

TEORÍA Y PRÁCTICA

Modelos Computacionales

<https://ismael-sallami.github.io>

Autor: Ismael Sallami Moreno



UNIVERSIDAD
DE GRANADA

21 de septiembre de 2025

Licencia

Este trabajo está licenciado bajo una [Licencia Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional](#).

Usted es libre de:

- **Compartir** — copiar y redistribuir el material en cualquier medio o formato.

Bajo los siguientes términos:

Reconocimiento Debe otorgar el crédito adecuado, proporcionar un enlace a la licencia e indicar si se han realizado cambios. Puede hacerlo de cualquier manera razonable, pero no de una manera que sugiera que tiene el apoyo del licenciante o lo recibe por el uso que hace.

NoComercial No puede utilizar el material para fines comerciales.

SinObraDerivada Si remezcla, transforma o crea a partir del material, no puede distribuir el material modificado.



Índice general

I Teoría	4
1 Introducción	5
2 Introducción a la Computación	6
2.1 Problema de la parada	6
2.2 Definiciones	6
2.3 Operaciones: Concatenación	7
2.4 Prefijos, Sufijos y subcadenas	7
2.5 Iteración y Palabra Inversa	8
2.6 Lenguajes	8
2.6.1 Ejemplos Adicionales de Lenguajes	8
2.6.2 Conjuntos Numerables	9
2.7 Un Conjunto No Numerable	9
2.8 Operaciones con Lenguajes: Concatenación	10
2.9 Propiedades de la Concatenación de Lenguajes	10
2.10 Iteración de Lenguajes y Clausura de Kleene	10
2.11 Operaciones con Lenguajes: Propiedades de Clausuras	11
2.12 Lenguaje Inverso	11
2.13 Cabecera de un Lenguaje	12
2.14 Homomorfismos entre Alfabetos	12
2.14.1 Consecuencias de la Definición	12
2.14.2 Ejemplo de Homomorfismo	12
2.14.3 Propiedades de Homomorfismos	13
2.15 Gramática Generativa	13
2.16 Lenguaje Generado: Idea Intuitiva	14
2.16.1 Ejemplo	14
2.17 Derivación y Lenguaje Generado	14
2.17.1 Derivación en un Paso	14
2.17.2 Secuencia de Derivación	14
2.17.3 Lenguaje Generado por una Gramática	15
2.18 Gramática Generativa: Ejemplo y Propiedades	15
2.18.1 Gramática Definida	15

2.18.2	Lenguaje Generado	15
2.18.3	Interpretación de las Variables	15
2.18.4	Propiedades del Lenguaje Generado	15
2.18.5	Demostración de la Primera Propiedad	16
2.18.6	Algoritmo de Generación	16
2.18.7	Condiciones de Garantía	16
2.19	Gramática Alternativa para el Mismo Lenguaje	17
2.19.1	Gramática que Incluye la Palabra Vacía	17
2.19.2	Gramática que Excluye la Palabra Vacía	17
2.20	Gramática Generativa: Ejemplo Adicional	17
2.20.1	Gramática Definida	17
2.20.2	Lenguaje Generado	18
2.20.3	Proceso de Derivación	18
2.20.4	Ejemplo de Derivación para $n = 3$	18
2.20.5	Propiedades del Lenguaje Generado	18
3	Relaciones de Ejercicios	20
3.1	Relación Tema 1: Modelos de Computación	20

Parte I

Teoría

Introducción

- Profesor: Serafín Moral
- Correo: smc@decsai.ugr.es
- El profesor recomienda ir a las tutorías y avisarle antes.
- J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979): un libro básico, se requiere ciertos conocimientos matemáticos para leerlo.
- M. Alfonseca, J. Sancho. M. Martínez, Teoría de Autómatas y Lenguajes Formales. Publicaciones R.A.E.C., Textos Cátedra (1997): básico y fácil de entender, está en la biblioteca en español.
- Los demás son una buena opción también.

La asignatura de Modelos de Computación se centra en el estudio de los fundamentos teóricos de la informática, explorando conceptos como autómatas, lenguajes formales, gramáticas y máquinas de Turing. Estos temas proporcionan las bases para comprender las capacidades y limitaciones de los sistemas computacionales, así como para analizar la complejidad de los problemas y los algoritmos que los resuelven. Es una materia esencial para quienes deseen profundizar en la teoría de la computación y su aplicación en el diseño de sistemas eficientes y correctos.

Introducción a la Computación

En el pasado, había teoremas que eran verdad, pero las matemáticas no eran capaces de demostrarlo. Turing puso solución a esto exponiendo que esta incompletitud de las matemáticas se corregía diciendo que este tipo de problemas eran indecidibles. En cuanto a la complejidad de los algoritmos, como vimos en algorítmica, podemos distinguir entre p (polinómico) y np (polinómico no determinista). Dentro de np , encontramos los np completos y los np difíciles (problemas que no puede resolver un ordenador convencional). Hoy día lo más importante es la complejidad algorítmica. Nadie ha demostrado que p sea distinto de np .

