

# Tema 1: Introducción a la Computación

Serafín Moral

Universidad de Granada

Septiembre, 2016

## Tema 1: Introducción a la Computación

- Breve introducción histórica a la computación
- Definiciones: palabras y lenguajes
- Operaciones con palabras y lenguajes
- Gramáticas
- Jerarquía de Chomsky

Vamos a tratar de responder varias preguntas:

- ¿Qué puede ser resuelto de forma automática?
- ¿Qué puede ser resuelto de forma eficiente?
- ¿Qué estructuras son comunes en la computación con palabras y símbolos y cómo se pueden procesar en un ordenador?

Consecuencia: *Estudio de Modelos de Computación*

- *Russell, Hilbert y Boole*: Los precursores
- *Turing y Church*: Los primeros años (30 y 40)
- *FORTRAN, COBOL, LISP*: Los primeros lenguajes
- *Rabin, Scott, Chomsky*: Autómatas y Lenguajes Formales (los 50).
- *Hartmanis, Lewis, Cook*: Complejidad Algorítmica (los 60)
- Complejidad (circuitos, tiempo-espacio), semántica y estructuras de datos (los 70)
- Complejidad paralela, computación distribuida y criptografía (los 80 y 90)

# El problema de la parada

¿Existe un programa que lea un programa y unos datos y nos diga si ese programa termina o cicla indefinidamente?

No existe, ya que si existiera (programa  $\text{Stops}(P, x)$ ) podríamos construir el algoritmo  $\text{Turing}(P)$  con entrada  $P$ .

```
L           If Stops(P,P) GOTO L
```

¿Cual es el resultado de  $\text{Turing}(\text{Turing})$ ?

Un **alfabeto** es un conjunto finito  $A$ . Sus elementos se llamarán **símbolos** o **letras**.

**Notación** - Alfabetos:  $A, B, C, \dots$

Símbolos:  $a, b, c, \dots$  o números.

## Ejemplos

- $A = \{0, 1\}$
- $B = \{ \langle 0, 0 \rangle, \langle 0, 1 \rangle, \langle 1, 0 \rangle, \langle 1, 1 \rangle \}$

Una **palabra** sobre el alfabeto  $A$  es una sucesión finita de elementos de  $A$ .

$$u = a_1 \dots a_n$$

donde  $a_i \in A$ ,  $\forall i = 1, \dots, n$ .

## Ejemplo

Si  $A = \{0, 1\}$  entonces  $0111$  es una palabra sobre este alfabeto.

El conjunto de todas las palabras sobre un alfabeto  $A$  se nota como  $A^*$ .

**Notación - Palabras:**  $u, v, x, y, z, \dots$

Si  $u \in A^*$ , entonces la longitud de la palabra  $u$  es el número de símbolos de  $A$  que contiene.

**Notación:**  $|u|$

Si  $u = a_1 \dots a_n$ , entonces  $|u| = n$ .

La palabra vacía es la palabra de longitud cero.

**Notación:**  $\varepsilon$

**Notación:** El conjunto de cadenas sobre un alfabeto  $A$  excluyendo la cadena vacía se nota como  $A^+$ .



# Operaciones: Concatenación

Si  $u, v \in A^*$ ,  $u = a_1 \dots a_n$ ,  $v = b_1 \dots b_m$ , se llama **concatenación** de  $u$  y  $v$  a la cadena  $u.v$  (o simplemente  $uv$ ) dada por  $a_1 \dots a_n b_1 \dots b_m$ .

## Ejemplo

Si  $u = 011, v = 1010$ , entonces  $uv = 0111010$

## Propiedades

- 1  $|u.v| = |u| + |v|$ ,  $\forall u, v \in A^*$
- 2 *Asociativa*.-  $u.(v.w) = (u.v).w$ ,  $\forall u, v, w \in A^*$
- 3 *Elemento Neutro*.-  $u.\varepsilon = \varepsilon.u = u$ ,  $\forall u \in A^*$

Estructura de **monoide**

# Prefijos y Sufijos

## Prefijo

Si  $u \in A^*$  entonces  $v$  es un **prefijo** de  $u$  si  $\exists z \in A^*$  tal que  $vz = u$   
Un prefijo  $v$  de  $u$  se dice propio si  $v \neq \varepsilon$  y  $v \neq u$ .

