Teoría y Práctica

Modelos Computacionales

https://ismael-sallami.github.io

https://elblogdeismael.github.io

Autor: **Ismael Sallami Moreno**

UNIVERSIDAD DE GRANADA

28 de septiembre de 2025

Licencia

Índice general

Índice de figuras

Índice de cuadros

Parte I

Teoría

Capítulo 1

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.

Capítulo 2

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 $u, v, x, y, z, \ldots$.

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 $u = a_1 a_2 \ldots a_n$, 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 $u, v \in A^*$, $u = a_1 \ldots a_n$, $v = b_1 \ldots b_m$, se llama concatenación de u y v a la cadena $u.v$ (o simplemente uv) dada por $a_1 \ldots a_n b_1 \ldots b_m$.

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**: $u.\varepsilon = \varepsilon.u = u$, $\forall u \in A^*$

Definición 2.9 . Monoide: Un monoide es una estructura algebraica $(M, \cdot, e)$ que consta de un conjunto M, una operación binaria asociativa $\cdot : M \times M \to M$, 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: $a \cdot e = e \cdot a = a$, $\forall a \in M$.

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

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

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

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. $L_6 = \{w \in \{0,1\}^* \mid w$ 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. $L_8 = \{w \in \{0,1\}^* \mid w$ 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.

Ejemplo 2.5 . A^* es siempre numerable. Si $A = \{a_1, \ldots, a_n\}$, 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 $b_1 \ldots b_k$ 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.

Demostración 2.1 . 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:

$$L_1L_2 = \{u_1u_2 \mid u_1 \in L_1, u_2 \in L_2\}$$

Ejemplo 2.8 . Si $L_1 = \{0^i1^i \mid i \geq 0\}$ y $L_2 = \{1^j0^j \mid j \geq 0\}$, entonces:

$$L_1L_2 = \{0^i1^i1^j0^j \mid i, j \geq 0\}$$

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**

$$L_1(L_2L_3) = (L_1L_2)L_3$$

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

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

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 2.10 . 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 2.11 . 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 2.12 . Si $L = \{0^i 1^i \mid i \geq 0\}$, 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 $h : A_1^* \to A_2^*$ se dice que es un **homomorfismo** si y solo si:

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

La transformación de uv debe de ser igual a la concatenación de la transformación de u y la transformación de v.

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:

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

Ejemplo 2.13 . Ejemplo de Homomorfismo

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

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

Entonces:

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

2.14.2 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 $h : A_1^* \to A_2^*$ es un homomorfismo, entonces:

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

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

Ejemplo 2.14 . Ejemplo de Composición Sea $h_1 : A_1^* \to A_2^*$ y $h_2 : A_2^* \to A_3^*$ 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 2.15 . 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.

Ejemplo 2.16 . Dada la gramática:

- $E \rightarrow E + E$
- $E \rightarrow E * E$
- $E \rightarrow (E)$

- $E \to a$
- $E \to b$
- $E \to c$

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 $\gamma \to \phi$ 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 ($\alpha \overset{*}{\Rightarrow} \beta$) si y solo si existe una sucesión de palabras $\gamma_1, \ldots, \gamma_n$ ($n \geq 1$) tales que:

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

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:

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

2.18 Gramática Generativa: Ejemplo y Propiedades

2.18.1 Gramática Definida

Sea la gramática $G = (V, T, P, S)$ definida como:
- $V = \{S, A, B\}$
- $T = \{a, b\}$

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

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

Demostración 2.2 . 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 $S \overset{*}{\Rightarrow} u$, tenemos:

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

- $S \to aSbS$
- $S \to bSaS$
- $S \to \varepsilon$

2.19.2 Gramática que Excluye la Palabra Vacía

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

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

2.20 Gramática Generativa: Ejemplo Adicional

2.20.1 Gramática Definida

Sea la gramática $G = (V, T, P, S)$ definida como:
- $V = \{S, X, Y\}$
- $T = \{a, b, c\}$
- $P = \{S \to abc, S \to aXbc, Xb \to bX, Xc \to Ybcc, bY \to Yb, aY \to aaX, aY \to aa\}$
- $S = S$

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**
   - $S \to abc$: Genera la palabra abc para $n = 1$.
2. **Caso General**
   - $S \to aXbc$: 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 $aY \to aa$:

$$aYbbcc \Rightarrow aabbcc$$

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

- 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.