2.1 Problema de la parada

Trata el tema de la existencia de que un programa que lea otro programa y unos datos y nos diga si este termina o cicla indefinidamente. Un programa es lo mismo que los datos, según Turing. En este caso, se pone como datos del programa el mismo programa.

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

Este programa lleva a una contradicción, ya que si termina no termina y si termina termina, cosa que no es coherente. Por ende, podemos concluir que si un programa se llama así mismo, en este caso, llegamos a una contradicción. Turing, con esto, llegó a la conclusión de que las matemáticas eran incompletas, ya que este programa no existe. La explicación es sencilla, basta con decir que es como un espejo, nunca va a pasar STOP ya que si eso pasa el programa nunca arrancaría.

2.2 Definiciones

Definición 2.1 . Alfabeto: Un alfabeto es un conjunto finito cuyos elementos se denominan símbolos o letras. Si los símbolos tienen varios elementos, se representan entre $\langle \rangle$.

Definición 2.2 . Palabra: Una palabra es una sucesión finita de elementos de un alfabeto A . Formalmente, $u = a_1 a_2 \dots a_n$, donde $a_i \in A$ para todo $i = 1, \dots, n$.

Definición 2.3 . Conjunto de Palabras: El conjunto de todas las palabras que se pueden formar sobre un alfabeto A se denota como A^* .

Definición 2.4 . Notación de Palabras: Las palabras se denotan comúnmente como u, v, x, y, z, \dots

Definición 2.5 . Longitud de una Palabra: Si $u \in A^*$, la longitud de la palabra u es el número de símbolos de A que contiene. - Notación: $|u|$ - Si $u = a_1 a_2 \dots a_n$, entonces $|u| = n$.

Definición 2.6 . Palabra Vacía: La palabra vacía es la palabra de longitud cero.

- Notación: ε

Definición 2.7 . Conjunto de Palabras no Vacías: El conjunto de cadenas sobre un alfabeto A excluyendo la palabra vacía se denota como A^+ .

2.3 Operaciones: Concatenación

Definición 2.8 . Concatenación de Palabras: 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 2.1 . 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

La concatenación de palabras sobre un alfabeto A junto con la palabra vacía ε forma un monoide.

Definición 2.9 . Monoide: Un monoide es una estructura algebraica (M, \cdot, e) que consta de un conjunto M , una operación binaria asociativa $\cdot : M \times M \rightarrow M$, y un elemento neutro $e \in M$ tal que: 1. ****Asociatividad****: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, $\forall a, b, c \in M$. 2. ****Elemento Neutro****: $a \cdot e = e \cdot a = a$, $\forall a \in M$.

2.4 Prefijos, Sufijos y subcadenas

Definición 2.10 . 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 $v \neq u$.

Definición 2.11 . 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 $v \neq u$.

Definición 2.12 . Subcadena: Si $u \in A^*$, entonces v es una subcadena de u si $\exists z_1, z_2 \in A^*$ tal que $z_1 v z_2 = u$.

Una subcadena v de u se dice propia si $v \neq \varepsilon$ y $v \neq u$.

2.5 Iteración y Palabra Inversa

Definición 2.13 . Iteración de una Cadena: La iteración n -ésima de una cadena (u^n) se define como la concatenación de la cadena consigo misma n veces. Si $u \in A^*$, entonces:

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

Ejemplo 2.2 . Si $u = 010$, entonces $u^3 = 010010010$.

Definición 2.14 . Palabra Inversa: Si $u = a_1 \dots a_n \in A^*$, entonces la palabra inversa de u es la cadena $u^{-1} = a_n \dots a_1 \in A^*$.

Ejemplo 2.3 . Si $u = 011$, entonces la palabra inversa de u es $u^{-1} = 110$.

2.6 Lenguajes

Definición 2.15 . Lenguaje: Un lenguaje sobre un alfabeto A es un subconjunto del conjunto de las cadenas sobre A , es decir, $L \subseteq A^*$. La palabra vacía siempre pertenece a un lenguaje.

Notación

- Lenguajes: L, M, N, \dots

Ejemplo 2.4 . 1) $L_1 = \{a, b, \varepsilon\}$

2) $L_2 = \{a^i b^j \mid i = 0, 1, 2, \dots\}$

3) $L_3 = \{uu^{-1} \mid u \in A^*\}$

4) $L_4 = \{a^{n^2} \mid n = 1, 2, 3, \dots\}$

2.6.1 Ejemplos Adicionales de Lenguajes

1. $L_5 = \{a^n b^m c^k \mid n, m, k \geq 0\}$: Palabras con cualquier número de a, b , y c en ese orden.
2. $L_6 = \{w \in \{0, 1\}^* \mid w \text{ tiene un número par de } 1\}$: Palabras binarias con un número par de unos.
3. $L_7 = \{a^n b^n c^n \mid n \geq 0\}$: Palabras con el mismo número de a, b , y c en ese orden.
4. $L_8 = \{w \in \{0, 1\}^* \mid w \text{ es un palíndromo}\}$: Palabras binarias que son palíndromos.
5. $L_9 = \{a^{2^n} \mid n \geq 0\}$: Sucesiones de a cuya longitud es una potencia de dos.

