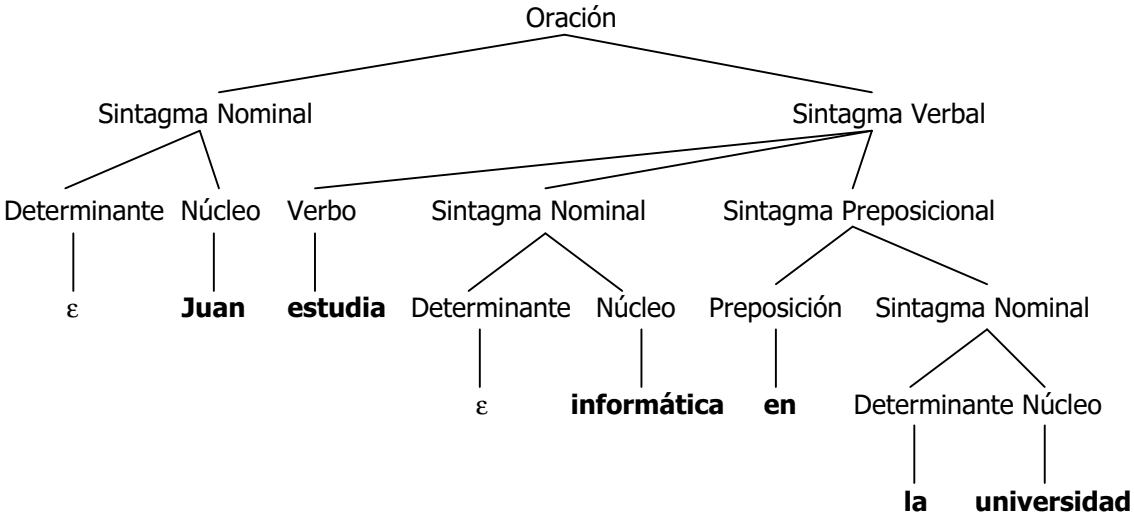
## Gramáticas

Análisis sintáctico de la frase "**Juan estudia informática en la universidad**"




<oración> → <sintagma nominal> <sintagma verbal>
<sintagma nominal> → <determinante> <núcleo>
<sintagma verbal> → <verbo> <sintagma nominal> <sintagma preposicional>
<sintagma preposicional> → <preposición> <sintagma nominal>
<determinante> → ε | **la**
<núcleo> → **Juan** | **informática** | **universidad**
<verbo> → **estudia**
<preposición> → **en**

**Definición:** Una Gramática es una cuádrupla **G ≡ (V, Σ, S, P)** donde

**V:** Es el conjunto de variables gramaticales o símbolos no terminales.
V es finito, $V \neq \emptyset$

**Σ:** Es el conjunto de símbolos terminales o alfabeto de la gramática. Σ es finito, $\Sigma \neq \emptyset$, $\Sigma \cap V = \emptyset$.

**S:** $S \in V$, es el símbolo de arranque o símbolo inicial de la gramática

**P:** Es el conjunto de reglas de producción.
$P \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ $P \neq \emptyset$
Los elementos de P son pares donde la primera componente es una variable sintáctica y la segunda una cadena de símbolos terminales (de Σ) y no terminales (de V). En el primer elemento del par ha de haber al menos un símbolo no terminal (de V)
Estas componentes se suelen representar de la forma:
$\alpha \rightarrow \beta$ o bien $\alpha ::= \beta$ $\alpha, \beta \in (V \cup \Sigma)^*$
Se suele especificar P listando todas sus producciones (reglas) colocando primero las del símbolo de arranque, S

## Convenios a la hora de escribir gramáticas

- Los símbolos **terminales** se representan mediante letras minúsculas del principio del alfabeto latino: **a, b, c...**
- Los símbolos **no terminales** se representan usando las primeras letras del alfabeto latino en mayúsculas: A, B, C, D,...
- Los **símbolos gramaticales** (terminales y no terminal) se representarán utilizando en mayúsculas las últimas letras del alfabeto latino: X, Y, Z, W,...
- Las **cadenas de símbolos terminales** se representarán usando en minúsculas las últimas letras del alfabeto latino: w, x, y, z,...
- Las **cadenas de símbolos gramaticales** (terminales y no terminales) se representarán usando letras del alfabeto griego: $\alpha$, $\beta$, $\gamma$, $\delta$, ...
- Si existen **diferentes alternativas** para un mismo símbolo no terminal, las diferentes alternativas se listan separadas por barras verticales: $A \rightarrow \alpha|\beta|...|\gamma|\zeta$

## Clasificación de Chomsky

- **Gramáticas estructuradas por frases, semi-Thue o de tipo 0:**

Son las más generales. Sus reglas no tienen ninguna restricción y son de la forma:

$$\alpha \rightarrow \beta$$

Donde $\alpha$ y $\beta$ son cadenas arbitrarias de símbolos gramaticales con $\alpha \neq \varepsilon$.

