# CAPITULO 5: GRAMÁTICAS

## 5.1. GRAMÁTICA EN UN LENGUAJE NATURAL:

En un lenguaje natural *la estructura de las frases, se describen por medio de una gramática* que agrupa las palabras en categorías sintácticas tales como sujetos, predicados, frases preposicionales, etcétera. Estas construcciones gramaticales surgen como un intento de explicar las formas admitidas por el Lenguaje, con las cuales se construyen las frases, aunque en su definición se presentarán excepciones gramaticales.

Para poder describir a los Lenguajes Naturales, podemos escribir reglas como las que enumeramos en el siguiente $P_n$ conjunto:

$P_n$ = {<oración> → <sujeto> <predicado>,
<sujeto> → <sustantivo> | <artículo> <sustantivo> | <artículo> <sustantivo> <adjetivo>,
<predicado> → <verbo> | <verbo> <modificador>,
<artículo> → el | la | los | las | un,
<sustantivo> → niño | madre | árbol | vehículos,
<adjetivo> → pequeño | alto | joven,
<verbo> → aprende | corre | juega | crece,
<modificador> →velozmente | risueño | temprano}

Vale la pena destacar que esta es "una versión muy simplificada" de la gramática castellana, y si bien da lugar a oraciones que son correctas en nuestro idioma, también puede generar oraciones que resulten incorrectas desde el punto de vista tanto semántico (de su significado), como sintáctico (de su estructura).

Este es el caso de:

El árbol juega (incorrecta semánticamente)

Los madre come risueño (incorrecta sintácticamente)

Estos errores se solucionan en la lengua castellana agregando reglas que regulan y restringen las combinaciones de terminales y que corresponden al análisis morfológico y semántico del lenguaje. (En los lenguajes formales no existe el análisis morfológico que es el de género, número y persona, es decir el que controla la concordancia entre sujeto y verbo o la concordancia entre artículo, sustantivo y adjetivo).

## 5.2. GRAMÁTICA EN UN LENGUAJE FORMAL:

**Un Lenguaje Formal *surge a partir de su gramática***, y por lo tanto no presenta excepciones en su definición. Esto es así, porque los Lenguajes Formales son los que se utilizan para que se comuniquen los hombres con las máquinas. De esta manera a partir de las gramáticas formales es como surgen los Lenguajes de Programación.
La gramática es un modelo matemático que incluye el conjunto de reglas que describen cómo se pueden generar todas las posibilidades combinatorias de un determinado lenguaje, y sólo las de dicho lenguaje. El mecanismo para generar las palabras es un proceso de derivación; a partir de una cadena, que podemos considerar como "la semilla" de todas las palabras del lenguaje, se la va afectando de sucesivas transformaciones (o derivaciones), de acuerdo con reglas precisas de la gramática hasta llegar a construir una cadena que se considera perteneciente al lenguaje.
Los objetivos de una gramática son:

- Definir si una sentencia pertenece o no al lenguaje.
- Describir estructuralmente las sentencias del lenguaje.

Podemos decir que las gramáticas formales son **descripciones** estructurales de las sentencias de los lenguajes, tanto formales (leng. de programación), como naturales (humanos). Es decir, es mucho más que una representación del lenguaje.

Para describir un lenguaje enumerábamos todas las palabras que lo forman. Ahora, con las gramáticas, el lenguaje puede quedar descrito por medio de:

- El *alfabeto* con el que están construidas sus palabras. Es un conjunto de símbolos denominados *terminales*. Por convención los escribiremos con letras minúsculas y dígitos: $\mathbf{V_T}$
- El *alfabeto* que será utilizado como auxiliar en el proceso de formación de cadenas (derivación), pero no formarán parte de las cadenas del lenguaje. Estos son los símbolos que permiten representar estados intermedios de la generación de las palabras (subconjuntos del lenguaje). Por convención se escriben con letras mayúsculas: $\mathbf{V_N}$
- Un símbolo inicial, "start", es un símbolo especial que pertenece a los no terminales. Es el símbolo del que partirá la obtención de cualquiera de las palabras del lenguaje (se lo denominado *axioma de la gramática*): **S**
- Un conjunto de producciones, que representan cómo se realiza la transformación desde los símbolos no terminales a las palabras del lenguaje (una producción es una regla de definición): P