## Sufijo

Si  $u \in A^*$  entonces  $v$  es un **sufijo** de  $u$  si  $\exists z \in A^*$  tal que  $zv = u$   
Un sufijo  $v$  de  $u$  se dice propio si  $v \neq \varepsilon$  y  $v \neq u$ .

# Iteración y Cadena Inversa

**Iteración**  $n$ -ésima de una cadena ( $u^n$ ) como la concatenación con ella misma  $n$  veces.

Si  $u \in A^*$  entonces

- $u^0 = \varepsilon$
- $u^{i+1} = u^i.u, \quad \forall i \geq 0$

## Ejemplo

Si  $u = 010$ , entonces  $u^3 = 010010010$ .

Si  $u = a_1 \dots a_n \in A^*$ , entonces la **cadena inversa** de  $u$  es la cadena  $u^{-1} = a_n \dots a_1 \in A^*$ .

## Ejemplo

Si  $u = 011$ , entonces  $u^{-1} = 110$ .

Un **lenguaje** sobre el alfabeto  $A$  es un subconjunto del conjunto de las cadenas sobre  $A$ :  $L \subseteq A^*$ .

**Notación** - Lenguajes:  $L, M, N, \dots$

**Ejemplos:**

- $L_1 = \{a, b, \epsilon\}$  *Tres palabras*
- $L_2 = \{a^i b^i \mid i = 0, 1, 2, \dots\}$  Una sucesión de  $a$  seguida de una de  $b$  de la misma longitud
- $L_3 = \{uu^{-1} \mid u \in A^*\}$  *Palíndromos de longitud par*
- $L_4 = \{a^{n^2} \mid n = 1, 2, 3, \dots\}$  Sucesiones de  $a$  de longitud un cuadrado perfecto

# Conjuntos Numerables

Un conjunto se dice **numerable** si existe una aplicación inyectiva de este conjunto en el conjunto de los números naturales, o lo que es lo mismo, se le puede asignar un número natural a cada elemento del conjunto de tal manera que dos elementos distintos tengan números distintos.

## Ejemplos

$A^*$  es siempre numerable. Si  $A = \{a_1, \dots, a_n\}$  entonces puedo asignar un número binario distinto de 0 y de la misma longitud a cada  $a_i$  de tal manera que símbolos distintos reciben números distintos y a cada palabra  $b_1 \dots b_k$  se le asigna el número cuya representación en binario es el que se obtiene sustituyendo cada  $b_i$  por su número binario.

**Ejemplo:** El conjunto de programas bien escritos en C es numerable.

# Un conjunto no numerable

**Ejemplo:** El conjunto de lenguajes sobre  $A^*$  (si  $A$  no es vacío) nunca es numerable.

Si lo fuese, y se pudiese asignar un número natural distinto  $f(L)$  a cada lenguaje  $L$ , sea  $a \in A$ . Definamos el lenguaje  $L$  formado por palabras de la forma  $a^i$  de acuerdo a lo siguiente: para cada  $i$  número natural:

- Si este número no es de un lenguaje, entonces  $a^i \in L$ .
- Si este número es del lenguaje,  $M$ , ( $i = f(M)$ )
  - Si  $a^i \notin M$  entonces  $a^i \in L$ .
  - Si  $a^i \in M$  entonces  $a^i \notin L$ .

$L$  no puede tener ningún número asociado. Si fuese  $j = f(L)$ , entonces la pertenencia de  $a^j$  a  $L$  es contradictoria:

- Si  $a^j \in L$  como  $j = f(L)$ , entonces  $a^j \notin L$
- Si  $a^j \notin L$  y  $j = f(L)$ , entonces  $a^j \in L$

Aparte de las operaciones de **unión** e **intersección** de lenguajes, dada su condición de conjuntos existe la operación de concatenación.

