TEMA 2. GRAMATICAS FORMALES:

Gramática Formal

Llamaremos Gramática Formal a la cuádrupla G = {$\sum_T$, $\sum_N$, S, P }

$\sum_T$ = Alfabeto de Símbolos Terminales: Símbolos que forman parte de las palabras del lenguaje generado por la gramática. (Alfabeto Principal)

$\sum_N$ = Alfabeto de Símbolos No Terminales: Símbolos que participan en la elaboración de las palabras. (Alfabeto Auxiliar)

S = Axioma. Partimos de este símbolo para generar todo el lenguaje de una gramática.

P = Conjunto de las reglas de producción, de la forma:

$$u ::= v; \quad u \in \sum^+, \quad v \in \sum^*, \quad u = xAy, \quad x,y \in \sum^*, \quad a \in \sum_N$$

- u no puede ser la palabra vacía.

PRODUCCIONES

También llamadas reglas de derivación. x:: = y (x produce a y)

Ejemplo: ∑ = (0, 1) En ∑ se podrían tener las siguientes producciones (000::= 010 / 10::= 01)

Derivación Directa (v → w): se dice que w es derivación directa de v o que v produce directamente w; si existen dos palabras u,z tales que v = zxu y w = zyu siendo x::= y ∈ P

Ejemplo: x::= y ∈ P   *Ca-ba-llo → Ca-me-llo*   *ba (x)::= me (y)*

***v** = u - x - z*   ***w** = u - y - z*

Derivación (v $\rightarrow_+$ w): se dice que w es derivación de v o que v produce w; si existe una secuencia de derivaciones directas que lleven de v a w.

Relación de Thue (v $\rightarrow_*$ w): se da si se verifica que v $\rightarrow_+$ w o v = w

Toda palabra se deriva de si misma con derivación de longitud 0

Ejemplo de una gramática definida por:

G = {$\sum_T$, $\sum_N$, S , P} $\sum_T$ = {0 , 1, 2 , 3..., 9}

$\sum_N$ = { A, B }   S = A

P = {A::= AB, A::= B, B::= 0, B::= 1, B::= 2, B::=3, B::= 4, B::= 5, B::= 6, B::= 7, B::= 8, B::= 9}

Para generar A → 42   A → AB → A2 → B2 → 42

Notación de Backus: si se tiene que u::=v y u::=w entonces se puede notar de la siguiente forma u::= v | w . Siguiendo el ejemplo anterior:
P = {A::= AB | A, B::= 0|1|2|3|4|5|6|7|8|9 }

Forma Sentencial: Dada una gramática $G = \{ \sum_T, \sum_N , S, P\}$.
x es forma sentencial de G si $S \to_* x$ (Si existe derivación desde el axioma hasta x). Ejemplo:

$A \to AB \to ABB$ ABB es forma sentencial de G

Se dice que x es palabra o sentencia de G cuando x es forma sentencial y $x \in \sum_T^*$

Lenguaje asociado a una gramática: Dada una gramática: $G = \{ \sum_T, \sum_N , S, P\}$

Se llama *" Lenguaje asociado a G"* o *" Lenguaje generado por G"* al conjunto

$$L(G) = \{x \mid S \Rightarrow xy \cap x \in \sum_T^*\}$$

Es decir, el conjunto de todas las sentencias o palabras de G.

$$L(G) = \{x \in \sum_T^* \mid S \Rightarrow_* x, \text{(siendo } \Rightarrow_* \text{ derivación)}\}$$

Ejemplo:

Dada la $G = (\sum_T = \{a, b, c\}, \sum_N = \{S, X\}, S, P\}$

$P = \{S ::= aX \quad X ::= bbX \mid c\}$

Podemos expresar el lenguaje generado por G como: $L(G) = \{ab^{2n}c / n \geq 0\}$

Ejemplos de derivaciones: $S \to aX \to abbX \to abbbbX \to abbbbc$

Recursividad: una producción es recursiva si aparece en el antecedente y en el consecuente de la producción. **$A ::= xAy$ $\quad x,y \in \sum^*$**

Recursiva a izquierdas: $A ::= Ay \quad x = \lambda$
Recursiva a derechas: $A ::= xA \quad y = \lambda$

Si la gramática tiene alguna producción recursiva será una gramática recursiva y si un lenguaje tiene una gramática recursiva es infinito.