  Las producciones son el "centro de las gramáticas"

  En P, cada producción es un par (α, β) tal que su vinculación responde a la forma general: $\alpha \rightarrow \beta$ donde α y β son cadenas y la $\rightarrow$ significaría **es sustituida por**

  α es el primer miembro o parte izquierda de la producción. Está integrada por símbolos no terminales, salvo la cadena vacía, es decir por símbolos que $\epsilon\ V_N^+$.

  Existirá, por lo menos un α que sea la cadena formada con el símbolo de inicio S.

  β es el segundo miembro o parte derecha de la producción. Está integrada por símbolos terminales y/o no terminales y/o la cadena vacía, es decir por símbolos que $\epsilon\ V^*$.

  Por lo tanto, al hablar de una producción estamos hablando de *una cadena que define a otra cadena*.

  Las producciones se pueden interpretar como reglas de reescritura donde α se reemplaza por β.

  También se pueden interpretar como llamadas a funciones, donde la función α llama a la función β.

  Dependiendo de las formas que tengan las producciones se obtienen diferentes tipos de gramáticas. Una clasificación de los distintos tipos, constituye lo que se conoce como jerarquías de Chomsky (se verá más adelante)

Sintetizando: una gramática utiliza dos conjuntos disjuntos de símbolos (el de los terminales $V_T$ y el de los no terminales $V_N$) y en general se la puede representar como un conjunto de 4 elementos: $G = \{V_T, V_N, P, S\}$

$V_T \cup V_N$ = alfabeto de símbolos gramaticales

$V_T \cap V_N = \{\ \}$

Una cadena del lenguaje ***no contendrá*** símbolos no terminales y el lenguaje generado por una gramática = L(G) es el conjunto de todas las palabras (o sentencias de la gramática) que se pueden obtener a partir del axioma o símbolo inicial de la gramática por aplicación de derivaciones. Cada derivación implica la transformación de una cadena en otra.

Así, podemos decir que, la derivación es un proceso que apunta a obtener las palabras del lenguaje, por lo tanto ***derivar una cadena es ir sustituyendo una a la vez, las subpartes de ella que coincidan con***

***el α de una producción por el β de la misma***. La palabra queda definitivamente formada cuando la constituyen sólo símbolos terminales (como se dijo antes)
Puedo considerar, en la cadena de partida, 3 parte constitutivas que identificaré como $\gamma\alpha\delta$ (contiene a un símbolo no terminal .)
Siendo $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ una producción para la gramática, significa que en la siguiente escritura de la cadena se sustituye α por β, es decir, la cadena $\gamma\boldsymbol{\alpha}\delta \Rightarrow \gamma\boldsymbol{\beta}\delta$ (deriva en . . .)

Es decir, partiendo del símbolo inicial, aplicando sucesivamente las distintas producciones (por eso también llamadas reglas de reescritura) alcanzo a formar las palabras definitivas.
Se llama ***forma sentencial*** de una gramática a una derivación de la misma (es decir a cualquier estadio intermedio de la formación de las palabras del lenguaje); se trata de una cadena formada por símbolos terminales y no terminales. Así S y $\gamma\alpha\delta$ son formas sentenciales.
Se llama ***sentencia*** o frase o instrucción del lenguaje a cualquier cadena que sea el resultado último de una derivación a partir de un símbolo inicial y que está formada sólo por símbolos terminales.

Ejemplo:

Dada la gramática: ***G = { {0, 1}, {A, B}, P, A}***

Donde:

P={(A→ B1), (A → 1), (B → A0)}

$V_T$ = {0, 1}

$V_N$ = {A, B}

S = A

Derivación para lograr una palabra del lenguaje descripto por esta gramática: Observemos que el símbolo inicial puede ser sustituido por una de dos cadenas distintas, ∴ a partir del inicio se realizará una u otra transformación.