---

Si  $L_1, L_2$  son dos lenguajes sobre el alfabeto  $A$ , la **concatenación** de estos dos lenguajes se define como,

$$L_1 L_2 = \{u_1 u_2 \mid u_1 \in L_1, u_2 \in L_2\}$$

## Propiedades

- $L\emptyset = \emptyset L = \emptyset$
- *Elemento Neutro.*-  $\{\epsilon\}L = L\{\epsilon\} = L$
- *Asociativa.*-  $L_1(L_2 L_3) = (L_1 L_2)L_3$

Si  $L_1 = \{0^i 1^i : i \geq 0\}$ ,  $L_2 = \{1^i 0^i : i \geq 0\}$  entonces,

$$L_1 L_2 = \{0^i 1^i 1^j 0^j : i, j \geq 0\}$$



La **iteración** de lenguajes se define de forma recursiva:

$$L^0 = \{\epsilon\}, \quad L^{i+1} = L^i L$$

Si  $L$  es un lenguaje sobre el alfabeto  $A$ , la *clausura de Kleene* de  $L$  es

$$L^* = \bigcup_{i \geq 0} L^i$$

---

$$L^+ = \bigcup_{i \geq 1} L^i$$

## Propiedades:

- $L^+ = L^*$  si  $\epsilon \in L$
  - $L^+ = L^* - \{\epsilon\}$  si  $\epsilon \notin L$
- 

## Ejemplo

Si  $L = \{0,01\}$ , entonces:

$L^*$  = Conjunto de palabras sobre  $\{0,1\}$  en las que un uno va siempre precedido de un cero.

$L^+$  = Conjunto de palabras sobre  $\{0,1\}$  en las que un uno va siempre precedido de un cero y distintas de la palabra vacía.

El **lenguaje inverso** de  $L$  es el lenguaje dado por:

$$L^{-1} = \{u \mid u^{-1} \in L\}$$

La **cabecera** de  $L$  es el lenguaje dado por

$$\text{CAB}(L) = \{u \mid u \in A^* \text{ y } \exists v \in A^* \text{ tal que } uv \in L\}$$

## Ejemlo

Si  $L = \{0^i 1^j : i \geq 0\}$ , entonces  $\text{CAB}(L) = \{0^i 1^j : i \geq j \geq 0\}$ .

Si  $A_1$  y  $A_2$  son dos alfabetos, una aplicación

$$h : A_1^* \rightarrow A_2^*$$

se dice que es un **homomorfismo** si y solo si

$$h(uv) = h(u)h(v)$$

## Consecuencias

- $h(\epsilon) = \epsilon$
- $h(a_1 \dots a_n) = h(a_1) \dots h(a_n)$

Si  $A_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ ,  $A_2 = \{0, 1\}$

$$\begin{array}{llll} h(0) = 0000, & h(1) = 0001, & h(2) = 0010, & h(3) = 0011 \\ h(4) = 0100, & h(5) = 0101, & h(6) = 0110, & h(7) = 0111 \\ h(8) = 1000 & h(9) = 1001 & & \end{array}$$

$$h(034) = 000000110100, \quad h(\varepsilon) = \varepsilon$$

¿Verdadero o falso?

- Si  $A$  es un alfabeto, la aplicación que transforma cada palabra  $u \in A^*$  en su inversa es un homomorfismo de  $A^*$  en  $A^*$ .

FALSO

- La transformación que a cada palabra sobre  $\{0,1\}^*$  le añade 00 al principio y 11 al final es un homomorfismo.

FALSO

Una **gramática generativa** es un cuádrupla  $(V, T, P, S)$  en la que

- $V$  es un alfabeto, llamado de **variables** o símbolos no terminales. Sus elementos se suelen representar con letras mayúsculas.
- $T$  es un alfabeto, llamado de **símbolos terminales**. Sus elementos se suelen representar con letras minúsculas.
- $P$  es un conjunto finito de pares  $(\alpha, \beta)$ , llamados **reglas de producción**, donde  $\alpha, \beta \in (V \cup T)^*$  y  $\alpha$  contiene, al menos un símbolo de  $V$ .

