UNIT - 2 (FORMAL LANGUAGES)

FORMAL GRAMMAR :- Describing rules of grammar in a rigorous mathematical way.

Formal Grammar was developed to make language translation using computer easy.

NOAM CHOMSKY gave mathematical model of a grammar in 1956, which became useful for describing computer languages. ex- ALGOL

Eg: To make calculator we need Grammar to define Algebra on Real No's.

GRAMMAR

A phrase-structure grammar (or simply a grammar) is (V_N, Σ, P, S), where

(i) V_N is a finite non-empty set whose elements are called VARIABLES.

(ii) Σ is a finite nonempty set whose elements are called TERMINALS.

(iii) $V_N \cap \Sigma = \phi$

(iv) S is a special variable (i.e. an elt. of V_N) called the START Symbol, and

(v) P is a finite set whose elements are $\alpha \rightarrow \beta$ where α & β are strings on $V_N \cup \Sigma$: α has atleast one symbol from V_N. The elements of P are called PRODUCTIONS or PRODUCTION RULES OR REWRITING RULES.

NOTE :- For Production Rules,

(i) Reverse substitution is not permitted i.e. $S \rightarrow AB$ is a production $\Rightarrow$ S can be replaced by AB but not vice-versa.

(ii) No inversion operation is permitted. eg :- $S \rightarrow AB$ is a production $\not\Rightarrow$ $AB \rightarrow S$ is a production.

Scanned with CamScanner

NOTATIONS :-

(i) If A is any set, then $A^* =$ set of all strings over A & $A^+ = A^* - \{\Lambda\}$ where Λ is an empty string.

(ii) $A, B, C, A_1, A_2, \ldots$ denote variables.

(iii) $a, b, c, \ldots$ " terminals.

(iv) $x, y, z, w, \ldots$ denote strings of terminals.

(v) $\alpha, \beta, \gamma, \ldots$ denote the elements of $(V_N \cup \Sigma)^*$

(vi) $X^0 = \Lambda$ for any symbol $X \in V_N \cup \Sigma$.

NOTE :- The productions are used to derive one string over $V_N \cup \Sigma$ from another string.

Def^n :- ONE STEP DERIVATION

If $\alpha \to \beta$ is a production in a grammar G and γ, δ are any 2 strings in $(V_N \cup \Sigma)$, then we say that $\gamma\alpha\delta$ directly derives $\gamma\beta\delta$ in G (we write $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$). This process is called One-Step Derivation.

eg :- $G = (\{S\}, \{0, 1\}, \{S \to 0S1, S \to 01\}, S)$

$S \to 0S1$; So, In 0^4S1^4, S can be replaced by $0S1$,

i.e., we have $0^4S1^4 \Rightarrow 0^40S11^4$

Def^n :- REFLEXIVE-TRANSITIVE CLOSURE

If α & β are strings on $V_N \cup \Sigma$, then we say that α derives β if $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$. Here, $\overset{*}{\underset{G}{\Rightarrow}}$ denotes the reflexive-transitive closure of the relation $\underset{G}{\Rightarrow}$ in $(V_N \cup \Sigma)^*$.

Means More than 1 step (≥ 2 steps)

Example :- $G = (\{S\}, \{0, 1\}, \{S \to 0S1, S \to 01\}, S)$.

$S \underset{G}{\Rightarrow} 0S1 \underset{G}{\Rightarrow} 0^2S1^2 \underset{G}{\Rightarrow} 0^3S1^3$; we have

$S \overset{*}{\underset{G}{\Rightarrow}} 0^3S1^3$ (We also have, $0^3S1^3 \overset{*}{\underset{G}{\Rightarrow}} 0^3S1^3$ as $\alpha \overset{*}{\underset{G}{\Rightarrow}} \alpha$)

Scanned with CamScanner

Def^n:- The language generated by a grammar G (denoted by L(G)) is defined as $\{w \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w\}$.

The elements of L(G) are called sentences.

OR L(G) is the set of all terminal strings derived from the start symbol, S.

Def^n:- If $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$, $\alpha \in (V_N \cup \Sigma)^*$, then α is called a Sentential form.

Note that all the elements of L(G) are sentential form but not vice-versa.

Def^n:- 2 Grammars G_1 & G_2 are said to be EQUIVALENT if $L(G_1) = L(G_2)$.

Remark:

(i) The string generated by the most recent application of production is called the working string.

(ii) The derivation of a string is complete when the working string cannot be modified. If the final string doesn't contain a variable, it is called a sentence in the language. If the final string contains a variable, it is called a sentential form & in this case the production generator gets stuck.

eg:- Production generator gets stuck in the grammar

$G = (\{S\}, \{0,1\}, \{S \to 0S1\}, S)$

(iii) If $A \to \alpha$ is a production where $A \in V_N$, then it is called an A-production.

