1 DEFINICIONES BÁSICAS E INTRODUCCIÓN A LENGUAJES FORMALES

El Volumen 1 de este libro trata la Sintaxis y la Semántica de los Lenguajes de Programación (LPs) desde el punto de vista de los que necesitan conocerla, sus usuarios, mientras que el Volumen 2 la analizará desde el punto de vista del compilador.

NOTA MUY IMPORTANTE

PARA UNA BUENA COMPRENSIÓN DE TODOS LOS TEMAS,
SIEMPRE RESUELVA LOS EJERCICIOS EN LA MEDIDA QUE VAN APARECIENDO.
NO LOS DEJE COMO ÚLTIMA TAREA DE CADA CAPÍTULO.

La SINTAXIS de un LENGUAJE DE PROGRAMACIÓN describe las combinaciones de símbolos que forman un programa sintácticamente correcto. Como esta SINTAXIS está basada en los LENGUAJES FORMALES, debemos comenzar con este concepto.

Los LENGUAJES FORMALES están formados por PALABRAS, las palabras son CADENAS y las cadenas están constituidas por CARACTERES de un ALFABETO. Esta frase involucra cinco términos (*lenguaje formal*, *palabra*, *cadena*, *carácter* y *alfabeto*) que son fundamentales en este campo. A continuación los analizamos detenidamente.

1.1 CARACTERES y ALFABETOS

Un CARÁCTER (también llamado SÍMBOLO) es el elemento constructivo básico. Es la entidad fundamental, indivisible, a partir de la cual se forman los alfabetos.

Un ALFABETO es un conjunto finito de caracteres. Se lo identifica, habitualmente, con la letra griega Σ (sigma), y con sus caracteres se construyen las cadenas de caracteres de un Lenguaje Formal.

Ejemplo 1
La letra a es un carácter o símbolo que forma parte del alfabeto español, del alfabeto inglés, etc.

Ejemplo 2
Los caracteres >, = y + son elementos del alfabeto de los operadores de muchos Lenguajes de Programación.

Ejemplo 3
El alfabeto $\Sigma = \{0, 1\}$ proporciona los caracteres utilizados en la construcción de los números binarios.

Un carácter de un alfabeto también puede ser "múltiple".

Ejemplo 4
El alfabeto Σ = {ab, cde} está integrado por dos caracteres: el carácter ab y el carácter cde.

* Ejercicio 1 *
Escriba el alfabeto que se requiere para construir el conjunto de los números enteros con signo en base 10.

1.2 CADENAS

Una CADENA, forma abreviada de la frase *cadena de caracteres*, es una secuencia finita de caracteres tomados de cierto alfabeto y colocados uno a continuación de otro. Es decir: una *cadena* se construye CONCATENANDO caracteres de un alfabeto dado.

Como sinónimo de *cadena* se usa, a veces, el término inglés *string*.

Ejemplo 5
abac (se lee "a-b-a-c") es una cadena formada con caracteres del alfabeto {a, b, c}.

Ejemplo 6
101110 ("uno-cero-uno-uno-uno-cero") es una cadena construida con caracteres del alfabeto {0, 1}.

Ejemplo 7
a es una cadena formada por un solo símbolo de cualquier alfabeto que contenga el carácter a.

➔ Para los conocedores del Lenguaje de Programación C: recuerden la diferencia que existe entre 'a' y "a". Es la misma diferencia que existe entre un carácter de un alfabeto y una cadena compuesta por ese único carácter.

* Ejercicio 2 *
Dado el alfabeto {0, 1, 2}, construya dos cadenas en la que cada uno de estos caracteres aparezca una sola vez.

* Ejercicio 3 *
Dado el alfabeto {ab, cde}, construya una cadena que tenga cuatro caracteres

1.2.1 LONGITUD DE UNA CADENA

La LONGITUD de una cadena S (se representa |S|) es la cantidad de caracteres que la componen.

Ejemplo 8
La longitud de la cadena abac es: |abac| = 4.

Ejemplo 9
La longitud de la cadena b es: |b| = 1.

1.2.2 CADENA VACÍA

La CADENA VACÍA, que se simboliza habitualmente con la letra griega ε (épsilon), es la cadena que no tiene caracteres. En otras palabras: la *cadena vacía* es la cadena de longitud 0 ($|\varepsilon| = 0$).

Nota 1
Este símbolo ε no forma parte de ningún alfabeto.

Nota 2
Algunos autores utilizan la letra griega λ (lambda) para representar a la cadena vacía.

1.2.3 UNA SIMPLIFICACIÓN: LA POTENCIACIÓN DE UN SÍMBOLO

En muchas ocasiones, una cadena puede contener un carácter que se repite un número determinado de veces.

Ejemplo 10
La cadena aaaabbbbbbb, construida sobre el alfabeto {a, b}, está formada por cuatro aes concatenadas con siete bes.

La POTENCIACIÓN de un carácter simplifica la escritura de cadenas como la del ejemplo anterior, de la siguiente manera:

Sea c un carácter cualquiera; entonces, c^n representa la concatenación del carácter c, consigo mismo, $n-1$ veces. En otras palabras, c^n representa la repetición del carácter c, n veces. Por lo tanto: $c^1 = c$, $c^2 = cc$, $c^3 = ccc$, etc.

Nota 3
Para un carácter, no existe la "potencia" 0.

Ejemplo 11
La cadena del Ejemplo 10 (aaaabbbbbbb) se puede escribir, más sintéticamente, así: a^4b^7.

Como se puede apreciar, esta simplificación también produce una mejor y más rápida lectura –y, por lo tanto, una mejor comprensión– de la cadena en cuestión.

* Ejercicio 4 *
Simplificando con el uso de la "potenciación", escriba la cadena que tiene 1300 aes seguidas de 846 bes seguidas de 257 aes.

1.2.4 CONCATENACIÓN DE DOS CADENAS

La operación de CONCATENACIÓN aplicada a cadenas (se escribe $S_1 \bullet S_2$ o, simplemente, S_1S_2) produce una nueva cadena formada por los caracteres de la primera cadena seguidos inmediatamente por los caracteres de la segunda cadena.

Ejemplo 12
Sean las cadenas $S_1 = aab$ y $S_2 = ba$; entonces, $S_1S_2 = aabba$.

Nota 4
La concatenación se escribe, normalmente, sin utilizar el operador correspondiente (•), como en el ejemplo anterior. Simplemente, se coloca una cadena a continuación de la otra.

Nota 5
La concatenación se extiende, fácilmente, a más de dos cadenas.

Ejemplo 13
$S_1 \bullet S_2 \bullet S_3 = S_1S_2S_3$.

* Ejercicio 5 *
Sean las cadenas $S_1 = aab$ y $S_2 = ba$. Obtenga la cadena $S_1S_2S_1S_2$.

➔ La concatenación NO ES CONMUTATIVA (excepto en casos muy especiales).

Ejemplo 14
La cadena aa concatenada con la cadena bb produce la cadena $aabb$, mientras que la cadena bb concatenada con la cadena aa produce la cadena $bbaa$. Estas dos cadenas son diferentes; en consecuencia, la concatenación no es siempre conmutativa.

