

Una **gramática** es un conjunto de reglas que nos permite generar todas las palabras válidas de un lenguaje. Es como un “recetario” para crear frases correctas.

## 4.1 Definición Formal:

$$G = (V, T, P, S)$$

Toda gramática tiene **4 componentes**:

**1.  $V$  (Variables o No Terminales - NT)** Son símbolos que se pueden sustituir por otras cosas. Se escriben en MAYÚSCULAS.

**Ejemplo:** S, A, B, E (Expresión), Número, etc.

**2.  $T$  (Terminales)** Son símbolos que aparecen en las palabras finales y NO se pueden sustituir. Se escriben en minúsculas o son caracteres específicos.

**Ejemplo:** a, b, 0, 1, +, -, números, palabras del lenguaje final

**3.  $P$  (Producciones o Reglas)** Son las reglas de transformación. Tienen la forma:  
**Variable → combinación de terminales y/o variables**

**4.  $S$  (Símbolo Inicial o Axioma)** Es la variable por la que SIEMPRE empezamos.

### Ejemplo Completo Simple:

Queremos crear una gramática para el lenguaje: **L = {a, aa, aaa, aaaa, ...}** (una o más 'a's)

$G = (V, T, P, S)$  donde:

```
V = {S}           // Solo tenemos una variable
T = {a}           // Solo tenemos un terminal
S = S             // Símbolo inicial
P = {
    S → aS        // S se puede convertir en 'a' seguido de S
    S → a          // S se puede convertir en solo 'a'
}
```

**¿Cómo generamos palabras?** - Para “a”:  $S \rightarrow a \checkmark$  - Para “aa”:  $S \rightarrow aS \rightarrow aa \checkmark$  - Para “aaa”:  $S \rightarrow aS \rightarrow aaS \rightarrow aaa \checkmark$

## 4.2 Clasificación de Gramáticas (Jerarquía de Chomsky)

Las gramáticas se clasifican de **menos restrictivas** a **más restrictivas**:

### Tipo 0: Sin Restricciones

- **Regla:**  $x \rightarrow y$ , donde x puede ser cualquier combinación (pero al menos un símbolo), y puede ser cualquier cosa (incluso vacío)
- Son las más potentes pero también las más difíciles de analizar

### **Tipo 1: Sensibles al Contexto**

- **Regla:**  $\alpha \rightarrow \beta$  donde  $|\alpha| \leq |\beta|$  (la parte derecha debe ser igual o más larga)
- **Importante:** La sustitución depende del “contexto” (los símbolos alrededor)
- Ejemplo:  $aSb \rightarrow aXYb$  (sustituimos S por XY, pero solo si está entre ‘a’ y ‘b’)

### **Tipo 2: Independientes del Contexto (GIC)**

- **Regla:**  $x \rightarrow y$  donde **x debe ser UN SOLO no terminal**
- La sustitución NO depende del contexto
- Ejemplo:  $S \rightarrow aSb$  (sustituimos S sin importar qué hay alrededor)

### **Tipo 3: Regulares**

- Las más restrictivas
- **Reglas lineales:**  $A \rightarrow aB$ ,  $A \rightarrow Ba$ , o  $A \rightarrow a$
- Ejemplo:  $S \rightarrow aS$ ,  $S \rightarrow a$

**Relación:** Tipo 3  $\subset$  Tipo 2  $\subset$  Tipo 1  $\subset$  Tipo 0

## **4.3. Gramáticas Regulares (Tipo 3)**

Son un subconjunto especial de las GIC.

### **Lineal por la Derecha:**

$A \rightarrow xB$  (terminal(es) seguido de una variable)  
 $A \rightarrow x$  (solo terminal(es))

**Ejemplo:** Lenguaje de palabras que terminan en ‘b’:

$S \rightarrow aS \mid bS \mid b$

- Genera: b, ab, aab, bb, abb, etc.

### **Lineal por la Izquierda:**

$A \rightarrow Bx$  (variable seguida de terminal(es))  
 $A \rightarrow x$  (solo terminal(es))

### **Ejemplo:**

$S \rightarrow Sa \mid Sb \mid a$

- Genera: a, aa, ab, aaa, aab, etc.

**Importante:** NO se pueden mezclar reglas lineales por la derecha con lineales por la izquierda en la misma gramática regular.

## **4.4 Lenguaje de una Gramática**

$$L(G) = \{w \in T^* \mid S \Rightarrow *w\}$$

Esto significa: El lenguaje generado por  $G$  es el conjunto de todas las palabras formadas **solo por terminales** que se pueden derivar desde  $S$ .

### Ejemplo:

$S \rightarrow aSb \mid$   
 $T = \{a, b\}$

**Derivaciones:** -  $S \Rightarrow \epsilon \Rightarrow$  palabra: "" (cadena vacía) -  $S \Rightarrow aSb \Rightarrow ab \Rightarrow$  palabra: "ab" -  $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb \Rightarrow$  palabra: "aabb" -  $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb \Rightarrow$  palabra: "aaabbb"

$$L(G) = \{a^n b^n \mid n \geq 0\} = \{\epsilon, ab, aabb, aaabbb, \dots\}$$

## 4.5 Árboles de Derivación

Un **árbol de derivación** es una representación visual de cómo se genera una palabra.

### Características:

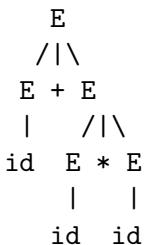
1. **Raíz:** Símbolo inicial  $S$
2. **Nodos internos:** Variables (no terminales)
3. **Hojas:** Terminales o  $\epsilon$
4. Si un nodo  $A$  tiene hijos  $X_1, X_2, \dots, X_k$ , entonces existe la regla  $A \rightarrow X_1 X_2 \dots X_k$

### Ejemplo:

Gramática:

$E \rightarrow E + E$   
 $E \rightarrow E * E$   
 $E \rightarrow (E)$   
 $E \rightarrow id$

Para la expresión: **id + id \* id**



**Lectura del árbol (hojas de izquierda a derecha):** id + id \* id ✓

## 4.6 Ambigüedad

Una gramática es **ambigua** si una misma palabra puede tener **DOS O MÁS árboles de derivación diferentes**.

