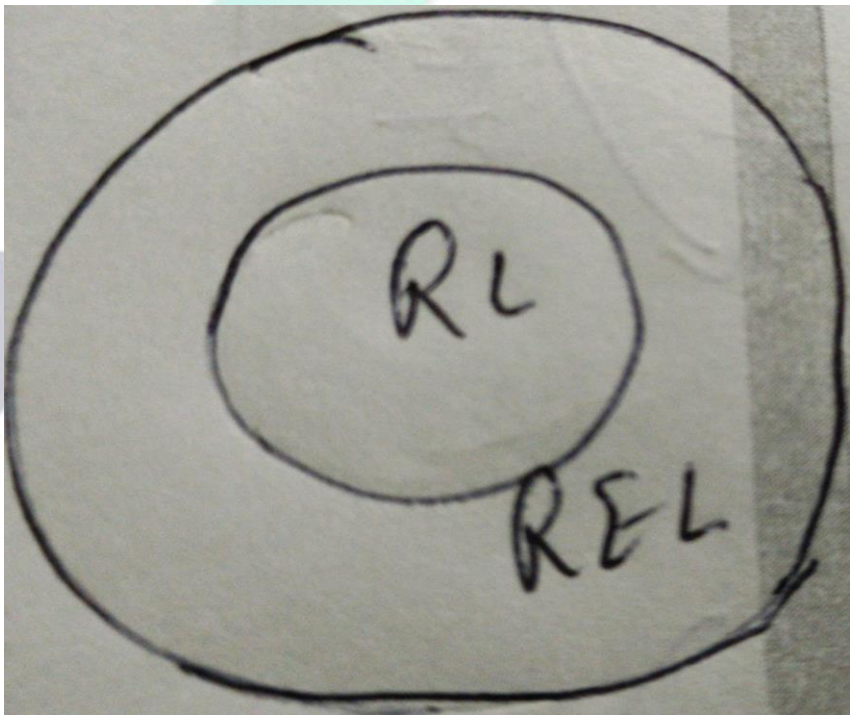
Turing Machine (TM) Part

Turing Machine

Content:

1. **Concept of Turing Machine**
2. **Algorithm of turing machine**
3. **Chomsky Hierarchy**

1. The language accepted by Turing Machine is **recursively enumerable language (REL).**
2. If we apply some restriction on Turing Machine, then the language accepted by Truing Machine is called **Recursive large(RL).**
3. All recursive languages are recursively enumerable languages .



4. Non- deterministic Turing Machine or Deterministic Turing Machine have same or equal powers.
5. **Recursively enumerable** language are closed under:
   a. Union

b. Intersection

c. Kleen star

It is not closer under:

1. Complement
2. Set difference

- If L is recursive then its complement is also recursive
- If L is RE , then its kleen star is also RE
- Two language are RE , then its concatenation is also RE
- Union of two RE language is also RE
- Recursively enumerable use unrestricted grammar

Decidability of turing Machine

Decidability is no for all this

1. No algorithm to check whether the machine will accept something or not means emptiness expression for Regular Expression cannot be decided by Turing Machine.
2. There does not exist a Turing Machine that can decide for any encoded Turing Machine T fed into it, whether or not the language T is finite or infinite.

THE CHOMSKY HIERARCHY

PHRASE-STRUCTRUE GRAMMARS

A phrase-structure grammar is a collection of three things:

1. A finite alphabet Σ of letters called terminals.

2. A finite set of symbols called non-terminal that includes the start symbol .S

3. A finite set of production of the forms

$$\text{String 1} \rightarrow \text{String 2}$$

Where string 1 can be any string of terminals and non-terminals that contains at least one non-terminals and where string 2 is any string of terminals and non-terminals what so ever.

A derivation in a phrase-structure grammar is a series of working string beginning with that star symbol S, which by making substitution according to the productions, arrives at a string of all terminals, at which point generation must stop.

TIPS

The language generated by a phrase structure grammar is the set of all strings of terminals that can be derived string at S.

Example: The following is a phrase structure grammar over Σ = {a, b} with non terminals A and B:

$$A \rightarrow BA/\in \qquad (P_1 \text{ say})$$

$$B \rightarrow aB/a \qquad (P_2 \text{ say})$$

$$aaaB \rightarrow ba \qquad (P_3 \text{ say})$$

Solution: The first production P_1 say that we can start with A and derive number of symbols of the type B, for example

$$A \rightarrow BA$$

$$\rightarrow BBA$$

$$\rightarrow BBBA$$

→ BBBBA

→ BBBB

The second production shows us that each B can be any string of a's (with at least one a):

B → aB

→ aaB

→ aaaB

→ aaaaB

→ aaaaa

The third production (P_3) says that any time we find three a's and B, we can replace these four symbols with the two-terminal string ba.