(iv) If $A \to \alpha_1$, $A \to \alpha_2$, ..., $A \to \alpha_m$ are A-productions, these productions are written as $A \to \alpha_1 | \alpha_2 | \alpha_3 | \ldots | \alpha_m$.

Scanned with CamScanner

EXAMPLES :-

If $G = (\{S\}, \{0,1\}, \{S \to 0S1, S \to \Lambda\}, S)$, find $L(G)$.

$S \underset{G}{\Rightarrow} \Lambda$; So, $\Lambda \in L(G)$.

For $n \geq 1$,

$$S \underset{G}{\Rightarrow} 0S1 \underset{G}{\Rightarrow} 0^2S1^2 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} 0^nS1^n \underset{G}{\Rightarrow} 0^n1^n$$

$\therefore \quad 0^n1^n \in L(G)$ for $n \geq 0$.

$\therefore \quad \{0^n1^n \mid n \geq 0\} \subseteq L(G)$.

Next, we need to show $L(G) \subseteq \{0^n1^n \mid n \geq 0\}$, we start with w in $L(G)$. The derivation of w can only start with the start symbol 'S'. $\therefore$ we start with S.

If $S \Rightarrow \Lambda$ is applied first, we get Λ. In this case, $w = \Lambda$. Otherwise, the first production to be applied is $S \to 0S1$. At any stage, if we apply $S \to \Lambda$, we get a terminal string. Also, the terminal string is obtained only by applying $S \Rightarrow \Lambda$. Thus, the derivation of w is of the form

$$S \overset{*}{\underset{G}{\Rightarrow}} 0^nS1^n \underset{G}{\Rightarrow} 0^n1^n \quad \text{for some } n \geq 1$$

i.e.

$$L(G) \subseteq \{0^n1^n \mid n \geq 0\}$$

$$\therefore \quad L(G) = \{0^n1^n \mid n \geq 0\}$$

Example :- Let $G = (\{S, C\}, \{a,b\}, P, S)$ where P consists of $S \to aCa$, $C \to aCa \mid b$. Find $L(G)$.

Sol^n :- $S \Rightarrow aCa \Rightarrow aba$. So, $aba \in L(G)$

$\therefore \quad S \overset{*}{\Rightarrow} a^nCa^n$ (by application of $C \to aCa$ $(n-1)$ times)

Scanned with CamScanner

Hence, $a^n b a^n \in L(G)$, where $n \geq 1$. $\therefore$

$$\{a^n b a^n \mid n \geq 1\} \subseteq L(G).$$

For converse part inclusion, As only S-production is - $S \to aCa$, this is the first production we have to apply in the derivation of any terminal string. If we apply, $C \to b$, we get aba. Otherwise, we have to apply aCa, either once or several times. So, we get - $a^n C a^n$ with a single variable C. To get a terminal string we have to replace C by b, by applying $C \to b$.

So, any derivation is of the form:-

$$S \overset{*}{\Rightarrow} a^n b a^n \text{ with } n \geq 1.$$

$$\therefore \quad L(G) \subseteq \{a^n b a^n \mid \cancel{\text{with}}\ n \geq 1\}$$

$$\therefore \quad L(G) = \{a^n b a^n \mid n \geq 1\}$$

Ques :- let L be the set of all palindromes over $\{a, b\}$. Construct a grammar G generating L.

Sol^n:- For constructing a grammar G generating the set of all the palindromes over $\{a, b\}$; we can use recursive def^n. to observe the following :-

(i) Λ is a palindrome.

(ii) a, b are palindromes.

(iii) If x is a palindrome axa, bxb are palindromes.

So, we define P as the set consisting of :-

(i) $S \to \Lambda$

(ii) $S \to a$, $S \to b$

(iii) $S \to aSa$ and $S \to bSb$

let $G = (\{S\}, \{a, b\}, P, S)$. Then,

$S \Rightarrow \Lambda$, $S \Rightarrow a$, $S \Rightarrow b$ $\quad \therefore \Lambda, a, b \in L(G)$

Scanned with CamScanner

If x is a <u>palindrome of even length</u>, then

$x = a_1 a_2 \ldots a_m a_m \ldots a_1$, where each a_i is either a or b.

Then, $S \overset{*}{\Rightarrow} a_1 a_2 \ldots a_m a_m a_{m-1} \ldots a_1$ by applying $S \to aSa$ or $S \to bSb$. Thus, $x \in L(G)$.

If x is a <u>palindrome of odd length</u>, then

$x = a_1 a_2 \ldots a_n c a_n \ldots a_1$, where a_i's and c are either a or b.