2.6.2 Conjuntos Numerables

Un conjunto se dice numerable si existe una aplicación inyectiva (corresponde cada elemento con su imagen) 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

Ejemplo 2.5 . 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**: Si $A = \{a, b\}$, podemos asignar $a = 01$, $b = 10$. Entonces, para la palabra ab , su representación binaria sería 0110.

Ejemplo 2.6 . El conjunto de programas bien escritos en C es numerable. Esto se debe a que los programas son cadenas finitas de un alfabeto finito, y por lo tanto, se pueden enumerar de manera similar a las palabras en A^* .

El hecho de que en el ordenador se trabaje con float, double, ... es porque como los números reales son conjuntos no numerables, un solo número real acabaría con toda la memoria del ordenador. Lo mismo pasa en los lenguajes, debemos de restringirnos a los lenguajes con los que podamos trabajar. Solo existe un conjunto numerable de programas.

2.7 Un Conjunto No Numerable

Ejemplo 2.7 . El conjunto de lenguajes sobre A^* (si A no es vacío) nunca es numerable.

Haremos la demostración por reducción al absurdo.

Si lo fuese, se podría 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 $a^j \in L$.

Por lo tanto, el conjunto de lenguajes sobre A^* no es numerable.

2.8 Operaciones con Lenguajes: Concatenación

Dada su condición de conjuntos, además de las operaciones de unión, intersección y complementario, los lenguajes también admiten 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:

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

Ejemplo

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

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

2.9 Propiedades de la Concatenación de Lenguajes

1. Propiedad de Aniquilación

$$L\emptyset = \emptyset L = \emptyset$$

2. Elemento Neutro

$$\{\varepsilon\}L = L\{\varepsilon\} = L$$

3. Asociatividad

$$L_1(L_2 L_3) = (L_1 L_2)L_3$$

2.10 Iteración de Lenguajes y Clausura de Kleene

La iteración de lenguajes se define de forma recursiva:

- $L^0 = \{\varepsilon\}$
- $L^{i+1} = L^i L, \forall i \geq 0$

Si L es un lenguaje sobre el alfabeto A , se definen las siguientes clausuras:

- **Clausura de Kleene:**

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

- **Clausura Positiva:**

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

Ejemplo

Si $L = \{a, b\}$:

- $L^0 = \{\varepsilon\}$
- $L^1 = \{a, b\}$
- $L^2 = \{aa, ab, ba, bb\}$
- $L^* = \{\varepsilon, a, b, aa, ab, ba, bb, \dots\}$
- $L^+ = \{a, b, aa, ab, ba, bb, \dots\}$

2.11 Operaciones con Lenguajes: Propiedades de Clausuras

1. Relación entre Clausura de Kleene y Clausura Positiva

- Si $\varepsilon \in L$, entonces $L^+ = L^*$.
- Si $\varepsilon \notin L$, entonces $L^+ = L^* \setminus \{\varepsilon\}$.

Ejemplo 2.8 . Si $L = \{0, 01\}$:

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

2.12 Lenguaje Inverso

Definición 2.16 . Lenguaje Inverso: El lenguaje inverso de un lenguaje L sobre un alfabeto A se define como:

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

Ejemplo

Si $L = \{011, 101\}$, entonces:

$$L^{-1} = \{110, 101\}$$

Propiedades

1. $(L^{-1})^{-1} = L$
2. $(L_1 \cup L_2)^{-1} = L_1^{-1} \cup L_2^{-1}$
3. $(L_1 L_2)^{-1} = L_2^{-1} L_1^{-1}$
4. $(L^*)^{-1} = (L^{-1})^*$

2.13 Cabecera de un Lenguaje

Definición 2.17 . Cabecera de un Lenguaje: La cabecera de un lenguaje L sobre un alfabeto A se define como:

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

Ejemplo

Si $L = \{0^i 1^j \mid i \geq 0\}$, entonces:

$$\text{CAB}(L) = \{0^i 1^j \mid i \geq j \geq 0\}$$

2.14 Homomorfismos entre Alfabetos

Definición 2.18 . 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:

$$h(uv) = h(u)h(v), \quad \forall u, v \in A_1^*$$

2.14.1 Consecuencias de la Definición

1. Imagen de la palabra vacía

$$h(\varepsilon) = \varepsilon$$

2. Imagen de una palabra

Si $u = a_1 a_2 \dots a_n \in A_1^*$, entonces:

$$h(u) = h(a_1)h(a_2) \dots h(a_n)$$

2.14.2 Ejemplo de Homomorfismo

Sea $A_1 = \{a, b\}$ y $A_2 = \{0, 1\}$. Definimos $h : A_1^* \rightarrow A_2^*$ como:

- $h(a) = 01$
- $h(b) = 10$

Entonces:

- $h(\varepsilon) = \varepsilon$
- $h(ab) = h(a)h(b) = 0110$
- $h(aba) = h(a)h(b)h(a) = 010110$