Inicio: A

Según la regla (A → 1) se llega a **1** (esta será una palabra del lenguaje)

Según la regla (A→ B1) se llega a **B1**, ésta cadena contiene un no terminal ∴ hay que transformarla aplicando otra producción: con (B → A0) se llega a **A01**, ésta cadena contiene un no terminal ∴ hay que derivar aplicando otra producción: con (A → 1) se llega a **101** (esta será una palabra del lenguaje).

Pero a partir de A01 se puede aplicar la producción (A→ B1) legando a B101, aplicando luego (B → A0) se llega a A0101, aplicando (A → 1) se termina en **10101**

Resumiendo:

La secuencia de derivaciones para generar una palabra es:

A ⇒ **1**

A ⇒ B1 ⇒ A01 ⇒ **101**

A ⇒ B1 ⇒ A01 ⇒B101 ⇒ A0101 ⇒**10101** (A0101: Forma sentencial; 10101: Sentencia)

A ⇒ B1 ⇒ A01 ⇒B101 ⇒ A0101 ⇒ B10101 ⇒ A010101 ⇒ **1010101**

Entonces: L = $\{1(01)^n$ siendo n = 0, 1, 2, ...} ={1, 101, 1010101, 101010101,…..}

| La gramática ***{{0, 1}, {A, B}, {(A→ B1), (A → 1), (B → A0)}, A}*** genera el lenguaje ***{1, 101, 1010101, 101010101,…..}*** |
|---|

Podemos plantear dos estrategias para reemplazar el símbolo no terminal:

- Derivación por la izquierda: Consiste en reemplazar el no terminal de más a la izquierda
- Derivación por la derecha: Consiste en reemplazar el no terminal de más a la derecha

Ejemplo:

Dada la gramática: ***G = ({a, b},{S, A, B}, S, P )***

Donde:

P = { (S→aABAa), (A→ε), (A →aA), (B→ε), (B→bB)}

Aplicando derivación por la izquierda la secuencia para generar algunas palabras del lenguaje puede ser:

S ⇒ aABAa ⇒ aBAa ⇒ aAa ⇒ **aa**

S ⇒ aABAa ⇒ aBAa ⇒ aabBAa ⇒ aabAa ⇒ aabaAa ⇒ **aabaa**

S ⇒ aABAa ⇒ aBAa ⇒ aabBAa ⇒ aabbBAa ⇒ aabbAa ⇒ **aabba**

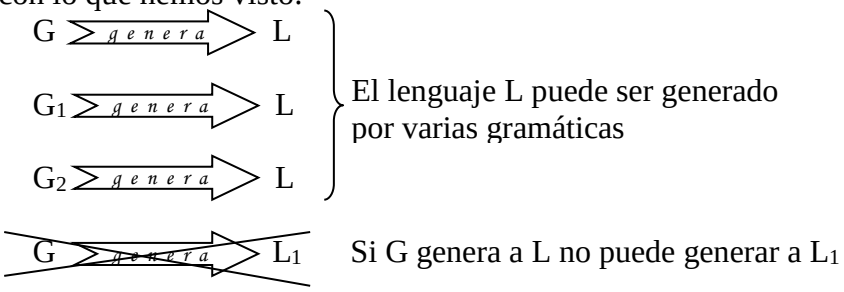
Aplicando derivación por la derecha la secuencia para generar algunas palabras del lenguaje puede ser:

S ⇒ aABAa ⇒ aABa ⇒ aAa ⇒ **aa**

En este caso se trata de una gramática que genera las palabras del lenguaje L={ $a^n b^m a^l$ siendo $n \geq 1$ y $l \geq 1$ y $m \geq 0$}

## 5.3. CLASIFICACIÓN DE LAS GRAMÁTICAS

De acuerdo con lo que hemos visto:




Toda gramática genera un único lenguaje, pero distintas gramáticas pueden generar el mismo lenguaje, por eso no podemos pensar en clasificar las gramáticas por el lenguaje que generan, por este motivo tenemos que pensar en clasificaciones basadas en ***la forma*** de la gramática, más que en la naturaleza del lenguaje que generan.