A continuación, veamos algunos casos especiales en los que la concatenación sí es conmutativa.

CASO 1: Si ambas cadenas son idénticas, entonces la concatenación es conmutativa.

Ejemplo 15
Si la primera cadena es ab y la segunda cadena también es ab, entonces: $ab \bullet ab$ (en cualquier orden) es $abab$.

CASO 2: Cuando ambas cadenas están compuestas por una repetición del mismo carácter, entonces la concatenación es conmutativa.

Ejemplo 16
Sea $S_1 = aa$ y sea $S_2 = aaa$. Entonces: $S_1S_2 = a^2a^3 = a^{2+3} = a^5$ y $S_2S_1 = a^3a^2 = a^{3+2} = a^5$. En consecuencia, $S_1S_2 = S_2S_1 = a^5$.

CASO 3:
➔ La cadena vacía (ε) es la IDENTIDAD para la concatenación. Esto es: para cualquier cadena S, $S \bullet \varepsilon = \varepsilon \bullet S = S$.

1.2.5 POTENCIACIÓN DE UNA CADENA

El supraíndice utilizado en la sección 1.2.3 para representar la repetición de un carácter se extiende fácilmente a una cadena, de esta manera: si S es una cadena, entonces S^n (con $n \geq 1$ y entero)

representa la cadena que resulta de CONCATENAR la cadena S, consigo misma, n-1 veces; es decir: la cadena final resulta de repetir la cadena original n veces. Por lo tanto:

$$S^1 = S,\ S^2 = SS,\ S^3 = SSS,\ \text{etc.}$$

➔ Esta definición se completa con la siguiente afirmación: S^0 es **ε** (la cadena vacía), para cualquier cadena S.

Ejemplo 17
Sea S = ab; entonces: $S^3 = SSS = (ab)^3 = ababab$.

Ejemplo 18
Nótese la diferencia que existe entre la cadena $(ab)^3$ y la cadena ab^3: $(ab)^3$ representa la cadena ababab, mientras que ab^3 es la cadena abbb.

* Ejercicio 6 *
¿Por qué el supraíndice 0 no es aplicable a caracteres pero sí a cadenas?

* Ejercicio 7 *
Demuestre que $(abc)^0 = (1234)^0$.

➔ Un caso particular: $\varepsilon^n = \varepsilon$ para cualquier valor entero de $n \geq 0$.

* Ejercicio 8 *
Demuestre que las cadenas $(ab^3)^3$ y $((ab)^3)^3$ son diferentes.

1.3 LENGUAJES NATURALES Y LENGUAJES FORMALES

Se denomina LENGUAJE NATURAL a todo lenguaje hablado y/o escrito que es utilizado por los seres humanos para comunicarse.

Ejemplo 19
El español, el inglés y el chino son los tres lenguajes naturales empleados por más personas en el mundo.

Los Lenguajes Naturales tienen cuatro características fundamentales:

1º EVOLUCIONAN con el paso del tiempo, incorporando nuevos términos y nuevas reglas gramaticales para mejorar y actualizar la comunicación;

> *Ejemplo 20*
> Las palabras pánfilo y zopenco no son utilizadas actualmente, aunque figuran en los diccionarios.
>
> * Ejercicio 9 *
> Escriban sinónimos de las palabras del Ejemplo 20, que se utilicen actualmente.

2º Sus REGLAS GRAMATICALES surgen después del desarrollo del lenguaje, para poder explicar su estructura, es decir: su sintaxis;

3º El SIGNIFICADO (o sea, la semántica) de cada palabra, de cada oración y de cada frase de un Lenguaje Natural es, en general, más importante que su composición sintáctica.

Ejemplo 21
"Donya Innes ah vaniado a su perro con javom pra labar la roppa." Esta oración tiene muchos errores, pero seguro que todos nosotros comprendemos su significado. En consecuencia, comprobamos que, en los Lenguajes Naturales, la semántica es más importante que la sintaxis.

Ejemplo 22
En general, los mensajes de texto tienen muchos errores sintácticos. A pesar de ello, las personas que se comunican a través de ellos comprenden su semántica, es decir: su significado.

4º Los Lenguajes Naturales son intrínsecamente AMBIGUOS, por lo siguiente: en todo proceso de comunicación existen un EMISOR y un RECEPTOR. El emisor piensa lo que quiere transmitir, pero no siempre lo pensado es lo que realmente comunica. El receptor percibe lo comunicado por el emisor y, de lo percibido, hay una comprensión que no necesariamente coincide con lo comunicado por el emisor, y menos aun, con lo que éste pensó comunicar. En este proceso interviene mucho el Lenguaje Natural.

Ejemplo 23
"Hay una llama en la cima de la montaña". La palabra llama se puede referir a un animal o a la existencia de fuego. Su interpretación dependerá, fundamentalmente, de la forma en la que la persona lo diga (tono de la voz) y de sus gestos.

Por otro lado, un **LENGUAJE FORMAL** es un conjunto de cadenas formadas con caracteres de un alfabeto dado, y tiene dos características fundamentales:

1) las cadenas que constituyen un Lenguaje Formal no tienen una semántica asociada, solo tienen una sintaxis dada por la ubicación de los caracteres dentro de cada cadena;
2) un Lenguaje Formal nunca es ambiguo.

Ejemplo 24
El Lenguaje Formal {Argentina, Holanda, Brasil} no está formado por los nombres de tres países (semántica). Solo consiste de tres cadenas, cada una de las cuales tiene los caracteres que están escritos y ubicados tal como se muestra (sintaxis).

* Ejercicio 10 *
Escriba el alfabeto mínimo a partir del cual se construye el Lenguaje Formal del ejemplo anterior.

Contrariamente a lo que ocurre con los Lenguajes Naturales, los LENGUAJES FORMALES están definidos por reglas gramaticales PREESTABLECIDAS y se deben ajustar rigurosamente a ellas. En consecuencia, un Lenguaje Formal *nunca puede evolucionar*.

Ejemplo 25
Sea $\Sigma = \{a\}$, un alfabeto con un solo carácter. Los que siguen son algunos Lenguajes Formales que se pueden construir sobre este alfabeto: $L_1 = \{a\}$; $L_2 = \{aa, aaa\}$; $L_3 = \{\varepsilon, a, a^{18}\}$.

Ninguno de estos lenguajes puede "evolucionar", es decir: no se puede modificar mediante el agregado de nuevas cadenas o quitando cadenas ya existentes en el lenguaje. Si así se hiciera, obtendríamos otros lenguajes.

➔ Observe cómo, en el Ejemplo 25, la letra a es utilizada en dos contextos diferentes, representando dos entidades distintas. Por un lado, a es un carácter de un alfabeto, mientras que, por otro lado, a es una cadena de un Lenguaje Formal. En consecuencia, el *contexto* en el que se encuentra un carácter, debe determinar, sin ambigüedades, el significado único de ese símbolo, ya sea como miembro de un alfabeto o bien como componente de un Lenguaje Formal.

