Grammer $\longrightarrow$ formal language

why not natural language; due to the ambigous nature.

Unit -2 Formal languages

Grammer: A grammer or phrase structure grammer is define using (V_n, Σ, P, S), where

starting symbol

$V_n \rightarrow$ finite non empty set, whose elements are called as variable

$\Sigma \rightarrow$ " " " ", whose elements are called terminals

$S \rightarrow$ a special variable, called starting symbol

$V_N \cap \Sigma = \phi$

$P \rightarrow$ finite set, whose elements are $\alpha \rightarrow \beta$; where α & β are string over $V_N \cup \Sigma$

α, has atleast one symbol from: V_n, $\alpha, \beta \in (V_N \cup \Sigma)^*$

the elements of P are called as production rule / Rewriting rules.

ex; $G_1 = (\{S\}, \{0,1\}, P, S)$ where

$P = \{S \to 0S1,\ S \to 01,\ S \to \Lambda\}$

$G_2 = (\{S, A\}, \{a, b\}, P, S)$

$P = \{S \to A,\ S \to a,\ A \to b\}$

• Remark: productⁿ rules cannot be reversed. i.e $S \to A \not\Rightarrow A \to S$

$(V_n \cup \Sigma)^+ = (V_n \cup \Sigma)^* \setminus \{\Lambda\}$

One-step derivation.

$\alpha \to \beta$

if α, β and γ and δ are any two strings in $(V_n \cup \Sigma)^*$ then we say $\gamma\alpha\delta$, directly derive $\gamma\beta\delta$ in G

$$(\gamma\alpha\delta \Rightarrow \gamma\beta\delta)$$

Reflexive - transitive closure.

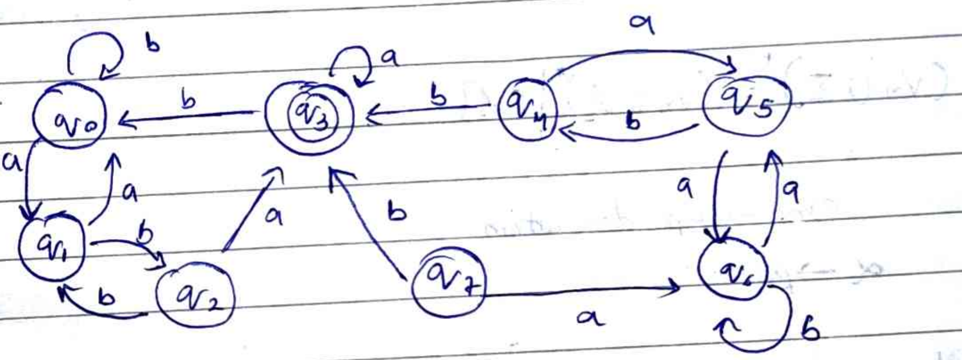
if α & β are strings on $V_n \cup \Sigma$, then we say α derives β if $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$; here $\overset{*}{\underset{G}{\Rightarrow}}$ represent the reflexive transitive closure of $\underset{G}{\Rightarrow}$ in $(V_n \cup \Sigma)^*$

The language generated by a grammer G, denoted by $L(G)$; is define $\{ w \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w \}$.

↳ elements are called - sentences...

unit - 1 Question:

Q:- minimize.



state / Σ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_3	q_0
q_4	q_5	q_3
q_5	q_6	q_4
q_6	q_5	q_6
q_7	q_6	q_3

0 - equivalent

$\Pi_0 = \{ \{q_3\}, \{q_0, q_1, q_2, q_4, q_5, q_6, q_7\} \}$

$\Pi_1 = \{ \{q_3\}, \{q_0, q_1, q_5, q_6\}, \{q_2\}, \{q_4, q_7\} \}$

$\Pi_2 = \{ \{q_3\}, \{q_2\}, \{q_0, q_6\}, \{q_4, q_7\}, \{q_1, q_5\} \}$.

ToC 26 - Aug - 2025

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

$G = (V_N, \Sigma, P, S)$ — $S \in V_N$

- V_N → Set of variables
  ↳ capital alphabets
- Σ → terminal
- P → $\alpha \to \beta$, $\alpha \in (V_N \cup \Sigma)^*$

Remark: If $S \overset{*}{\Rightarrow} \alpha$, $\alpha \in (V_N \cup \Sigma)^*$, then alpha is called the sentential form

$\text{sentential form} \xrightarrow{\times} \text{sentences}$

$\text{sentences} \to \text{sentential form}$

sentences → $w \in \Sigma^*$ (only terminal)

Note: all the elements of $L(G)$ are sentential form, but not vice versa...