Noam Chomsky, hacia fines de los años 50, clasificó las gramáticas y los lenguajes dentro de cuatro familias jerárquicamente ordenadas como modelos potenciales del lenguaje natural.

Esta clasificación se establece aumentando las restricciones sobre la forma de las producciones y se las nombra como:

Gramática de tipo 0

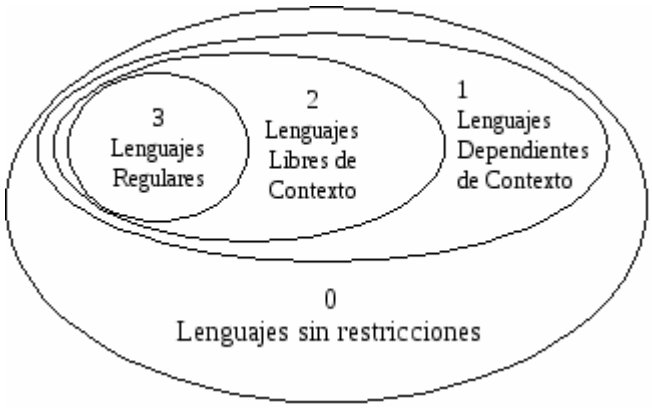
Gramática de tipo 1

Gramática de tipo 2

Gramática de tipo 3

Cada gramática de un determinado tipo surge de aplicar ciertas restricciones al tipo anterior. De tal modo que los lenguajes generados por ellas guardan entre sí la misma relación

Podríamos representarlo gráficamente de la siguiente manera:




.

Vimos que una producción quedaba definida por reglas de la forma $\alpha \rightarrow \beta$, entonces los tipos de gramática (y por lo tanto las clases de lenguajes que se puedan generar) quedan fijadas por las limitaciones que tengan las cadenas $\alpha$ y $\beta$. Así tenemos, en la jerarquía de Chomsky:

**Tipo 1 o Gramática Dependiente del Contexto**: en este caso cada producción es de la forma $\boldsymbol{\alpha A \beta \rightarrow \alpha \gamma \beta}$ donde:

$\mathbf{A} \in V_N$ (1)

$\boldsymbol{\alpha}, \boldsymbol{\beta} \in (V_T \cup V_N)^*$ es decir $\alpha$ y $\beta$ son cadenas que contienen a cualquier símbolo de la gramática, (2) sea o no terminal e inclusive a la cadena vacía (4)

$\boldsymbol{\gamma} \in (V_T \cup V_N)^+$ es decir $\boldsymbol{\gamma}$ contiene símbolos terminales y no terminales (3) y NO podría ser vacía (5)

> ***A** puede ser sustituido por $\boldsymbol{\gamma}$ si está acompañada de $\boldsymbol{\alpha}$ por la izquierda y de $\boldsymbol{\beta}$ por la derecha.*

Sobre estas producciones se impone la restricción de que cualquier derivación contiene mayor cantidad de símbolos que la cadena en la que se hizo la sustitución.
Esta propiedad es fácilmente demostrable.
Siendo que:

por la definición de estas producciones $\Rightarrow |\boldsymbol{A}| \geq \boldsymbol{1} \Rightarrow |\boldsymbol{\alpha A \beta}| \geq \boldsymbol{1}$

por la característica definida para $\gamma$, $\gamma \neq \varepsilon \Rightarrow |\boldsymbol{\gamma}| \geq \boldsymbol{1} \Rightarrow |\boldsymbol{\alpha \gamma \beta}| \geq \boldsymbol{1}$

$\therefore$ se demuestra que $|\boldsymbol{\alpha A \beta}| \leq |\boldsymbol{\alpha \gamma \beta}|$

Podría decirse entonces, que la característica de este tipo de gramática es que:
- *el miembro izquierdo de las producciones <u>contiene al menos un símbolo no terminal</u>*
- *la cadena sustituta, que está en el miembro derecho de las producciones <u>nunca será vacía</u>, por lo tanto es una secuencia de símbolos terminales y/o no terminales*

y que la restricción es que:
- *la cantidad de símbolos que forman la cadena del lado izquierdo es <u>menor o a lo sumo igual</u> que la cantidad de símbolos de la cadena del lado derecho.*

