# TECANA AMERICAN UNIVERSITY

**Bachelor of Science in Computer Science**

INFORME IV

*Teoría de la Computación*

*Por la presente, doy fe que soy la única autora de esta investigación, y que su contenido es consecuencia de mi trabajo académico.*

Eugenia Bahit

Buenos Aires, 1 de Junio de 2021

# ÍNDICE DE CAPÍTULOS

# ÍNDICE DE TABLAS



# ÍNDICE DE GRÁFICOS



**AVISO DE COPYRIGHT:** Excepto donde se indique lo contrario, todo el contenido de este trabajo, incluidas las tablas y gráficos, son de elaboración propia.

# INTRODUCCIÓN Y OBJETIVOS

En el estudio teórico de la computación confluyen tres teorías: la teoría de autómatas, que establece modelos matemáticos para procesos computacionales, la teoría de la computabilidad, que intenta responder qué tan factible es o no resolver un problema, y la teoría de la complejidad qué intenta responder a qué tan difícil es resolver un problema computacionalmente. El objetivo de este trabajo es ofrecer una visión general de los principales temas que conforman estas tres teorías, con el fin futuro de servir de base de conocimiento para su estudio en profundidad.

Para alcanzar este objetivo, se dividirá el trabajo en cinco capítulos con los siguientes objetivos específicos:

- Capítulo I: ofrecer las bases matemáticas para la comprensión y estudio de las teorías de autómatas, de la computabilidad, y de la complejidad.
- Capítulo II: definir las bases formales con las que las teorías de autómatas, de la computabilidad, y de la complejidad operan y son descritas.
- Capítulo III: entender cómo la Teoría de Autómatas influye en el procesamiento de la información.
- Capítulo IV: conocer el objeto de estudio de la Teoría de la Computabilidad.
- Capítulo V: conocer el objeto de estudio de la Teoría de la Complejidad.

# CAPÍTULO I

# FUNDAMENTOS MATEMÁTICOS

Las teorías de autómatas, de la computabilidad, y de la complejidad en las que se fundamenta gran parte de la informática teórica, tienen como base las matemáticas discretas. A fin de facilitar la comprensión de los temas venideros, este capítulo se enfocará en las teorías de conjuntos, de funciones, y en el principio de inducción matemática, áreas de estudio principales de la mencionada base. Se hace notar que a pesar de ser un tema clave en la Teoría de Autómatas, no se abarca la teoría de gráficos por restricciones impuestas para la presentación del informe.

## Teoría de conjuntos

La teoría de conjuntos define tanto las colecciones de elementos, como sus relaciones y operaciones. En el contexto de la informática teórica se utiliza como base de la descripción formal de cualquier colección de elementos independientemente del contexto en el cual se describa. Los conceptos fundamentales se definen a continuación.

**CONJUNTOS Y ELEMENTOS.** Un ***conjunto*** es una colección de objetos distintos y bien definidos. Por ***elementos*** se hace referencia a cada uno de los objetos miembros de un conjunto. Es frecuente denotar a los conjuntos por letras mayúsculas y a sus elementos por letras minúsculas, tal que $a$ es el elemento mientras que $A$ el conjunto.

**PERTENENCIA.** Cualquier elemento que se encuentra dentro de un conjunto, se dice que pertenece a dicho conjunto y se denota como $a \in A$. Para denotar lo contrario (que $a$ no pertenece al conjunto $A$) se emplea el símbolo de no pertenencia tal que $a \notin A$.

**DEFINICIÓN DE CONJUNTOS.** Los conjuntos pueden definirse en dos formas posibles:

1. *Forma tabular:* donde el conjunto se define como una lista de sus elementos. Por ejemplo, el conjunto de los ocho primeros números de la serie de Fibonacci se denotaría como: $\{1, 1, 2, 3, 5, 8, 13, 21\}$.