Ejemplo de Derivación para $n = 3$

$$S \Rightarrow aXbc \Rightarrow abXc \Rightarrow abYbcc \Rightarrow aYbbcc \Rightarrow aaXbbcc \Rightarrow aabXbcc$$
$$\Rightarrow aabYbbccc \Rightarrow aaYbbbccc \Rightarrow aaaXbbbccc \Rightarrow aaabbbbccc$$

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.**

Podemos encontrar gramáticas independientes del contexto y dependientes del contexto.

Definición 2.20 (Gramática Independiente del Contexto). Una gramática es independiente del contexto si todas sus reglas de producción tienen la forma $\alpha \rightarrow \beta$, donde $\alpha \in V$ y $\beta \in (V \cup T)^*$. Es decir, el lado izquierdo de cada regla de producción está compuesto por un único símbolo no terminal.

Definición 2.21 (Gramática Dependiente del Contexto). Una gramática es dependiente del contexto si sus reglas de producción tienen la forma $\gamma \rightarrow \beta$, donde $\gamma, \beta \in (V \cup T)^*$ y γ contiene al menos un símbolo no terminal. En este caso, γ puede depender del contexto en el que aparece para ser sustituido por β.

Cuando se inventa un lenguaje, se está inventando un lenguaje independiente del contexto, es lo más lógico naturalmente.

2.20.4 Jerarquía de Chomsky

- 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 $\beta \neq \varepsilon$, en cuyo caso S no aparece a la derecha de las reglas. Es lo que se conoce como lenguaje dependientes del contexto.

Ejemplo es esta regla de producción:

$$Xc \rightarrow Ybcc$$

Donde se ve que para que se de esta regla de producción debe de haber un c antes y después, se puede pensar que es como un invariante.

– Tipo 2: Si cualquier producción tiene la forma

$$A \rightarrow \alpha$$

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

Lenguajes Independientes del Contexto

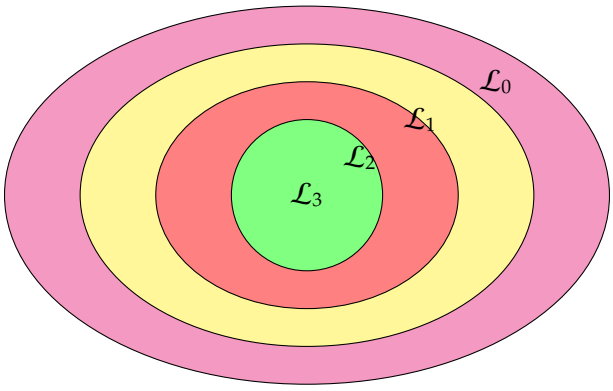
– Tipo 3: Si toda regla tiene la forma

$$A \rightarrow uB \quad \text{o} \quad A \rightarrow u$$

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

Conjuntos Regulares

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



Como podemos observar la familia $\mathcal{L}_0$ es la más amplia, destacando que la $\mathcal{L}_2$ es la de independientes y la de $\mathcal{L}_3$ de los regulares.

Ejercicio Resuelto 2.20.1. *Demostrar que la gramática* $G = (\{S\}, \{a, b\}, \{S \rightarrow \varepsilon, S \rightarrow aSb\}, S)$ *genera el lenguaje* $L = \{a^i b^i \mid i = 0, 1, 2, \ldots\}$.

Demostración 2.3 . La gramática G tiene dos reglas de producción: $S \rightarrow \varepsilon$ y $S \rightarrow aSb$. Procedemos a demostrar que genera el lenguaje L siguiendo los pasos de derivación.