El par  $(\alpha, \beta)$  se suele representar como  $\alpha \rightarrow \beta$ .

- $S$  es un elemento de  $V$ , llamado **símbolo de partida**.



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

- $V = \{E\}$
- $T = \{+, *, (, ), a, b, c\}$
- $P$  está compuesto por las siguientes reglas de producción

$$\begin{array}{lll} E \rightarrow E + E, & E \rightarrow E * E, & E \rightarrow (E), \\ E \rightarrow a, & E \rightarrow b, & E \rightarrow c \end{array}$$

- $S = E$

# Lenguaje Generado: idea intuitiva

Una gramática sirve para determinar un lenguaje.

$$\begin{array}{lll} E \rightarrow E + E, & E \rightarrow E * E, & E \rightarrow (E), \\ E \rightarrow a, & E \rightarrow b, & E \rightarrow c \end{array}$$

Las palabras son las de  $T^*$  que se obtienen a partir del símbolo inicial efectuando pasos de derivación. Cada paso consiste en elegir una parte de la palabra que coincide con la parte izquierda de una producción y sustituir esa parte por la derecha de la misma producción.

## Ejemplo

$$\begin{array}{l} EE \Rightarrow E * E \Rightarrow (E) * E \Rightarrow (E + E) * E \Rightarrow (a + E) * E \Rightarrow \\ (a + b) * E \Rightarrow (a + b) * b \end{array}$$

Palabra Generada

Gramática  $G = (V, T, P, S)$  y dos palabras  $\alpha, \beta \in (V \cup T)^*$   
 $\beta$  es **derivable** a partir de  $\alpha$  **en un paso** ( $\alpha \Rightarrow \beta$ )

si y solo si

existe una producción  $\gamma \rightarrow \varphi$  tal que

- $\alpha$  contiene a  $\gamma$  como subcadena.
- $\beta$  se obtiene sustituyendo  $\gamma$  por  $\varphi$  en  $\alpha$ .

$\beta$  es **derivable** de  $\alpha$  ( $\alpha \xRightarrow{*} \beta$ ),  
si y solo si existe una sucesión de palabras  $\gamma_1, \dots, \gamma_n$  ( $n \geq 1$ )  
tales que

$$\alpha = \gamma_1 \Rightarrow \gamma_2 \Rightarrow \dots \Rightarrow \gamma_n = \beta$$

Lenguaje generado por una gramática  $G = (V, T, P, S)$  al conjunto de cadenas formadas por símbolos terminales y que son derivables a partir del símbolo de partida.

Es decir,

$$L(G) = \{u \in T^* \mid S \xRightarrow{*} u\}$$

# Ejemplo

$G = (V, T, P, S)$ , donde  $V = \{S, A, B\}$ ,  $T = \{a, b\}$ , el símbolo de partida es  $S$  y las reglas son

$$\begin{array}{llll} S \rightarrow aB, & S \rightarrow bA, & A \rightarrow a, & A \rightarrow aS, \\ A \rightarrow bAA, & B \rightarrow b, & B \rightarrow bS, & B \rightarrow aBB \end{array}$$

Esta gramática genera el lenguaje

$$L(G) = \{u \mid u \in \{a, b\}^+ \text{ y } N_a(u) = N_b(u)\}$$

donde  $N_a(u)$  y  $N_b(u)$  son el número de apariciones de símbolos  $a$  y  $b$ , en  $u$ , respectivamente.

Esto es fácil de ver interpretando,

- $A$  palabra con una  $a$  de más
- $B$  palabra con una  $b$  de más
- $S$  palabra con igual número de  $a$  que de  $b$ .

Hay que demostrar dos cosas:

- Todas las palabras generadas por la gramática tienen el mismo número de  $a$  que de  $b$ .
- Cualquier palabra con el mismo número de  $a$  que de  $b$  es generada.

