Pumping Lemma for Context free Language

The language which can be represented by CFG is called Context free Language (CFL).
Pumping Lemma is used to check that given language is not a CFL.

Let CFG $G (V_N, \Sigma, P, S)$ exist which derive a string w such that

$$|w| \geq n,$$

where n is a constant number. Then string w can be written into the form.

$$w = uvxyz$$

Subjected to the following condition

1) $|vxy| \leq n$ $\qquad u\underbrace{vxy}z$

ie middle portion $\leq n$

(ii) $v, y \neq \lambda$ it means $|vy| > 0$

$\therefore$ string v and y will be pumped.

(iii) for $i \geq 0$ $\quad uv^ixy^iz \in L$ ie

for every $i \geq 0$

string uv^ixy^iz must be present into the language generated by CFG.

Q. Prove that language

$L = \{a^n b^n c^n \mid n \geq 1\}$ is not CFL

Sol.

Step 1: Assume that given language is a CFL.

Step 2: chose a string w such that

$|w| \geq n$, where n is any positive no.

$w = a^n b^n c^n$

$|w| = (n+n+n) = |3n| \geq n$.

Step 3. Represent the string w in the form of $uvxyz$ with the condition that

(i) $|vxy| \leq n$

(ii) $vy \neq \lambda$

$w = a^n b^n c^n$

$w = \underbrace{a^p}_{u} \underbrace{a^q}_{v} \underbrace{a^r}_{x} \underbrace{b^p}_{y} \underbrace{b^q b^r c^n}_{z}$ (we assume $n = p+q+r$

where $q > 0, p > 0$, since $vy \neq \lambda$.

Step 4. Check that for $i > 0$

$uv^i xy^i z \notin L$

for $i = 2$

$uv^2xy^2z = a^p (a^q)^2 a^r (b^p)^2 b^q b^r c^n$

$= a^p a^q . a^q a^r b^p b^p b^q b^r c^n$

$= a^n . a^q b^n b^p c^n = a^{n+q} b^{n+p} c^n$

Here $p, q > 0$

Therefore the string $a^{n+q} b^{n+p} c^{n}$ doesn't have an equal no. of a, equal no. of b and equal no. of c. Hence This is a contradiction of our assumption.

So, Given Language (L) is not a CFL.

Closure property of CFL.

1) Union ($L_1 \cup L_2$):

If L_1 and L_2 are two CFL, then $L_1 \cup L_2$ is also a CFL.
Hence, Union operation is closed under CFL.

Proof: Let L_1 is a CFL, then it must be represen-ted by a CFG $G_1(V_{N_1}, \Sigma_1, P_1, S_1)$

Let L_2 is a CFL, then it must be represented by a CFG $G_2(V_{N_2}, \Sigma_2, P_2, S_2)$

If $L_1 \cup L_2$ is also a CFL, then it must be represented by a CFG.

CFG for language $L_1 \cup L_2$ can be constructed as follows:

CFG $G(V_N, \Sigma, P, S)$

$V_N = V_{N_1} \cup V_{N_2} \cup \{S\}$

$\Sigma = \Sigma_1 \cup \Sigma_2$

P is given as follows

$S \rightarrow S_1 | S_2$ and all productions of P_1, P_2

$S = \{S\}$

Since, language $L_1 \cup L_2$ is represented by above CFG. Hence $L_1 \cup L_2$ is a CFL.

2) Concatenation

If L_1 and L_2 is a CFL, then $L_1 L_2$ is also a CFL
i.e Concatenation is the closed operation for CFL.

Proof: Let L_1 is a CFL, then it must be represented by CFG $G_1(V_{N_1}, \Sigma_1, P_1, S_1)$

Let L_2 is a CFL, then it must be represented by CFG $G_2(V_{N_2}, \Sigma_2, P_2, S_2)$

If $L_1 L_2$ is also a CFL, then it must be represented by a CFG.

CFG for the language $L_1 L_2$ can be constructed as follows:

CFG $G(V_N, \Sigma, P, S)$

$V_N = V_{N_1} \cup V_{N_2} \cup \{S\}$

$\Sigma = \Sigma_1 \cup \Sigma_2$

P is given as follows

$S \rightarrow S_1 S_2$ and ~~and~~ all the productions of P_1 & P_2.

$S = \{S\}$

Since Language $L_1 L_2$ is represented by above CFG. Hence $L_1 L_2$ is a context free language.

3) Closure $(L,^*)$

The Kleene closure of a CFL is also a CFL. Hence, Kleene closure is a closed operation for CFL.

Proof:

Let L_1 is a CFL, then it must be represented by a CFG $G_1 (V_N, \Sigma, P, S)$

If L_1^* is also a CFL, then it must be represented by a CFG.