- **Caso Base:** Si aplicamos la regla $S \to \varepsilon$, obtenemos la palabra vacía ε, que pertenece al lenguaje L para $i = 0$.
- **Caso General:** Si aplicamos la regla $S \to aSb$, obtenemos una palabra de la forma aSb. Si continuamos aplicando esta regla, podemos generar palabras de la forma:

  $$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow \ldots$$

  Finalmente, al aplicar $S \to \varepsilon$, obtenemos:

  $$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow \ldots \Rightarrow a^i b^i$$

  donde $i \geq 0$. Por lo tanto, todas las palabras de la forma $a^i b^i$ pertenecen al lenguaje L.
- **Unicidad:** Observemos que cada vez que aplicamos la regla $S \to aSb$, se añade exactamente un símbolo a al principio y un símbolo b al final de la palabra. Esto asegura que el número de a's y b's siempre sea igual. Además, la única forma de terminar la derivación es aplicando $S \to \varepsilon$, lo que garantiza que no se generen palabras adicionales. Por lo tanto, las únicas palabras generadas por G son las de la forma $a^i b^i$, con $i \geq 0$.

Conclusión: La gramática G genera exactamente el lenguaje $L = \{a^i b^i \mid i = 0, 1, 2, \ldots\}$.

Ejercicio Resuelto 2.20.2. *Encontrar el lenguaje generado por la gramática $G = (\{S, A, B\}, \{a, b\}, P, S)$, donde P contiene las siguientes producciones:*

$$\begin{aligned} S &\to aAB \\ B &\to a \\ Ab &\to SBb \\ Aa &\to SaB \\ B &\to SA \\ B &\to ab \end{aligned}$$

Resultado: *El lenguaje generado por esta gramática es el* ***lenguaje vacío****. Esto se debe a que nunca se puede llegar a generar una palabra compuesta únicamente por símbolos terminales.*

Demostración 2.4 . Analicemos las producciones de la gramática:

- La producción inicial $S \to aAB$ introduce las variables A y B.
- Cada vez que se sustituye A, aparece S nuevamente, como se observa en las producciones $Ab \to SBb$ y $Aa \to SaB$.
- De manera similar, cada vez que se sustituye S, aparece A, como se observa en la producción inicial $S \to aAB$.
- Por lo tanto, las variables S y A se refieren mutuamente de forma cíclica, sin posibilidad de reducirse a símbolos terminales.
- Aunque B tiene producciones que incluyen símbolos terminales ($B \to a$ y $B \to ab$), estas producciones no pueden ser utilizadas sin que S o A aparezcan nuevamente en la derivación.

Por lo tanto, no es posible generar ninguna palabra que consista únicamente en símbolos terminales,

lo que implica que el lenguaje generado por la gramática es el lenguaje vacío ($\emptyset$).

Ejercicio Resuelto 2.20.3. *Encontrar una gramática independiente del contexto para generar cada uno de los siguientes lenguajes:*

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

   Solución 2.20.0. . La gramática es:

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

   donde P contiene las siguientes producciones:

   $$S \longrightarrow Sb \mid aSb \mid \varepsilon$$

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

   Solución 2.20.0. . La gramática es:

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

   donde P contiene las siguientes producciones:

   $$S \longrightarrow aSb \mid T$$
   $$T \longrightarrow bTa \mid \varepsilon$$

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

   Solución 2.20.0. . La gramática es:

   $$G = (\{S_1\}, \{a, b\}, P, S)$$

   donde P contiene las siguientes producciones:

   $$S_1 \longrightarrow aS_1b$$
   $$S_1 \longrightarrow \varepsilon$$

   El lenguaje L se puede generar añadiendo $S \longrightarrow S_1S_1$

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

   Solución 2.20.0. . No es posible construir una gramática independiente del contexto para este lenguaje, ya que requiere contar dos cantidades i y j de forma independiente y luego repetirlas en el mismo orden, lo cual excede las capacidades de las gramáticas independientes del contexto.

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

   Solución 2.20.0. . Para no repetir la definición de gramática, se va a obviar.

