

4. Gramáticas Independientes del Contexto

Copyright (c) 2025 Adrián Quiroga Linares Lectura y referencia permitidas; reutilización y plagio prohibidos

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:

Una gramática se define siempre como:

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

1. **V (Variables/No Terminales):** Los "ingredientes intermedios". Se escriben en MAYÚSCULAS (S, A, B).
2. **T (Terminales):** El "plato final". Símbolos que forman las cadenas reales ($a, b, 0, 1$). **Nunca** aparecen a la izquierda de una flecha en gramáticas estándar (Context Free).
3. **P (Producciones):** Las reglas de sustitución. Estructura: Cabeza \rightarrow Cuerpo.
4. **S (Axioma):** El estado inicial (siempre pertenece a V).

4.2 La Jerarquía de Chomsky

Chomsky clasificó las gramáticas según **cuán estrictas son sus reglas**.

- **Regla de oro:** Cuanto mayor es el número ($0 \rightarrow 3$), **más restricciones** tiene y **menos potente** es.

Ejemplo	Tipo	Nombre	¿Qué ves a la IZQUIERDA de la flecha?	La Regla de Oro (En español)
<code>aAb -> c</code>	Tipo 0	Sin Restricciones	Cualquier cosa.	Vale todo. Puedes borrar, cambiar, mezclar... Caos total.

(3 cosas se convierten en 1. Se puede encoger).

Ejemplo	Tipo	Nombre	¿Qué ves a la IZQUIERDA de la flecha?	La Regla de Oro (En español)
<code>xAy -> xBy</code> (La 'A' cambia a 'B' solo si está entre x e y).	Tipo 1	Sensible al Contexto	Un grupo de cosas. (Contexto)	"No puedes encoger". Lo que entra debe ser igual o menor a lo que sale. Necesitas "vecinos" para cambiar.
<code>A -> aBc</code> (Siempre que veas una A, cámbiala).	Tipo 2	Indep. del Contexto (GIC)	UNA SOLA variable. (Mayúscula)	"Sustitución simple". Cambias una variable por lo que quieras, sin importar qué tenga al lado.
<code>A -> aB</code> ó <code>A -> a</code> (La variable siempre va a un extremo).	Tipo 3	Regular	UNA SOLA variable. (Mayúscula)	"Cola india". Muy rígido. Solo puedes poner un terminal y (opcional) una variable al final.

Nota: Tipo 3 \subset Tipo 2 \subset Tipo 1 \subset Tipo 0. *Toda gramática regular es independiente del contexto, pero no al revés.*

4.3. Gramáticas Regulares (Tipo 3)

Son las que generan los Lenguajes Regulares (los mismos que vimos en el Tema 2 y 3).

Estructura Rígida:

- **Lineal Derecha:** $A \rightarrow \text{terminal} \cdot \text{Variable}$ (ej: $A \rightarrow aB$) o $A \rightarrow \text{terminal}$ ($A \rightarrow a$).
- **Lineal Izquierda:** $A \rightarrow \text{Variable} \cdot \text{terminal}$ (ej: $A \rightarrow Ba$).

¡Ojo! No puedes mezclar reglas lineales por derecha e izquierda en la misma gramática. Si lo haces, se convierte en Tipo 2 (GIC) y deja de ser Regular.

4.4 Lenguaje de una Gramática

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

Se lee así: *"El lenguaje L generado por la gramática G es el conjunto de cadenas $w...$ "*

1. $w \in T^*$:

- Esto impone la condición principal: la cadena resultante w debe estar formada **únicamente por símbolos Terminales** (T).
- El asterisco (*) es el **Cierre de Kleene**, que significa "cualquier combinación de terminales".
- **¿Por qué es importante?** Porque durante la derivación intermedia puedes tener cosas como aSb (que mezcla terminales y variables). La fórmula te dice que eso **no** es parte del lenguaje final todavía; solo lo es cuando desaparecen todas las letras mayúsculas (Variables).

2. $|$:

- Significa "tal que" o "que cumplen la condición de que...".