So, $S \overset{*}{\Rightarrow} a_1 \ldots a_n S a_n \ldots a_1 \Rightarrow x$ by applying $S \to aSa$, $S \to bSb$ and finally $S \to a$ or $S \to b$. Thus, $x \in L(G)$. This proves

$$L = L(G)$$

<u>Ques</u> Find a grammar generating $L = \{a^n b^n c^i \mid n \geq 1, i \geq 0\}$

<u>Sol^n</u>:- $L = L_1 \cup L_2$ where

$L_1 = \{a^n b^n \mid n \geq 1\}$

$L_2 = \{a^n b^n c^i \mid n \geq 1, i \geq 1\}$

We construct L_1 by recursion & L_2 by concatenating the elts. of L_1 & c^i, $i \geq 1$. Define P as the set of the following productions:-

$S \to A$, $A \to ab$, $A \to aAb$, $S \to Sc$

Let $G = (\{S, A\}, \{a, b, c\}, P, S)$. For $n \geq 1$, $i \geq 0$, we have

$S \overset{*}{\Rightarrow} Sc^i \Rightarrow Ac^i \overset{*}{\Rightarrow} a^{n-1} A b^{n-1} c^i \Rightarrow a^{n-1} ab b^{n-1} c^i \Rightarrow a^n b^n c^i$

$\therefore \{a^n b^n c^i \mid n \geq 1, i \geq 0\} \subseteq L(G)$

To prove reverse inclusion, note that only S-productions are $S \to Sc$ & $S \to A$. If we begin with $S \to A$, we have to apply $A \overset{*}{\Rightarrow} a^{n-1} A b^{n-1} \Rightarrow a^n b^n$ & so $a^n b^n c^0 \in L(G)$

If we start with $S \to Sc$ & ~~Sos~~ apply it repeatedly to get Sc^i. But to get terminal string we have to apply $S \to A$. As $A \overset{*}{\Rightarrow} a^n b^n$, the resulting terminal -

Scanned with CamScanner

string is $a^n b^n c^i$. Thus, we have shown that

$$L(G) \subseteq \{a^n b^n c^i \mid n \geq 1, i \geq 0\}$$

$$\boxed{\therefore \quad L(G) = \{a^n b^n c^i \mid n \geq 1, i \geq 0\}}$$

Ques:- Let $G = (\{S, A_1, A_2\}, \{a, b\}, P, S)$, where P consists of $S \to aA_1A_2a$, $A_1 \to baA_1A_2b$, $A_2 \to A_1ab$, $aA_1 \to baa$, $bA_2b \to abab$. Test whether $w = \underline{baabba}baaa\underline{bbaba}$ is in $L(G)$.

We begin with an S-production & at every stage apply a suitable production which is likely to derive w.

$$S \Rightarrow \underline{aA_1}A_2a \Rightarrow baa\underline{A_2}a \Rightarrow baa\underline{A_1}aba$$

$$\Rightarrow baab\underline{aA_1}A_2baba \Rightarrow baabbaa\underline{A_2}baba$$

$$\Rightarrow baabbaa\underline{A_1}abbaba \Rightarrow baabbabaaabbaba = w$$

$$\Rightarrow w \in L(G).$$

Ques:- If the grammar G is given by the production $S \to aSa \mid bSb \mid aa \mid bb \mid \Lambda$, show that

(i) $L(G)$ has no strings of odd length,

(ii) any string in $L(G)$ is of length $2n$, $n \geq 0$.

(iii) # of strings of length $2n$ is 2^n.

Sol^n:- On application of any production, (except $S \to \Lambda$), a variable is replaced by two terminals & at most one variable. So, every step in any derivation increases the # of terminals by 2 except that involving $S \to \Lambda$. Thus, (i) & (ii) have been proved.

To prove (iii), consider any string w of length $2n$. Then, it is of the form $a_1a_2 \ldots a_n a_n \ldots a_2a_1$ involving 'n' parameters' $a_1, a_2, \ldots, a_n$. Each a_i can be either a or b. So, the # of such strings is 2^n. Hence, (iii) is proved.

Scanned with CamScanner

CHOMSKY CLASSIFICATION OF LANGUAGES

We can classify grammars by considering the form of the productions. Chomsky classified grammars into 4 types (Type 0-3).

TYPE - 0 GRAMMAR

A Type 0 Grammar is any phrase structure grammar without any restrictions.

DEFN :- In a production of the form $\phi A \psi \rightarrow \phi \alpha \psi$, where A is a variable, ϕ is called a left context and ψ the right context; $\phi \alpha \psi$ is the replacement string.

Ex :- (a) $ab\underline{A}bcd \longrightarrow ab\underline{AB}bcd$

$\phi = ab$ (left context) ; $\psi = bcd$ (Right context)

$\alpha = AB$

(b) $A\underline{C} \longrightarrow A$; A is left context, Λ is the right context

$\alpha = \Lambda$.

(c) $C \longrightarrow \Lambda$; The left & right contexts are Λ ; $\alpha = \Lambda$.

- A production without any restriction is called a Type - 0 production.
- A production of the form $\phi A \psi \longrightarrow \phi \alpha \psi$ is called a Type 1 production if $\alpha \neq \Lambda$. (In type 1 production, erasing of A is not permitted.)