* Ejercicio 11*
Escriba un Lenguaje Formal con cuatro cadenas de longitud cinco sobre el alfabeto {c, p}.

1.3.1 PALABRA

Una *cadena* es vacía o bien está compuesta por una sucesión de uno o más caracteres que pertenecen a un alfabeto dado, como hemos visto ya en muchos ejemplos.

Además, una *cadena* puede ser un miembro de un Lenguaje Formal, pero también puede no serlo. Por ello, incorporamos una nueva definición: "si una *cadena* pertenece a un determinado Lenguaje Formal, decimos que la cadena es una PALABRA de ese lenguaje". La pertenencia de una cadena a un Lenguaje Formal le da la propiedad de ser PALABRA de ese lenguaje en particular.

Ejemplo 26
Refiriéndonos al Ejemplo 25, observamos que el lenguaje L_1 tiene una sola palabra, a, mientras que el lenguaje L_3 tiene tres palabras: ε, a, a^{18}. La cadena aa no es una palabra de ninguno de estos dos lenguajes, pero sí lo es del lenguaje L_2.

Ejemplo 27
Sea el siguiente Lenguaje Formal descripto por EXTENSIÓN: L = {101, 1001, 10001, 100001}. Utilizando el operador "supraíndice", este lenguaje puede ser descripto por COMPRENSIÓN, en forma más compacta, así: $L = \{10^n1 / 1 \leq n \leq 4\}$. Entonces, la cadena 1001 ("uno-cero-cero-uno") es una palabra del lenguaje L, mientras que 1100 ("uno-uno-cero-cero") es una cadena construida con caracteres del mismo alfabeto pero no es una palabra de L.

Nota 6
Un Lenguaje Formal puede ser descripto por extensión, por comprensión, mediante una frase en un Lenguaje Natural (castellano, en nuestro caso) o mediante otras formas especiales que veremos más adelante.

Ejemplo 28
Sea el lenguaje L del Ejemplo anterior. Este Lenguaje Formal puede ser descripto mediante una frase en castellano (Lenguaje Natural), de la siguiente manera: "L es un lenguaje de cuatro palabras, cada una de las cuales comienza con el carácter 1, termina con otro carácter 1, y en medio tiene una secuencia de uno a cuatro 0s".

Nota 7
Si al describir un Lenguaje Formal no se indica explícitamente sobre qué alfabeto está construido, se considera que el alfabeto está formado por los caracteres que forman las palabras del lenguaje. Por caso: en el Ejemplo 27, el lenguaje L está construido sobre el alfabeto {0, 1}.

➔ Si analizamos las tres formas de describir un Lenguaje Formal (por comprensión, por extensión o mediante una frase en Lenguaje Natural), seguramente consideraremos que son más simples las dos que figuran en el Ejemplo 27. Sin embargo, hay muchos casos en los que la dificultad para describir un Lenguaje Formal con "simbología matemática" es evidente. En estos casos, debemos utilizar una frase en Lenguaje Natural, *evitando ambigüedades*, u otras herramientas que veremos más adelante.

Ejemplo 29
El lenguaje $L = \{a^{2i}b^i / 0 \leq i \leq 2\}$ está formado por tres palabras: si $i = 0$, $a^{2*0}b^0 = a^0b^0$ (ausencia de aes y ausencia de bes) representa la palabra vacía (ε); si $i = 1$, la palabra es $a^{2*1}b^1 = aab$; y si $i = 2$, la palabra es $a^{2*2}b^2 = aaaabb$. En este caso, obviamente es más sencilla la descripción por extensión que por comprensión.

* Ejercicio 12 *
Describa por comprensión al siguiente lenguaje:
L = {ε, b, bb, bbb, bbbb, bbbbb, bbbbbb, bbbbbbb, bbbbbbbb}.

* Ejercicio 13 *
Describa mediante una frase al lenguaje del ejercicio anterior.

Ejemplo 30
Sea el Lenguaje Formal sobre el alfabeto {a, b, c} formado por todas las palabras de longitud 30 que comienzan con a, terminan con b y, en medio, tienen exactamente tres ces. Una palabra de este lenguaje es: $a^{20}cbbcacb^4$.

* Ejercicio 14 *
Intente describir por comprensión o por extensión el Lenguaje Formal del Ejemplo 30.

1.3.2 PROPIEDADES DE LAS PALABRAS

Dado que una PALABRA es una cadena que pertenece a un determinado Lenguaje Formal, todos los conceptos sobre *cadena* explicados anteriormente se aplican también a las palabras. Por lo tanto, hablaremos de: longitud de una palabra, palabra vacía, concatenación de dos o más palabras, potenciación de una palabra, con el mismo significado ya visto.

Ejemplo 31
Sea el lenguaje $L = \{(abc)^n / 0 \leq n \leq 3\} = \{\varepsilon, abc, abcabc, abcabcabc\}$. Entonces:
– La longitud de la palabra abc es $|abc| = 3$.
– La palabra vacía ε es un miembro de este lenguaje.
– La concatenación de las palabras abc y abcabc produce otra palabra de este lenguaje: abcabcabc. En cambio, la concatenación de la palabra abcabc consigo misma produce la cadena abcabcabcabc, que no es una palabra de este lenguaje.

– La potencia $(abc)^2$ es una palabra del lenguaje.

➔ La concatenación de dos palabras produce una *cadena* que no siempre es una *palabra* del lenguaje, como se observa en el ejemplo anterior y en el ejemplo que sigue.

Ejemplo 32
Sea el lenguaje $L = \{a^{2n+1} / 0 \leq n \leq 200\}$. Este Lenguaje Formal está formado por 201 palabras, cada una de las cuales tiene un número impar de letras a. Si $n = 0$, la palabra es a; ésta es la palabra de menor longitud de L. Si $n = 200$, la palabra es a^{401}; ésta es la palabra de mayor longitud de L.

Si se concatenan dos palabras cualesquiera de L, el resultado será una cadena con una cantidad par de letras a, ya que la suma de dos números impares produce un número par. En consecuencia, la concatenación de dos palabras cualesquiera de este lenguaje produce una cadena que nunca es una palabra de L.

* Ejercicio 15 *
Describa mediante una frase al lenguaje del Ejemplo 32.

* Ejercicio 16 *
Sea el lenguaje $L = \{a^{2n} / 0 \leq n \leq 200\}$. Escriba, por comprensión, el Lenguaje Formal que se obtiene realizando todas las concatenaciones de dos palabras cualesquiera del lenguaje dado.

1.3.3 CARDINALIDAD DE UN LENGUAJE FORMAL

La cardinalidad de un Lenguaje Formal es la cantidad de palabras que lo componen.

Ejemplo 33
$L = \{a, ab, aab\}$ es un lenguaje de cardinalidad 3 sobre el alfabeto $\{a, b\}$.

Ejemplo 34
El lenguaje $L = \{\varepsilon\}$ es un lenguaje muy especial, pues está formado solo por la palabra vacía. Su cardinalidad es 1, ya que contiene una palabra.