2. *Forma de construcción del conjunto:* donde se define el conjunto a partir de la forma en la que éste se construye, es decir, describiendo sus propiedades características. Cualquier conjunto puede definirse a través de una descripción verbal de sus propiedades (por ejemplo: $A = \{a | a \text{ es un número natural mayor que } 5\}$) o de forma más simbólica (como por ejemplo: $A = \{a | a \in \mathbb{N} \wedge a > 5\}$, que se lee como *"'a' pertenece al conjunto de los números naturales [ $\mathbb{N}$ ] y es mayor que cinco"*). Notar que las anotaciones $x|x$ y $x:x$ son equivalentes, y se lee "equis tal que equis...". Por ejemplo, $\{x | x \in A\}$ puede ser leído como *"equis, tal que equis pertenece al conjunto A"*.

**IGUALDAD.** Dos conjuntos son iguales si comparten exactamente los mismos miembros. Por ejemplo, los siguientes conjuntos son iguales dado que en un conjunto no importa el orden de sus elementos y los elementos repetidos son en realidad un mismo elemento: $\{1,3,5\}=\{5,1,3\}=\{1,1,3,5,5,3\}$ (aquí el último conjunto es en realidad el conjunto formado por los números 1, 3 y 5). Se dice entonces, que $A$ y $B$ son iguales se comparten todos sus elementos. En caso contrario, se dice que $A$ y $B$ son desiguales. Para $A=\{1,2,3\}$, $B=\{3,1,2\}$ y $C=\{2,3,4\}$, $A=B$ y $A\neq C$, $B\neq C$.

**CONJUNTOS DISJUNTOS.** Dos o más conjuntos que no tienen en común ningún elemento. Para $A=\{1,2,3\}$ y $B=\{9,7\}$ se dice que $A$ y $B$ son disjuntos.

**SUBCONJUNTO.** Es cualquier conjunto cuyos elementos se encuentran además en otro conjunto. Si todos los elementos de un conjunto están dentro de otro, se dice que el primero es subconjunto del segundo. Para $A=\{1,2,3\}$ y $B=\{3,1,2,4\}$ se dice que $A\subset B$ ( $A$ es subconjunto de $B$ ) y para $A=\{1,2,3,4\}$ y $B=\{3,1,2,4\}$ se dice que $A\subseteq B$ ( $A$ es subconjunto igual de $B$ ). Para decir lo mismo pero desde la perspectiva del conjunto "que contiene" al otro (desde la perspectiva de $B$ ) se dice que $B\supset A$ ( $B$ es el superconjunto de $A$ o $B$ contiene a $A$ ). Para $A\subset B$ no existe igualdad entre A y B, mientras que sí la existe para $A\subseteq B$. Cuando no existe igualdad se dice que $A$ es un subconjunto propio de $B$.

**CONJUNTO VACÍO.** Es aquel conjunto que no tiene ningún elemento. Se define como $\{\}$ y se denota por $\emptyset$. El conjunto vacío es subconjunto de todos los conjuntos.

**CONJUNTO UNIVERSAL.** Es aquel conjunto que se compone de todos los conjuntos existentes, y se denota por $U$.

**CARDINALIDAD.** La cardinalidad de un conjunto es la cantidad de elementos que tiene dicho conjunto. Para un conjunto $A=\{a,b,c\}$, su cardinalidad se denota por $|A|$ y es igual a $3$. Dado que los conjuntos con un número infinito de elementos —denominados *conjuntos infinitos*—, al igual que aquellos finitos pueden tener un tamaño variable, también poseen una cardinalidad determinada por un número denominado *número cardinal*. Por el *Teorema de Schröder-Bernstein*, para dos conjuntos infinitos $A$ y $B$, si $|A|\leq|B|$ y $|B|\leq|A|$, entonces $|A|=|B|$.

**CONJUNTOS NUMERABLES Y NO NUMERABLES.** Se dice que un conjunto es numerable cuando o bien es un conjunto finito, o bien existe una correspondencia uno a uno desde los elementos del conjunto de los números naturales ($\mathbb{N}$) a este. Un conjunto no es numerable en caso contrario.