TYPE 1 GRAMMAR OR CONTEXT-SENSITIVE OR CONTEXT-DEPENDENT GRAMMAR

A grammar is called a Type-1 Grammar if all its productions are Type-1 productions. The production $S \rightarrow \Lambda$ is allowed in Type-1 grammar, but in this case, S doesn't appear on the RHS of any production.

Scanned with CamScanner

TYPE - 1 ~~GRAMMAR~~ LANGUAGE

The language generated by Type 1 Grammar is called a Type - 1 or context - sensitive language.

<u>Remark</u>: In context - sensitive grammar G, $S \to \Lambda$ is allowed for including Λ in $L(G)$. Apart from $S \to \Lambda$, all other productions do not decrease the length of the working string.

i.e. $|\phi A \psi| \leq |\phi \alpha \psi|$ as $\alpha \neq \Lambda$

Note that if $\alpha \longrightarrow \beta$ is any production, ~~$\alpha, \beta \in V_N^*$~~ such that $|\alpha| \leq |\beta|$, then it need not be Type - 1 production

eg :- $BC \longrightarrow CB$ is not of Type - 1.

However, such productions can be replaced by productions of Type - 1.

<u>Theorem</u>:- Let G be a Type 0 Grammar. ~~For constructing~~ Then, we can find an equivalent grammar G_1 in which each production is either of the form $\alpha \longrightarrow \beta$, $\alpha, \beta \in V_N^*$ (strings of variables), or of the form $A \longrightarrow a$, $A \in V_N$, $a \in \Sigma$. G_1 is of Type 1, 2 or 3 according to as G is of Type 1, 2 or 3.

<u>PROOF</u>:- We construct G_1 as follows. For constructing productions of G_1, consider a production $\alpha \longrightarrow \beta$ in G, where α or β has some terminals. In both α & β we replace every terminal by a new variable C_a & get α' and β'. Thus, corresponding to every ~~α and~~ $\alpha \to \beta$, where α or β contains some terminals, we construct $\alpha' \longrightarrow \beta'$ & productions of the form $C_a \longrightarrow a$ for every terminal a appearing in α or β.

The productions of G_1 are the new productions, we have obtained through above construction. For G_1 the variables are the variables of G together with the new variables (C_a). The terminals & start symbol of G_1 are those of G. G_1 satisfies the required cond^ns & is equivalent to G. So, $L(G) = L(G_1)$.

Scanned with CamScanner

Def^n :- MONOTONIC GRAMMAR (or LENGTH INCREASING)

A Grammar $G = (V_N, \Sigma, P, S)$ is monotonic if every production in P is of the form $\alpha \to \beta$ with $|\alpha| \leq |\beta|$ or $S \to \Lambda$. In the second case $(S \to \Lambda)$, S doesn't appear on the RHS of any production in P.

Theorem :- Every monotonic Grammar G is equivalent to a type 1 Grammar.

Proof :- First we apply previous theorem to get an equivalent grammar G_1. Next, we construct G' equivalent to G_1 as follows : Consider a production $A_1A_2 \cdots A_m \longrightarrow B_1B_2 \cdots B_n$ with $n \geq m$ in G_1. If $m = 1$, then above production is of Type 1 (with left & right context being Λ). Suppose $m \geq 2$. Corresponding to $A_1A_2 \cdots A_m \longrightarrow B_1B_2 \cdots B_n$, we construct the following Type 1 productions introducing the new variables $C_1, C_2, \ldots, C_m$.

$$\underline{A_1} A_2 \cdots A_m \longrightarrow \underline{C_1} A_2 \cdots A_m$$

$$C_1 \underline{A_2} \cdots A_m \longrightarrow C_1 \underline{C_2} A_3 \cdots A_m$$

$$C_1 C_2 \underline{A_3} \cdots A_m \longrightarrow C_1 C_2 C_3 A_4 \cdots A_m \cdots$$

$$\vdots$$

$$C_1 C_2 \cdots C_{m-1} \underline{A_m} \longrightarrow C_1 C_2 \cdots C_m B_{m+1} B_{m+2} \cdots B_n$$

$$\underline{C_1} C_2 \cdots C_{m-1} C_m \xrightarrow{B_{m+1} \cdots B_n} B_1 C_2 \cdots C_m B_{m+1} \cdots B_n$$

$$B_1 \underline{C_2} \cdots C_{m-1} A_m \longrightarrow B_1 B_2 \cdots C_m B_{m+1} \cdots B_n$$

$$\cdots$$

$$B_1 B_2 \cdots \underline{C_m} B_{m+1} \cdots B_n \longrightarrow B_1 B_2 \cdots \underline{B_m} \cdots B_n$$

The above construction can be explained as follows :- The production $A_1A_2 \cdots A_m \longrightarrow B_1B_2 \cdots B_n$ is not of Type 1 as we replace more than one symbol on LHS. In the

Scanned with CamScanner

chain of productions we have constructed, we replace A_1 by C_1, A_2 by C_2 ..., A_m by $C_m B_{m+1} \cdots B_n$. Afterwards, we start replacing C_1 by B_1, C_2 by B_2, etc. As we replace only one variable at a time, these productions are Type-1.