2.14.3 Propiedades de Homomorfismos

1. Preservación de la Concatenación

$$h(u.v) = h(u)h(v), \quad \forall u, v \in A_1^*$$

2. Homomorfismo Inverso

Si $h : A_1^* \rightarrow A_2^*$ es un homomorfismo, entonces:

$$h(u^{-1}) = h(u)^{-1}, \quad \forall u \in A_1^*$$

3. Composición de Homomorfismos

Si $h_1 : A_1^* \rightarrow A_2^*$ y $h_2 : A_2^* \rightarrow A_3^*$ son homomorfismos, entonces su composición $h_2 \circ h_1 : A_1^* \rightarrow A_3^*$ también es un homomorfismo.

Ejemplo de Composición

Sea $h_1 : A_1^* \rightarrow A_2^*$ y $h_2 : A_2^* \rightarrow A_3^*$ definidos como:

- $h_1(a) = 01, h_1(b) = 10$
- $h_2(0) = x, h_2(1) = y$

Entonces, para $u = ab \in A_1^*$:

- $h_1(u) = 0110$
- $h_2(h_1(u)) = h_2(0110) = xyxy$

2.15 Gramática Generativa

Definición 2.19 . Una gramática generativa es una cuádrupla (V, T, P, S) donde:

- 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 (α, β) , llamados reglas de producción, donde $\alpha, \beta \in (V \cup T)^*$ y α contiene al menos un símbolo de V .
 - El par (α, β) se suele representar como $\alpha \rightarrow \beta$.
- S : Es un elemento de V , llamado símbolo de partida.

Tiene la misma potencia que un lenguaje, por ende, podemos pensar que es similar a un lenguaje de programación aunque pensemos que no.

Ejemplo 2.9 . Sea la gramática $G = (V, T, P, S)$ definida como:

- $V = \{S, A\}$
- $T = \{a, b\}$

- $P = \{S \rightarrow aA, A \rightarrow b\}$
- $S = S$

Esta gramática genera el lenguaje $L = \{ab\}$.

2.16 Lenguaje Generado: Idea Intuitiva

Una gramática sirve para determinar un lenguaje. Las palabras generadas pertenecen al conjunto de símbolos terminales T^* y 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.

2.16.1 Ejemplo

Dada la gramática:

- $E \rightarrow E + E$
- $E \rightarrow E * E$
- $E \rightarrow (E)$
- $E \rightarrow a$
- $E \rightarrow b$
- $E \rightarrow c$

Derivación de una palabra:

$$E \Rightarrow E * E \Rightarrow (E) * E \Rightarrow (E + E) * E \Rightarrow (a + E) * E \Rightarrow (a + b) * E \Rightarrow (a + b) * b$$

Palabra Generada: $(a + b) * b$

2.17 Derivación y Lenguaje Generado

2.17.1 Derivación en un Paso

Dada una gramática $G = (V, T, P, S)$ y dos palabras $\alpha, \beta \in (V \cup T)^*$, se dice que β es derivable a partir de α en un paso ($\alpha \Rightarrow \beta$) si y solo si existe una producción $\gamma \rightarrow \phi$ tal que α contiene a γ como subcadena y β se obtiene sustituyendo γ por ϕ en α .

2.17.2 Secuencia de Derivación

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

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

2.17.3 Lenguaje Generado por una Gramática

El lenguaje generado por una gramática $G = (V, T, P, S)$ es el conjunto de cadenas formadas por símbolos terminales que son derivables a partir del símbolo de partida S . Formalmente:

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

2.18 Gramática Generativa: Ejemplo y Propiedades

2.18.1 Gramática Definida

Sea la gramática $G = (V, T, P, S)$ definida como:

- $V = \{S, A, B\}$
- $T = \{a, b\}$
- $P = \{S \rightarrow aB, S \rightarrow bA, A \rightarrow a, A \rightarrow aS, A \rightarrow bAA, B \rightarrow b, B \rightarrow bS, B \rightarrow aBB\}$
- $S = S$

2.18.2 Lenguaje Generado

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 los símbolos a y b en u , respectivamente.

2.18.3 Interpretación de las Variables

- A : Representa palabras con una a de más.
- B : Representa palabras con una b de más.
- S : Representa palabras con igual número de a que de b .

2.18.4 Propiedades del Lenguaje Generado

1. Todas las palabras generadas tienen el mismo número de a que de b .
2. Cualquier palabra con el mismo número de a que de b puede ser generada.

2.18.5 Demostración de la Primera Propiedad

Consideremos $N_{a,A}(\alpha)$ (número de a más número de A) y $N_{b,B}(\alpha)$ (número de b más número de B). Para una derivación $S \xRightarrow{*} u$, tenemos:

- Al inicio: $N_{a,A}(S) = N_{b,B}(S) = 0$.
- Al aplicar cualquier regla $\alpha_1 \rightarrow \alpha_2$, si $N_{a,A}(\alpha_1) = N_{b,B}(\alpha_1)$, entonces $N_{a,A}(\alpha_2) = N_{b,B}(\alpha_2)$.