Para lo primero, podemos considerar  $N_{a,A}(\alpha)$  (número de  $a$  + número de  $A$ ) y  $N_{b,B}(\alpha)$  (número de  $b$  + número de  $B$ ) y tener en cuenta lo siguiente para una generación  $S \xRightarrow{*} u$ :

- Al principio de la generación tenemos:  $N_{a,A}(S) = N_{b,B}(S) = 0$
- Al aplicar cualquier regla  $\alpha_1 \Rightarrow \alpha_2$ , si  $N_{a,A}(u\alpha_1) = N_{b,B}(\alpha_1)$ , entonces  $N_{a,A}(\alpha_2) = N_{b,B}(\alpha_2)$
- Luego al final  $N_{a,A}(u) = N_{b,B}(u)$ , y como  $u$  no contiene variables,  $N_a(u) = N_b(u)$ , como se quería demostrar.



# Algoritmo de Generación

*Generación por la izquierda, un símbolo cada vez.*

- Para generar una  $a$ 
  - Si  $a$  último símbolo de la palabra, aplicar  $A \rightarrow a$
  - Si no es el último símbolo
    - Si la primera variable es  $S$  aplicar  $S \rightarrow aB$
    - Si la primera variable es  $B$  aplicar  $B \rightarrow aBB$
    - Si la primera variable es  $A$ 
      - Si hay más variables aplicar  $A \rightarrow a$
      - Si no hay más, aplicar  $A \rightarrow aS$
- Para generar una  $b$ 
  - Si  $b$  último símbolo de la palabra, aplicar  $B \rightarrow b$
  - Si no es el último símbolo
    - Si la primera variable es  $S$  aplicar  $S \rightarrow bA$
    - Si la primera variable es  $A$  aplicar  $A \rightarrow bAA$
    - Si la primera variable es  $B$ 
      - Si hay más variables aplicar  $B \rightarrow b$
      - Si no hay más, aplicar  $B \rightarrow bS$

# Condiciones de Garantía

- Las palabras generadas tienen primero símbolos terminales y después variables.
- Se genera un símbolo de la palabra en cada paso de derivación
- Las variables que aparecen en la palabra pueden ser:
  - Una cadena de  $A$  (si hemos generado más  $b$  que  $a$ )
  - Una cadena de  $B$  (si hemos generado más  $a$  que  $b$ )
  - Una  $S$  si hemos generado las mismas  $a$  que  $b$
- Antes de generar el último símbolo tendremos como variables:
  - Una  $A$  si tenemos que generar  $a$
  - Una  $B$  si tenemos que generar  $b$
- Entonces aplicamos la primera opción para generar los símbolos y la palabra queda generada.

Sea  $G = (\{S, X, Y\}, \{a, b, c\}, P, S)$  donde  $P$  tiene las reglas,

$$\begin{array}{llll} S \rightarrow abc & S \rightarrow aXbc & Xb \rightarrow bX & Xc \rightarrow Ybcc \\ bY \rightarrow Yb & aY \rightarrow aaX & aY \rightarrow aa & \end{array}$$

# Lenguaje Generado

$$\begin{array}{llll} S \rightarrow abc & S \rightarrow aXbc & Xb \rightarrow bX & Xc \rightarrow Ybcc \\ bY \rightarrow Yb & aY \rightarrow aaX & aY \rightarrow aa & \end{array}$$

Esta gramática genera el lenguaje:  $\{a^n b^n c^n \mid n = 1, 2, \dots\}$ .

---

Inicialmente tenemos dos posibilidades:

$$S \Rightarrow abc, \quad S \Rightarrow aXbc$$

Con la primera generamos la palabra  $abc$ .

Si seguimos la segunda opción, sólo se puede seguir cómo:

$$aXbc \Rightarrow abXc \Rightarrow abYbcc \Rightarrow aYbbcc$$

# Lenguaje Generado: $\{a^n b^n c^n \mid n = 1, 2, \dots\}$

$$\begin{array}{llll} S \rightarrow abc & S \rightarrow aXbc & Xb \rightarrow bX & Xc \rightarrow Ybcc \\ bY \rightarrow Yb & aY \rightarrow aaX & aY \rightarrow aa & \end{array}$$

Ya tenemos  $abc$ .