We repeat the construction for every production in G_1 which is not of Type 1. For the new grammar G', the variables are the variables of G_1 together with new-variables. The productions of G' are the new Type 1-productions obtained through the above construction. The terminals & start symbol of G' are those of G_1.

G' is context-sensitive & from the construction it is easy to see that $L(G') = L(G_1) = L(G)$.

<u>Def^n</u>:- A <u>type 2 production</u> is a production of the form - $A \to \alpha$, where $A \in V_N$ and $\alpha \in (V_N \cup \Sigma)^*$. In other words, the LHS has no left or right context.

<u>example</u>:- $S \to Aa$, $A \to a$, $B \to abc$, $A \to \Lambda$

TYPE 2 GRAMMAR OR CONTEXT-FREE GRAMMAR

A grammar is called a Type-2 grammar if it contains only type-2 productions. It is also called context-free (as A can be replaced by α in any context).

TYPE 2 LANGUAGE

A language generated by a context-free grammar is called a Type 2 language or a context-free language.

TYPE 3 PRODUCTION

A production of the form $A \to a$ or $A \to aB$, where $A, B \in V_N$ and $a \in \Sigma$, is called a Type 3 production.

Scanned with CamScanner

Defn :- TYPE 3 GRAMMAR OR REGULAR GRAMMAR

A grammar is called a type 3 or regular grammar if all its productions are type 3 productions. A production $S \rightarrow \Lambda$ is allowed in type 3 grammar, but in this case S doesnot appear on the RHS of any production.

TYPE 3 LANGUAGE

The language generated by Type-3 Grammar is called Type 3 language or Regular Language.

Ques:- Find the highest type number which can be applied to the following productions :- (∵ Type 3 Grammar ⇒ Type 2 ⇓ Type 1 ⇓ Type 0 Grammar)

(a) $S \rightarrow Aa$ (Type 2), $A \rightarrow c \mid Ba$ (Type 3 ✓, Type 2), $B \rightarrow abc$ (Type 2)

So, highest type no. is Type 2.

IN TERMS OF RESTRICTIONS :-

Type 0 < Type 1 < Type 2 < Type 3

⟶ Increasing Restrictions

(b) $S \rightarrow ASB \mid d$ (Type 2, Type 3), $A \rightarrow aA$ (Type 3) } Highest Type No. = Type 2

(c) $S \rightarrow aS \mid ab$ (Type 3, Type 2) } Highest Type No. = Type 2.

LANGUAGES & THEIR RELATION

$\mathcal{L}_0$ = Type 0 languages

$\mathcal{L}_{cs1}$ = Type 1 " or content-sensitive

$\mathcal{L}_{cf1}$ = Type 2 " " " -free

$\mathcal{L}_{r1}$ = Type 3 " or Regular languages

Scanned with CamScanner

From the definitions of various languges ; it follows

$$\boxed{\mathcal{L}_{r1} \subsetneq \mathcal{L}_{cf1} \subsetneq \mathcal{L}_{cs1} \subsetneq \mathcal{L}_0}$$

Ques :- Classify the following grammars :-

i) $G = (\{S\}, \{0,1\}, \{S \to 0S1, S \to \Lambda\}, S)$ Contexet free.

ii) $G = (\{S\}, \{a\}, \{S \to SS\}, S)$ Contexet free

iii) $G = (\{S, C\}, \{a, b\}, P, S)$; $P = \{S \to aCa, C \to aCa \mid b\}$ Contexet free

iv) $G = (\{S\}, \{a, b\}, \{S \to aS \mid bS \mid a \mid b\}, S)$ Regular

Remark :-

i) Two grammars of different types may generate the same language.

example :- ★ G : $S \to aS \mid bS \mid a \mid b$ $\to$ $L(G) = \{a, b\}^+$.
(Regular Grammar)

★ G' : $S \to SS \mid aS \mid bS \mid a \mid b$ $\boxed{L(G') = L(G) = \{a, b\}^+}$
(Contexet - free)

ii) The type of a given grammar is easily decided by the nature of productions. But to decide on the type of a given subset of Σ^*, it is more difficult because same set of strings may be generated by grammars of different types as seen in Remark (i).

ALGORITHM TO TEST whether $\omega \in L(G)$ where $|\omega| = n$.

1. Construct $W_1, W_2, \ldots$ using

a) $W_0 = \{S\}$

b) $W_{i+1} = W_i \cup \{\beta \in (V_N \cup \Sigma)^* \mid \exists\, \alpha \in W_i \text{ s.t. } \alpha \Rightarrow \beta \text{ and } |\beta| \le n\}$

Terminate the construction when $W_{k+1} = W_k$ for the first time.

2. If $\omega \in W_k$, then $\omega \in L(G)$. Otherwise, $\omega \notin L(G)$. (As $|W_k| \le N$, testing whether $\omega \in W_k$ requires at most N steps).