## Operaciones sobre los conjuntos

**UNIÓN.** La unión de dos conjuntos es el resultado de incluir el total de elementos de cada conjunto en uno nuevo. La unión de dos conjuntos $A$ y $B$ se define como: $A\cup B=\{x|x\in A\vee x\in B\}$.

**INTERSECCIÓN.** La intersección de dos conjuntos es el conjunto formado por los elementos en común de dos o más conjuntos. La intersección de dos conjuntos $A$ y $B$ se define como: $A\cap B=\{x|x\in A\wedge x\in B\}$.

**DIFERENCIA.** La diferencia de dos conjuntos es el conjunto formado por aquellos elementos de un conjunto que no se encuentran presentes en el otro conjunto. La diferencia de dos conjuntos $A$ y $B$ se define como: $A-B=\{x|x\in A\wedge x\notin B\}=A-(A\cap B)$ .

**COMPLEMENTO.** El complemento de un conjunto es el resultado de la diferencia entre el conjunto universal y los elementos del conjunto. El complemento de un conjunto $A$ se define como $\bar{A}=U-A$ . El complemento de un conjunto también es denotado por $A'$ .

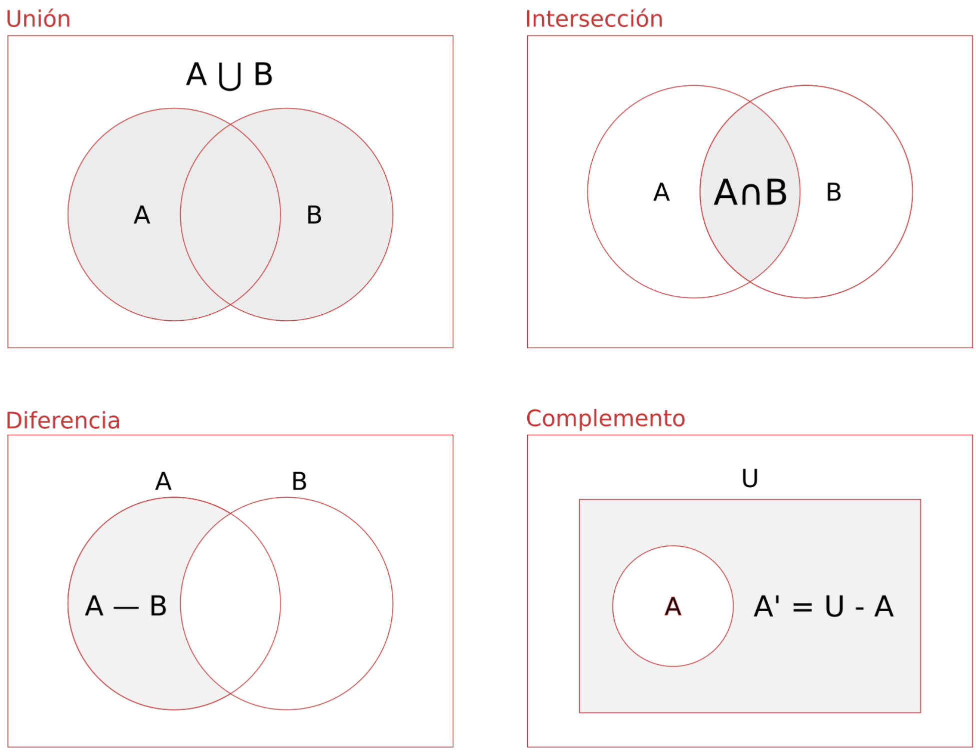


***Gráfico 1:*** *Diagramas de Venn de operaciones con conjuntos*

**POTENCIA.** La potencia de un conjunto es el conjunto de todos sus subconjuntos. Para un conjunto A, su potencia puede ser denotada tanto como $2^A$ como $P(A)$ . Dado que la cardinalidad de un conjunto es $|A|$ , la cardinalidad de la potencia de $P(A)$ queda determinada por $|P(A)|=2^{|A|}$. Así, para un conjunto $A$ con cardinalidad $4$, la cardinalidad de la potencia será $2^4=16$. Por ejemplo, la potencia de un conjunto $A=\{a,b\}$ es $P(A)=\{\emptyset,\{a\},\{b\},\{a,b\}\}$.