Las reglas de producción serían:

$$S_1 \to aS_1b$$
$$S_1 \to \varepsilon$$

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

Solución 2.20.0. . La gramática es:

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

donde P contiene las siguientes producciones:

$$S \to aSa \mid bSb \mid \varepsilon$$

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

Solución 2.20.0. . La gramática es:

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

donde P contiene las siguientes producciones:

$$S \to aSc \mid bSc \mid \varepsilon$$

Ejercicio Resuelto 2.20.4. *Determinar si la gramática $G = (\{S,A,B\}, \{a,b,c,d\}, P, S)$, donde P es el conjunto de reglas de producción:*

$$S \to AB$$
$$A \to Ab$$
$$A \to a$$
$$B \to cB$$
$$B \to d$$

genera un lenguaje de tipo 3.

Solución: *Esta gramática genera el lenguaje:*

$$L = \{ab^i c^j d \mid i,j \in \mathbb{N}\}$$

Este lenguaje se puede generar mediante la siguiente gramática:

$$S \to aB$$
$$B \to bB \mid C$$
$$C \to cC \mid d$$

Como esta gramática cumple las restricciones de las gramáticas de tipo 3 (producciones de la forma $A \to uB$

o $A \rightarrow u$, donde $u \in T^*$ y $A, B \in V$), el lenguaje generado es de tipo 3.

Capítulo 3

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:

$$S \to XYX$$

$$X \to aX \mid bX \mid \epsilon$$

$$Y \to bbb$$

Solución 3.1.1 (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:

$$S \to aX$$

$$X \to aX \mid bX \mid \epsilon$$

Solución 3.1.1 (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 3.1.1 (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 3.1.1 (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 $S \to SS$). 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 $S \to XaXaX$ y $X \to bX \mid \epsilon$.
  - Una concatenación de cadenas en R. Si $u, v \in R$, entonces $uv \in L$ porque la regla $S \to SS$ 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:

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

Determinar el lenguaje que genera.

Solución 3.1.2 (Ejercicio 2.a). Cada vez que aplicamos $S \to abAS$ generamos un bloque abA adicional y dejamos un S al final para poder repetir la expansión. Tras m aplicaciones de $S \to abAS$ obtenemos la forma $(abA)^m S$.

Cada bloque abA puede convertirse o bien en $baab$ aplicando la regla $abA \to baab$, o bien en abb aplicando primero $A \to b$ (porque $abA \Rightarrow abb$). Finalmente $S \to a$. 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 $S \to abAS$ se tiene $(abA)^m S$ (prueba por inducción sobre m: base $m = 0$ trivial; paso: si $S \Rightarrow (abA)^k S$ entonces aplicando $S \to abAS$ al S final obtenemos $(abA)^{k+1} S$).

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

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

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

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

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

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

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

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

$$V = \{\langle numero \rangle, \langle digito \rangle\}$$
$$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$
$$S = \langle numero \rangle$$

- $\langle numero \rangle \to \langle numero \rangle \langle digito \rangle$
- $\langle numero \rangle \to \langle digito \rangle$
- $\langle digito \rangle \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Determinar el lenguaje que genera.

Solución 3.1.2 (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:
   - $\langle numero \rangle \to \langle numero \rangle \langle digito \rangle$ permite construir cadenas de longitud arbitraria añadiendo dígitos.
   - $\langle numero \rangle \to \langle digito \rangle$ 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 $\langle digito \rangle \to 0 \mid 1 \mid \ldots \mid 9$.

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:

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

Determinar el lenguaje generado por la gramática.

Solución 3.1.2 (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:
   - $S \to aS$ permite añadir un número arbitrario de a al principio.
   - $S \to aA$ 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:
   - $A \to bA$ permite añadir un número arbitrario de b.
   - $A \to b$ 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.*

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 dos símbolos b consecutivos.

b) 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.*

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 contiene un número impar de símbolos a.

b) 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.*

a) 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.

b) 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 $S \to SS$ $S \to XXX$ $X \to aX \mid Xa \mid b$ 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 $f : A^* \to B^*$, $g : A^* \to B^*$, 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 $C_i = S_i^*$.*

a) ¿Es S_1 siempre, nunca o a veces igual a C_2? Justificar la respuesta.