Scanned with CamScanner

Ques :- Consider the grammar G given by
$S \to 012$, $2A_1 \to A_12$, $1A_1 \to 11$. Test whether
(a) $00112 \in L(G)$ and (b) $001122 \in L(G)$.

Solⁿ :- (a) To test whether $w = 00112 \in L(G)$, we construct $W_0, W_1, \ldots$ etc. $|w| = 5$.

$W_0 = \{S\}$; $W_1 = \{012, S, 0SA_12\}$

$W_2 = \{012, S, 0SA_12\}$. As $W_2 = W_1$, we terminate

(Although, $0SA_12 \to 0012A_12$, we can't include it in W_2 as length > 5). Then $00112 \notin W_2 \Rightarrow 00112 \notin L(G)$.

(b) $w = 001122$ $|w| = 6$.

$W_0 = \{S\}$; $W_1 = \{012, S, 0SA_12\}$

$W_2 = \{012, S, 0SA_12, 0012A_12\}$

$W_3 = \{012, S, 0SA_12, 0012A_12, 001A_122\}$

$W_4 = \{012, S, 0SA_12, 0012A_12, 001A_122, 001122\}$

$W_5 = W_4$; we terminate

Then, $001122 \in W_4$. Thus, $001122 \in L(G)$.

OPERATION ON LANGUAGES

We consider the effect of applying set operations on $\mathcal{L}_0$, $\mathcal{L}_{cs1}$, $\mathcal{L}_{cf1}$, $\mathcal{L}_{r1}$. Let A and B be any 2 sets of strings. Then, the concatenation AB of A & B is defined by :-

$AB = \{uv \mid u \in A, v \in B\}$ (Here, uv is the concatenation of strings u and v)

We define A^1 as A & A^{n+1} as A^nA $\forall n \geq 1$

The transpose set A^T of A is defined as :-

$A^T = \{u^T \mid u \in A\}$ (Reversing the string)

Scanned with CamScanner

Theorem: Each of the classes $\mathcal{L}_0$, $\mathcal{L}_{cs1}$, $\mathcal{L}_{cf1}$, $\mathcal{L}_{r1}$ is closed under union.

Proof:- Let L_1 and L_2 be 2 languages of the same type i. Then, we can find the grammars

$$G_1 = (V_N', \Sigma_1, P_1, S_1) \quad \text{and} \quad G_2 = (V_N'', \Sigma_2, P_2, S_2) \quad \text{of}$$

type i generating L_1 and L_2 respectively. So, any production in G_1 or G_2 is either $\alpha \to \beta$ $(\alpha, \beta \in V_N)$ or $A \to a$, where $A \in V_N$, $a \in \Sigma$. (Using Theorem).

We can further assume that $V_N' \cap V_N'' = \phi$. (This is achieved by renaming the variables of V_N'' if they occur in V_N')

Define a new grammar G_u as follows:

$$G_u = (V_N' \cup V_N'' \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_u, S)$$

where S is a new symbol, i.e. $S \notin V_N' \cup V_N''$.

$$P_u = P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}$$

We prove $L(G_u) = L_1 \cup L_2$ as follows:

If $w \in L_1 \cup L_2$, then $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} w$ or $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w$. Therefore,

$S \underset{G_u}{\Rightarrow} S_1 \overset{*}{\underset{G_1}{\Rightarrow}} w$ or $S \underset{G_u}{\Rightarrow} S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w$ i.e. $w \in L(G_u)$.

Thus, $L_1 \cup L_2 \subseteq L(G_u)$.

<u>To prove</u> $L(G_u) \subseteq L_1 \cup L_2$, consider a derivation of w. The first step should be $S \Rightarrow S_1$ or $S \Rightarrow S_2$. If $S \Rightarrow S_1$ is the first step, in the subsequent steps S_1 is changed. As $V_N' \cap V_N'' = \phi$ these steps should involve only the variables of V_N' and the productions we apply are in P_1. So, $S \overset{*}{\underset{G_1}{\Rightarrow}} w$. Illy, if the first step is $S \Rightarrow S_2$ then, $S \underset{G_u}{\Rightarrow} S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w$. Thus, $L(G_u) = L_1 \cup L_2$. Also, $L(G_u)$ is of type 0 or type 2 according to as L_1 and L_2 are of type 0 or type 2. If $\Lambda \notin L_1 \cup L_2$, then $L(G_u)$ is of type 3 or type 1 according to as L_1 and L_2 are of type 3 or type 1.

Suppose $\Lambda \in L_1$. In this case, define

Scanned with CamScanner

$G_u = (V_N' \cup V_N'' \cup \{S, S'\}, \Sigma_1 \cup \Sigma_2, P_u, S')$

where (i) S' is the new symbol, i.e. $S' \notin V_N' \cup V_N'' \cup \{S\}$, and

(ii) $P_u = P_1 \cup P_2 \cup \{S' \to S, S \to S_1, S \to S_2\}$. So, $L(G_u)$ is of type 1 or type 3 according to as L_1 and L_2 are of type 3 or type 1.