**PRODUCTO CARTESIANO.** El producto cartesiano de dos conjuntos es la asociación de cada uno de los elementos de un conjunto con los del otro, donde cada par de elementos es un par ordenado. El producto cartesiano de dos conjuntos A y B, se define como $A\times B=\{(a,b) \mid a\in A\wedge b\in B,\forall a\wedge\forall b\}$. Lo anterior se lee como sigue: *el producto cartesiano de 'A y B' es el conjunto de tuplas 'a b' tal que 'a' pertenece al conjunto 'A' y 'b' pertenece al conjunto 'B', para todo elemento 'a' y para todo elemento 'b'*. Por ejemplo, el producto cartesiano de los conjuntos $A=\{a,b\}$ y $B=\{1,2\}$ es $A\times B=\{(a,1),(a,2),(b,1),(b,2)\}$.

**CONCATENACIÓN.** La concatenación de dos conjuntos es el resultado de unir cada elemento de un conjunto con cada elemento del otro conjunto. Para dos conjuntos $A$ y $B$, la concatenación de sus elementos se define como $A\circ B=\{x|x=ab,\ \forall a\in A\ \wedge\ \forall b\in B\}$. El proceso de unión de los elementos es similar al del producto cartesiano pero el resultado de la unión no es una tupla de dos elementos sino un único elemento. Así, mientras que el producto cartesiano para dos elementos $a$ y $b$ es $(a,b)$, la concatenación es $ab$.

## Teoría de funciones

**FUNCIÓN.** Dados dos conjuntos ( $A$ y $B$ ), si existe una ley matemática (es decir, una *regla*) que permita que a cada elemento del primero le corresponda un único elemento del segundo, dicha regla se denomina función, mapa, transformación u operación (indistintamente) "de $A$ en $B$ " y se denota por $f$ , tal que $f: A \to B$ representa la *función de A en B*.

> *Una **función** es una regla que asigna a cada elemento de un conjunto A un único elemento de un conjunto B".*
> *—Marilina Carena[F01]*

**IMAGEN.** Se denomina imagen a cada uno de los elementos correspondidos (cada $b \in B$ correspondido por un elemento $a \in A$ ). Así, $b$ es la imagen de $a$ , denotada como $f(a)$ (leído como *"efe de 'a'"*) y $a$ es la *preimagen* de $b$ . Esto significa que $b$ es el resultado de aplicar $f$ a $a$ .

**VARIABLES DEPENDIENTES E INDEPENDIENTES.** Se denomina variable dependiente a la imagen ( $b$ ) e independiente a la preimagen ( $a$ ).

**DOMINIO, CODOMINIO Y RANGO.** Se denomina dominio al conjunto de elementos para los que existe correspondencia, y codominio, al conjunto de los elementos que son correspondidos. Así, $A$ es el dominio de la función

$f: A \rightarrow B$ mientras que $B$, el codominio. Se denomina rango al conjunto de imágenes de la función, $f(A)$.

**FUNCIONES K-ARIAS.** Cuando una función se define sobre más de una variable independiente, se dice que es *k*-aria. Así, la función $f: A_1 \times \ldots \times A_k \rightarrow B$ con imagen $f(a_{1,} \ldots, a_k)$, se dice *k*-aria. Es frecuente denotar funciones *k*-arias por $f^{(k)}$. Si $k=1$ se dice función unaria, si $k=2$, función binaria, si $k=3$, función ternaria, y así sucesivamente.

**FUNCIONES TOTALES Y PARCIALES.** Una función es total cuando está definida para todos los elementos del dominio, y parcial, cuando solo se define para algunos.