Por lo tanto, al final de la derivación, $N_{a,A}(u) = N_{b,B}(u)$. Como u no contiene variables, $N_a(u) = N_b(u)$, lo que demuestra la propiedad.

2.18.6 Algoritmo de Generación

La generación de palabras se realiza por la izquierda, un símbolo a la vez:

- **Para generar una a :**
 - Si a es el último símbolo, 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 es el último símbolo, 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 $B \rightarrow bS$.

2.18.7 Condiciones de Garantía

1. Las palabras generadas tienen primero símbolos terminales y después variables.
2. Se genera un símbolo de la palabra en cada paso de derivación.
3. Las variables que aparecen en la palabra pueden ser:

- Una cadena de A (si se han generado más b que a).
- Una cadena de B (si se han generado más a que b).
- Una S (si se han generado el mismo número de a y b).

Antes de generar el último símbolo, las variables serán:

- Una A si se necesita generar una a .
- Una B si se necesita generar una b .

En este caso, se aplica la primera opción para generar los símbolos, y la palabra queda generada.

2.19 Gramática Alternativa para el Mismo Lenguaje

2.19.1 Gramática que Incluye la Palabra Vacía

Esta gramática genera todas las palabras con el mismo número de símbolos a que b , incluyendo la palabra vacía:

- $S \rightarrow aSbS$
- $S \rightarrow bSaS$
- $S \rightarrow \epsilon$

2.19.2 Gramática que Excluye la Palabra Vacía

Si no se desea incluir la palabra vacía, se puede usar la siguiente gramática:

- $S \rightarrow SS$
- $S \rightarrow ab$
- $S \rightarrow ba$
- $S \rightarrow aSb$
- $S \rightarrow bSa$

2.20 Gramática Generativa: Ejemplo Adicional

2.20.1 Gramática Definida

Sea la gramática $G = (V, T, P, S)$ definida como:

- $V = \{S, X, Y\}$
- $T = \{a, b, c\}$
- $P = \{S \rightarrow abc, S \rightarrow aXbc, Xb \rightarrow bX, Xc \rightarrow Ybcc, bY \rightarrow Yb, aY \rightarrow aaX, aY \rightarrow aa\}$
- $S = S$

2.20.2 Lenguaje Generado

Esta gramática genera el lenguaje:

$$L(G) = \{a^n b^n c^n \mid n \geq 1\}$$

2.20.3 Proceso de Derivación

1. Caso Base

- $S \rightarrow abc$: Genera la palabra abc para $n = 1$.

2. Caso General

- $S \rightarrow aXbc$: Introduce la variable X para generar palabras de mayor longitud.

A partir de $aXbc$, el proceso es el siguiente:

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

En este punto, se tienen dos opciones:

- Aplicar $aY \rightarrow aa$:

$$aYbbcc \Rightarrow aabbcc$$

Genera la palabra $a^2 b^2 c^2$.

- Aplicar $aY \rightarrow aaX$:

$$aYbbcc \Rightarrow aaXbbcc$$

Introduce nuevamente la variable X , permitiendo repetir el proceso para generar palabras más largas.

3. Iteración del Proceso

- En cada iteración, la variable X se mueve hacia la frontera $b - c$, donde se añade una b y una c , y X se transforma en Y .
- La variable Y se mueve hacia la frontera $a - b$, donde se elige entre añadir una a o una aX , permitiendo continuar el proceso.

2.20.4 Ejemplo de Derivación para $n = 3$

$$\begin{aligned} S &\Rightarrow aXbc \Rightarrow abXc \Rightarrow abYbcc \Rightarrow aYbbcc \Rightarrow aaXbbcc \Rightarrow aabXbcc \\ &\Rightarrow aabYbbccc \Rightarrow aaYbbbccc \Rightarrow aaaXbbbccc \Rightarrow aaabbbbccc \end{aligned}$$

2.20.5 Propiedades del Lenguaje Generado

1. Todas las palabras generadas tienen la forma $a^n b^n c^n$.
2. El proceso de derivación asegura que el número de a , b , y c es siempre igual.

3. El lenguaje generado es un subconjunto de $\{a, b, c\}^*$ con la restricción de igualdad en las cantidades de a , b , y c .

Relaciones de Ejercicios

3.1 Relación Tema 1: Modelos de Computación

Ejercicio 3.1.1. Descripción de lenguajes generados por gramáticas.

- a) Describir el lenguaje generado por la siguiente gramática:

$$S \rightarrow XYX$$

$$X \rightarrow aX \mid bX \mid \epsilon$$

$$Y \rightarrow bbb$$

Solución (Ejercicio 1.a). El lenguaje generado por la gramática está compuesto por cadenas que tienen la forma:

- 1) Una secuencia de cero o más a o b (generada por X).
- 2) Seguido por bbb (generado por Y).
- 3) Seguido nuevamente por una secuencia de cero o más a o b (generada por X).