- **Gramáticas sensibles al contexto o de tipo 1:**

Sus reglas son de la forma:

$$\alpha \rightarrow \beta$$

con la restricción $|\beta| \geq |\alpha|$. Se puede demostrar que este tipo de reglas son equivalentes a las de la forma:

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

- **Gramáticas independientes del contexto (CFG) o de tipo 2:**

Sus reglas tienen la forma

$$A \rightarrow \alpha$$

La mayor parte de los lenguajes de programación están generados por una CFG

- **Gramáticas regulares o de tipo 3:**

Si todas las producciones de una gramática son de la forma:

$$A \rightarrow wB \mid w \qquad \text{o bien}$$
$$A \rightarrow Bw \mid w$$

Donde A y B son variables sintácticas y w una cadena arbitraria de terminales (podría ser vacía), se dice que la gramática es lineal por la derecha (izquierda). Una gramática lineal por la izquierda o por la derecha se denomina una gramática regular.

**Simplificación de una Gramática Independiente del Contexto**

## *Eliminación de símbolos inútiles*

Etapa 1

$V' = \varnothing$

**Para cada** producción de la forma $A \rightarrow w$ **hacer**

$V' = V' \cup \{A\}$

**while** (V' cambie) **do**

**Para cada** producción de la forma $B \rightarrow \alpha$ **hacer**

**if** (todas las variables de $\alpha$ pertenecen a V')

$V' = V' \cup \{B\}$

Eliminar todas las variables ($C \in V$) que estén en V y no en V'

Eliminar todas las producciones donde aparezca una variable de las eliminadas en el paso anterior

Etapa 2

- V'' y J son conjuntos de variables (de V)
- J son las variables pendientes de analizar
- T' es un conjunto de símbolos terminales (de $\Sigma$)

$J = \{S\}$

$V'' = \{S\}$

$T' = \varnothing$

**while** ($J \neq \varnothing$) **do**

Extraer un no terminal A de J: $J = J - \{A\}$

**Para cada** producción de la forma $A \rightarrow \alpha$

**begin**

**Para cada** no-terminal B en $\alpha$

**if** ($B \notin V''$) then

**begin**

$J = J \cup \{B\}$

$V'' = V'' \cup \{B\}$

**end**

Poner todos los terminales de $\alpha$ en T'

**end**

Eliminar todas las variables que no estén en V'' y todos los terminales que no estén en T'.

Eliminar todas las producciones donde aparezca un símbolo o variable de los eliminados.

## *Cálculo del conjunto de variables anulables, H*

$H = \varnothing$
**Para cada** producción de la forma $A \rightarrow \varepsilon$
    $H = H \cup \{A\}$
**while** (H cambie) **do**
    **Para cada** producción $B \rightarrow A_1A_2 \ldots A_n$ donde $A_i \in H \; \forall \; i = 1, 2, \ldots n$ **hacer**
        $H = H \cup \{B\}$

## *Eliminación de producciones vacías*

Eliminar todas las producciones vacías ($A \rightarrow \varepsilon$)
**Para cada** producción de la forma $A \rightarrow \alpha_1\alpha_2 \ldots \alpha_n$ $(\alpha_i \in (\Sigma \cup V))$
    Eliminar la producción $A \rightarrow \alpha_1\alpha_2 \ldots \alpha_n$
    Añadir todas las producciones $A \rightarrow \beta_1\beta_2 \ldots \beta_n$
    donde: $\beta_i = \alpha_i$ si $\alpha_i$ no es anulable
    $(\beta_i = \alpha_i) \vee (\beta_i = \varepsilon)$ si $\alpha_i$ es anulable
    $\beta_i$ no es $\varepsilon \; \forall$ i (no se introducen producciones nulas)