b) 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.*

3.2 Relación de problemas 1 bis

Ejercicio 3.2.1. *Calcula, de forma razonada, gramáticas que generen cada uno de los siguientes lenguajes:*

- ***SENCILLOS***
  - *a)* $\{u \in \{0,1\}^* \text{ tales que } |u| \leq 4\}$
  - *b) Palabras con 0's y 1's que no contengan dos 1's consecutivos y que empiecen por un 1 y terminen por dos 0's.*
  - *c) El conjunto vacío.*
  - *d) El lenguaje formado por los números naturales.*
  - *e)* $\{a^n \in \{a,b\}^* \text{ con } n \geq 0\} \cup \{a^n b^n \in \{a,b\}^* \text{ con } n \geq 0\}$
  - *f)* $\{a^n b^{2n} c^m \in \{a,b,c\}^* \text{ con } n, m > 0\}$
  - *g)* $\{a^n b^m a^n \in \{a,b\}^* \text{ con } m, n \geq 0\}$
  - *h) Palabras con 0's y 1's que contengan la subcadena 00 y 11.*
  - *i) Palíndromos formados con las letras a y b.*
- ***DIFICULTAD MEDIA***
  - *a)* $\{uv \in \{0,1\}^* \text{ tales que } u^{-1} \text{ es un prefijo de } v\}$

    Solución 3.2.1. media.a.

    $$S \to XY$$
    $$X \to 0X0 \mid 1X1 \mid \varepsilon$$
    $$Y \to 1Y \mid 0Y \mid \varepsilon$$

    Esta gramática no es de tipo 3, ya que las reglas de producción no cumplen con las restricciones de las gramáticas regulares, por ende, es de tipo 2.
  - *b)* $\{ucv \in \{a,b,c\}^* \text{ tales que } u \text{ y } v \text{ tienen la misma longitud}\}$

    Solución 3.2.1. media.b.

    $$S \to c$$
    $$S \to aSa \mid aSb \mid aSc \mid bSa \mid bSb \mid bSc \mid cSa \mid cSb \mid cSc$$
  - *c)* $\{u1^n \in \{0,1\}^* \text{ donde } |u| = n\}$

    Solución 3.2.1. media.c.

    $$S \to 1S1 \mid 0S0 \mid \varepsilon$$

    Esta gramática genera cadenas de la forma $u1^n$, donde $|u| = n$, ya que cada vez que se añade un 1 a la derecha, se añade un símbolo correspondiente a u a la izquierda. La gramática no es de tipo 3, ya que las reglas de producción no cumplen con las restricciones de las gramáticas regulares, por lo que es de tipo 2.

d) $\{a^n b^n a^{n+1} \in \{a, b\}^* \text{ con } n \geq 0\}$

Solución 3.2.1. media.d.

$$S \rightarrow aSb$$
$$S \rightarrow aA$$
$$A \rightarrow aA$$
$$A \rightarrow a$$

Esta gramática genera cadenas de la forma $a^n b^n a^{n+1}$, donde $n \geq 0$. La primera regla $S \rightarrow aSb$ asegura que el número de a y b en las primeras dos partes de la cadena sea igual. La regla $S \rightarrow aA$ transfiere el control a A, que genera la parte final a^{n+1}. La gramática no es de tipo 3, ya que las reglas de producción no cumplen con las restricciones de las gramáticas regulares, por lo que es de tipo 2.

– ***DIFÍCILES***

a) $\{a^n b^m c^k \text{ tales que } k = m + n\}$

Solución 3.2.1. difícil.a.

$$S \rightarrow aSbC \mid \varepsilon$$
$$C \rightarrow aC \mid bC \mid \varepsilon$$

Esta gramática genera cadenas de la forma $a^n b^m c^k$ tales que $k = m + n$. La regla $S \rightarrow aSbC$ asegura que por cada a y b generados, se genera un símbolo adicional en C. La regla $C \rightarrow aC \mid bC \mid \varepsilon$ permite completar la parte final de la cadena con cualquier combinación de a y b.