Por lo tanto, el lenguaje generado es:

$$L = \{w_1 bbb w_2 \mid w_1, w_2 \in \{a, b\}^*\}$$

Donde w_1 y w_2 son cadenas arbitrarias (incluyendo la cadena vacía) formadas por los símbolos a y b. Otra forma es demostrándolo mediante doble inclusión (manera más matemática).

- b) Describir el lenguaje generado por la siguiente gramática:

$$S \rightarrow aX$$

$$X \rightarrow aX \mid bX \mid \epsilon$$

Solución (Ejercicio 1.b). El lenguaje generado por la gramática está compuesto por cadenas que tienen la forma:

- 1) Una a inicial (generada por S).
- 2) Seguido por una secuencia de cero o más a o b (generada por X).

Por lo tanto, el lenguaje generado es:

$$L = \{a w \mid w \in \{a, b\}^*\}$$

Donde w es una cadena arbitraria (incluyendo la cadena vacía) formada por los símbolos a y b . Otra forma de demostrarlo es mediante doble inclusión.

c) Describir el lenguaje generado por la siguiente gramática:

$$S \rightarrow XaXaX$$

$$X \rightarrow aX \mid bX \mid \epsilon$$

Solución (Ejercicio 1.c). El lenguaje generado por la gramática está compuesto por cadenas que tienen la forma:

- 1) Una secuencia de cero o más a o b (generada por X).
- 2) Seguido por una a .
- 3) Seguido nuevamente por una secuencia de cero o más a o b (generada por X).
- 4) Seguido por otra a .
- 5) Seguido nuevamente por una secuencia de cero o más a o b (generada por X).

Por lo tanto, el lenguaje generado es:

$$L = \{w_1 a w_2 a w_3 \mid w_1, w_2, w_3 \in \{a, b\}^*\}$$

Donde w_1 , w_2 y w_3 son cadenas arbitrarias (incluyendo la cadena vacía) formadas por los símbolos a y b . Otra forma de demostrarlo es mediante doble inclusión.

d) Describir el lenguaje generado por la siguiente gramática:

$$S \rightarrow SS \mid XaXaX \mid \epsilon$$

$$X \rightarrow bX \mid \epsilon$$

Solución (Ejercicio 1.d). El lenguaje generado por la gramática está compuesto por cadenas que tienen las siguientes características:

- 1) La gramática permite generar la cadena vacía (ϵ).
- 2) También permite generar cadenas de la forma $w_1 a w_2 a w_3$, donde w_1 , w_2 , y w_3 son cadenas formadas únicamente por el símbolo b (generadas por X).
- 3) Además, permite concatenar arbitrariamente las cadenas generadas en los puntos anteriores debido a la regla $S \rightarrow SS$.

Por lo tanto, el lenguaje generado es:

$$L = \{\epsilon\} \cup \{w_1 a w_2 a w_3 \mid w_1, w_2, w_3 \in \{b\}^*\} \cup \{uv \mid u, v \in L\}$$

Donde w_1 , w_2 , y w_3 son cadenas arbitrarias (incluyendo la cadena vacía) formadas por el símbolo b , y u, v son cadenas generadas por la gramática. Otra forma de demostrarlo es mediante doble inclusión.

Demostración por doble inclusión:– **Primera inclusión** ($L \subseteq R$):

Sea $w \in L$. Según las reglas de la gramática, w puede ser:

- La cadena vacía (ϵ), que claramente pertenece a R .
- Una cadena de la forma $w_1 a w_2 a w_3 a$, donde $w_1, w_2, w_3 \in \{b\}^*$. Estas cadenas también pertenecen a R por definición.
- Una concatenación de cadenas en L (por la regla $S \rightarrow SS$). Si $u, v \in L$, entonces $uv \in R$ porque R es cerrado bajo concatenación.

Por lo tanto, $w \in R$, y se cumple que $L \subseteq R$.

– **Segunda inclusión** ($R \subseteq L$):

Sea $w \in R$. Según la definición de R , w puede ser:

- La cadena vacía (ϵ), que claramente puede ser generada por la gramática.
- Una cadena de la forma $w_1 a w_2 a w_3 a$, donde $w_1, w_2, w_3 \in \{b\}^*$. Estas cadenas pueden ser generadas por la regla $S \rightarrow XaXaX$ y $X \rightarrow bX \mid \epsilon$.
- Una concatenación de cadenas en R . Si $u, v \in R$, entonces $uv \in L$ porque la regla $S \rightarrow SS$ permite concatenar cadenas generadas por la gramática.

Por lo tanto, $w \in L$, y se cumple que $R \subseteq L$.

Dado que $L \subseteq R$ y $R \subseteq L$, se concluye que $L = R$.

Ejercicio 3.1.2. *Determinar lenguajes.*

a) Dada la gramática $G = (\{S, A\}, \{a, b\}, P, S)$ donde:

$$P = \{S \rightarrow abAS, abA \rightarrow baab, S \rightarrow a, A \rightarrow b\}$$

Determinar el lenguaje que genera.