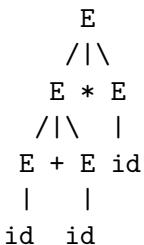
### Ejemplo Clásico:

Gramática:

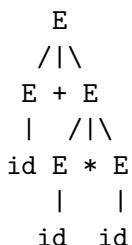
$$\begin{aligned} E &\rightarrow E + E \\ E &\rightarrow E * E \\ E &\rightarrow id \end{aligned}$$

Para la palabra: **id + id \* id**

**Árbol 1:** (interpretado como  $(id + id) * id$ )



**Árbol 2:** (interpretado como  $id + (id * id)$ )



**¡DOS ÁRBOLES DIFERENTES! → Gramática AMBIGUA**

**¿Por qué es un problema?**

En programación, “ $2 + 3 * 4$ ” debe dar **14** (no 20), por la precedencia de operadores. Una gramática ambigua no respeta esto.

**Solución: Crear una gramática no ambigua**

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow (E) \mid id \end{aligned}$$

Esta gramática asegura que \* tiene más precedencia que +.

---

## 4.7 Ejemplos de GIC

**Ejemplo 1: Palíndromos sobre {0,1}**

$$S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid$$

**Derivaciones:** -  $S \Rightarrow \epsilon \rightarrow ""$  -  $S \Rightarrow 0 \rightarrow "0"$  -  $S \Rightarrow 1 \rightarrow "1"$  -  $S \Rightarrow 0S0 \Rightarrow 00 \rightarrow "00"$  -  $S \Rightarrow 1S1 \Rightarrow 10S01 \Rightarrow 10101 \rightarrow "10101"$  ✓

**L(G) = {palabras que se leen igual de izquierda a derecha que de derecha a izquierda}**

**Ejemplo 2: L = {0^n1^n | n ≥ 0}**

$S \rightarrow 0S1 \mid$

**Derivaciones:** -  $S \Rightarrow \epsilon \rightarrow ""$  -  $S \Rightarrow 0S1 \Rightarrow 01 \rightarrow "01"$  -  $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 0011 \rightarrow "0011"$

**L(G) = {ε, 01, 0011, 000111, ...}**

## 4.8 Formas Normales

Las formas normales simplifican las gramáticas para facilitar el análisis.

**Pasos para convertir a Forma Normal:**

**Paso 1: Eliminar producciones ε (excepto S → ε) Ejemplo:**

Original:

$S \rightarrow AB$

$A \rightarrow aA \mid$

$B \rightarrow bB \mid$

Paso 1: Identificar variables anulables (que pueden generar )

- A es anulable ( $A \rightarrow \cdot$ )
- B es anulable ( $B \rightarrow \cdot$ )
- S es anulable (porque  $S \rightarrow AB$  y tanto A como B son anulables)

Paso 2: Crear nuevas reglas eliminando variables anulables

$S \rightarrow AB \mid A \mid B \mid$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

**Paso 2: Eliminar producciones unitarias (A → B) Ejemplo:**

Original:

$S \rightarrow A$

$A \rightarrow B \mid a$

$B \rightarrow b$

Las producciones unitarias son:  $S \rightarrow A$  y  $A \rightarrow B$

Eliminándolas:

$S \rightarrow a \mid b$  (sustituyendo lo que generan A y B)

$A \rightarrow b \mid a$

$B \rightarrow b$

**Paso 3: Eliminar símbolos inútiles No generadores:** Símbolos que no pueden convertirse en terminales. **No alcanzables:** Símbolos a los que no se puede llegar desde S.

**Ejemplo:**

$S \rightarrow AB \mid a$   
 $A \rightarrow b$   
 $C \rightarrow c$   
 $D \rightarrow SD$

- C no es alcanzable (no hay regla que lo genere desde S)
- D no es generador (genera D infinitamente pero nunca terminales)
- Eliminar C y D

## 4.9 Forma Normal de Chomsky (FNC)

**Reglas permitidas:**

1.  $A \rightarrow BC$  (dos no terminales)
2.  $A \rightarrow a$  (un solo terminal)
3.  $S \rightarrow \epsilon$  (solo si  $\epsilon$  está en el lenguaje)

**Ventajas:**

- Árbol binario
- Para una palabra de longitud n, se necesitan exactamente **2n - 1** pasos

**Ejemplo de conversión:**

**Original:**

$S \rightarrow ASA \mid aB$   
 $A \rightarrow B \mid S$   
 $B \rightarrow b \mid$

**En FNC:**

$S \rightarrow AA \mid TB \mid$   
 $A \rightarrow SA$   
 $A \rightarrow b \mid AA \mid TB$   
 $B \rightarrow b$   
 $T \rightarrow a$

## 4.10 Forma Normal de Greibach (FNG)

### Reglas permitidas:

1.  $A \rightarrow a\alpha$  donde  $a$  es terminal y  $\alpha$  es una cadena de variables (puede ser vacía)
2.  $S \rightarrow \epsilon$  (solo si  $\epsilon$  está en el lenguaje)

### Ventajas:

- Para una palabra de longitud  $n$ , se necesitan exactamente  $n$  pasos
- Útil para analizadores sintácticos descendentes

### Ejemplo:

$S \rightarrow aAB \mid b$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

Todas las producciones empiezan con un terminal