3. $S \Rightarrow^* w$:

- S : Debes empezar obligatoriamente desde el **Símbolo Inicial**.
- \Rightarrow^* : La flecha doble con asterisco significa **"se deriva en cero o más pasos"**. Es decir, no importa si tardas 1 paso o 1.000 pasos en llegar.
- w : Debes llegar a la palabra final.

Ejemplo:

```
S → aSb | ε
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"
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaSbbb \Rightarrow aaabbb \rightarrow$ palabra: "aaabbb"

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

4.5 Árboles de Derivación vs Derivaciones

Derivar es el proceso de construir una cadena de texto válida (una palabra) aplicando las reglas de la gramática paso a paso.

Se empieza con el **Símbolo Inicial** (generalmente llamado S) y se van sustituyendo las variables (letras mayúsculas) por lo que dictan las reglas (producciones) hasta que

solo quedan **terminales** (letras minúsculas o símbolos finales que ya no se pueden cambiar).

Es básicamente un juego de "**buscar y reemplazar**":

- **Entrada:** El símbolo inicial S .
- **Acción:** Eliges una variable presente en tu cadena actual, buscas una regla que le aplique y la sustituyes.
- **Fin:** Cuando ya no quedan variables (letras mayúsculas), has terminado la derivación.

Es importante distinguir entre el **proceso** y la **estructura**.

1. **Derivación (Texto):** Es la secuencia de pasos paso a paso.

- $S \Rightarrow aA \Rightarrow abB \Rightarrow abb$

2. **Árbol de Derivación (Gráfico):** Es la estructura jerárquica.

- Raíz: S .
- Hojas: La palabra final (a, b, b) .
- Nodos: Variables intermedias.

Gramática:

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow id$ (donde 'id' es un número o variable)

Objetivo: Generar el árbol para la cadena: $id + id * id$

Empezamos con el símbolo inicial E .

Miramos nuestra cadena objetivo ($id + id * id$). Vemos que hay una suma principal (o una multiplicación, depende de cómo decidamos estructurarlo, pero asumamos la suma primero para este ejemplo).

Aplicamos la regla $E \rightarrow E + E$.

- De la raíz E salen tres hijos: E , $+$ y E .

Ahora tenemos:

- Un E a la izquierda.
- Un $+$ en el medio (ya es terminal, se queda quieto).
- Un E a la derecha.

Miramos la cadena objetivo: la primera parte es solo id. Así que al E de la izquierda le aplicamos la regla $E \rightarrow id$.

Para la segunda parte, necesitamos $id * id$. Así que al E de la derecha no lo convertimos en id directamente, sino que le aplicamos la regla de multiplicación:
 $E \rightarrow E * E$

Ahora el árbol ha crecido. Los nuevos E que creamos para la multiplicación se convierten cada uno en **id** usando la regla $E \rightarrow id$.

4.6 Ambigüedad

Definición de Examen: Una gramática es ambigua si existe **al menos una cadena** que tiene **dos o más árboles de derivación distintos**. Pueden tener derivaciones distintas y no ser ambiguas, por ejemplo:

1. $S \rightarrow AB$
2. $A \rightarrow a$
3. $B \rightarrow b$

Queremos generar la cadena: ab Podemos hacerlo con dos "derivaciones" diferentes (cambiando el orden en que sustituimos las letras):

Derivación 1 (Por la izquierda):

1. $S \Rightarrow AB$