Solución (Ejercicio 2.a). Cada vez que aplicamos $S \rightarrow abAS$ generamos un bloque abA adicional y dejamos un S al final para poder repetir la expansión. Tras m aplicaciones de $S \rightarrow abAS$ obtenemos la forma $(abA)^m S$.

Cada bloque abA puede convertirse o bien en $baab$ aplicando la regla $abA \rightarrow baab$, o bien en abb aplicando primero $A \rightarrow b$ (porque $abA \Rightarrow abb$). Finalmente $S \rightarrow a$. Por tanto, cada bloque se convierte en $baab$ o en abb y al final queda una a .

De aquí se deduce la forma general de las cadenas generadas:

$$L(G) = \{xa \mid x \in \{baab, abb\}^*\},$$

es decir, en notación de expresiones regulares:

$$L(G) = (baab \mid abb)^* a.$$

Prueba formal (dos sentidos)

$$1) L(G) \subseteq (baab \mid abb)^*a$$

Tras m aplicaciones de $S \rightarrow abAS$ se tiene $(abA)^m S$ (prueba por inducción sobre m : base $m = 0$ trivial; paso: si $S \Rightarrow (abA)^k S$ entonces aplicando $S \rightarrow abAS$ al S final obtenemos $(abA)^{k+1} S$).

Para cada uno de los m factores abA podemos aplicar $abA \rightarrow baab$ (obteniendo $baab$) o bien aplicar $A \rightarrow b$ (obteniendo abb). Por tanto, la parte antes de la última S es una concatenación de $baab$ y abb .

Finalmente $S \rightarrow a$. Por tanto, toda cadena derivable tiene la forma (bloques $baab$ o abb) seguida de a .

$$2) (baab \mid abb)^*a \subseteq L(G)$$

Sea $w = b_1 b_2 \dots b_m a$ con cada $b_i \in \{baab, abb\}$.

Expandimos S m veces con $S \rightarrow abAS$ para obtener $(abA)^m S$.

Para cada i : si $b_i = baab$ aplicamos la regla $abA \rightarrow baab$ sobre el i -ésimo factor; si $b_i = abb$ aplicamos $A \rightarrow b$ en ese factor (convirtiendo abA en abb).

Finalmente aplicamos $S \rightarrow a$. Eso produce exactamente w . Por tanto, cualquier cadena del lado derecho puede derivarse.

b) Sea la gramática $G = (V, T, P, S)$ donde:

$$V = \{\langle \text{numero} \rangle, \langle \text{digito} \rangle\}$$

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

$$S = \langle \text{numero} \rangle$$

- $\langle \text{numero} \rangle \rightarrow \langle \text{numero} \rangle \langle \text{digito} \rangle$
- $\langle \text{numero} \rangle \rightarrow \langle \text{digito} \rangle$
- $\langle \text{digito} \rangle \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Determinar el lenguaje que genera.

Solución (Ejercicio 2.b). El lenguaje generado por la gramática está compuesto por cadenas que tienen las siguientes características:

- 1) La gramática permite generar cadenas formadas por uno o más dígitos, ya que:
 - $\langle \text{numero} \rangle \rightarrow \langle \text{numero} \rangle \langle \text{digito} \rangle$ permite construir cadenas de longitud arbitraria añadiendo dígitos.
 - $\langle \text{numero} \rangle \rightarrow \langle \text{digito} \rangle$ permite terminar la construcción con un único dígito.
- 2) Cada dígito es uno de los símbolos terminales $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, según la regla $\langle \text{digito} \rangle \rightarrow 0 \mid 1 \mid \dots \mid 9$.

Por lo tanto, el lenguaje generado es el conjunto de todas las cadenas no vacías de dígitos, es decir:

$$L = \{w \mid w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^+\}.$$

En notación de expresiones regulares, el lenguaje puede escribirse como:

$$L = [0 - 9]^+.$$

c) Sea la gramática $G = (\{A, S\}, \{a, b\}, S, P)$ donde las reglas de producción son:

- $S \rightarrow aS$
- $S \rightarrow aA$
- $A \rightarrow bA$
- $A \rightarrow b$

Determinar el lenguaje generado por la gramática.

Solución (Ejercicio 2.c). El lenguaje generado por la gramática está compuesto por cadenas que tienen las siguientes características:

- 1) La gramática permite generar cadenas que comienzan con uno o más símbolos a , ya que:
 - $S \rightarrow aS$ permite añadir un número arbitrario de a al principio.
 - $S \rightarrow aA$ permite terminar la secuencia de a y pasar a generar b .
- 2) Después de la secuencia de a , la gramática genera uno o más símbolos b , ya que:
 - $A \rightarrow bA$ permite añadir un número arbitrario de b .
 - $A \rightarrow b$ permite terminar la secuencia de b .

Por lo tanto, el lenguaje generado es el conjunto de todas las cadenas que consisten en una secuencia no vacía de a seguida de una secuencia no vacía de b , es decir:

$$L = \{a^n b^m \mid n \geq 1, m \geq 1\}.$$

En notación de expresiones regulares, el lenguaje puede escribirse como:

$$L = a^+ b^+.$$

Ejercicio 3.1.3. Gramáticas de tipo 2 y tipo 3.