1.3.4 SUBLENGUAJES

Dado que un Lenguaje Formal es un conjunto, un SUBLENGUAJE es un subconjunto de un Lenguaje Formal dado.

Ejemplo 35
Sea $L_1 = \{a, ab, aab\}$. Entonces, $L_2 = \{ab, aab\}$ es un sublenguaje de L_1, mientras que $L_3 = \{\ \}$ es el sublenguaje vacío de L_1.

➔ El sublenguaje vacío, habitualmente representado con el símbolo ø, es un sublenguaje de cualquier Lenguaje Formal.

➔ No se debe confundir el sublenguaje vacío, que tiene cardinalidad 0, con el lenguaje que solo contiene la palabra vacía, que tiene cardinalidad 1.

1.4 LENGUAJES FORMALES INFINITOS

Todos los lenguajes ejemplificados hasta el momento han sido LENGUAJES FORMALES FINITOS, es decir: lenguajes con un número finito de palabras. Sin embargo, los Lenguajes Formales también pueden ser INFINITOS, lo que significa que estos lenguajes pueden tener una cantidad infinita de palabras, aunque cada una de ellas debe ser de longitud finita; no existen las palabras de longitud infinita.

Nota 8
Las propiedades de las palabras que han sido ejemplificadas anteriormente se aplican también a los lenguajes infinitos.

Ejemplo 36
$L = \{a^n / n \geq 1\}$ es un Lenguaje Formal infinito ya que no existe un límite superior para el supraíndice n. Cada palabra de este lenguaje está formada por una secuencia de una o más aes. Por ello, la concatenación de dos palabras cualesquiera de este lenguaje producirá siempre otra palabra del lenguaje L. Por esta propiedad, se dice que este lenguaje L tiene una particularidad especial: es cerrado bajo la concatenación.

Ejemplo 37
El lenguaje $L = \{ab^n / n \geq 0\}$ es otro Lenguaje Formal infinito. Este lenguaje está formado por la palabra a y todas aquellas palabras que comienzan con una a, seguida de una secuencia de una o más bes. Este lenguaje no es cerrado bajo la concatenación ya que, por caso: si consideramos sus dos palabras de menor longitud – a y ab – y las concatenamos, obtenemos la cadena aab, que no es una palabra de este lenguaje.

* Ejercicio 17 *
Sea el lenguaje $L = \{ab^na / n \geq 1\}$.
a) Escriba las tres palabras de menor longitud.
b) Describa este lenguaje mediante una frase en castellano.

1.4.1 LENGUAJE UNIVERSAL SOBRE UN ALFABETO

Dado un alfabeto Σ, el LENGUAJE UNIVERSAL sobre este alfabeto es un Lenguaje Formal infinito que contiene todas las palabras que se pueden formar con los caracteres del alfabeto Σ, más la palabra vacía. Se lo representa con la notación Σ*, que se lee "sigma clausura" o "sigma estrella".

➔ Importante: acabamos de introducir un nuevo operador que representamos con el supraíndice *. Este es un operador unario (opera sobre un solo operando), como lo es la potenciación en matemáticas. Se lo denomina "clausura de Kleene" (se pronuncia *klaini*) o "estrella de Kleene", y será muy utilizado en los próximos capítulos.

Una propiedad fundamental del Lenguaje Universal es que es cerrado bajo concatenación.

Ejemplo 38
Si Σ = {a, b}, entonces el Lenguaje Universal para este alfabeto está formado por la palabra vacía, todas las palabras que solo tienen aes, todas las palabras que solo tienen bes, y todas las palabras formadas por aes y bes concatenadas en cualquier orden:

$$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, ..., aabaabbbab, \ldots\},$$

Afirmación: La concatenación de palabras de Σ^* siempre produce una palabra de este Lenguaje Universal sobre el alfabeto {a, b}.

* Ejercicio 18 *
¿Puede encontrar un caso en que la afirmación anterior sea falsa? Justifique su respuesta.

➔ Cualquier lenguaje L sobre el alfabeto Σ es un sublenguaje de Σ^*. Por lo tanto, existen infinitos Lenguajes Formales sobre un alfabeto dado.

1.4.2 LENGUAJES FORMALES INFINITOS MÁS COMPLEJOS

Todos los ejemplos vistos hasta ahora se refieren al tipo de Lenguaje Formal llamado LENGUAJE REGULAR. Pero existen otros tipos de Lenguajes Formales infinitos que ejemplificamos a continuación.

Ejemplo 39
Los que siguen son ejemplos de tres diferentes tipos de Lenguajes Formales infinitos, ninguno de los cuales es un Lenguaje Regular:

$L_1 = \{a^n b^n / n \geq 1\}$. Las tres palabras de menor longitud de este lenguaje son: ab, aabb y aaabbb.

$L_2 = \{a^n b^n c^n / n \geq 1\}$. Las tres palabras de menor longitud de este lenguaje son: abc, aabbcc y aaabbbccc.

$L_3 = \{a^n / n$ es una potencia positiva de 2$\}$. Las tres palabras de menor longitud de este lenguaje son: aa, aaaa y aaaaaaaa.

En el próximo capítulo veremos porqué hay diferentes tipos de Lenguajes Formales y porqué los lenguajes del Ejemplo 39 no son LENGUAJES REGULARES.

Ciertos Lenguajes Formales están relacionados con la sintaxis de los Lenguajes de Programación.
Un Lenguaje de Programación (LP) tiene palabras reservadas, nombres creados por el programador, constantes enteras y reales, caracteres de puntuación, operadores aritméticos, operadores lógicos, declaraciones, expresiones, sentencias, etc.
Todos estos elementos constituyen diferentes Lenguajes Formales, algunos infinitos y otros finitos.

* Ejercicio 19 *
Para un LP que conozca, escriba dos ejemplos de cada una de las entidades (palabras reservadas, expresiones, etc.) mencionadas en el recuadro anterior. Indique si corresponden, según su criterio, a un Lenguaje Formal finito o a un Lenguaje Formal infinito. Explique su decisión.

1.5 IMPLEMENTACIÓN EN C

Investigue y construya, en LENGUAJE C, la función que realiza cada operación solicitada:

* Ejercicio 20 *
(a) Calcula la longitud de una cadena;
(b) Determina si una cadena dada es vacía.
(c) Concatena dos cadenas.

* Ejercicio 21 *
Construya un programa de testeo para cada función del ejercicio anterior.

2 GRAMÁTICAS FORMALES Y JERARQUÍA DE CHOMSKY

Un LENGUAJE FORMAL, sea finito o infinito, es un conjunto de palabras con una connotación sintáctica, sin semántica. Existen ciertas estructuras que tienen la habilidad de GENERAR todas las palabras que forman un Lenguaje Formal. Estas estructuras se denominan GRAMÁTICAS FORMALES.

2.1 GRAMÁTICA FORMAL

Una Gramática Formal es, básicamente, un conjunto de PRODUCCIONES, es decir: *reglas de reescritura* que se aplican para obtener las palabras del Lenguaje Formal que la Gramática Formal en cuestión genera. Una Gramática Formal genera un determinado Lenguaje Formal, único.