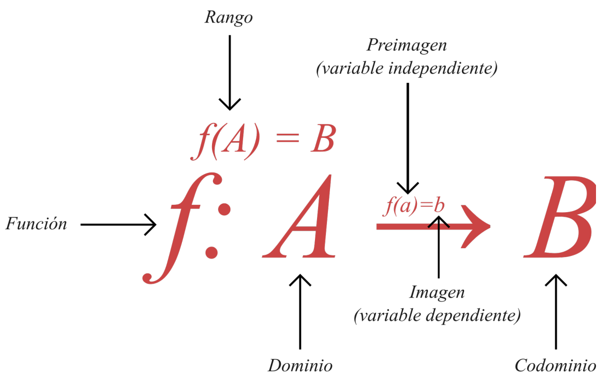


***Gráfico 2:*** *Partes de una función matemática.*

Las funciones, también pueden ser vistas desde una perspectiva informática como *relaciones de entrada y salida*, donde $f$ es un objeto que recibe una variable independiente como entrada y produce una variable dependiente como salida. Se trata de la misma definición vista desde una perspectiva diferente. Por lo tanto, también pueden ser descriptas en diferentes formas, no solo como en la imagen previa: también puede emplearse una tabla, como en el siguiente ejemplo. Para una función $f: A \rightarrow B$ donde $A=\{1,2,3\}$ y $B=\{10,20,30\}$ (para $B$ cumpliendo la regla: $B=\{b | b=a*10, \forall a \in A\}$ ), puede describirse $f$ mediante la siguiente tabla:

***Tabla 1:*** *Descripción de la función f:A→B*

| *a* | *f(a)=b* |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 3 | 30 |

**FUNCIÓN IDENTIDAD.** Aquella función para la cual se cumple $f(x)=x, \quad \forall x \in X$ . Es decir, una función cuyo dominio y codominio son idénticos.

**FUNCIÓN CARACTERÍSTICA.** Para dos conjuntos $A$ y $B$ , siendo $B \subseteq A$ , la función $f: A \rightarrow \{0,1\}$ es *función característica para* $B$ , si para todo $a \in A$ se cumple que $f(a)=1$ cuando $a \in B$ , y $f(a)=0$ si $a \notin B$ .

## Principio de inducción matemática

La inducción matemática es un método empleado para demostrar formalmente —y con certeza lógica— que un teorema se cumple para un número infinito de casos. De esta forma, una declaración $S$ sobre cualquier número natural $n$

(denotada como $S(n)$), es verdadera para todo $n \in \mathbb{N}$. Este principio establece que:

1. $S(1)$ es verdadero.
2. Que si $S(k)$ es verdadero para $k=n$, y $S(k+1)$ es también verdadero, entonces $S(n)$ será verdadero para cualquier valor de $n \in \mathbb{N}$.

Dicho de otro modo, si $A$ es un subconjunto de enteros positivos para el que se cumple que $1 \in A$, se asume como verdadero que $a \in A \Rightarrow a+1 \in A$, y por lo tanto, $A = \mathbb{N}$.

El método inductivo propone demostrar entonces, que si una declaración es válida para $n$ lo es también para $n+1$, siguiendo tres pasos:

***Base de la inducción:*** prueba que la declaración sea verdadera para el número $n$ más bajo posible (comúnmente, $n=0$ o $n=1$).

***Hipótesis inductiva:*** se asume que $S(n)$ es verdadera para algún valor de $n$, y $n=k$.

***Paso inductivo:*** prueba que la declaración también es verdadera cuando $n=k+1$.

Lo anterior puede verse en el ejemplo descrito a continuación. En el mismo, se intentará demostrar que el siguiente teorema es válido para cualquier $n$.