TIPOS DE GRAMÁTICAS:

Chomsky clasificó las gramáticas en cuatro grandes grupos (G_0, G_1, G_2, G_3), cada uno de los cuales incluye a los siguientes: $(G_3 \subseteq G_2 \subseteq G_1 \subseteq G_0)$

Se dice que dos gramáticas son equivalentes si describen el mismo lenguaje.

Gramáticas Tipo 0:

Son gramáticas que generan los "lenguajes sin restricciones", sus producciones son de la forma:

$u ::= v; \quad u \in \sum^+ \quad v \in \sum^* \quad u = xAy \quad x,y \in \sum^* \quad A \in \sum_N$

Puede demostrarse que todo lenguaje representado por una gramática de tipo 0 puede ser descrito también por una gramática de estructura de frases, cuyas producciones tienen la forma: $xAy ::= xvy$ donde $x, y, v \in \sum^*$ y $A \in \sum_N$

Pueden ser λ

A es no terminal

Ejemplo 1: Dada la gramática $G = \{ \{a, b, c\}, \{A, B, C\}, A, P \}$

$$P \left| \begin{array}{l} A ::= aABC \mid abC \\ CB ::= BC \\ bB ::= bb \\ bC ::= b \end{array} \right.$$

La producción $CB ::= BC$, no está en forma de estructura de frases. Las demás producciones si están en forma de estructura de frases.

Se cambian por:

$$\begin{array}{l} CB ::= XB \\ XB ::= XY \\ XY ::= BY \\ BY ::= BC \end{array}$$

La gramática queda entonces:

$$P \left| \begin{array}{l} A ::= aABC \mid abC \\ CB ::= XB \\ XB ::= XY \\ XY ::= BY \\ BY ::= BC \\ bB ::= bb \\ bC ::= b \end{array} \right.$$

El lenguaje representado es $\{a^n b^n, n \geq 1\}$

Es una gramática de Tipo 0, está en forma de estructura de frases.

Gramáticas Tipo 1:

Pueden ser λ

A es no terminal

Sus producciones son de la forma: xAy::= xvy; donde $v \in \Sigma^+$ $x,y \in \Sigma^*$ $A \in \Sigma_N$

Por tanto, estas gramáticas no pueden contener reglas compresoras. Se admite una excepción en el hecho de que la regla S::= λ puede pertenecer al conjunto de producciones de una gramática de Tipo 1.

Los lenguajes representados por estas gramáticas se llaman "lenguajes dependientes de contexto"

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

$P = \{S ::= aSb \mid ab\}$ $L(G) = \{a^n b^n, n \geq 1\}$

Gramáticas Tipo 2:

Sus producciones son de la forma: A::= v; donde $v \in \Sigma^*$ $A \in \Sigma_N$ (v no puede ser igual a λ a menos que S::= λ).

Los lenguajes representados por las gramáticas de Tipo 2 se llaman "lenguajes independientes del contexto". Es una gramática no compresora a menos que S::= λ.

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

$P = \{S ::= aSb \mid ab\}$ $L(G) = \{a^n b^n, n \geq 1\}$

Es la misma gramática que el ejemplo de Tipo 1 y genera el mismo lenguaje que el de Tipo0. En general, un mismo lenguaje puede describirse mediante muchas gramáticas diferentes.