The following is a summary of one possible derivation in this grammar:

A → BBBBBB

→ aaaaBBBBB

→ aabaBBBB

→ aabaaaBB

→ aabbaBB

→ aabbaaaB

→ aabbba

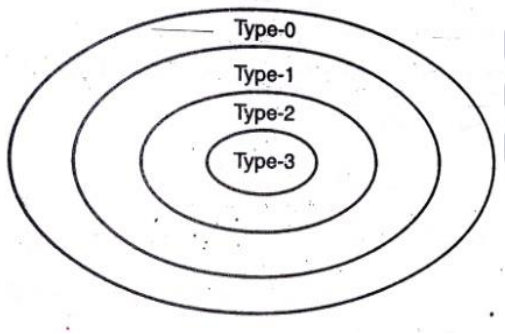
All CFG's are phrase-structure grammars in which we restrict ourselves as to what we put on the left side of productions. So, all CFLs can be generated by phrase-structure grammars.

CHOMSKY HIERARCHY

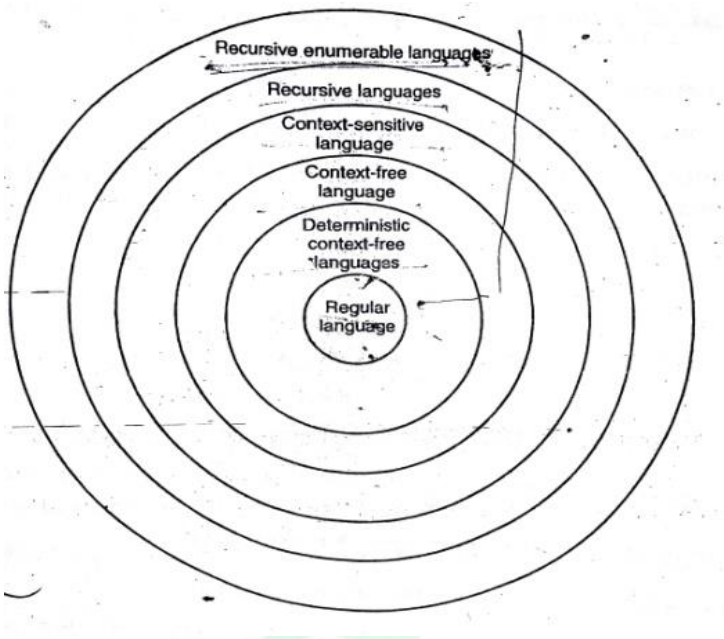
We can exhibit the relationship between grammars by the Chomsky Hierarchy. Noum Chomsky, a founder of formal language theory, provided an initial classification in to four language types:

Type - 0	(Unrestricted grammar)
Type - 1	(Context sensitive grammar)
Type - 2	(Context free grammar)
Type - 3	(Regular grammar)

Type 0 languages are those generated by unrestricted grammars, that is, the recursively enumerable languages. Type 1 consist of the context-sentive languages, Type 2 consists of the context-free languages and Type 3 consists of the regular languages. Each languages family of type k is a proper subset of the family of type k - 1. Following diagram shows the original Chomsky Hierarchy.



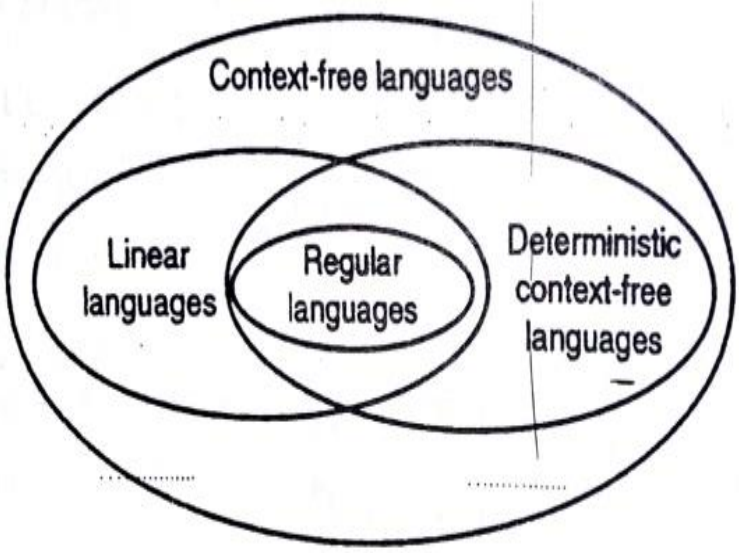
We have also met several other language families that can be fitted in to this picture. Including the families of deterministic context-free languages (LDCF), and recursive languages (LREC). The modified Chomsky Hierarchy can be seen in below figure.