**TEOREMA:** $4+9+14+19+\ldots+(5n-1)=\frac{n}{2}(3+5n)$

**BASE:** Se toma $1$ como menor valor posible de $n$ y se sustituye en todas sus apariciones, a partir del término general de la progresión aritmética ( $5n-1$ ): $((5x1)-1)=\frac{1}{2}(3+(5x1))$ . Luego, se resuelve la ecuación a ambos lados:

$$
\begin{gathered}
((5)-1)=\frac{1}{2}(3+(5)) \\
4=\frac{1}{2}(8) \\
4=\frac{8}{2} \\
4=4
\end{gathered}
$$

Y así se demuestra que el teorema es válido para $n=1$ .

**HIPÓTESIS:** El teorema se asume verdadero para $n=k$ :

$$4+9+14+19+\ldots+(5k-1)=\frac{k}{2}(3+5k)$$

**INDUCCIÓN:** se debe demostrar que el teorema también es válido cuando sea $n=k+1$ , es decir, no solo para $(5k-1)$ , sino también, para $(5(k+1)-1)$ . Se demostrará entonces $4+9+14+19+\ldots+(5k-1)+(5(k+1)-1)$ . Para ello, se comienza sustituyendo todas las apariciones de $k$ por $k+1$ a la derecha para resolver ambos lados:

$$4+9+14+19+\ldots+(5k-1)+(5(k+1)-1)=\frac{k+1}{2}(3+5(k+1))$$

Por la base, se sabe que $(5n-1)=\frac{n}{2}(3+5n)$ y por la hipótesis, se sabe que $n=k$ , por lo que se reemplaza $(5k-1)$ (el equivalente a $(5n-1)$ ) por:

$\frac{k}{2}(3+5k)$ obteniendo: $(\frac{k}{2}(3+5k))+(5(k+1)-1)=\frac{k+1}{2}(3+5(k+1))$ .

Se resuelve entonces a partir de allí, comenzando por el primer término de la izquierda. Se multiplica entonces $\frac{k}{2}(3)=\frac{3k}{2}$ y $\frac{k}{2}(5k)=\frac{5k^2}{2}$ (porque $k \cdot k=k^2$ ) , y luego el segundo, multiplicando $5 \cdot k=5k$ , $5 \cdot 1=5$ que $-1=4$ . Se obtiene entonces:

$$\frac{3k}{2}+\frac{5k^2}{2}+5k+4=\frac{k+1}{2}(3+5(k+1))$$

Se realiza ahora la suma para lo que se iguala primero el denominador para $5k$ como $10k/2$ ya que $10k/2=5k/1$ , y se suman los términos de mismo grado obteniendo:

$$\frac{13k}{2}+\frac{5k^2}{2}+4=\frac{k+1}{2}(3+5(k+1))$$

Se resuelve ahora el lado derecho, multiplicando primero el paréntesis y luego sumándolo, donde $5 \cdot 1k=5k$ , $5 \cdot 1=5$ , y luego, $3+5=8$ :

$$\frac{13k}{2}+\frac{5k^2}{2}+4=\frac{k+1}{2}(5k+8)$$

Se multiplica ahora $k+1/2$ por $5k$ y luego por $8$ para finalmente poder multiplicar los términos de forma independiente:

$$\frac{13k}{2}+\frac{5k^2}{2}+4=\frac{5k(k+1)}{2}+\frac{8(k+1)}{2}$$

Multiplicando los términos de forma independiente y manteniendo el denominador, se obtiene que $5k \cdot k=5k^2$ más $5k \cdot 1=5k$ , y que $8 \cdot k=8k$

más $8 \cdot 1 = 8$ . Ahora, se pueden simplificar los dos primeros términos (donde $8/2 = 4$ ). Sin embargo, se mantiene el primero (para poder sumarlo al último, del mismo grado, donde $8k + 5k = 13k$ ) y solo se simplifica el segundo, obteniendo que:

$$\begin{aligned}\frac{13k}{2} + \frac{5k^2}{2} + 4 &= \frac{8k}{2} + \frac{8}{2} + \frac{5k^2}{2} + \frac{5k}{2} \\ &= \frac{8k}{2} + 4 + \frac{5k^2}{2} + \frac{5k}{2} \\ &= \frac{13k}{2} + 4 + \frac{5k^2}{2}\end{aligned}$$

Por la propiedad conmutativa de la suma, se reorganizan los términos(pasando el $4$ al final) y se obtiene una igualdad, demostrando la validez formal del argumento, es decir, demostrando que el teorema es válido también para $k+1$ .