Puede verse que las producciones de este tipo de gramática son tales que las sustituciones sólo pueden efectuarse en cierto contexto (el símbolo 'A' puede transformarse en 'γ' si y sólo si γ está precedido por 'α', y le sigue 'β'), esto es, son gramáticas dependientes (sensibles) al contexto, por tanto, pueden hacer que un sintagma sea sistemáticamente igual a otro.

Todo lenguaje formal generado por una Gramática dependiente del Contexto y que no puede ser generado por una gramática de menor jerarquía, se llama Lenguaje Dependiente del Contexto

Si bien γ no puede ser la cadena vacía se admite una excepción en la regla **S** → ε (siendo S el axioma de la gramática). Como consecuencia se tiene que la palabra vacía pertenece al lenguaje generado por la gramática sólo si contiene esta regla.

Con los lenguajes generados por este tipo de gramática se hacen los analizadores sintácticos (transforma su entrada en un árbol de derivación). El análisis sintáctico convierte el texto de entrada en otras estructuras (comúnmente árboles) que son más útiles para el posterior análisis y capturan la jerarquía implícita de la entrada.

Ejemplos:

1) ***G = ({a, b}, {S, A}, S, P )***
P = { *(S → aS), (S → aA) (A → ba), (A → aS), (A → b)*}

En estas producciones el 1er. miembro es un no terminal (cumple con ① ) y está sólo (α y β son vacíos, cumple con ④ ). El 2do. miembro de las producciones contiene terminales y no terminales (cumple con ③ ) y estos están solos, es decir, van acompañados por vacío igual que en el primer miembro (cumple con ④ ).

2) ***G = ({a, b}, {S, A, B,C}, S, P )***
P = { *(S → aB), (S → bA) (A → a), (A → aS), (A → bAA), (B → b), (B → bS), (B → aBB)*}

3) NO es gramáticas de tipo 1:
***G = ({a, b}, {S, A}, S, P )***
P = { *(S → abAS), (S → a) (A → b), (**ab**A → baab)*}

En esta producción no se respeta el contexto. Debería ser: *abA → abab*

***$G_1$ = ({a, b, c}, {S, B,C}, S, P )***
P = { *(S → aSBC), (S → aBC), (CB → BC), (bB → bb), (bC → bc), (cC → cc), (aB → ab)*}

En esta producción no se respeta el contexto.

Para generar el lenguaje a partir de esta gramática:
*S ⇒ aSBC ⇒ aaBCBC ⇒ aabCBC ⇒ aabBCC ⇒ aabbCC ⇒ aabbcC ⇒* $\boxed{aabbcc}$

*S ⇒ aSBC ⇒ aaSBCBC ⇒ aaaBCBCBC ⇒ aaaBBCCBC ⇒ aaaBBCBCC ⇒ aaabBCBCC ⇒ aaabBBCCCC ⇒ aaabbBCCCC ⇒ aaabbbCCCC ⇒ aaabbbcCCC ⇒ aaabbbccCC ⇒ aaabbbcccC ⇒* $\boxed{aaabbbccc}$

∴ L={ *aabbcc, aaabbbccc, …* }, es decir L={ $a^n\ b^n\ c^n$ *siendo* $n \geq 1$ }

La producción *CB → BC* puede ser sustituida por:
*(CB → XB), (XB → XY), (XY → BY), (BY → BC)* entonces tendríamos:

***$G_2$ = ({a, b, c}, {S, B,C}, S, P )***
P = {*(S → aSBC), (S → aBC), (bB → bb), (bC → bc), (cC → cc), (aB → ab), (CB → XB), (XB → XY), (XY → BY), (BY → BC)*}

$G_2$ es equivalente a $G_1$ pues generaría el mismo lenguaje, pero $G_2$ sería una gramática de tipo 1