CFG for language L_1^* can be constructed as follows:

CFG $G (V_N, \Sigma, P, S)$

$V_N = V_{N_1} \cup \{S\}$

$\Sigma = \Sigma_1$

P is defined by and all the production of L_1

$S \to SS_1 \mid \lambda$

$S = \{S\}$

Since L_1^* is represented by above CFG. Hence, Kleene closure is a closed operation for CFL.

Scanned with CamScanner

4.) Intersection

If L_1 and L_2 is a CFL, then $L_1 \cap L_2$ is not a CFL ie

Intersection is not a closed operation for CFL

Proof: Let L_1 is a CFL

$$L_1 = \{ a^n b^n c^i \mid n, i \geq 0 \}$$

Let L_2 is a CFL

$$L_2 = \{ a^i b^n c^n \mid n, i \geq 0 \}$$

then $L_1 \cap L_2 = \{ a^n b^n c^n \mid n \geq 0 \}$.

Here $L_1 \cap L_2$ is not a CFL, because it requires comparison among the three things.

Hence, intersection of two CFL is not a CFL.

5) Complement ($\bar{L}$)

Complement of a CFL is not a CFL. Hence, Complement is not closed operation for CFL.

Proof: Assume that complement of a CFL is also a CFL.

If L_1 is a CFL, then $\bar{L_1}$ is also a CFL.

If L_2 is a CFL, then $\bar{L_2}$ is also a CFL.

$\bar{L_1} \cup \bar{L_2}$ is also a CFL. Since, Union of two CFL's is also a CFL.

$\overline{\bar{L_1} \cup \bar{L_2}}$ is also a CFL.

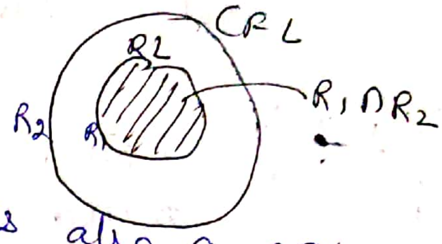
$$\overline{\bar{L_1} \cup \bar{L_2}} = \bar{\bar{L_1}} \cap \bar{\bar{L_2}}$$

$$= L_1 \cap L_2$$

Intersection of two CFL is not a CFL. This is the contradiction from the above assumption.