Ejemplo 1
Con una simbología simple, que será aclarada posteriormente, podemos representar las producciones de una Gramática Formal que genera el lenguaje L = {ab, ac}. Estas producciones se pueden escribir así: S –> ab y S –> ac.

Para construir las PRODUCCIONES de una Gramática Formal se requieren tres tipos de símbolos:
- los símbolos "productores", como es S en el Ejemplo 1;
- los símbolos que forman las palabras del lenguaje generado, como son los caracteres a, b y c en el Ejemplo 1; y
- ciertos símbolos especiales que llamaremos *metasímbolos*, como –> en el Ejemplo 1.

Ejemplo 2
Sea el lenguaje L = {a}, formado por una sola palabra. Este lenguaje es generado por una Gramática Formal con una única producción: S –> a (se lee "S produce a" o "S es reemplazada por a"). La "flecha" será llamada OPERADOR DE PRODUCCIÓN.

Si bien en estos dos primeros ejemplos se observa que cada producción genera una palabra del lenguaje, esto no siempre es así.

Ejemplo 3
El Lenguaje Formal L = {aa, ab} puede ser generado por una Gramática Formal con dos producciones: S –> aa y S –> ab. Pero este Lenguaje Formal también puede ser generado por la Gramática Formal con producciones: S –> aT, T –> a y T –> b.

Si utilizamos el segundo conjunto de producciones, ¿cómo obtenemos las dos palabras del lenguaje? Comenzamos con la producción S –> aT ("S produce a seguido de T" o "S produce a concatenado con T") para obtener el carácter a con el que comienzan ambas palabras de este Lenguaje Formal. Luego, si aplicamos la producción T –> a obtenemos la palabra aa y si aplicamos T –> b obtenemos la palabra ab.

➔ Toda producción está formada por tres partes: el lado izquierdo, el lado derecho, y el operador de producción, que significa que el lado izquierdo de la producción "produce" –o "es reemplazado por" o "equivale a"– el lado derecho de la misma.

Ejemplo 4
En la producción S −> aT, el *lado izquierdo* está constituido por el símbolo S, mientras que su *lado derecho* está formado por la concatenación del carácter a con el símbolo T. El operador de producción indica que el símbolo S es reemplazado por la secuencia aT.

➔ Una Gramática Formal puede tener, entre sus producciones, una muy particular que se denomina PRODUCCIÓN-ÉPSILON y que se escribe, por ejemplo: S −> ε. Esta notación significa que S es reemplazada por "nada" o que genera la palabra vacía. Veamos los dos casos.

Ejemplo 5
El Lenguaje Formal L = {aa, ε}, que contiene a la palabra vacía, puede ser generado por una Gramática Formal con dos producciones: S −> aa y S −> ε. En este caso, la producción-épsilon genera la palabra vacía.

Ejemplo 6
Veamos, ahora, el segundo caso. El Lenguaje Formal L = {aa, aab} puede ser generado por una Gramática Formal con las siguientes producciones: S −> aaT, T −> ε y T −> b.

* Ejercicio 1 *
a) Describa el orden en que se aplican estas producciones para generar las palabras del lenguaje del Ejemplo 6.
b) ¿La producción T −> ε genera la palabra vacía del lenguaje?

2.1.1 DEFINICIÓN FORMAL DE UNA GRAMÁTICA FORMAL

Toda Gramática Formal es una 4-upla (V_N, V_T, P, S), donde:

– V_N es el vocabulario de noterminales; es un conjunto finito de "productores".

– V_T es el vocabulario de terminales, caracteres del alfabeto sobre el cual se construyen las palabras del Lenguaje Formal que es generado por la gramática descripta; también es un conjunto finito.

– P es el conjunto finito de producciones, y

– S $\in$ V_N es un noterminal especial llamado axioma. Es el noterminal a partir del cual siempre deben comenzar a aplicarse las producciones que generan las palabras de un determinado Lenguaje Formal.

➔ Cuando decimos "una Gramática Formal genera un Lenguaje Formal" significa que esa gramática es capaz de generar todas las palabras del Lenguaje Formal, pero que, a su vez, no genera cadenas que están fuera del lenguaje.

Ejemplo 7
Retomamos el Ejemplo 3, que describe las producciones de una gramática que genera el lenguaje {aa, ab}. La definición formal de esta gramática es la 4-upla:

G = ({S, T}, {a, b}, {S −> aT, T −> a, T −> b}, S).

Dado que el noterminal S es el axioma, la generación de cualquier palabra del lenguaje mencionado comienza con una producción que tenga al noterminal S en su lado izquierdo; en este caso, hay una única producción con esta característica: S −> aT. Esta única producción indica que el símbolo inicial S es reemplazado, obligatoriamente, por la secuencia aT, por lo que toda palabra generada

por esta gramática debe comenzar con el carácter a. Todavía no se ha obtenido una palabra del lenguaje, porque T es un noterminal y sabemos que una palabra debe estar formada solo por caracteres o debe ser la palabra vacía. En consecuencia, debe haber una o más producciones que tengan al noterminal T en su lado izquierdo, y se debe reemplazar a T por su lado derecho. En este caso, el noterminal T tiene dos producciones que representan dos opciones: la producción T −> a significa que el noterminal T es reemplazado por el carácter a, mientras que la producción T −> b representa que el noterminal T es reemplazado por el carácter b. Estas dos producciones para el noterminal T generan dos procesos diferentes:

(1) la aplicación de las producciones S −> aT y T −> a genera la palabra aa;
(2) la aplicación de las producciones S −> aT y T −> b genera la palabra ab.

Ejemplo 8
Realizamos una pequeña variante sobre el Ejemplo 7. Supongamos una Gramática Formal con la siguiente definición:

G = ({S, T}, {a}, {S −> aT, T −> a, T −> ε}, S).

* Ejercicio 2 *
¿Qué Lenguaje Formal genera la gramática del Ejemplo 8? Justifique.

Nota 1
Al describir Gramáticas Formales para aplicaciones teóricas, como las que estamos viendo en este capítulo, es muy común utilizar las siguientes convenciones:
– se denomina S al axioma,
– todo noterminal es representado mediante una letra mayúscula,
– el vocabulario de terminales está formado por letras minúsculas y dígitos.

Nota 2
Agregamos otro conjunto que, si bien no figura en la definición formal de una Gramática Formal, será de gran utilidad en aplicaciones que veremos más adelante. Nos referimos al conjunto de METASÍMBOLOS, que no son terminales ni son noterminales, pero que son símbolos que ayudan a representar las producciones de una Gramática Formal. Uno de ellos ya lo hemos visto: −>, el operador de producción.

Ejemplo 9
Sea la definición formal del Ejemplo 7: G = ({S, T}, {a, b}, {S −> aT, T −> a, T −> b}, S). Es habitual describir esta Gramática Formal mediante la escritura de sus producciones únicamente. De ser así, por convención se entiende que el noterminal que aparece en el lado izquierdo de la primera producción es el axioma y, normalmente, se lo llama S. Por otro lado, si para un noterminal dado existe más de una producción, se utiliza el *metasímbolo* "barra vertical" para indicar "ó", es decir, describir dos o más producciones con el mismo noterminal en su lado izquierdo. Entonces, las producciones de esta gramática se escriben, generalmente, de la siguiente manera:

S −> aT
T −> a | b

Esta Gramática Formal está formada por tres producciones. Los metasímbolos −> y | colaboran en la escritura de estas producciones en forma más compacta.