Def^n: 2 Grammars G_1 & G_2 are s.t.b equivalent iff

$$L(G_1) = L(G_2)$$

Remarks: the string generated by the most recent applicat

↳ the derivat^n of a string is complete, when the working str Cannot be modified. If the string doesn't ~~cannot~~ conta any variable, it is called

(i) find $L(G)$ where $G = \{ \{S, C\}, \{a, b\}, P, S\}$ & P
$P = \{ S \to aCa, \; C \to aCa \mid b \}$.

(ii) let L be the set of all palindromes over $\{a, b\}$. Construct a grammer generating L.

(iii) find a grammer generating $L = \{ a^n b^n c^i \mid n \geq 1, \, i \geq 0 \}$.

(i) $S \Rightarrow aCa \Rightarrow a^2Ca^2 \overset{*}{\Rightarrow} a^nCa^n \Rightarrow a^nba^n$.

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

(ii) $S \to \Lambda$

$S \to a \mid b$ $\qquad S \to a \mid b \mid \Lambda$

$S \to aSa \mid bSb$

$S \Rightarrow aSa \Rightarrow abSba$

$\Downarrow^*$

S

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

$P: (S \to aSa \mid bSb \mid a \mid b \mid \Lambda)$.

(iii) $L = \{ a^n b^n c^i \mid n \geq 1 ; i \geq 0 \}$

$L = \{ab, abc, a^2b^2c^2, \dots$

$S \to ab \mid abc$

$P = \{ S \to Sc \mid A$

$A \to aAb \mid ab$

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

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

Q: let $G = (\{S, A_1, A_2\}, \{a, b\}, P, S)$ where

$P = \{ S \to aA_1A_2a,\ A_1 \to baA_1A_2b,\ A_2 \to A_1ab,$
$aA_1 \to baa,\ bA_2b \to abab \}$.

$\omega = baabbabaaabbaba\dots$

ToC 1 Sep 2025

$$G = (V_n, \Sigma, P, S) \longrightarrow L(G)$$

$$\alpha - \beta \;;\; \alpha, \beta \in (V_n \cup \Sigma)^*$$

Chomsky classification of language...

we can classify the languages by classifying the grammers using the form of productⁿ rules...

chomsky classify grammer into four types:-

Type 0 - unrestricted grammer.
Type 2 - Context free grammer.
Type 1 - context sensitive grammer.
Type 3 - Regular Grammer.

Type 0 - Grammer :

A type 0 grammer is any grammer without any restriction (type 0 productⁿ are the productⁿ without any restriction)...

$$\begin{array}{ccccc} \text{Type 0} & \longrightarrow & \text{Type 0} & \longrightarrow & \text{Type 0} \\ \downarrow & & \downarrow & & \downarrow \\ \text{Productions} & \longrightarrow & \text{Grammer} & \longrightarrow & \text{language} \end{array}$$

TOC 1 Sep 2025

$$G = (V_n, \Sigma, P, S) \longrightarrow L(G)$$

$$\alpha \rightarrow \beta \; ; \; \alpha, \beta \in (V_n \cup \Sigma)^*$$

Chomsky classification of language...

we can classify the languages by classifying the grammers using the form of product^n rules...

chomsky classify grammer into four types:-

Type 0 - unrestricted grammer.
Type 2 - Context free grammer.
Type 1 - context sensitive grammer.
Type 3 - Regular Grammer.

Type 0 - Grammer :

A type 0 grammer is any grammer without any restriction (type 0 product^n are the product^n without any restriction)...

Type 0 → Type 0 → Type 0

Productions → Grammer → language

$$\underset{\text{Type } i}{\text{Product}^m} \longrightarrow \underset{\text{Type } i}{\text{Grammer}} \longrightarrow \underset{\text{Type } i}{\text{language}} \Big\} \; i = 0,1,2,3$$

↳ ~~Type 0~~

In a production of the form

$\phi A \psi \to \phi \alpha \psi$ where

$A \in V_n$ & $\alpha \in (V_n \cup \Sigma)^*$

ϕ is called the left context & ψ is called the right context.

↳ $\phi \alpha \psi$ is called the replacement string

	ϕ	ψ
$SA \to Sb$	S	—
$ASB \to AbB$	A	B
$S \to b$	—	—
$aSA \to aSa$	aS	—

ex;

Type 1:

A production of form $\phi A \psi \to \phi \alpha \psi$; $\alpha \neq \Lambda$

↳ it contains type 1 product^{ms} only.