---

Habíamos generado también:  $aYbbcc$ . En este momento podemos aplicar dos reglas:

- $aY \rightarrow aa$ , en cuyo caso producimos  $aabbcc = a^2b^2c^2 \in L(G)$
- $aY \rightarrow aaX$ , en cuyo caso producimos  $aaXbbcc$

A partir de  $aaXbbcc$  repetimos el proceso: sólo se puede mover la  $X$  a la derecha hasta la frontera  $b-c$ . Entonces se añade una  $b$  y una  $c$  y se cambia  $X$  por  $Y$ . Después, se mueve la  $Y$  a la izquierda hasta que encuentra la frontera  $a-b$ . Entonces, tiene dos opciones: añadir sólo  $a$ , obteniendo la siguiente palabra, o  $aX$  con lo que se vuelven a generar las otras palabras.

- **Tipo 0** Cualquier gramática. Sin restricciones.  
**Lenguajes recursivamente enumerables.**
- **Tipo 1** Si todas las producciones tienen la forma

$$\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$$

donde  $\alpha_1, \alpha_2, \beta \in (V \cup T)^*$ ,  $A \in V$ , y  $\beta \neq \epsilon$ , excepto posiblemente la regla  $S \rightarrow \epsilon$ , en cuyo caso  $S$  no aparece a la derecha de las reglas. **Lenguajes dependientes del contexto.**

- **Tipo 2** Si cualquier producción tiene la forma

$$A \rightarrow \alpha$$

donde  $A \in V, \alpha \in (V \cup T)^*$ .

**Lenguajes Independientes del Contexto**

- **Tipo 3** Si toda regla tiene la forma

$$A \rightarrow uB \text{ ó } A \rightarrow u$$

donde  $u \in T^*$  y  $A, B \in V$ .

**Conjuntos Regulares**

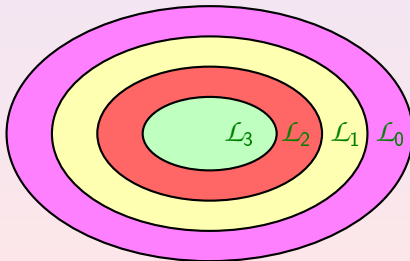
# Clases de Lenguajes

Un lenguaje se dice que es de **tipo**  $i$  ( $i = 0, 1, 2, 3$ ) si y solo si es generado por una gramática de tipo  $i$ .

La clase o familia de lenguajes de tipo  $i$  se denota por  $\mathcal{L}_i$ .

## Propiedad

$$\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$$





Demostrar que la gramática

$$G = (\{S\}, \{a, b\}, \{S \rightarrow \epsilon, S \rightarrow aSb\}, S)$$

genera el lenguaje  $L = \{a^i b^i \mid i = 0, 1, 2, \dots\}$

---

Inicialmente tenemos dos opciones:

$$S \Rightarrow \epsilon, \quad S \Rightarrow aSb$$

Con eso generamos la palabra vacía, o continuamos generando.

Otra vez hay dos opciones:

$$S \Rightarrow aSb \Rightarrow ab, \quad S \Rightarrow aSb \Rightarrow aaSbb$$

Demostrar que la gramática

$$G = (\{S\}, \{a, b\}, \{S \rightarrow \epsilon, S \rightarrow aSb\}, S)$$

genera el lenguaje  $L = \{a^i b^i \mid i = 0, 1, 2, \dots\}$

---

Si seguimos este procedimiento, nos encontramos que podemos ir generando todas las palabras de la forma  $a^i b^i$ , y siempre nos queda la palabra  $a^i S b^i$  para seguir generando las palabras de mayor longitud.

Por otra parte, estas son las únicas palabras que se pueden generar.

Encontrar el lenguaje generado por la gramática

$G = (\{A, B, C\}, \{a, b\}, P, S)$  donde  $P$  contiene las siguientes producciones

$$\begin{array}{lll} S \rightarrow aAB & bB \rightarrow a & Ab \rightarrow SBb \\ Aa \rightarrow SaB & B \rightarrow SA & B \rightarrow ab \end{array}$$

El resultado es el **Lenguaje vacío**: nunca se puede llegar a generar una palabra con símbolos terminales. Siempre que se sustituye  $S$  aparece  $A$ , y siempre que se sustituye  $A$  aparece  $S$ .