# CAPÍTULO II

# TEORÍA DE LENGUAJES FORMALES

La teoría de lenguajes formales constituye la base de la teoría de autómatas. Este capítulo es una breve exposición de la misma y tiene por objetivo, al igual que el capítulo anterior, facilitar la comprensión de los temas venideros.

## Elementos de la teoría de lenguajes formales

**ALFABETO, SÍMBOLOS Y CADENA.** Un *alfabeto* $\Sigma$ es un conjunto finito de elementos. Un elemento de un alfabeto es un símbolo. Un *símbolo* es una entidad abstracta que no puede ser definida formalmente. Una *cadena* $\alpha$ es una secuencia ordenada de símbolos de un alfabeto $\Sigma$. Dependiendo de la naturaleza de su estudio, una cadena también puede ser denominada, *palabra* o *sentencia*. Una cadena de $n$ símbolos es una cadena de longitud $n$, o lo que es igual una cadena $|\alpha|$. Una cadena cualquiera se denota genéricamente por $\alpha$, mientras que una cadena concreta se denota por una de las últimas letras minúsculas del alfabeto latino: $u, v, w, x, y, z$.

**CADENA VACÍA.** Una *cadena vacía* se denota por $\epsilon$ (épsilon)[1] y se define como una cadena de longitud cero, tal que $|\epsilon|=0$.

**PREFIJOS.** Es prefijo de una cadena, cualquier secuencia de símbolos desde el inicio de la cadena, incluidos $\epsilon$ y la propia cadena. Se denomina *prefijo propio*

1 Obsérvese que algunos/as autores/as denotan la cadena vacía por $\lambda$ (lambda).

*de la cadena* a cualquier secuencia de símbolos desde el inicio de la cadena, exceptuando a $\alpha$ .

**SUFIJOS.** Es sufijo de una cadena, cualquier secuencia de símbolos hacia el final de la cadena, incluidos $\epsilon$ y la propia cadena. Se denomina *sufijo propio de la cadena* a cualquier secuencia de símbolos hacia el final de la cadena, exceptuando a $\alpha$ .

**LENGUAJE.** Un *lenguaje* $L$ es el conjunto de todas las cadenas de un alfabeto. Formalmente, se denomina lenguaje $L$ al conjunto de cadenas sobre un alfabeto $\Sigma$ , tal que $\Sigma^* = \{\alpha | \alpha_1, \alpha_2 \ldots, \alpha_n\}$ . Dado un alfabeto $\Sigma$ , un lenguaje $L$ se definirá entonces como cada uno de los subconjuntos de $\Sigma^*$ (leído «cerradura estrella de sigma»), donde $\Sigma^*$ es el conjunto de todas las cadenas $\alpha \in \Sigma$ , y $\Sigma^+$ (leído como «cerradura positiva de sigma») es la diferencia entre $\Sigma^* - \epsilon$ .

**GRAMÁTICA.** Una *gramática* es un conjunto finito de reglas que permiten generar cadenas sintácticamente válidas. Una gramática $G$ es una 4-tupla (*cuádrupla*) de la forma $(V, T, P, S)$ donde:

$V$ es un conjunto finito no vacío de *variables* denominadas *no terminales* (por ello a veces se denota por $N$ ), y en el que se cumple que $V \cap T = \emptyset$ .

$T$ es un conjunto finito no vacío de elementos llamados *terminales*. A veces denotado por $\Sigma$ .

$P$ es un conjunto finito de reglas sintácticas con la forma $\alpha \rightarrow \beta$, donde $\alpha, \beta \subset (V \cup T)^*$ ( $\{V \cup T\}$ se conoce como vocabulario de $L$ ), $\alpha \neq \epsilon$ y $\alpha$ tiene al menos un elemento de $V$.