* Ejercicio 3 *
Sea la Gramática Formal con producciones:

S −> aT | bQ
T −> a | b
Q −> a | ε

a) ¿La gramática con estas seis producciones genera la cadena bab? Justifique.
b) ¿Cuál es el Lenguaje Formal generado por esta gramática? Justifique.

2.2 LA JERARQUÍA DE CHOMSKY

En 1956 y 1959, el lingüista norteamericano Noam Chomsky publicó dos trabajos sobre los Lenguajes Naturales que, aplicados al área de los Lenguajes Formales, produjeron lo que se conoce como "Jerarquía de Chomsky".

Esta Jerarquía de Chomsky establece una clasificación de cuatro tipos de Gramática Formales que, a su vez, generan cuatro tipos diferentes de Lenguajes Formales.

Las Gramáticas Formales se clasifican según las restricciones que se imponen a sus producciones, y la Jerarquía de Chomsky establece estos cuatro niveles:

- Gramáticas Regulares o Gramáticas Tipo 3
- Gramáticas Independientes del Contexto o Gramáticas Tipo 2
- Gramáticas Sensibles al Contexto o Gramáticas Tipo 1
- Gramáticas Irrestrictas o Gramáticas Tipo 0

Por otro lado, estos cuatro tipos de gramáticas generan los siguientes tipos de Lenguajes Formales:

Gramática Formal	**Lenguaje Formal**
Gramática Regular	Lenguaje Regular
Gramática Independiente del Contexto	Lenguaje Independiente del Contexto
Gramática Sensible al Contexto	Lenguaje Sensible al Contexto
Gramática Irrestricta	Lenguaje Irrestricto

A continuación, analizamos las diferencias que existen entre los cuatro tipos de Gramáticas Formales.

2.2.1 GRAMÁTICA REGULAR (GR)

Sus producciones tienen las siguientes restricciones:
- el lado izquierdo debe tener un solo noterminal,
- el lado derecho debe estar formado por:
  a) un solo terminal, o
  b) un terminal seguido por un noterminal,
  c) o "épsilon".

Ejemplo 10
Sea la gramática G = ({S, X}, {a, b}, {S –> aX, X –> b}, S).
Las producciones de esta gramática cumplen con las restricciones mencionadas arriba. Analícelas. Por lo tanto, esta Gramática Formal es una GR.

También es válida una GR en la que se invierte el orden en el lado derecho de aquellas producciones que tienen dos símbolos. Por lo tanto, una segunda definición para las GRs sería:

Una Gramática Formal es una GR si sus producciones tienen las siguientes restricciones:
- el lado izquierdo debe tener un solo noterminal,
- el lado derecho debe estar formado por:
  a) un solo terminal, o
  b) un noterminal seguido de un terminal, o
  c) "épsilon".

Ejemplo 11
La Gramática Formal ({S, X}, {a, b}, {S –> Xa, X –> b}, S) es una GR porque cumple con las restricciones de esta segunda definición.

En general: sean v y v' noterminales y sea t un terminal. Entonces, las producciones de una GR pueden tener estos formatos:

1) v –> t, 2) v –> tv' y 3) v –> ε o 1) v –> t, 2) v –> v't y 3) v –> ε

Sin embargo, debemos tener cuidado con "la mezcla" de ambas definiciones porque la Gramática Formal resultante no será una GR.

Ejemplo 12
Sea G = ({S, X}, {a, b}, {S –> Xa, S –> bX, X –> b}, S). Esta Gramática Formal no es una GR porque el noterminal S produce "noterminal terminal" y también produce "terminal noterminal".

* Ejercicio 4 *
a) Escriba las producciones de una GR que genere el Lenguaje Regular $L = \{a^n b / 1 \leq n \leq 3\}$.
b) Escriba la definición formal de la GR desarrollada en el punto anterior.

* Ejercicio 5 *
a) Escriba las producciones de una GR que genere el Lenguaje Regular $L = \{a^n b^n / 0 \leq n \leq 2\}$.
b) Escriba la definición formal de la GR desarrollada en el punto anterior.

2.2.1.1 GRAMÁTICA REGULAR QUE GENERA UN LENGUAJE REGULAR INFINITO

Las GRs que hemos visto anteriormente generan Lenguajes Regulares (LRs) finitos. Pero las GRs, aun teniendo un número finito de producciones, pueden generar LRs infinitos. Para ello existen las llamadas PRODUCCIONES RECURSIVAS: producciones en las que el noterminal que figura en el lado izquierdo también figura en el lado derecho.

Ejemplo 13
La producción T –> aT es una producción recursiva.

Ejemplo 14
Sea el Lenguaje Formal infinito $L = \{a^n b / n \geq 1\}$. Construyamos las producciones de una GR que genere este lenguaje:

1 S –> aS
2 S –> aT
3 T –> b

Habitualmente, estas producciones se escriben en forma más compacta de la siguiente manera:

S –> aS | aT
T –> b

Análisis: La producción (1) genera tantas aes como se necesiten por ser una producción recursiva, aunque no es obligatorio que esta producción siempre sea elegida ya que el axioma S tiene dos producciones. La producción (2) termina la recursividad y produce una última a, o bien produce la única a si la producción (1) no es utilizada. La producción (3) finaliza la generación de una palabra, reemplazando el noterminal T que está en el lado derecho de la producción (2) por el carácter b con el que debe terminar toda palabra del LR que genera esta gramática.

* Ejercicio 6 *
Indique cuál es la mínima palabra del LR del Ejemplo 14 y muestre cómo la genera.

* Ejercicio 7 *
Escriba la sucesión de producciones que aplicaría para generar la palabra aaab del LR definido en el Ejemplo 14.

2.2.1.2 GRAMÁTICA QUASI–REGULAR (GQR)

Estas gramáticas están muy vinculadas a la sintaxis de los Lenguajes de Programación, como veremos en el próximo capítulo. Son gramáticas similares a una GR, pero donde un conjunto de terminales es reemplazado por un noterminal en una o varias producciones. Una GQR abrevia la escritura de una GR y siempre puede ser re-escrita como una GR.

Ejemplo 15
Sea la gramática con las siguientes producciones:

S –> N | NS
N –> a | b | c

Esta gramática es una GQR, porque las dos producciones del noterminal S no corresponden a la definición de una GR pero sí responden a la definición dada en el párrafo anterior. El noterminal N genera los terminales a, b y c; por lo tanto, N representa al conjunto {a, b, c}.

Ejemplo 16
Re-escribamos las producciones de la GQR del ejemplo anterior como producciones de una GR. Para ello, debemos reemplazar al noterminal N por cada uno de los terminales que representa. Resultan, entonces, las siguientes producciones:

S –> a | b | c | aS | bS | cS

Como se observa, la GQR tiene cinco producciones, mientras que la GR equivalente tiene seis producciones.

