7. Gramáticas GSC Y GSR

7.1 Gramáticas Sin Restricciones (GSR) - Tipo 0

Como su nombre indica, son las gramáticas más flexibles y potentes. No tienen límites en la forma de sus reglas, salvo que siempre debe haber algo que sustituir.

Definición y Reglas
Una GSR se define formalmente como $G = (NT, T, S, P)$.

Sus producciones tienen la forma genérica:

$$x \rightarrow y$$

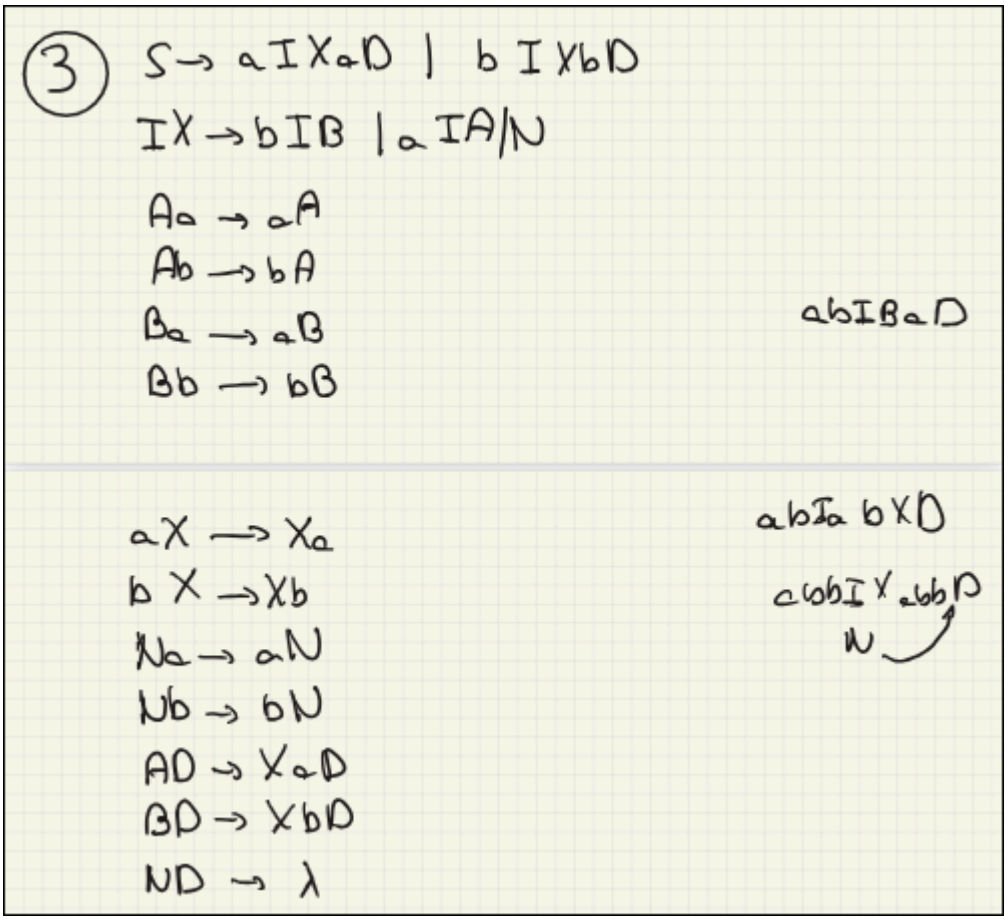
Donde:

- x **(Izquierda):** Es una cadena de símbolos terminales y no terminales (NT/T). La única condición es que **no puede estar vacía** (debe tener al menos un símbolo).
- y **(Derecha):** Es cualquier cadena de símbolos terminales y no terminales. **Puede ser vacía**.

Características Clave

- **Libertad total:** Puedes reemplazar un grupo de símbolos por otro grupo, acortar la cadena, alargarla o borrar símbolos completamente.
- **Lenguaje que generan:** Generan los **Lenguajes Recursivamente Enumerables (LRE)**.
- **Máquina Equivalente:** Son reconocidos por la **Máquina de Turing** estándar.