La gramática no es de tipo 3, ya que las reglas de producción no cumplen con las restricciones de las gramáticas regulares, por lo que es de tipo 2.

b) Palabras que son múltiplos de 7 en binario.

Solución 3.2.1. difícil.b. Para generar palabras que son múltiplos de 7 en binario, podemos construir una gramática que simule un autómata finito determinista (AFD) con 7 estados, donde cada estado representa el residuo de la división del número binario leído hasta el momento entre 7. El estado inicial es el residuo 0, y el estado final también es el residuo 0. Como residuo entendemos el resultado de realizar la operación del módulo 7.

- S: residuo 0 (inicio y final)
- A: residuo 1
- B: residuo 2
- C: residuo 3
- D: residuo 4
- E: residuo 5
- F: residuo 6

$$
\begin{aligned}
S &\to 0S \mid 1A \mid \varepsilon \\
A &\to 0B \mid 1C \\
B &\to 0D \mid 1E \\
C &\to 0F \mid 1S \\
D &\to 0A \mid 1B \\
E &\to 0C \mid 1D \\
F &\to 0E \mid 1F
\end{aligned}
$$

En esta gramática:

- S es el estado inicial (residuo 0).
- A, B, C, D, E, F, G representan los estados correspondientes a los residuos 1, 2, 3, 4, 5, y 6, respectivamente.
- Las reglas de producción simulan las transiciones del AFD según el residuo actual y el siguiente bit leído.

Esta gramática no es de tipo 3, ya que las reglas de producción no cumplen con las restricciones de las gramáticas regulares, por lo que es de tipo 2.

Para entenderlo mejor debemos de tener en cuenta cual es la ecuación que genera el siguiente estado:

Si el número actual (en decimal) es n, al leer un bit $b \in \{0, 1\}$ el nuevo número es:

$$n' = 2n + b.$$

Por tanto, el nuevo residuo es:

$$r' = (2 \cdot r + b) \mod 7,$$

donde r es el residuo actual ($r = n \mod 7$).

Estado	**Residuo** r	**Ejemplo (binario)** n	**al leer 0:** $2n \to$ residuo	**nueva letra**	**al leer 1:** $2n+1 \to$ residuo	**nueva letra**
S	0	ε (vacía)	$2 \cdot 0 = 0 \mod 7 = 0$	S	$2 \cdot 0 + 1 = 1 \mod 7 = 1$	A
A	1	1	$2 \cdot 1 = 2 \mod 7 = 2$	B	$2 \cdot 1 + 1 = 3 \mod 7 = 3$	C
B	2	10	$2 \cdot 2 = 4 \mod 7 = 4$	D	$2 \cdot 2 + 1 = 5 \mod 7 = 5$	E
C	3	11	$2 \cdot 3 = 6 \mod 7 = 6$	F	$2 \cdot 3 + 1 = 7 \mod 7 = 0$	S
D	4	100	$2 \cdot 4 = 8 \mod 7 = 1$	A	$2 \cdot 4 + 1 = 9 \mod 7 = 2$	B
E	5	101	$2 \cdot 5 = 10 \mod 7 = 3$	C	$2 \cdot 5 + 1 = 11 \mod 7 = 4$	D
F	6	110	$2 \cdot 6 = 12 \mod 7 = 5$	E	$2 \cdot 6 + 1 = 13 \mod 7 = 6$	F

- ***EXTREMADAMENTE DIFÍCILES (no son libres de contexto)***

  a) $\{ww \text{ con } w \in \{0, 1\}^*\}$

  Solución 3.2.1. extremadamente difícil.a. El lenguaje $\{ww \mid w \in \{0, 1\}^*\}$ no es libre de

contexto, ya que requiere comparar dos partes arbitrariamente largas de una cadena para verificar que son iguales. Esto se puede demostrar utilizando el lema de bombeo para lenguajes libres de contexto.