➔ Dos Gramáticas Formales son EQUIVALENTES si generan el MISMO Lenguaje Formal.

Ejemplo 17
Las gramáticas de los ejemplos 15 y 16 son equivalentes.

* Ejercicio 8 *
a) Escriba una GR que genere cualquier secuencia de uno o más dígitos decimales.
b) Escriba una GQR que genere cualquier secuencia de uno o más dígitos decimales.
c) Compare la cantidad de producciones de ambas gramáticas.

* Ejercicio 9 *
Escriba las producciones de una GQR que genere un LR infinito cuyas palabras son secuencias de tres o más dígitos octales (en base 8).

* Ejercicio 10 *
Transforme las producciones de la GQR obtenida en el Ejercicio 9 en producciones de una GR equivalente.

2.2.2 GRAMÁTICA INDEPENDIENTE DEL CONTEXTO (GIC)

A diferencia de las GRs, las GICs no tienen restricciones con respecto a la forma del lado derecho de sus producciones, aunque sí se requiere que el lado izquierdo de cada producción siga siendo un único noterminal.

> El origen de la formalización de las GICs se encuentra en Chomsky [1956]: "Three models for the description of language". [*Hopcroft y Ullman, "Introduction to Automata Theory, Languages and Computation", 1979, Addison-Wesley*]

Ejemplo 18
Supongamos que a una GR le agregamos la producción S –> ba. Esta nueva producción posee dos terminales en su lado derecho y, como ya hemos visto, este no es un formato válido para una GR. La nueva Gramática Formal es una GIC.

En general: sean v y v' noterminales y sea t un terminal. Entonces las producciones de una GIC corresponden a este formato general:

v –> (v' + t)*, donde v y v' pueden representar el mismo noterminal

La expresión (v' + t)* (recuerden el operador "estrella" del capítulo anterior) representa ε y cualquier secuencia de noterminales y/o terminales.

Ejemplo 19
Teniendo en cuenta la convención mencionada antes (letra mayúscula para noterminales, y letras minúsculas o dígitos para terminales), he aquí una serie de producciones correctas para las GICs:

A −> ε	producción-épsilon
B −> a	lado derecho con un terminal
C −> ab2c5	lado derecho con cinco terminales
D −> ZXZ	lado derecho con tres noterminales
E −> aEZbRgh	lado derecho con cuatro terminales y tres noterminales

* Ejercicio 11 *
a) ¿Una GR es siempre una GIC? Justifique.
b) ¿Una GIC es siempre una GR? Justifique.

2.2.2.1 GIC QUE GENERA UN LENGUAJE INFINITO

Las GICs pueden generar Lenguajes Independientes del Contexto (LICs) infinitos utilizando PRODUCCIONES RECURSIVAS.

Ejemplo 20
La GIC con producciones:

S −> aSb | a

genera un LIC infinito; la primera producción es recursiva y la segunda producción debe utilizarse para terminar la recursividad.

* Ejercicio 12 *
a) ¿Cuál es la mínima palabra generada por la GIC del ejemplo anterior? Justifique.
b) ¿Cuál es la palabra que le sigue en longitud? Justifique.

* Ejercicio 13 *
Describa, por comprensión, el LIC generado por la GIC del Ejemplo 20.

* Ejercicio 14 *
Sea el lenguaje $L_9 = \{a^n b^{n+1} / n \geq 0\}$. Escriba las producciones de una GIC que genere L_9.

* Ejercicio 15 *
Sea el lenguaje $L_{10} = \{a^{2n} b^{n+1} a^r / n \geq 1, r \geq 0\}$. Escriba la definición formal de una GIC que genere el lenguaje L_{10}.

➔ Por las restricciones establecidas sobre los dos tipos de Gramáticas Formales vistas hasta ahora, la GR y la GIC, es fácil notar que toda GR también es una GIC; la inversa no es cierta.

* Ejercicio 16 *
Demuestre que una GQR es un caso particular de una GIC.

La frase "independiente del contexto" refleja que, como el lado izquierdo de cada producción solo puede contener un único noterminal, toda producción de una GIC puede aplicarse sin importar el contexto donde se encuentre dicho noterminal.

Estas GICs son de gran utilidad en la representación de la sintaxis
de los Lenguajes de Programación; ampliaremos más adelante.

2.2.3 GRAMÁTICA IRRESTRICTA

Estas son las Gramáticas Formales más amplias. Sus producciones tienen la forma general: $\alpha \rightarrow \beta$, donde tanto α como β pueden ser secuencias de noterminales y/o terminales, con $\alpha \neq \varepsilon$.

Ejemplo 21 [de Hopcroft & Ullman, 1979]
La que sigue es una Gramática Irrestricta que genera el lenguaje

$$\{a^i / i \text{ es una potencia positiva de 2}\}:$$

$$G = (\{S, A, B, C, D, E\}, \{a\}, \{S \rightarrow ACaB, Ca \rightarrow aaC, CB \rightarrow DB, CB \rightarrow E,$$
$$aD \rightarrow Da, AD \rightarrow AC, aE \rightarrow Ea, AE \rightarrow \varepsilon\}, S)$$

* Ejercicio 17 *
Compruebe que la gramática escrita en el Ejemplo 21 puede generar la palabra aaaa. Indique, paso a paso, cada producción que aplica y explique porqué utiliza la producción elegida.

2.2.4 GRAMÁTICA SENSIBLE AL CONTEXTO (GSC)

Una GSC es una Gramática Irrestricta, con la siguiente restricción en las longitudes: $|\alpha| \leq |\beta|$.

Ejemplo 22
La Gramática Formal del Ejemplo 21 no es una GSC porque tiene dos producciones que violan la restricción impuesta sobre las longitudes: $CB \rightarrow E$ y $AE \rightarrow \varepsilon$.

Cada tipo de Gramática Formal genera un tipo de Lenguaje Formal cuyo nombre deriva del nombre de la correspondiente gramática, como ya hemos visto anteriormente. En este libro nos interesarán, fundamentalmente: las GICs, las GQRs, los Lenguajes Regulares (LRs) y los Lenguajes Independientes del Contexto (LICs).

2.3 EL PROCESO DE DERIVACIÓN

La derivación es el proceso que permite obtener cada una de las palabras de un Lenguaje Formal a partir del axioma de una Gramática Formal que lo genera, y aplicando, sucesivamente, las producciones convenientes de esa gramática. El mismo proceso también permite descubrir si una cadena dada no es una palabra del lenguaje.

Existen diferentes formas de representar una derivación:

1) en forma horizontal, utilizando el símbolo **=>** para relacionar un paso de la derivación con el paso siguiente;

2) en forma vertical, reemplazando, en cada paso, un noterminal por su lado derecho para producir una nueva línea de esta derivación;

3) en forma de árbol, donde el axioma de la gramática es la raíz del árbol.

En esta sección solo utilizaremos la derivación vertical, aplicándola a GQRs y a GICs.