2. $S \Rightarrow aB$ (Sustituyo A primero)
3. $S \Rightarrow ab$ (Sustituyo B después)

Derivación 2 (Por la derecha):

1. $S \Rightarrow AB$
2. $S \Rightarrow Ab$ (Sustituyo B primero)
3. $S \Rightarrow ab$ (Sustituyo A después)

¿Son derivaciones diferentes?

SÍ. La lista de pasos es distinta. En una escribí la 'a' antes y en la otra escribí la 'b' antes.

¿Son árboles diferentes?

NO. Si dibujas el árbol, el resultado es idéntico en ambos casos:

- La raíz es S .
- De S salen dos ramas: A y B .
- De A cuelga una a .
- De B cuelga una b .

Caso típico: Operaciones matemáticas sin paréntesis ni precedencia.

- $3 + 4 * 5 \rightarrow$ ¿Es $(3+4)*5$ o $3+(4*5)$? Si la gramática permite ambos árboles, es mala (ambigua).

4.7 Protips para ejercicios

Patrón 1: El Espejo y la Cebolla (Palíndromos y $a^n b^n$)

Si necesitas que el principio coincida con el final, o que la cantidad de letras del principio sea igual a la del final, usas la **recursividad envolvente**.

La Regla de Oro:

$$S \rightarrow x S y$$

Esto genera x a la izquierda y y a la derecha, sincronizados.

Ejemplo Práctico Palíndromo:

1. Si añado una 'a' al principio, debo añadir una 'a' al final: $S \rightarrow aSa$
2. Si añado una 'b' al principio, debo añadir una 'b' al final: $S \rightarrow bSb$
3. ¿Cómo termino? Con el centro. Puede ser 'a', 'b', o vacío (ε): $S \rightarrow a \mid b \mid \varepsilon$

Gramática final: $S \rightarrow aSa \mid bSb \mid a \mid b \mid \varepsilon$

Patrón 2: Ecuaciones Lineales ($k = i + j$ o $k = i + 2j$)

Estos ejercicios te piden relacionar contadores.

- **Truco:** Traduce la ecuación a "**quién consume a quién**".
- **Orden:** Fíjate MUY bien en el orden de las letras en el alfabeto ($a^i b^j c^k$).

Caso A: Suma simple ($k = i + j$ en $a^i b^j c^k$)

Significa que por cada 'a' hay una 'c', Y por cada 'b' hay una 'c'. Pero las 'a' están lejos de las 'c'.

- Solución: Anidamiento. Tratamos la cadena como $a^i (b^j c^j) c^i$.
- Las 'a' envuelven a todo el bloque de 'b' y 'c'.
- Las 'b' se emparejan con las 'c' del medio.

Caso B: Multiplicación ($k = i + 2j$ en $a^i b^j c^k$)

- Interpretación:
 - Por cada 1 'a', genero 1 'c'. (Exterior)
 - Por cada 1 'b', genero 2 'c's. (Interior)
- Diseño:

1. Estado inicial (S): Genera 'a' izquierda y 'c' derecha. Cuando acaben las 'a', pasamos al bloque central (B).

$$S \rightarrow aSc \mid B$$

2. Estado central (B): Genera 'b' izquierda y dos 'c' derecha.

$$B \rightarrow bBcc \mid \varepsilon$$

Patrón 3: La "O" Lógica (Unión de Casos)

Si ves un "o", una coma, o condiciones alternativas ($i = j$ o $j = k$), **NO** intentes hacerlo todo en una sola regla. Divide y vencerás.

Estrategia: El símbolo inicial solo sirve para elegir camino.

$$S \rightarrow S_1 \mid S_2$$

Ejemplo ($a^i b^j c^k$ donde $i = j$ O $j = k$):

- Camino 1 (S_1): $i = j$ (y k va por libre). Cadena tipo $a^n b^n c^m$.
 - Empareja 'a' y 'b'. La 'c' se genera aparte libremente.
 - $S_1 \rightarrow XC$
 - $X \rightarrow aXb \mid \varepsilon$ (pareja a-b)
 - $C \rightarrow cC \mid \varepsilon$ (c libre)
- Camino 2 (S_2): $j = k$ (y i va por libre). Cadena tipo $a^m b^n c^n$.
 - La 'a' va libre al principio. Luego empareja 'b' y 'c'.