Sin embargo, podemos describir una gramática de tipo 0 (gramática general) que genera este lenguaje. Una posible gramática es:

$$S \to X\#X$$
$$X \to 0X0 \mid 1X1 \mid \epsilon$$

En esta gramática:

- $S \to X\#X$ asegura que la cadena generada tiene la forma $w\#w$, donde w es una cadena de ceros y unos.
- $X \to 0X0$ y $X \to 1X1$ generan cadenas de ceros y unos de forma recursiva, asegurando que las dos partes de la cadena son idénticas.
- $X \to \epsilon$ permite terminar la generación de la cadena.

El símbolo # es un marcador que separa las dos partes de la cadena. Este lenguaje no es regular ni libre de contexto, pero puede ser generado por una gramática de tipo 0.

b) $\{a^{n^2} \in \{a\}^* \text{ con } n \geq 0\}$

Solución 3.2.1. extremadamente difícil.b. El lenguaje $\{a^{n^2} \mid n \geq 0\}$ no es libre de contexto, ya que requiere contar el número de símbolos a y verificar que este número es un cuadrado perfecto. Esto no puede lograrse con una gramática libre de contexto.

Sin embargo, podemos describir una gramática de tipo 0 (gramática general) que genera este lenguaje. Una posible gramática es:

$$S \to A\#A$$
$$A \to aA \mid \epsilon$$

Esta gramática utiliza un marcador # para separar las partes de la cadena y asegura que el número total de a generados corresponde a un cuadrado perfecto. La gramática no es regular ni libre de contexto, pero puede ser generada por una gramática de tipo 0.

c) $\{a^p \in \{a\}^* \text{ con } p \text{ primo}\}$

Solución 3.2.1. extremadamente difícil.c. El lenguaje $\{a^p \mid p \text{ es primo}\}$ no es libre de contexto ni regular, ya que requiere verificar que el número de símbolos a es un número primo, lo cual no puede lograrse con una gramática libre de contexto.

Sin embargo, podemos describir una gramática de tipo 0 (gramática general) que genera este lenguaje. Una posible gramática es:

$$S \rightarrow aS \mid P$$
$$P \rightarrow a^2 \mid a^3 \mid a^5 \mid a^7 \mid \ldots$$

En esta gramática:

- S genera cadenas de longitud arbitraria.
- P genera cadenas cuya longitud corresponde a un número primo.

La gramática no es regular ni libre de contexto, pero puede ser generada por una gramática de tipo 0. Podemos ver que es infinita, por ende podemos concluir con que aunque el lenguaje $\{a^p \mid p \text{ primo}\}$ no es regular y por tanto no es libre de contexto, es decidible[1], por lo que pertenece a la clase recursiva; existe por tanto alguna gramática tipo-0 o máquina de Turing[2] que lo reconozca, pero no hay una gramática regular ni libre de contexto que lo genere.

d) $\{a^n b^m \in \{a, b\}^* \text{ con } n \leq m^2\}$

Solución 3.2.1. extremadamente difícil.d. El lenguaje $\{a^n b^m \mid n \leq m^2\}$ no es libre de contexto, ya que requiere comparar dos partes de una cadena y verificar que una es menor o igual al cuadrado de la otra. Esto no puede lograrse con una gramática libre de contexto.

Sin embargo, podemos describir una gramática de tipo 0 (gramática general) que genera este lenguaje. Una posible gramática es:

$$S \rightarrow aSbb \mid \epsilon$$

En esta gramática:

- $S \rightarrow aSbb$ asegura que por cada símbolo a generado, se generan dos símbolos b, lo que garantiza que $n \leq m^2$.
- $S \rightarrow \epsilon$ permite terminar la generación de la cadena.

La gramática no es regular ni libre de contexto, pero puede ser generada por una gramática de tipo 0.

[1] Un lenguaje es decidible si existe un algoritmo que, dado cualquier elemento del alfabeto, puede determinar en un tiempo finito si pertenece o no al lenguaje.

[2] Una máquina de Turing es un modelo computacional abstracto que consiste en una cinta infinita dividida en celdas, un cabezal de lectura/escritura que se mueve sobre la cinta, y un conjunto de estados que determinan las transiciones basadas en el símbolo leído. Es capaz de simular cualquier algoritmo y se utiliza para definir formalmente la computabilidad.

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.