Ampliamos, entonces, lo indicado para la DERIVACIÓN VERTICAL. Esta derivación producirá una tabla con una sola columna y, eventualmente, una segunda columna con comentarios. En la primera línea estará siempre el axioma de la gramática utilizada. Las líneas sucesivas se obtendrán reemplazando un solo noterminal, de lo logrado hasta el momento, por el lado derecho de la producción más conveniente.

Ejemplo 23
Sea la GIC con producciones S −> aSb y S −> ab. Esta GIC genera el lenguaje $\{a^n b^n / n \geq 1\}$. Una de las palabras de este LIC es aaabbb. Verifiquemos esta situación mediante una derivación vertical:

Derivación Vertical	**Comentario**
S	aplicamos la primera producción y obtenemos:
aSb	aplicamos, nuevamente, la primera producción y obtenemos:
aaSbb	aplicamos la segunda producción y obtenemos:
aaabbb	

Ejemplo 24
En cambio, la cadena aaabb no es una palabra del LIC generado en el Ejemplo 23 y, por lo tanto, no la podremos derivar. Comprobemos esta situación:

Derivación Vertical	**Comentario**
S	aplicamos la primera producción y obtenemos:
aSb	aplicamos, nuevamente, la primera producción y obtenemos:
aaSbb	no tenemos forma de obtener la tercera a sola, ¿por qué?
??	

Por lo tanto, no podemos derivar aaabb y, en consecuencia, ésta no es una palabra del LIC generado por la GIC dada.

* Ejercicio 18 *
(a) Dada la GIC del Ejemplo 23, determine, aplicando una derivación vertical, si la cadena aaabbbb es una palabra del LIC generado por esta GIC.
(b) Investigue y dibuje la misma derivación, pero en forma de árbol.

➔ Antes de proseguir con este tema, introducimos una definición que facilitará la descripción de lo que sigue. Denominaremos CADENA DE DERIVACIÓN a la cadena de terminales y noterminales que se forma en cada paso de la derivación vertical, antes de obtener la palabra (si existe).

Ejemplo 25
En la derivaciones de los ejemplo 23 y 24, las dos cadenas de derivación son: aSb y aaSbb.

En el Ejemplo 23, la GIC tiene un solo noterminal en el lado derecho de una producción. En el próximo capítulo veremos gramáticas que tienen más de un noterminal en el lado derecho de una o más producciones. Por ello, definiremos dos tipos de derivaciones verticales:

➔ DERIVACIÓN VERTICAL A IZQUIERDA: en cada paso de la derivación se reemplaza el noterminal que se encuentra primero (de izquierda a derecha) en la cadena de derivación.

➔ DERIVACIÓN VERTICAL A DERECHA: en cada paso de la derivación se reemplaza el noterminal que se encuentra primero (de derecha a izquierda) en la cadena de derivación.

Ejemplo 26
Sea la GIC con producciones:

1 S –> ST
2 S –> ab
3 T –> aaT
4 T –> b

Una derivación vertical a izquierda produciría la siguiente tabla para obtener la palabra abaabaab:

Derivación Vertical a Izquierda	**Comentario**
S	Se aplica la producción (1)
ST	Se aplica la producción (1)
STT	Se aplica la producción (2)
abTT	Se aplica la producción (3)
abaaTT	Se aplica la producción (4)
abaabT	Se aplica la producción (3)
abaabaaT	Se aplica la producción (4)
abaabaab	

Una derivación vertical a derecha produciría la siguiente tabla para obtener la misma palabra:

Derivación Vertical a Derecha	**Comentario**
S	Se aplica la producción (1)
ST	Se aplica la producción (3)
SaaT	Se aplica la producción (4)
Saab	Se aplica la producción (1)
STaab	Se aplica la producción (3)
SaaTaab	Se aplica la producción (4)
Saabaab	Se aplica la producción (2)
abaabaab	

➔ Si una palabra se puede obtener mediante una derivación vertical a izquierda, entonces también se puede obtener mediante una derivación vertical a derecha.

Si una cadena de derivación tiene más de un noterminal, la próxima cadena de derivación se obtiene reemplazando un ÚNICO noterminal en la cadena de derivación anterior, tal como se observa en ambas tablas del Ejemplo 26. Nunca se pueden reemplazar dos o más noterminales en un solo paso.

* Ejercicio 19 *
Sea el LIC infinito $L = \{a^n bc^n b / n \geq 1\}$. Escriba la Definición Formal de una GIC que genere este Lenguaje Formal.

* Ejercicio 20 *
Dada la GIC construida en el ejercicio anterior, utilice derivación vertical a izquierda para determinar si las siguientes cadenas son o no palabras del LIC generado:

a) aaabcccb
b) aabbccb
c) aaabcccbb
d) aaccb

2.4 INTRODUCCIÓN A LAS GQRs, LAS GICs Y LOS LENGUAJES DE PROGRAMACIÓN

Tanto las GQRs como las GICs son de gran utilidad en la representación de la sintaxis de los Lenguajes de Programación. Si bien este es un tema que desarrollaremos en el próximo capítulo, he aquí unos ejemplos para introducirlo:

Ejemplo 27
Supongamos que un Lenguaje de Programación tiene nombres de variables que deben comenzar con una letra minúscula entre a y d, que puede estar seguida por letras en este mismo intervalo y por dígitos entre 2 y 6. El lenguaje de estos nombres es un LR infinito que puede ser generado por una GQR con estas producciones:

S –> L | SL | SD
L –> a | b | c | d
D –> 2 | 3 | 4 | 5 | 6

* Ejercicio 21 *
Escriba una GR equivalente a la GQR del ejemplo anterior. ¿Cuál es más sencilla para ser leída? Justifique su respuesta.

* Ejercicio 22 *
Dada la GQR del Ejemplo 27, determine si las siguientes cadenas son nombres válidos:

ab23
2a3b

Verifíquelo aplicando: a) derivación a izquierda y b) derivación a derecha.

Ejemplo 28

Supongamos un Lenguaje de Programación en el que sus expresiones aritméticas están formadas por los números enteros 2 y 6, el operador de suma y siempre terminan con un "punto y coma". Algunas expresiones aritméticas de este Lenguaje de Programación son:

6;
2+2+6;

Vamos a construir una GIC que genere la totalidad de estas expresiones aritméticas. Para ello, debemos definir el vocabulario de noterminales, el vocabulario de terminales, el axioma y el conjunto de producciones. Supongamos que el vocabulario de noterminales está formado por: S (el axioma), E (de expresión) y T (de término). Los terminales son 2, 6, + y ; (punto y coma). Las producciones de la GIC que genera el lenguaje de estas expresiones aritméticas son:

S −> E;
E −> T | E+T
T −> 2 | 6

* Ejercicio 23 *

a) Determine, aplicando derivación a izquierda, si 6; es una expresión correcta.
b) Determine, aplicando derivación a izquierda, si 6+6+6+; es una expresión correcta.
c) Determine, aplicando derivación a derecha, si 6+6+6+; es una expresión correcta.
d) Determine, aplicando derivación a izquierda, si 6+6+6+2; es una expresión correcta.