**Tipo 2 o Gramática Independiente o libres del Contexto**: en este caso cada producción es de la forma $\mathbf{A} \rightarrow \boldsymbol{\gamma}$ donde:

$\mathbf{A} \in V_N$

$\boldsymbol{\gamma} \in (V_T \cup V_N)^+$ es decir $\gamma$ NO podría ser vacía

| |
|---|
| ***A** puede sustituirse por $\boldsymbol{\gamma}$ independientemente de las cadenas por las que esté acompañada.* |

La característica de una gramática de tipo 2 es:
- *el miembro izquierdo de las producciones <u>es un símbolo no terminal</u>*
- *la cadena sustituta, que está en el miembro derecho de las producciones <u>nunca será vacía</u>, por lo tanto empieza con un terminal o no terminal*

y la restricción es:
- *la cantidad de símbolos que forman la cadena del lado izquierdo es siempre <u>uno</u>.*

La mayor parte de los lenguajes de programación pueden describirse mediante gramáticas de este tipo, es decir son lenguajes libres de contexto.

Un ejemplo típico del uso de una gramática libre de contexto en los lenguajes de programación es la **descripción de expresiones aritméticas**.

**Asociatividad de los operadores**

En la mayoría de los lenguajes de programación los cuatro operadores aritméticos (suma, resta, multiplicación y división), son **asociativos por la izquierda**. Por ejemplo decimos que el operador + se asocia por izquierda porque un operando con signos positivos a ambos lados de él pertenece al operador que está su izquierda:

Ej:

A + B +5, tenemos + B+, el operando B con signo positivos a ambos lados pertenece al operador que está a su izquierda, es decir primero se realiza A +B y luego se suma 5.

Se podría representar: (A+B) +5.

**Asociatividad por derecha**

Por ejemplo la exponenciación y el operador de asignación en C

Ejemplo:

a=b=c es lo mismo que a= (b=c)

3 + 5**2 -7/3 es lo mismo que 3 + (5**2) -7/3

**Precedencia de los operadores**

Dada la expresión A +B*5 hay dos posibles formas de resolver si no se establece un orden de prioridad de los operadores: (A+B)*5 ó A + (B*5**). Las reglas que definen la precedencia relativa de los operadores son necesarias cuando hay más de un tipo de operador presente en la expresión.**

Ejemplo

Para la siguiente gramática la asociatividad y precedencia es la siguiente:

MENOR PRECEDENCIA asociativo por izquierda: + -
MAYOR PRECEDENCIA asociativo por derecha: * /

No Terminales: expr, term, factor, digito, donde :

factor → digito | (expr)
term → term*factor
| term/factor
| factor
expr → expr + expr
| expr – term
| term

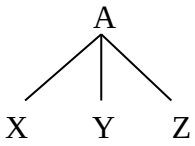
La gramática resultante es:

expr→ expr + expr
term→term*factor | term/factor | factor
factor → digito | ( expr)
digito → 0| 1 | 2 | 3 | 4 |……| 9

**Árbol de análisis sintáctico.**

El análisis sintáctico (parsing) consiste en determinar como una cadena de terminales deriva a partir del símbolo **S** (star o comienzo) de la gramática. Si no puede derivarse entonces habrá que obtener los errores dentro de la cadena.

**Un árbol de análisis sintáctico muestra en forma grafica**, la manera en que el símbolo inicial de una gramática deriva a una cadena en el lenguaje. Por ejemplo A → XYZ, un árbol de análisis sintáctico podría tener un nodo interior etiquetado como A con tres hijos llamados XYZ




**Terminología de árbol**

- Un árbol consiste en uno o más nodos
- Uno de los nodos es el raíz, todos los demás tienen un padre, el cual está por encima del nodo hijo.
- Dos nodos que derivan de un mismo padre se llaman hermanos
- Un nodo sin hijos se llama hoja. Los nodos que tienen uno o mas hijos se llaman nodos interiores.
- Un descendiente de un nodo N, es el mismo, un hijo de N o un hijo de un hijo de N y así sucesivamente. Decimos que el nodo N es un ancestro de M, si M es descendiente de N.