When $\Lambda \in L_2$, the proof is similar.

Theorem:- Each of the classes $\mathcal{L}_0, \mathcal{L}_{cs1}, \mathcal{L}_{cf1}, \mathcal{L}_{r1}$ is closed under concatenation.

Proof:- Let L_1 and L_2 be 2 languages of type i. Then, we get $G_1 = (V_N', \Sigma_1, P_1, S_1)$ and $G_2 = (V_N'', \Sigma_2, P_2, S_2)$ of the same type i. We have to prove that L_1L_2 is of type i.

Construct a new grammar G_{con} as follows:

$$G_{con} = (V_N' \cup V_N'' \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_{con}, S) \text{ where } S \notin V_N' \cup V_N''$$

$$P_{con} = P_1 \cup P_2 \cup \{S \to S_1S_2\}$$

We prove $L_1L_2 = L(G_{con})$. If $\omega = \omega_1\omega_2 \in L_1L_2$, then

$$S_1 \overset{*}{\underset{G_1}{\Rightarrow}} \omega_1 \quad , \quad S_2 \overset{*}{\underset{G_2}{\Rightarrow}} \omega_2 .$$

So, $$S \underset{G_{con}}{\Rightarrow} S_1S_2 \overset{*}{\underset{G_{conc}}{\Rightarrow}} \omega_1\omega_2 \qquad \therefore L_1L_2 \subseteq L(G_{con}).$$

If $\omega \in L(G_{con})$, then the first step in the derivation of ω is $S \Rightarrow S_1S_2$. As $V_N' \cap V_N'' = \phi$ & the productions in G_1 & G_2 involve only the variables (except those of form $A \to a$), we have $\omega = \omega_1\omega_2$ where $S \overset{*}{\underset{G_1}{\Rightarrow}} \omega_1$ & $S \overset{*}{\underset{G_2}{\Rightarrow}} \omega_2$. Thus, $L_1L_2 = L(G_{con})$.

Also, G_{con} is of type 0 or 2 according to as G_1 and G_2 are of type 0 or 2. The above construction is sufficient when when G_1 & G_2 are of type 3 or 1 provided $\Lambda \notin L_1 \cup L_2$.

Suppose G_1 & G_2 are of type 1 or 3 and $\Lambda \in L_1$ or $\Lambda \in L_2$.

Let $L_1' = L_1 - \{\Lambda\}$, $L_2' = L_2 - \{\Lambda\}$. Then

Scanned with CamScanner

$$L_1L_2 = \begin{cases} L_1'L_2' \cup L_2' & \text{if } \Lambda \in L_1 \text{ but } \Lambda \notin L_2 \\ L_1'L_2' \cup L_1' & \text{if } \Lambda \notin L_1 \text{ but } \Lambda \in L_2 \\ L_1'L_2' \cup L_1' \cup L_2' \cup \{\Lambda\} & \text{if } \Lambda \in L_1 \text{ and } \Lambda \in L_2 \end{cases}$$

As shown already, $\mathcal{L}_{cs1}$ and $\mathcal{L}_{r1}$ are closed under union. L_1L_2 is of type 1 or 3 according as $\mathcal{L}_1$ & $\mathcal{L}_2$ are of type 1 or 3.

Theorem:- Each of the classes $\mathcal{L}_0, \mathcal{L}_{cs1}, \mathcal{L}_{cf1}, \mathcal{L}_{r1}$ is closed under the transpose operation.

Proof:- Let L be a language of type i. Then, $L = L(G)$, where G is of type i.

We construct a new grammar G^T as follows:-

$G^T = (V_N, \Sigma, P^T, S)$, where the productions of P^T are constructed by reversing the symbols on L.H.S. and RHS of every production in P. Symbolically, $\alpha^T \rightarrow \beta^T$ is in P^T if $\alpha \rightarrow \beta$ is in P.

From the construction it is obvious that G^T is of type 0, 1 or 2 according to as G is of type 0, 1 or 2 and $L(G^T) = L^T$.

(For regular grammar, proof is given later).