## *Eliminación de producciones unitarias*

- H será el conjunto de parejas (A, B) tales que $A \Rightarrow B$

$H = \varnothing$
**Para cada** producción de la forma $A \rightarrow B$
    $H = H \cup \{(A, B)\}$
**while** (H cambie) **do**
    **Para cada** par de parejas de la forma (A, B) (B, C)
        **if** $((A, C) \notin H)$ then
            $H = H \cup \{(A, C)\}$
Eliminar todas las producciones unitarias
**Para cada** producción $B \rightarrow \alpha$
    **Para cada** pareja $(A, B) \in H$
        Añadir a la gramática una producción $A \rightarrow \alpha$

## Escritura de una gramática en forma normal de Chomsky

**Para cada** producción de la forma $A \to \alpha_1\alpha_2 \ldots \alpha_n$ $(\alpha_i \in (\Sigma \cup V), n \geq 2)$

Para cada $\alpha_i$ si $\alpha_i = a \in \Sigma$

Se añade la producción $C_a \to a$

Se cambia $\alpha_i$ por $C_a$ en la producción $A \to \alpha_1\alpha_2 \ldots \alpha_n$

Para cada producción de la forma $A \to B_1B_2 \ldots B_m$ $(m \geq 3)$

Añadir (m-2) variables $D_1, D_2, \ldots, D_{m-2}$ (distintas para cada producción)

Reemplazar la producción $A \to B_1B_2 \ldots B_m$ por las producciones

$A \to B_1D_1$, $D_1 \to B_2D_2$ , ..., $D_{m-2} \to B_{m-1}B_m$

## Eliminación de la recursividad por la izquierda

***Recursividad inmediata:***

Sustituir las reglas de la forma: $A \to A\alpha \mid \beta$

por el par de reglas: $A \to \beta \mid \beta T$

$T \to \alpha \mid \alpha T$

***Recursividad indirecta:***

Ordenar los símbolos no terminales en algún orden arbitrario: $V=\{A_1, A_2, \ldots, A_n\}$

**for** i:= 1 to n **do**

begin

**for** h := 1 to i-1 **do**

**begin**

Sustituir cada producción $A_i \to A_h\gamma$ por las producciones

$A_i \to \delta_1\gamma \mid \delta_2\gamma \mid \ldots \mid \delta_k\gamma$ donde

$A_h \to \delta_1 \mid \delta_2 \mid \ldots \mid \delta_k$ son las producciones de $A_h$

**end**

Eliminar la recursividad inmediata de $A_i$

**end**

**Nota:** el algoritmo, además de eliminar la recursividad por la izquierda garantiza que si $(A_i \to A_h\alpha) \in P$, entonces $i < h$.

**Conversión a forma normal de Greibach**

1. Ordenar los símbolos no terminales $V=\{A_1, A_2, ..., A_n\}$ de modo que toda producción $(A_i \rightarrow \alpha) \in P$ verifique que el primer símbolo de $\alpha$ sea un terminal o una variable $A_h$ tal que $i < h$ (este orden se consigue con el algoritmo de eliminación de la recursividad por la izda.)
2. i := n - 1
3. **while** $(i \neq 0)$ **do**
   **begin**
   a) Sustituir las reglas de la forma $A_i \rightarrow A_h\alpha$ con $i < h$ por
      $A_h \rightarrow \beta_1\alpha \mid \beta_2\alpha \mid ... \mid \beta_m\alpha$
      donde $A_h \rightarrow \beta_1 \mid \beta_2 \mid ... \mid \beta_m$ son todas las producciones de $A_h$
      (las cadenas $\beta_i$ comienzan todas por un símbolo terminal)
   b) i := i - 1
   **end**
4. En este punto todas las reglas (salvo $S \rightarrow \varepsilon$, si existe) comienzan por un símbolo terminal. Para cada regla $A \rightarrow aX_1X_2...X_k$, sustituir todos los $X_i$ que sean terminales por nuevos símbolos no terminales $Z_i$ y añadir la producción $Z_i \rightarrow X_i$

**Algoritmo de análisis CYK (Cocke-Younger-Kasami)**

Entrada: una cadena $w=a_1a_2...a_n$ y una CFG escrita en forma normal de Chomsky.
Salida: el algoritmo determina si $w \in L(G)$: $w \in L(G) \Leftrightarrow S \in V_{1n}$

```
for i := 1 to n do
      V_i1 := {A | A → a    es una producción y el iésimo símbolo de w es a}
for j := 2 to n do
      for i := 1 to n - j + 1 do
            begin
            V_ij := ∅;
            for k := 1 to j - 1 do
                  V_ij := V_ij ∪ {A|A → BC  es una producción, B∈V_ik, C∈V_i+k,j-k};
            end
```