**Ejemplo**

La siguiente gramática describe las expresiones que consisten en dígitos, signos positivos y negativos. Estas expresiones pueden llamarse como: “listas de dígitos separadas por signos positivos y negativos”. La siguiente gramática describe la sintaxis de estas expresiones. Las producciones son:

lista → lista+digito (1)
| → lista – digito (2)
| → digito (3)
digito→ 0 | 1 | 2 | 3 | 4 | ……| 9 (4)
lista →lista + digito | lista – digito | digito (5)

donde:

$V_N$ = {lista, digito}
$V_T$ = {+,-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
lista es el símbolo de comienzo o start ya que sus producciones se dan primero.

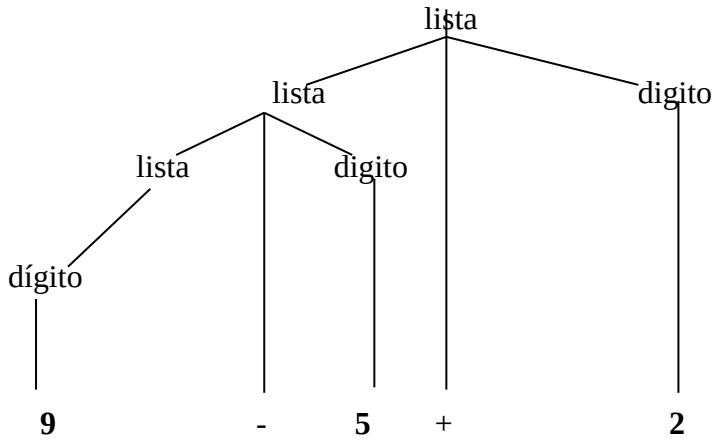
**Derivaciones**

Una gramática deriva o produce cadenas empezando con el símbolo inicial y sustituyendo en forma repetida un no terminal por su producción. Las derivaciones, **son las cadenas de terminales** que pueden producirse del símbolo inicial del lenguaje definido por la gramática.

**Árbol**

La producción lista→ lista +digito (1), de la gramática anterior y atendiendo a sus producciones podemos concluir en

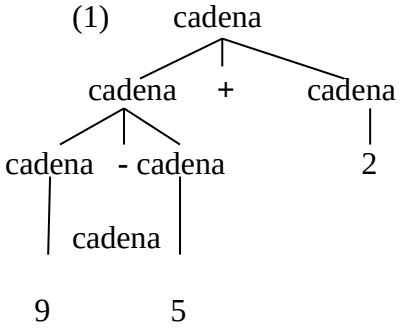
lista → lista – digito +digito (2)
digito→digito – digito +digito (3)

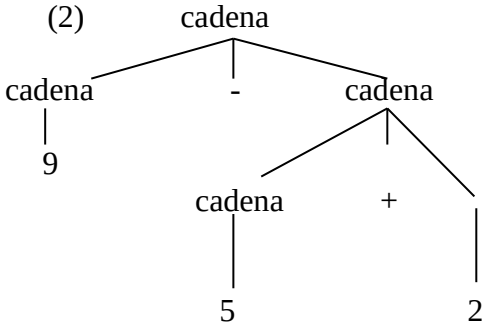



**Ambigüedad**

Una gramática puede tener más de un árbol de análisis sintáctico que genere una cadena de terminales, en este caso la gramática es ambigua. Debe tratar de evitarse con reglas adicionales para resolver las ambigüedades.
Suponemos que utilizamos **una sola cadena no terminal** y que **no** diferenciamos entre los dígitos y las listas. Podemos escribir la siguiente gramática:
cadena → cadena + cadena | cadena – cadena | 0|1|2|3|4|5|6|7|8|9

Los dos árboles para 9 -5 +2 corresponde a las dos formas de aplicar paréntesis a la expresión:
(9 - 5) + 2 (1)
9 – (5 +2) (2)