Ejemplo:



7.2 Gramáticas Sensibles al Contexto (GSC) - Tipo 1

Estas gramáticas imponen restricciones sobre las GSR. La idea principal es que las sustituciones dependen de lo que rodea a la variable ("el contexto") y la cadena nunca se hace más pequeña.

Definición y Reglas
Una GSC tiene producciones de la forma:

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

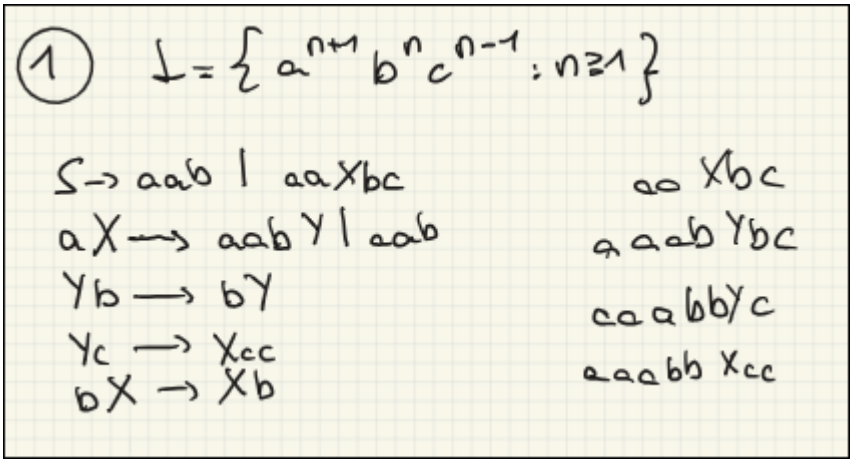
Desglosemos qué significa esto:

1. A: Es un **símbolo no terminal** (la variable que vamos a cambiar).
2. α **y** β **(El Contexto):** Son cadenas de terminales o no terminales (pueden ser vacías). Representan lo que está a la izquierda y derecha de A.
3. γ **(El Cambio):** Es una cadena **no vacía**. Es por lo que sustituimos a A.

La Regla de Oro: No Contracción
La característica más importante para identificar una GSC es la longitud.

- La longitud de la parte derecha (y) debe ser **igual o mayor** que la de la parte izquierda (x).
- Esto significa que la gramática **nunca "encoge"** la cadena (excepto posiblemente para generar la cadena vacía λ si el lenguaje lo permite).

Ejemplo:



Relación con Máquinas

- **Lenguaje que generan:** Lenguajes Sensibles al Contexto (LSC).
- **Máquina Equivalente:** Son reconocidos por los **Autómatas Linealmente Acotados (ALA)**. Es una Máquina de Turing (MT) no determinista con su cinta limitada por ambos extremos, siendo el tamaño de la cinta fijo . Imagina una MT normal, que tiene papel infinito para hacer cálculos. Un ABA, en cambio, solo tiene el espacio que ocupa la palabra de entrada (y quizás unas pocas casillas más fijas a los lados). No puede pedir más papel. Para que la cabeza lectora no se "caiga" de la cinta, tiene dos símbolos especiales (marcadores) en los extremos izquierdo y derecho que le dicen "hasta aquí puedes llegar" .

La fórmula del Lenguaje $L(M)$. L(M) significa **"El Lenguaje de la Máquina M"**. En términos sencillos: es la lista VIP de palabras que la máquina acepta.

$$L(M) = \{w \in \Sigma^+ : q_0[w] \vdash^* [x_1 q_f x_2], q_f \in F \dots\}$$

- `[` **(Marcador izquierdo):** Es el muro de la izquierda.
  - La fórmula $\delta(q_i, [) = (q_j, [, D)$ significa: "Si la cabeza está en el estado q_i y lee el muro `[`, pasa al estado q_j, **deja el muro intacto** (`[`) y se mueve a la **Derecha** (D)".
  - En resumen: **Rebota hacia dentro**. No puede atravesarlo ni borrarlo.
- `]` **(Marcador derecho):** Es el muro de la derecha.
  - La fórmula $\delta(q_i,]) = (q_j,], I)$ significa: "Si lee el muro `]`, **lo deja intacto** y se mueve a la **Izquierda** (I)".
  - En resumen: **Rebota hacia dentro**.