Otros ejemplo de Gramática de tipo 2:

$G = (\sum_T = \{0, 1\}, \sum_N = \{S, A, B\}, S, P)$

$P = \{S ::= 0B \mid 1A; \quad A ::= 0 \mid 0S \mid 1AA; \quad B ::= 1 \mid 0S \mid 0BB\}$

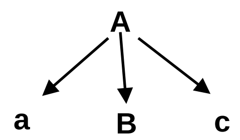
L(G) = {Palabras con el mismo número de ceros que unos}

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

$P = \{S ::= aSa \mid bSb \mid c\}$ $L(G) = \{xcx^{-1} / x \in \{a, b\}^*\}$

Arboles de derivación: dada una gramática $G = \{\sum_T, \sum_N, S, P\}$ una derivación de la palabra u se puede representar mediante un árbol.

- La raíz es S
- Si se aplica una producción del tipo A::= aBc se expresa:

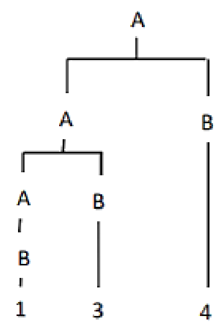


- Hay tantos hijos del nodo A como símbolos tenga la parte derecha de la producción
- Las hojas leídas de izquierda a derecha constituyen una forma sentencial
- Si todos los vértices son símbolos de $\sum_T$ entonces se tiene una sentencia $u \in L(G)$

Sea la gramática G= { { 0, 1 , 2 , 3...,9 } , { A , B } , A , P }

$$P = \begin{cases} A ::= AB \mid B \\ B ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{cases}$$

Y sea la derivación: $A \to AB \to ABB \to BBB \to 1BB \to 13B \to 134$



Ambigüedad:

Una palabra o sentencia es ambigua si tiene, al menos, dos árboles de derivación diferentes o dos derivaciones por la izquierda distintas.
Una gramática es ambigua si contiene al menos una sentencia o palabra ambigua.

Derivación por la Izquierda: sucede si y solo si encada paso de la derivación se reemplaza el símbolo no terminal situado más a la izquierda posible.

$G = (\sum_T = \{a, b\}, \sum_N = \{S\}, S, P)$ $P = \{S ::= SS \mid ab\}$ En este caso:

- S → **S**S → ab**S** → ab**S**S → abab**S** → ababab
- S → **S**S → **S**SS → ab**S**S → abab**S** → ababab

La gramática es ambigua porque x = ababab es palabra ambigua, tiene dos derivaciones por la izquierda diferentes o dos árboles de derivación distintos.

Gramática Bien Formada: es una gramática que no tiene reglas innecesarias, ni símbolos inaccesibles, ni reglas no generativas, ni reglas de redenominación, ni reglas reductoras y, si se tiene $S ::= \lambda$, hay que eliminar el axioma inducido.

La **depuración** de una gramática consiste en una serie de transformaciones para obtener una gramática equivalente bien formada a partir de una que no lo es.

1. Reglas innecesarias: son de la forma $A ::= A$ con $A \in \sum_N$. Estas reglas se eliminan.

2. Símbolos inaccesibles*:* se quiere una gramática conexa, los símbolos no- terminales tienen que ser accesibles desde el axioma. Por ello, todo símbolo no-terminal inaccesible desde del axioma se elimina. Para su eliminación se utiliza una matriz booleana y el siguiente algoritmo:

   Se marca el axioma y se marcan los símbolos no terminales inducidos por el axioma o por variables ya marcadas. Si no se pueden marcar más símbolos, los no marcados son inaccesible. Se eliminan esos símbolos y las producciones dónde están.