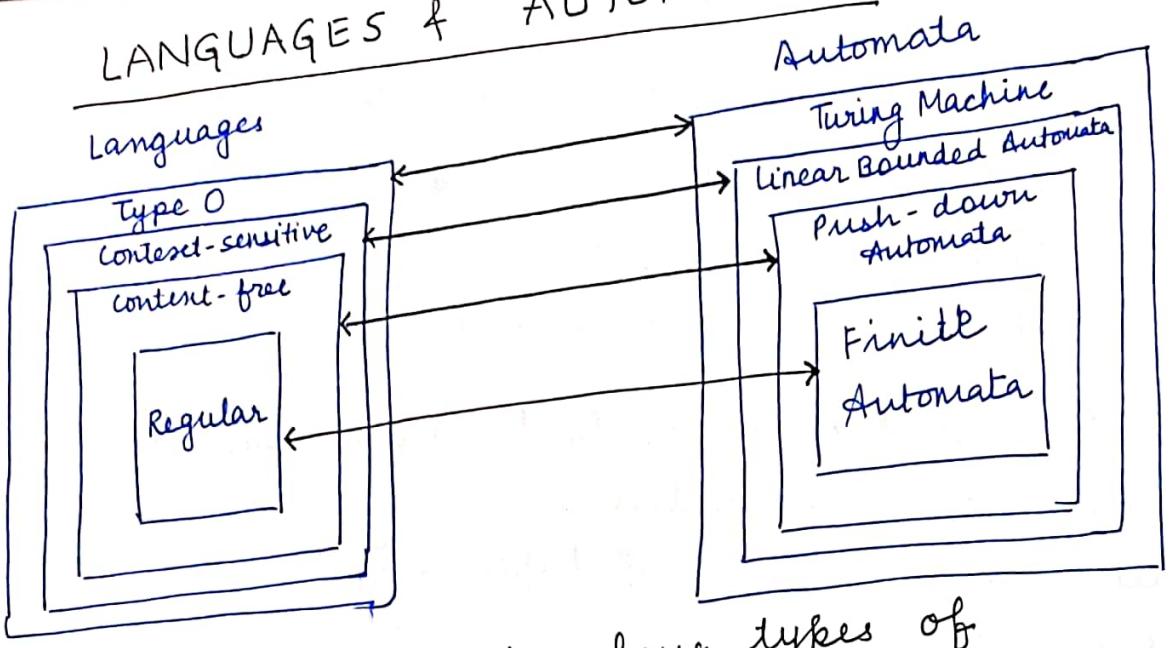
Theorem:- (i) Each of the families $\mathcal{L}_{01}, \mathcal{L}_{cs1}, \mathcal{L}_{r1}$ is closed under intersection.

(ii) $\mathcal{L}_{cf1}$ is not closed under intersection. But the intersection of a context free language and a regular language is context free.

Proof:-

Scanned with CamScanner

LANGUAGES & AUTOMATA



Accepting Devices for four types of languages

Ques :- Construct a context free grammar generating :-

(i) $L_1 = \{ a^n b^{2n} \mid n \geq 1 \}$ (ii) $L_2 = \{ a^m b^n \mid m > n,\ m, n \geq 1 \}$

Solⁿ :- (i) $G = (V_N, \Sigma, P, S)$ where $V_N = \{S\}$

$\Sigma = \{a, b\}$

$P = \{ S \rightarrow aSbb,\ S \rightarrow abb \}$

(ii) $P = \{ S \rightarrow aS \mid aA,\ A \rightarrow aAb,\ A \rightarrow ab \}$

(iii) $L = \{ w \in \{a,b\}^* \mid$ the # of a's in w is divisible by 3 $\}$

Consider, a string over $\{a,b\}$ having 3 a's. These 3 a's appear amidst the strings of b. A typical string is $b^m a b^n a b^s a b^t$. For generating b^m, we can have the productions $S \rightarrow bS$. For getting the first a, we can have $S \rightarrow aA$. For getting b^n afterwards, we have $A \rightarrow bA$. For getting the second a, we can have $A \rightarrow aB$. For getting b^s, we have $B \rightarrow bB$. For getting the third a & repetition of this pattern, we can have $B \rightarrow a \mid aS$

Now, we construct G as follows:

$G = (\{S, A, B\}, \{a, b\}, P, S)$ where P consists of

Scanned with CamScanner

$S \to bS \mid aA$, $A \to bA \mid aB$, $B \to bB \mid a$, $B \to aS$.

A string in L is of the form $y_1 y_2 \cdots y_n$ where each y_i is of the form $b^m a b^n a b^s a$ for some nonnegative integers m, n & s.

$S \overset{*}{\Rightarrow} b^m S \Rightarrow b^m a A \overset{*}{\Rightarrow} b^m a b^n A \Rightarrow b^m a b^n a B \overset{*}{\Rightarrow} b^m a b^n a b^s S$.

Hence, G is the required grammar.

(iv) $\mathcal{L}(G) = \{ w \in \{a, b\}^* \mid w$ has an equal # of a's and b's$\}$

$S \to aB \mid bA$, $A \to aS \mid bAA \mid a$ $B \to bS \mid aBB \mid b$

(v) $\mathcal{L}(G) = \{ w \in \{a, b\}^* \mid w$ has more a's than b's$\}$

$S \to a \mid aS \mid Sa \mid bSS \mid SSb \mid SbS$

(vi) Set of all non-palindromes over $\{a, b\}$.

$S \to aSa \mid bSb \mid A$

$A \to aBb \mid bBa$

$B \to aB \mid bB \mid \Lambda$

Scanned with CamScanner