We know that L = {w : n_a (w) = n_b(w) } is deterministic, but not linear. On the other hand, the language

$$L = \{a^n b^m\} \cup \{a^n b^{2n}\}$$

Is linear, but not deterministic. The relationship between, linear, deterministic context-free and non-deterministic context-free language is shown in below figure.



In the last we can summarise our discussion as follows:

Type	Name of languages generated	Production Restrictions A → B	Acceptor
0	Unrestricted (reversively-enumerable)	A = any string with non-terminal B = any string	Turing machine
1	Context-sentive	A = any string with non terminals B = any string as long as or longer than A.	Linear bound Automata
2	Context-free	A = one non-terminal B = any string	Pushdown Automata
3	Regular	A = one non-terminal B = aX or B = a, where 'a' is a terminal and X is non-terminal.	Finite automata

UNRESTRICTED (TYPE-0) GRAMMARS

The unrestricted grammar is defined as

$$G = (Vn, Vt, P, S)$$

Where

Vn = a finite set of non-terminal

Vt = a finite set of terminals

S = starting non-terminal, $S \in Vn$

And P is set of productions of the following form

$$\alpha \rightarrow \beta$$

where α and β are arbitrary string of grammar symbols with $\alpha \neq \in$. These grammars are knows as type-0, phase-structure or unrestricted grammars.

Context - Sensitive grammar (CSL):

Let G = (V_n, V_t, P, S) be context sentive grammar

Where

V_n = finite set of non-terminals

V_t = finite set of terminals

S = starting non-terminals $S \in V_n$

And P is the set of rules called productions defined as

$$\alpha \rightarrow \beta$$

where β is at least as long as α that is clearly

$$|\alpha| \leq |\beta|$$

The term "context-sentive" comes from a normal form for these grammars, where each production is of the form $\alpha_1 \rightarrow \alpha_1 \beta \alpha_2$, with $\beta \neq \in$. Replacement of variable A by string β is permitted in the "context" of α_1 and α_2.

For example, language $L = \{a^n b^n c^n : n \geq 0\}$ is context sensitive. We can show this exhibiting a context sentive grammar for the language L_1 grammar as

$$S \rightarrow abc/aAbc$$

$$Ab \rightarrow bA$$

$$Ac \rightarrow Bbcc$$

$$bB \rightarrow Bb$$

$$aB \rightarrow aa/aaA$$

"A type 0 grammar is called length-increasing if for all rules $\alpha \rightarrow \beta$ in P we have $|\alpha| \leq |\beta|$" (same as context sensitive grammar).

So we can say that $\in$ free type-0 grammar is said to be length increasing grammar.

LINEAR BOUNDED AUTOMATA

A Linear Bounded Automata (LBA) is a non-deterministic turing machine satisfying the following two conditions.

(a) Its input alphabet includes two special symbols $\$_1$ nd $\$_2$ the left and right end marker, respectively.

(b) The LBA has no more left from $\$_1$ and no right form $\$_2$ nor may it print another symbol over $\$_1$ or $\$_2$.

The linear bound automata is simply a turning machine which instead of having potentially infinite tape on which to compute, is restricted to the portion of the tape containing the input plus the two tape squares bolding the end markers.

LBA will be denoted as

$T_m = \{Q, \Sigma, \Gamma, \delta, q_0, h, \$_1, \$_2)$ where Q, Σ, δ, S are as for non-deterministic turning machine; $\$_1$ and $\$_2$ are symbols in Γ, the left and right end markers.

The $L(T_m)$ the languages accepted by turning machine is

$\{W/W$ is in $\Sigma*$ and $(q_0, \$_1\ W\ \$_2) *\vdash_{T_m} (h, \$_1\ y\ \$_2\}$

Here y is for yes and h shows halting of LBA after accepting the string W.

RELATIONS BETWEEN CLASSES OF LANGUAGES

The four classes of languages-recursive sets, context sensitive languages, context-free languages and regular sets are often referred to as language of types 0, 1, 2 and 3 respectively. We can find the relation between all those sets of language through following theorems:

Theorems:

1. Every context sensitive language is recursive.
2. There is at least one language L that is recursive but not context-sensitive.

gradeup SUPER

Gradeup UGC NET
Super Superscription

Features:

1. 7+ Structured Courses for UGC NET Exam
2. 200+ Mock Tests for UGC NET & MHSET Exams
3. Separate Batches in Hindi & English
4. Mock Tests are available in Hindi & English
5. Available on Mobile & Desktop

Gradeup Super Subscription, Enroll Now

gradeup.co