   P = { S::= aB | bc   B::= bA | b   C::= cC |C   D::= dC}

	S	A	B	C	D
S					
A					
B					
C					
D					

Símbolos accesibles desde S: B, C

Símbolos accesibles desde B: C, A

Símbolos accesibles desde C: A

Símbolos accesibles desde A: Ø

   El símbolo D no es accesible desde el axioma así que se elimina:
   P = { S::= aB | bc   B::= bA | b   C::= cC |C}

3. Reglas superfluas o no generativas*:* son producciones que tienen en su cadena símbolos no-terminales que no participan en la elaboración de palabras. Ejemplo: C::= cC

   Para eliminarlas se utiliza el algoritmo:

   Sea $P_G = \{A ::= x \ / \ \exists\ S \rightarrow_+ uAV \rightarrow_+ uxv \rightarrow_+ y \in \sum_T^* \}$ (el conjunto de las producciones generativas)

   - Paso 1: $P_0 = N_0 = \emptyset$, $i = 0$
   - Paso 2: $P_{i+1} = \{A ::= x \ / \ x \in (\sum_T \cup N_i)^* \}$ $N_{i+1} = \{A \in \sum_N \ / \ \exists\ A ::= x \in P_{i+1} \}$
   - Paso 3: Si $N_{i+1} \neq N_i$ hacer $i = i+1$ y volver al paso 2
   - Paso 4: Si $N_{i+1} = N_i$ entonces $P_G = P_{i+1}$

   Se eliminarán todas las producciones que no se encuentre en el conjunto P_G

$P = \{S ::= AB \mid A \mid CS1 \mid 0E \quad A ::= 0AS \mid \lambda \mid A0 \mid C \quad B ::= B1 \mid 1 \quad E ::= E1\}$

Se empieza: $i = 0$, $P_0 = N_0 = \emptyset$

Producciones que a la derecha tienen solo terminales o símbolos en N_i siendo $N_0 = \emptyset$

Iteración 1: $P_1 = \{B ::= 1 \quad A ::= \lambda\}$

$N_1 = \{A, B\}$ (es distinto de N_0 así que seguimos)

Iteración 2: $P_2 = \{B ::= 1 \quad A ::= \lambda \quad B ::= B1 \quad A ::= A0 \quad S ::= AB \quad S ::= A\}$

$N_2 = \{A, B, S\}$ (es distinto de N_1 así que seguimos)

Producciones que a la derecha tienen solo terminales o símbolos en N_i siendo $N_1 = \{A, B\}$

Iteración 3: $P_3 = \{B ::= 1 \quad A ::= \lambda \quad B ::= B1$
$A ::= A0 \quad S ::= AB \quad S ::= A \quad A ::= 0AS\}$

Producciones que a la derecha tienen solo terminales o símbolos en N_i siendo $N_2 = \{A, B, S\}$

$N_3 = \{A, B, S\} = N_2$ (Se para el algoritmo)

Se eliminan las producciones que no estén en P_G ($=P_3$)

$P' = \{ S ::= AB \mid A \quad A ::= 0AS \mid \lambda \mid A0 \quad B ::= B1 \mid 1\}$

También se puede utilizar otro método:

1. Se marcan los símbolos no terminales para los que existe una producción del tipo $A ::= x \,/\, x \in \Sigma^*_T$ o todos los símbolos no terminales que contenga "x" ya estén marcados.
2. Los símbolos no marcados son superfluos y se eliminan ellos y las producciones dónde estén.

$P = \{S ::= AB$	A	$CS1$	$0E$		$A ::= 0AS$	λ	$A0$	C	$B ::= B1$	1	$E ::= E1\}$
9	8	7	X	X	5	10	4	6	X	2	3	1	X

Como se ve en el ejemplo, se van seleccionando los símbolos no terminales y de los que vienen. Asimismo, después de marcar esos se abren otras opciones (como en el 8 o el 10) Finalmente, los símbolos no marcados se descartan

4. <u>Reglas reductoras</u>: hay que eliminar las reglas de la forma $A ::= \lambda$ con $A \neq S$. Si $\lambda \in L(G)$ entonces $S ::= \lambda$.