 - $S_2 \rightarrow AY$
 - $A \rightarrow aA \mid \varepsilon$ (a libre)
 - $Y \rightarrow bYc \mid \varepsilon$ (pareja b-c)

Patrón 4: Desigualdades (El Truco del "Sobra Algo")

Las gramáticas no saben hacer "mayor que". Solo saben hacer "igual". **Truco**

Matemático:

- $k > i \rightarrow$ significa $k = i + m$ (donde $m \geq 1$).
- Es decir: "Hay tantas 'k' como 'i', y luego **sobran** más 'k'".

Ejemplo ($a^i(b+c)^k$ donde $k > i$): Vamos a simplificar $(b+c)$ llamándolo X . La estructura es $a^i X^k$.

1. Parte equilibrada: Por cada 'a', pongo una X .
2. Parte sobrante: Añado más X al final (o al lado de las X).

$$S \rightarrow aSX \mid A \quad (\text{Emparejo a con X})$$

$$A \rightarrow XA \mid X \quad (\text{Genero las X sobrantes, al menos una})$$

$$X \rightarrow b \mid c \quad (\text{Defino qué es } X)$$

Truco para "Distinto" (\neq). "Distinto" significa "Mayor que" O "Menor que".

$$S \rightarrow S_{mayor} \mid S_{menor}$$

Haces dos gramáticas (como en el Patrón 4) y las unes.

Patrón 5: Desorden y Mezcla (Conteo $N(a) = N(b)$)

Aquí NO hay orden $a^i b^j$. Las letras pueden estar mezcladas "aababb...". Esto se resuelve con **Inserción Relativa**.

Regla Maestra para $N(a) = N(b)$: Si quiero mantener el equilibrio, donde ponga una 'a', debo poner una 'b'. Pero como no hay orden, la 'b' puede ir antes, después, o alrededor.

La forma estándar segura es:

$$S \rightarrow aSbS \mid bSaS \mid SS \mid \varepsilon$$

Significado: Si pongo 'a', debo "deber" una 'b' (el estado S intermedio se encarga de resolver esa deuda).

Ejemplo ($N(0) = N(1) + 1$): Esto es: "Equilibrado + un 1 extra".

- Definimos un equilibrio perfecto B (mismo nº de 0 y 1).

$$B \rightarrow 0B1B \mid 1B0B \mid \varepsilon$$

(Ojo: simplificado para el concepto, a veces se requiere más rigor con SS).

- Estado Inicial: Es el equilibrio B , pero forzando un '0' extra en algún sitio.

$$S \rightarrow B0B$$

4.7 Simplificación de Gramáticas (GIC)

Antes de pasar a formas normales, **siempre** debes limpiar la gramática en este orden estricto:

1. Eliminar Producciones ε (Vacías)

Si $A \rightarrow \varepsilon$, entonces A es "anulable".

Método:

- Busca todas las variables que pueden volverse ε (directa o indirectamente).
- Si tienes $S \rightarrow Ab$, y A es anulable, añade una nueva regla $S \rightarrow b$ (versión donde A desaparece).
- Borra las reglas $A \rightarrow \varepsilon$ originales (salvo si $S \rightarrow \varepsilon$ es necesario para el lenguaje)

2. Eliminar Producciones Unitarias ($A \rightarrow B$)

Reglas que solo cambian el nombre de la variable sin añadir terminales.

Método:

1. Si $A \rightarrow B$ y $B \rightarrow \text{algo}$, entonces añade $A \rightarrow \text{algo}$.
2. Borra $A \rightarrow B$.

3. Eliminar Símbolos Inútiles

Se hace en dos pasadas:

1. **No Generadores:** Variables que entran en bucle y nunca llegan a terminales ($A \rightarrow aA$). Bórralas.
2. **Inalcanzables:** Variables a las que no puedes llegar empezando desde S . Bórralas.

Ejemplo que usa los pasos 1 y 2:

a) $S \rightarrow XC \mid AY \mid C \mid X \mid Y \mid A$
 $X \rightarrow aXb \mid ab$
 $C \rightarrow Cc \mid c$
 $A \rightarrow Aa \mid a$