Se traduce así:

1. Tomas una palabra de entrada w.
2. La encierras entre los muros: `[w]`.
3. Empiezas en el estado inicial q_0.
4. La máquina procesa ($\vdash^*$) moviéndose dentro de esos muros.

5. Si la máquina llega a un estado final ($q_f \in F$) manteniendo los muros intactos ($[x_1 \dots x_2]$), entonces **acepta** la palabra.

7.3 Protips

7.3.1 El Mensajero (Commutation)

Problema: Tienes variables mezcladas (ej: $ABAB$) pero necesitas ordenarlas (ej: $AABB$). En una Gramática Independiente del Contexto (CFG) no puedes cambiar el orden una vez generado. En una GSC, sí.

La Estrategia: Creas una regla que permite que dos variables intercambien lugares.

Cómo:

1. **Generación:** Generas pares o tríos desordenados.
2. **Tráfico:** Usas reglas de la forma $XY \to YX$ para mover la variable "mensajera" a su posición correcta.
3. **Conversión:** Solo cuando están en orden, se convierten en terminales.

Ejemplo Práctico: $L = \{a^n b^n c^n\}$

Queremos generar $aabbcc$.

Intentamos generar bloques ABC.

1. **Regla de Inicio:** $S \to aSBc \mid abc$ (Esto genera algo como $a(abc)Bc \to aabcBc$).
   - *Problema:* Tenemos una c antes de una B. El orden está mal: $\dots cBc \dots$
2. **Regla del Mensajero:** $cB \to Bc$.
   - *Traducción:* Si una c ve una B a su derecha, le dice "Pasa tú primero". La B viaja a la izquierda sobre la c.

Derivación visual:

$$aab\mathbf{cB}c \xrightarrow{cB \to Bc} aab\mathbf{Bc}c$$

> **Resultado:** Ahora las B están con las b y las c con las c.

7.3.2 El Muro (Boundary)

Problema: En las GSC, las reglas son peligrosas. Si tienes la regla $A \to a$, podrías aplicarla demasiado pronto, antes de que la cadena esté ordenada.

La Estrategia: Usar símbolos especiales (Centinelas) que marcan el principio o el final de la cadena. Las variables no se convierten en terminales hasta que tocan "El Muro", y luego el cambio se propaga como un efecto dominó.

Cómo funciona:

1. **Colocar el Muro:** Tu regla inicial pone topes. $S \to \#T\#$.
2. **Contacto:** Una variable solo cambia si toca el muro. $B\# \to b\#$.
3. **Propagación:** El cambio se contagia de derecha a izquierda (o viceversa). $Ab \to ab$.

Ejemplo Práctico: Convertir variables a letras ordenadamente

Imagina que tienes la cadena $AAABBB$. Quieres pasarla a minusculas ($aaabbb$) pero solo si **todo** está listo.

1. Colocamos muro al final: $AAABBB\#$
2. **Regla de Gatillo:** $B\# \to b\#$ (La última B toca el muro y se transforma).
3. Regla de Dominó: $Bb \to bb$ (Una B mayúscula ve una b minúscula a su derecha y se transforma).

$$AAAB\mathbf{Bb} \to AAAB\mathbf{bb}$$

4. Regla de Cruce de Frontera: $Ab \to ab$ (El cambio pasa de las B a las A).

$$AA\mathbf{Ab}bb \to AA\mathbf{ab}bb$$

> **Por qué es útil:** Garantiza que no te queden letras sueltas en medio de variables no procesadas.

7.3.2 El Clonador

Problema: Necesitas duplicar información exacta, como en el lenguaje $L = \{ww\}$ (ej: $abcabc$) o a^{2n}.

La Estrategia: Una variable genera un par (el original y la copia) y una de ellas actúa como "Mensajero" viajando hasta su nueva posición.

Cómo funciona:

Para hacer ww (copiar la palabra exacta):

1. **Generar Pares:** Por cada letra que quieras añadir, generas su par.
   - Si quieres 'a', generas X_aY_a.
2. **Transporte:** Y_a es el clon. Debe viajar saltando sobre las X hasta llegar a la segunda mitad de la palabra.
3. **Regla de Salto:** $Y_aX_b \to X_bY_a$ (El clon salta sobre otras letras base).

Ejemplo Práctico: $L = \{ww\}$ con $\Sigma = \{0, 1\}$

Queremos generar 0101.