Formas de eliminarlas (siempre que $x, y \neq \lambda$):

Si se tiene:				
$A ::= \lambda$ $B ::= xAy$	$\in P_1$	$\Rightarrow$	$B ::= xAy$ $B ::= xy$	$\in P_2$
Si se tiene: $A ::= \lambda$ $B ::= xAyAz$	$\in P_1$	$\Rightarrow$	$B ::= xAyAz$ $B ::= xyAz$ $B ::= xAyz$ $B ::= xyz$	$\in P_2$

5. Reglas de redenominación: hay que eliminar las reglas de la forma A::= B / A, B ∈ $\sum_N$. Para ello se utiliza el método de simplificación de derivaciones.

Si se tiene: A :: = B B ::= x	∈ P_1	⇒	Se **incluye** A::= x (Por cada regla de tipo B::= x)	∈ P_2	En este caso se elimina de P_2 A::= B
Si se tiene: A ::= B_1 B_1 ::= B_2 B_2 := ... . ::= B_n B_n ::= x	∈ P_1	⇒	Se **incluye** A::= x	∈ P_2	Se eliminan de P_2 A::= B1 … a Bn::= x

6. Axioma inducido: si se tiene en la gramática la producción S::= λ y S aparece en la derecha de una producción se dice que el axioma esta inducido.

Si G = ($\sum_T$, $\sum_N$, S, P) con axioma inducido, se construye otra gramática sin axioma inducido: G' = ($\sum_T$, $\sum_N$'= {$\sum_N$ ∪ S'} S', P') con P' = P ∪ {S'::= S}

No hay un orden establecido predeterminado para la depuración de gramáticas, pero es aconsejable realizar primero los pasos que implican eliminación de reglas y símbolos (reglas no generativas y símbolos no accesibles) y posteriormente eliminar el axioma inducido y las reglas reductoras y de redenominación.

Sin embargo, quizás haya que repetir algún proceso varias veces hasta conseguir la depuración de una gramática.

Gramáticas Tipo 3: los lenguajes generados por estas gramáticas se denominan "Lenguajes Regulares". No son compresoras.

Para toda gramática lineal derecha existe una gramática lineal izquierda equivalente y para toda gramática lineal izquierda existe una gramática lineal derecha equivalente.

a) Gramáticas lineales por la izquierda: las reglas de producción son de la forma:

A::= a; A::= Va dónde A, V ∈ $\sum_N$ (A y V son no terminales) y a ∈ $\sum_T$ (a es terminal)

b) Gramáticas lineales por la derecha: cuyas reglas de producción son de la forma:

A::= a; A::= aV dónde A, V ∈ $\sum_N$ (A y V son no terminales) y a ∈ $\sum_T$ (a es terminal)

Ejemplo Tipo 3:

$G = (\sum_T = \{0, 1\}, \sum_N = \{A, B\}, A, P)$

$P = \{A ::= B1 \mid 1 \qquad B ::= A0\}$

Gramática lineal izquierda:

$-L(G) = \{1(01)^n \,/\, n \geq 0\}$

$-L(G) = (10)^*1 = 1(01)^*$

$G = (\sum_T = \{0, 1\}, \sum_N = \{A, B\}, A, P)$

$P = \{A ::= 1B \mid 1 \qquad B ::= 0A\}$

Gramática lineal derecha:

$-L(G) = \{1(01)^n \,/\, n \geq 0\}$

Ejercicio Extra:

Construir las gramáticas que generan los siguientes lenguajes, indicando de qué tipo es la gramática obtenida y cuando sea posible expresar L mediante expresión regular.

$L = \{a^m b^n \,/\, m > 0 , n \geq 0\}$
$L = aa^*b^*$, es lenguaje regular. La gramática que lo genera debe ser lineal

Una gramática lineal derecha que lo genera es:
$G = (\sum_T = \{a, b\}, \sum_N = \{S, A\}, S, P)$ $P = \{S ::= aS \mid aA \mid a \qquad A ::= bA \mid b\}$