Encontrar una gramática libre del contexto para generar cada uno de los siguientes lenguajes

- 1  $L = \{a^i b^j \mid i, j \in \mathbb{N}, i \leq j\}$
- 2  $L = \{a^i b^j a^j b^i \mid i, j \in \mathbb{N}\}$
- 3  $L = \{a^i b^i a^j b^j \mid i, j \in \mathbb{N}\}$
- 4  $L = \{a^i b^i \mid i \in \mathbb{N}\} \cup \{b^i a^i \mid i \in \mathbb{N}\}$
- 5  $L = \{uu^{-1} \mid u \in \{a, b\}^*\}$
- 6  $L = \{a^i b^j c^{i+j} \mid i, j \in \mathbb{N}\}$

donde  $\mathbb{N}$  es el conjunto de los números naturales incluyendo el 0.

$$L = \{a^i b^j \mid i, j \in \mathbb{N}, i \leq j\}$$

$$S \rightarrow aSb$$

$$S \rightarrow \epsilon$$

$$S \rightarrow Sb$$

$$L = \{a^i b^j a^j b^i \mid i, j \in \mathbb{N}\}$$

Este lenguaje es generado por la siguiente gramática:

$$S \rightarrow aSb$$

$$S \rightarrow B, \quad B \rightarrow bBa, \quad B \rightarrow \epsilon$$

$$L = \{a^i b^i a^j b^j \mid i, j \in \mathbb{N}\}$$

Podemos generar  $\{a^i b^i \mid i \in \mathbb{N}\}$  con:

$$S_1 \rightarrow aS_1b, \quad S_1 \rightarrow \varepsilon$$

El lenguaje  $L$  se puede generar añadiendo:

$$S \rightarrow S_1 S_1$$

siendo  $S$  el símbolo inicial.

$$L = \{a^i b^i \mid i \in \mathbb{N}\} \cup \{b^i a^i \mid i \in \mathbb{N}\}$$

Podemos generar  $\{a^i b^i \mid i \in \mathbb{N}\}$  con:

$$S_1 \rightarrow aS_1b, \quad S_1 \rightarrow \varepsilon$$

y  $\{b^i a^i \mid i \in \mathbb{N}\}$  con

$$S_2 \rightarrow bS_2a, \quad S_2 \rightarrow \varepsilon$$

El lenguaje  $L$  se puede generar añadiendo:

$$S \rightarrow S_1, \quad S \rightarrow S_2$$

siendo  $S$  el símbolo inicial.



$$L = \{uu^{-1} \mid u \in \{a,b\}^*\}$$

Este lenguaje se genera con la gramática:

$$S \rightarrow aSa, \quad S \rightarrow bSb, \quad S \rightarrow \varepsilon$$

$$L = \{a^i b^j c^{i+j} \mid i, j \in \mathbb{N}\}$$

Este lenguaje se genera con la gramática:

$$S \rightarrow aSc, \quad S \rightarrow B,$$

$$B \rightarrow bBc, \quad B \rightarrow \varepsilon$$

Determinar si la gramática  $G = (\{S, A, B\}, \{a, b, c, d\}, P, S)$  donde  $P$  es el conjunto de reglas de producción:

$$\begin{array}{lll} S \rightarrow AB & A \rightarrow Ab & A \rightarrow a \\ B \rightarrow cB & B \rightarrow d & \end{array}$$

genera un lenguaje de tipo 3.

Esta gramática genera el lenguaje:  $\{ab^i c^j d : i, j \in \mathbb{N}\}$

Y este lenguaje se puede generar mediante la gramática:

$$S \rightarrow aB, \quad B \rightarrow bB, \quad B \rightarrow C, \quad C \rightarrow cC, \quad C \rightarrow d$$

Como esta gramática es de tipo 3, el lenguaje lo es.