1. **Semilla:** $S \to C_0S \mid C_1S \mid \dots$ (Donde C_0 representa el par "Original 0 + Clon 0").
   - Digamos que C_0 en realidad genera X_0Y_0.
2. **Generación:** Generas $X_0Y_0X_1Y_1$.
   - Aquí tienes "Original 0", "Clon 0", "Original 1", "Clon 1".
   - Orden actual: 0, 0, 1, 1.
   - Orden deseado: 0, 1, 0, 1.
3. **Movimiento (Mensajero):** El Y_0 (Clon 0) debe moverse a la derecha.
   - Regla: $Y_0X_1 \to X_1Y_0$.
   - Cadena: $X_0\mathbf{Y_0}\mathbf{X_1}Y_1 \Rightarrow X_0\mathbf{X_1}\mathbf{Y_0}Y_1$.
4. **Finalización:** Ahora tienes $X_0X_1Y_0Y_1$ (Grupo 1 separado de Grupo 2). Los conviertes a terminales.

7.4 Identificar Grámaticas y Lenguajes

Gramáticas

¿Qué ves a la IZQUIERDA?	¿Qué ves a la DERECHA?	TIPO
Solo 1 Variable ($A \to \dots$)	Estricto: a ó aB	**Tipo 3 (Regular)**
Solo 1 Variable ($A \to \dots$)	Libre: aAb, BC, etc.	**Tipo 2 (GIC)**
Grupo ($aA \to \dots$)	Longitud $\geq$ Izquierda	**Tipo 1 (Sensible)**
Grupo ($aA \to \dots$)	Longitud < Izquierda	**Tipo 0 (Irrestricta)**

Lenguajes

1. Lenguajes Regulares (Tipo 3)

> Memoria: Nula o Finitud.
>
> Clave: "No necesito contar hasta el infinito".

- **Patrones:** Empieza por, termina en, contiene la subcadena "aba".
- **Conteo:** Solo cuenta hasta un número fijo. (Ej: "Longitud par", "múltiplo de 3").
- **NO PUEDE:** Contar cosas indefinidas y compararlas.
- *Ejemplo:* $L = \{\text{cadenas de a y b que terminan en ab}\}$.

2. Lenguajes Independientes del Contexto (Tipo 2)

> Memoria: Una Pila (Stack).
>
> Clave: "Puedo comparar DOS cosas (o anidar)".

- **Patrones:** Equilibrio, espejos.
- **Conteo:** $a^n b^n$ (mismo número de a's que de b's).
- **Estructura:** Paréntesis equilibrados `(( ))`, palíndromos (ww^R).
- **Límite:** Solo puedo relacionar **dos** contadores a la vez o estructuras anidadas. No puedo cruzar relaciones.
- *Ejemplo:* $L = \{a^n b^n \mid n \geq 0\}$.

$L = \{a^i b^j c^k \mid i=j \text{ o } j=k, i, j, k > 0\}.$

$L = \{a^i b^i a^j b^j \mid i, j > 0\},$

$L = \{a^i b^j c^k / k = i + (2 * j)\}$

3. Lenguajes Sensibles al Contexto (Tipo 1)

> Memoria: Cinta acotada (puedo leer y volver atrás).
>
> Clave: "Puedo comparar TRES cosas o COPIAR".

- **Patrones:** Relaciones triples o cruzadas.
- **Conteo:** $a^n b^n c^n$ (Tres cantidades iguales).
- **Copia:** ww (una palabra repetida exactamente igual, ej: "papa", "mama"). *Nota: El palíndromo es Tipo 2, la copia exacta es Tipo 1.*
- *Ejemplo:* $L = \{a^n b^n c^n \mid n \geq 1\}$.

$L = \{a^{n+1} b^n c^{n-1} : n \geq 1\}.$

4. Lenguajes Recursivamente Enumerables (Tipo 0)

> Memoria: Ordenador completo.
>
> Clave: "Cualquier algoritmo lógico".

- Si te dan un problema lógico complejo que no tiene restricciones de estructura simple. Generalmente, en los exámenes, se centran en los tres anteriores, salvo que pregunten por problemas de parada o indecidibilidad.

$L = \{ww : w \in \{a, b\}^+\}.$