- a) Encontrar una gramática de tipo 2 para el lenguaje de palabras en las que el número de b no es tres. Determinar si el lenguaje generado es de tipo 3.
- b) Encontrar una gramática de tipo 2 para el lenguaje de palabras que tienen 2 ó 3 b . Determinar si el lenguaje generado es de tipo 3.

Ejercicio 3.1.4. Gramáticas para lenguajes específicos.

- a) Encontrar una gramática de tipo 2 para el lenguaje de palabras que no contienen la subcadena ab . Determinar si el lenguaje generado es de tipo 3.
- b) Encontrar una gramática de tipo 2 para el lenguaje de palabras que no contienen la subcadena baa . Determinar si el lenguaje generado es de tipo 3.

Ejercicio 3.1.5. Lenguaje con más a que b . Encontrar una gramática libre de contexto que genere el lenguaje

sobre el alfabeto $\{a, b\}$ de las palabras que tienen más a que b (al menos una más).

Ejercicio 3.1.6. Gramáticas regulares o libres de contexto.

- Encontrar, si es posible, una gramática regular (o, si no es posible, una gramática libre de contexto) que genere el lenguaje L sobre el alfabeto $\{a, b\}$ tal que $u \in L$ si, y solamente si, u no contiene dos símbolos b consecutivos.
- Encontrar, si es posible, una gramática regular (o, si no es posible, una gramática libre de contexto) que genere el lenguaje L sobre el alfabeto $\{a, b\}$ tal que $u \in L$ si, y solamente si, u contiene dos símbolos b consecutivos.

Ejercicio 3.1.7. Propiedades de lenguajes.

- Encontrar, si es posible, una gramática regular (o, si no es posible, una gramática libre de contexto) que genere el lenguaje L sobre el alfabeto $\{a, b\}$ tal que $u \in L$ si, y solamente si, u contiene un número impar de símbolos a .
- Encontrar, si es posible, una gramática regular (o, si no es posible, una gramática libre de contexto) que genere el lenguaje L sobre el alfabeto $\{a, b\}$ tal que $u \in L$ si, y solamente si, u no contiene el mismo número de símbolos a que de símbolos b .

Ejercicio 3.1.8. Gramáticas para palabras con restricciones.

- Dado el alfabeto $A = \{a, b\}$, determinar si es posible encontrar una gramática libre de contexto que genere las palabras de longitud impar, y mayor o igual que 3, tales que la primera letra coincida con la letra central de la palabra.
- Dado el alfabeto $A = \{a, b\}$, determinar si es posible encontrar una gramática libre de contexto que genere las palabras de longitud par, y mayor o igual que 2, tales que las dos letras centrales coincidan.

Ejercicio 3.1.9. Regularidad de un lenguaje. Determinar si el lenguaje generado por la gramática $S \rightarrow SS$ $S \rightarrow XXXX \rightarrow aX \mid Xa \mid b$ es regular. Justificar la respuesta.

Ejercicio 3.1.10. Numerabilidad de un lenguaje. Dado un lenguaje L sobre un alfabeto A , ¿es L^* siempre numerable? ¿nunca lo es? ¿o puede serlo unas veces sí y otras, no? Proporcionar ejemplos en este último caso.

Ejercicio 3.1.11. Propiedades de L^* . Dado un lenguaje L sobre un alfabeto A , caracterizar cuándo $L^* = L$. Es decir, dar un conjunto de propiedades sobre L de manera que L cumpla esas propiedades si y sólo si $L^* = L$.

Ejercicio 3.1.12. Igualdad de homomorfismos. Dados dos homomorfismos $f : A^* \rightarrow B^*$, $g : A^* \rightarrow B^*$, se dice que son iguales si $f(x) = g(x)$, $\forall x \in A^*$. ¿Existe un procedimiento algorítmico para comprobar si dos homomorfismos son iguales?

Ejercicio 3.1.13. Lenguajes S_i y C_i . Sea $L \subseteq A^*$ un lenguaje arbitrario. Sea $C_0 = L$ y definamos los lenguajes S_i y C_i , para todo $i \geq 1$, por $S_i = C_{i-1}^+$ y $C_i = S_i^*$.

- ¿Es S_1 siempre, nunca o a veces igual a C_2 ? Justificar la respuesta.
- Demostrar que $S_2 = C_3$, cualquiera que sea L . (Pista: Demostrar que C_2 es cerrado para la concatenación).

Ejercicio 3.1.14. Numerabilidad de lenguajes finitos. Demostrar que, para todo alfabeto A , el conjunto de los

lenguajes finitos sobre dicho alfabeto es numerable.

Bibliografía

- [1] Ismael Sallami Moreno, **Estudiante del Doble Grado en Ingeniería Informática + ADE**, Universidad de Granada, 2025.
- [2] Universidad de Granada, *Diapositivas de la asignatura*, Curso 2025/2026.