 $Y \rightarrow bYc \mid bc \mid C$

b) $S \rightarrow XC \mid AY \mid Cc \mid c \mid aXb \mid ab \mid bYc \mid bc \mid Aa \mid a$
 $X \rightarrow aXb \mid ab$
 $C \rightarrow Cc \mid c$
 $A \rightarrow Aa \mid a$
 $Y \rightarrow bYc \mid bc \mid Cc \mid c$

4.8 Formas Normales (FNC-Chomsky)

Objetivo: Estandarizar la gramática para que todas las reglas sean "cortas" y binarias. Es fundamental para el algoritmo CYK (análisis sintáctico).

Solo se permiten 2 tipos de reglas:

1. $A \rightarrow BC$ (Dos variables).
2. $A \rightarrow a$ (Un terminal).

Algoritmo de Conversión a FNC (Práctico)

Supongamos que ya has simplificado la gramática (paso 4.8).

Paso 1: Terminales solitarios en reglas mixtas. Si tienes $S \rightarrow aB$, eso está prohibido (mezcla terminal y variable).

- Crea una variable nueva: $X_a \rightarrow a$.
- Cambia la regla a: $S \rightarrow X_a B$.

Paso 2: Acortar cadenas largas. Si tienes $S \rightarrow ABC$ (3 variables), es demasiado largo.

- Rompe la cadena creando variables intermedias.
- $S \rightarrow AZ$
- $Z \rightarrow BC$

Ejemplo Rápido:

Original: $S \rightarrow aSb$

1. Crear variables para terminales: $X_a \rightarrow a$, $X_b \rightarrow b$.
2. Sustituir: $S \rightarrow X_a S X_b$.
3. Romper cadena de 3:
 - $S \rightarrow X_a Y$
 - $Y \rightarrow S X_b$

Resultado FNC:

$$\begin{aligned} S &\rightarrow X_a Y \\ Y &\rightarrow S X_b \\ X_a &\rightarrow a \\ X_b &\rightarrow b \end{aligned}$$

4.9 Forma Normal de Greibach (FNG)

La **FNG** es otra forma de estandarizar gramáticas (como la de Chomsky), pero con una filosofía distinta: "**Producir letra a letra**".

Imagina una máquina expendedora. En la **FNG**, cada vez que la gramática hace un movimiento (aplica una regla), está **obligada** a soltar exactamente **una moneda (símbolo terminal)** y quedarse con el cambio (variables).

La regla siempre tiene esta forma:

$$A \rightarrow a\alpha$$

- A : Variable actual.
- a : **Un solo terminal** (la moneda que suelta).
- α : Una cadena de cero o más variables (el cambio que te queda por procesar).

Ejemplo: $S \rightarrow aAB$ (Soltó una 'a', le queda procesar A y B).

Problema 1: Eliminación de Símbolos Inútiles

Enunciado: Encontrar gramática equivalente a:

$$S \rightarrow AB \mid CA$$

$$A \rightarrow a$$

$$B \rightarrow BC \mid AB$$

$$C \rightarrow aB \mid b$$

Objetivo: Limpiar la gramática. Para ello buscamos variables que sean "basura" (no generan terminales) o "fantasmas" (nadie llega a ellas).

Paso 1: Detectar símbolos NO Generadores. (¿Qué variables son capaces de convertirse en una cadena de solo terminales al final?)

1. **A es generador:** $A \rightarrow a$ (directo).
2. **C es generador:** $C \rightarrow b$ (directo).
3. **S es generador:** $S \rightarrow CA \rightarrow ba$ (indirecto).
4. **¿Y la B?**

- Sus reglas son: $B \rightarrow BC$ y $B \rightarrow AB$.
- Fíjate bien: Para deshacerte de B , necesitas aplicar una regla. Pero **todas** sus reglas vuelven a invocar a B (BC o AB).
- Es un bucle infinito: $B \rightarrow AB \rightarrow aB \rightarrow aAB \dots$ Nunca desaparece.
- **Conclusión:** B es un símbolo inútil (no generador).

Paso 2: Eliminar la B