**Tipo 3 o Gramática Lineal o Regular:** en este caso cada producción es de la forma
**A → αB** y **A → α**
o
**A → Bα** y **A → α**

donde:
**A, B** $\epsilon$ $V_N$
**α** $\epsilon$ $V_T$

La característica de una gramática de tipo 3 es:
- *el miembro izquierdo de las producciones es un símbolo no terminal*
- *la cadena sustituta, que está en el miembro derecho de las producciones nunca será vacía, por lo tanto puede ser una secuencia de terminales con un no terminal como sufijo (gramática regular por la derecha) o con un no terminal como prefijo (gramática regular por la izquierda) o simplemente una secuencia de terminales*

y la restricción es:
- *la cantidad de símbolos que forman la cadena del lado izquierdo es uno.*

Ejemplos:
1) por la derecha
***G = ({a, b}, {S, A}, S, P )***
P = { *(S → bbS), (S → aaA) (A → aaA), (A → bb)*}

El lenguaje generado:
$S \Rightarrow aaA \Rightarrow aaaaA \Rightarrow aaaaaaA \Rightarrow \boxed{aaaaaabb}$

$S \Rightarrow aaA \Rightarrow aaaaA \Rightarrow \boxed{aaaabb}$

$S \Rightarrow bbS \Rightarrow bbaaA \Rightarrow bbaaaaA \Rightarrow \boxed{bbaaaabb}$

$S \Rightarrow bbS \Rightarrow bbaaA \Rightarrow \boxed{bbaabb}$

$\therefore$ L={ $(bb)^n (aa)^k$ *siendo* $n \geq 0$ *y* $k \geq 1$ }

2) por la izquierda
$G_2$ = ({C, D}, {a}, P, S)
donde $P_2$ es el siguiente conjunto de producciones
S → ε
S → Ca
C→ Da
C→ a
D→ Ca
donde L(G2) = { $a^{2n}$ con n >=0}

Un ejemplo de lenguaje regular es el conjunto de todos los números binarios.

**Tipo 0 o Gramática sin restricciones**: en este caso cada producción es de la forma $\alpha \rightarrow \beta$ donde:

$\alpha \in (V_T \cup V_N)^+ \Rightarrow$ no habrá producción de la forma $\varepsilon \rightarrow \beta$

$\beta \in (V_T \cup V_N)^*$

Ejemplos:

1) G2 = (Vn={S},Vt= {0,1}, S, P)

   donde P :

   S→ 000S1111

   0S1→ 01

   se concluye que

   $L(G2) = \{ 0^{(3n)} 1^{(3n)} , n>=1\}$

2) $G_{8=}$ ( Vn= {A,S}, Vt={a,b}, S, P)

   donde P:

   S→aS

   S → aS

   A→ bA

   A→b

   se concluye

   $L(G8)= a\ a^*\ b\ b^*$

Cuadro que sintetiza la clasificación de gramáticas según sus producciones:

| Tipo | Forma de las producciones | | |
|---|---|---|---|
| 0 | $\alpha A \beta \longrightarrow \omega$ | | Generales |
| 1 | $\alpha A \beta \longrightarrow \omega$ con $\omega \neq \varepsilon$<br>$S \longrightarrow \varepsilon$, $\lvert\alpha A \beta\rvert \leq \lvert\omega\rvert$ | | Dependientes del contexto |
| 2 | $A \longrightarrow \omega$ con $\omega \neq \varepsilon$<br>$S \longrightarrow \varepsilon$ | | Libres del contexto |
| 3 | $A \longrightarrow aB$<br>$A \longrightarrow a$<br>$S \longrightarrow \varepsilon$ | Lineales a la derecha | Regulares |
| | $A \longrightarrow Ba$<br>$A \longrightarrow a$<br>$S \longrightarrow \varepsilon$ | Lineales a la izquierda | |

Modelos matemáticos mediante los cuales se resuelven estos lenguajes:

Los lenguajes regulares (de tipo 3) mediante AF

Los lenguajes libres de contexto (de tipo 2) mediante A de pila

Los lenguajes dependientes de contexto (de tipo 1) mediante A lineales

Los lenguajes irrestrictos (de tipo 0) mediante la máquina de Turing