Hence, complement of a CFL is not a CFL

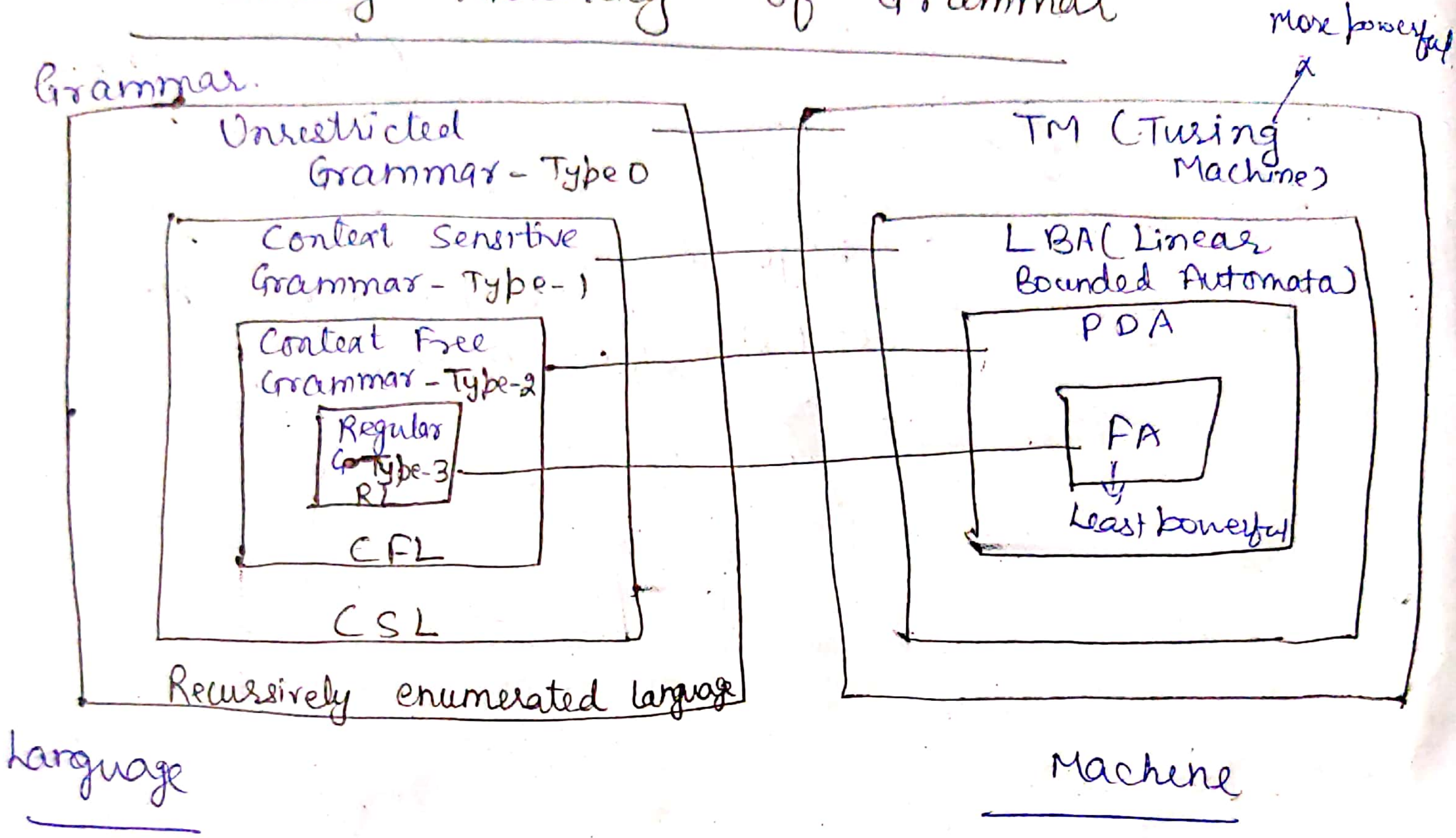
6.) Intersection of CFL and RL



Intersection of a CFL and a RL is also a CFL.
Since, all the Regular Languages can be represented by Context free Grammar.

Types of Grammar

Chomsky Hierarchy of Grammar



Four types of grammar

1) Unrestricted Grammar (Type 0)
2) Context sensitive Grammar (Type 1)
3) Context free Grammar (Type 2)
4) Regular Grammar (Type 3)

(1) Unrestricted Grammar (Type 0)

A grammar is said to be unrestricted grammar if the production of the grammar doesn't have any restriction. The language corresponding to this grammar is called Recursively Enumerable language (REL), which is accepted by ~~Rs~~ Turing machine (TM).

$$G = (V_N, \Sigma, P, S\}$$

$$\alpha \longrightarrow B$$

$$\alpha \in (\Sigma \cup V_N)^* V_n (\Sigma \cup V_N)^*$$

$$B \in (\Sigma \cup V_N)^*$$

R.E.G

$\downarrow$

R.E.L

$\uparrow$

T.M

Non-Contracting Grammar

2) Context Sensitive Grammar (Type 1)

It's also known length increasing grammar.

A grammar is said to be Context sensitive, if any production of the grammar is of the form

$$\alpha \longrightarrow \beta$$

$$\alpha \in (\Sigma \cup V_N)^* V_n (\Sigma \cup V_n)^*$$

$$\beta \in (\Sigma \cup V_N)^+$$

It means null production not allowed

It means

$$\phi A \psi \rightarrow \phi \alpha \psi$$

where $\phi, \psi, \in (V_N \cup \Sigma)^*$

$\alpha \in (V_N \cup \Sigma)^+$

$A \in V_N$

$$\alpha \neq \lambda$$

CSG
↓
CSL
↓
LBA

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

The language corresponding to Context Sensitive grammar is called Context Sensitive Language.

$AaB \rightarrow bb$ ✗ not Type 1 grammar (Context, length)

* $S \rightarrow \epsilon$ is allowed as an exception. Null only produce by S start symbol.

If we produce null using S production then S can't appear on R.H.S production, for recursion

Context free grammar (Type-2)

A grammar is said to be CFG, if every production of the grammar is of the form

$$\alpha \longrightarrow \beta$$

$\alpha \in V_N$ $\qquad |\alpha| = 1$

$\beta \in (V_N \cup \Sigma)^*$

The language corresponding to the CFG is called Context Free language, which is accepted by Push-down automata (PDA)

CFG

↓

CFL

↓

PDA

* when a grammar is R.G, then its definitely CFG, but the vice versa is not true.

Regular Grammar (Type-3)

A grammar is said to be regular Grammar if every production of the grammar is of the form

Left Linear Grammar	Right Linear Grammar
$A \to a$ or $A \to Ba$	$A \to a$ $A \to aB$
$A, B \in V_n$, $a \in \Sigma^*$ $\lvert A\rvert = \lvert B\rvert = 1$	$A, B \in V_N$

$A \to B\alpha \mid \beta$

$A, B \in V$
$\alpha, \beta \in \Sigma^*$

In production, the production rules are either RLG or LLG, not both.

eg. $\left.\begin{array}{l} A \to Ba \mid a \\ B \to aB \mid a \end{array}\right]$ $\left.\begin{array}{l} LLG \\ RLG \end{array}\right]$ not type 3. because one is LLG or other is RLG.

Regular grammar generates Regular Language which is accepted by finite automata.