Borramos todas las reglas donde aparezca B (tanto a la izquierda como a la derecha).

- Borrar $S \rightarrow AB$ (contiene B).
- Borrar todo el bloque de $B \rightarrow \dots$
- Borrar $C \rightarrow aB$ (contiene B).

Gramática resultante:

$$S \rightarrow CA$$

$$A \rightarrow a$$

$$C \rightarrow b$$

Paso 3: Símbolos Inaccesibles

Desde S , ¿podemos llegar a todos?

- $S \rightarrow CA$ (Llegamos a C y a A).
- Todo es accesible.

Resultado Final (Problema 1):

$$S \rightarrow CA$$

$$A \rightarrow a$$

$$C \rightarrow b$$

Problema 2: Paso a Forma Normal de Chomsky (FNC)**Enunciado:**

$$S \rightarrow ASB \mid \varepsilon$$

$$A \rightarrow aAS \mid a$$

$$B \rightarrow SbS \mid A \mid bb$$

Este es un ejercicio largo. Se hace en 3 fases estrictas.

Fase 1: Eliminar producciones ε (Vacías)

La única regla vacía es $S \rightarrow \varepsilon$. S es "anulable".

Debemos reescribir las reglas imaginando "qué pasa si S desaparece".

1. En $S \rightarrow ASB$:

- Si la S del centro desaparece: queda AB .
- Nueva regla: $S \rightarrow ASB \mid AB$.

2. En $A \rightarrow aAS$:

- Si la S del final desaparece: queda aA .
- Nueva regla: $A \rightarrow aAS \mid aA$.

3. En $B \rightarrow SbS$:

- Si la 1ª S desaparece: bS .
- Si la 2ª S desaparece: Sb .
- Si ambas desaparecen: b
- Nueva regla: $B \rightarrow SbS \mid bS \mid Sb \mid b$.

Gramática tras Fase 1 (sin ε):

(Nota: asumimos $S \rightarrow \varepsilon$ se elimina o se mantiene aparte si el lenguaje acepta vacío, aquí lo quitamos para limpiar).

$$S \rightarrow ASB \mid AB$$

$$A \rightarrow aAS \mid aA \mid a$$

$$B \rightarrow SbS \mid bS \mid Sb \mid b \mid A \mid bb$$

Fase 2: Eliminar producciones Unitarias ($X \rightarrow Y$)

Buscamos reglas donde una variable lleva a una sola variable.

Aquí detectamos: $B \rightarrow A$.

- Como B se convierte en A , B debe heredar todo lo que hace A .
- Producciones de A : $\{aAS, aA, a\}$.
- Añadimos eso a B y borramos $B \rightarrow A$.

*Gramática tras Fase 2:

$$S \rightarrow ASB \mid AB$$

$$A \rightarrow aAS \mid aA \mid a$$

$$B \rightarrow SbS \mid bS \mid Sb \mid b \mid bb \mid aAS \mid aA \mid a$$

Fase 3: Convertir a FNC

Reglas permitidas: $Var \rightarrow VarVar$ o $Var \rightarrow terminal$.

1. Crear variables para terminales:

Creamos $X_a \rightarrow a$ y $X_b \rightarrow b$.

Sustituimos todos los terminales en reglas largas.

2. Ajustar reglas largas (Romper cadenas):

- $S \rightarrow ASB \Rightarrow S \rightarrow AZ_1$, donde $Z_1 \rightarrow SB$.
- $A \rightarrow aAS \Rightarrow Sustituir 'a' : A \rightarrow X_aAS \Rightarrow A \rightarrow X_aZ_2$, donde $Z_2 \rightarrow AS$.
- $B \rightarrow SbS \Rightarrow B \rightarrow SZ_3$, donde $Z_3 \rightarrow X_bS$.
- (Y así con el resto...).

Resultado Final (Esquemático):

Variables auxiliares: $X_a \rightarrow a$, $X_b \rightarrow b$.

Transformación de **S**:

$$S \rightarrow AZ_1 \mid AB \quad (Z_1 \rightarrow SB)$$

Transformación de **A**:

$$A \rightarrow X_aZ_2 \mid X_aA \mid a \quad (Z_2 \rightarrow AS)$$

Transformación de **B**:

$$B \rightarrow SZ_3 \mid X_bS \mid SX_b \mid b \mid X_bX_b \mid X_aZ_2 \mid X_aA \mid a \quad (Z_3 \rightarrow X_bS)$$