↳ the product^m $S \to \Lambda$ is allowed in type 1 grammer but, in this case, S shouldn't come on the RHS of any Production...

Error

$S \to \Lambda$

$S \to aSb$ ✗

the language generated by type 1 grammer is called context ~~free~~ / type 1 language....

↳ sensitive...

Type 2 :

↳ A product^n of the form.

$$A \to \alpha \ ; \ A \in V_N \ \& \ \alpha \in (V_N \cup \Sigma)^*$$

called type 2 product^n...

Type 3 :

↳ A product^n of the form → terminal followed by variable...

$$A \to a, \ A \to aB \ ; \ A, B \in V_N \ \& \ a \in \Sigma$$

↳ similar to type 1 if $S \to \Lambda$ then S shouldn't come at RHS....

$$\text{Type 3} \subseteq \text{Type 2} \subseteq \text{Type 1} \subseteq \text{Type 0}$$

Find the highest type of product^n which can be applied to the following production?

) $S \to Aa$, $A \to c \mid Ba$, $B \to abc$

) $S \to ASB \mid d$, $A \to aA$

$S \to aS \mid ab$.

Thrm: let G be a type 0 grammer, then, we can find an equivalent grammer G' in which each production is of the type $\alpha \to \beta$; $\alpha, \beta \in V_N^*$ or of the form $A \to a$, $A \in V_N, a \in \Sigma$. G' is type 1, 2, or 3 according to G, is of type 1, 2, or 3..

4-September 2025

Operation on languages

$L = \{ w \in \Sigma^* \mid cond^n \}$.

L_1 & L_2

Union: $L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ or } w \in L_2 \}$.

Concatenation: $L_1 L_2 = \{ w_1 w_2 \mid w_1 \in L_1, w_2 L_2 \}$

Transpose: $L^T = \{ w^T \mid w \in L \}$.

$w = abc$

$w^T = cba...$

L_0 : unrestricted language.. L_{cs} = context sensitive

L_{cf} : Context free L_r = regular.

new; $L_1 \to G_1 : (V_N, \Sigma_1, P_1, S_1)$

$L_2 \to G_2 : (V_N', \Sigma_2, P_2, S_2)$

* $L(G) = L_1 \cup L_2$

$G = (V_N \cup V_N', \Sigma_1 \cup \Sigma_2, P, S)$

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

Thm: each of the classes $\mathcal{L}_0, \mathcal{L}_{cf}, \mathcal{L}_{cs}, \& \mathcal{L}_r$ is closed under Union...

Proof: let $L_1 \& L_2$ be 2 languages of same type 'i' then, we can find the grammer $G_1 = (V_N', \Sigma_1, P_1, S_1)$ &

$G_2 = (V_N'', \Sigma_2, P_2, S_2)$ of type 'i' generating $L_1 \& L_2$ respectively

so, any product^n in G_1 or G_2 is either $\alpha \to \beta$, $\alpha, \beta \in V_N^*$ or $A \to a$; where $A \in V_N$ & $a \in \Sigma$...

we can further assume $V_N' \cap V_N'' = \phi$ (achieved by rearranging the ~~vertic~~ veriables of V_N'' if they occur in V_N')...

define a new grammer G_N as follow:

$G_N = (V_N' \cup V_N'', \Sigma_1 \cup \Sigma_2, P_N, S)$ where S is the new starting symbol; i.e; $S \notin V_N' \cup V_N''$.

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

we prove $L(G_N) = L_1 \cup L_2$ as follows:

If $w \in L_1 \cup L_2$, then $w \in L_1$ or $w \in L_2$ i.e;

$$S_1 \overset{*}{\Rightarrow} w \quad \text{or} \quad S_2 \overset{*}{\Rightarrow} w$$

$\therefore$ $S \Rightarrow S_1 \overset{*}{\Rightarrow} w$ or $S \Rightarrow S_2 \overset{*}{\Rightarrow} w$

i.e; $w \in L(G_N)$ thus $L_1 \cup L_2 \subseteq L(G_N)$

Next to prove $L(G_n) \subset L_1 \cup L_2$; consider a derivation of w.

The first step is $S \Rightarrow S_1$ 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 variables of V_N' and the product^n of P_1.

So, $S \overset{*}{\underset{G_1}{\Rightarrow}} w$ similarly, if first step is $S \overset{*}{\underset{G_2}{\Rightarrow}} w$.

Thus, $L(G_m) \subseteq L_1 \cup L_2$

$\square$

Hence $L(G_N) = L_1 \cup L_2$