$S$ es un símbolo llamado símbolo de inicio para el que se cumple $S \in V$.

**GRAMÁTICA INDEPENDIENTE DEL CONTEXTO.** Es una gramática para la que se cumple que $P$ tiene la forma $A \rightarrow \alpha$, para $A$ siendo una no terminal cualquiera, y $\alpha$ una forma sentencial.

La siguiente tabla (tabla 2) resume los conceptos precedentes.

***Tabla 2:*** *Resumen de definiciones formales y denotación de elementos de la Teoría de Autónomas*

| | SÍMBOLO | DEFINICIÓN FORMAL O DENOTACIÓN |
|---|---|---|
| CADENA | $\alpha$ | … |
| CADENA VACÍA | $\epsilon$ | $\epsilon = \emptyset$ |
| LENGUAJE | $L$ | $\Sigma^* = \{\alpha \vert \alpha_1, \alpha_2 \ldots, \alpha_n\}$ |
| GRAMÁTICA | $G$ | $G = \{V, T, P, S\}$<br>$P = \alpha \rightarrow \beta$ |
| GRAMÁTICA INDEPENDIENTE DEL CONTEXTO | $G$ | $G = \{V, T, P, S\}$<br>$P = A \rightarrow \alpha$ |

La teoría de lenguajes formales es un desarrollo posterior a las teorías que hoy, dependen de ellas. Esto implica que al momento de proponer los autómatas y las máquinas de Turing, se hizo necesario el desarrollo de teorías que permitiesen validar científicamente dichos conceptos.

## CAPÍTULO III

# TEORÍA DE AUTÓMATAS

El objetivo de este capítulo es abordar el funcionamiento de los autómatas finitos partiendo de una base de conocimientos previos ya consolidada. Para ello, el capítulo será divido en tres etapas: una primera etapa introductoria donde se presentarán en lenguaje natural, las definiciones de los conceptos básicos del modo más categórico posible; en la segunda etapa, se presentarán las definiciones formales de los conceptos introducidos; y finalmente, tras consolidar los conocimientos tanto a nivel conceptual como formal, se pondrá el foco en el funcionamiento de los autómatas y su relación con el capítulo precedente.

## Definiciones previas

**AUTÓMATAS. AUTÓMATAS FINITOS.** Los *autómatas* son modelos computacionales. Un modelo es un esquema teórico (abstracto) elaborado para comprender y estudiar el comportamiento de un sistema complejo. Computacional, se refiere a que dicho sistema es un sistema que lleva a cabo acciones de cómputo. Se puede inferir entonces, que un ***autómata*** es un esquema teórico que permite estudiar y comprender el comportamiento de un sistema de cómputo complejo. Dado que la capacidad de memoria de los sistemas de cómputo es limitada, se dice que los autómatas son ***autómatas finitos***, pues definen modelos para ordenadores con una capacidad limitada de memoria. En términos de aplicación, los autómatas finitos son los modelos teóricos empleados para el desarrollo de circuitos secuenciales[2].

**ESTADOS Y TRANSICIONES. FUNCIÓN DE TRANSICIÓN.** En un modelo básico, cada entrada (*input*) es transformada secuencialmente en una salida (*output*). El proceso de transformación de una entrada en una salida se denomina *transición,* y es llevado a cabo por una función conocida como *función de transición*. El resultado de la ejecución de dicho función es lo que se conoce como *estado*.

**AUTÓMATAS FINITOS DETERMINISTAS Y NO DETERMINISTAS.** Los autómatas son modelos computacionales que toman decisiones a partir de unos datos de entrada. Esas decisiones, se traducen en cambios de estado. Cuando a partir de una entrada el autómata debe decidir entre múltiples opciones, significa que a partir de una entrada puede derivar en diferentes estados. En cambio, si por cada entrada solo puede arribar a un único estado, significa que solo existe una opción (la decisión entonces, no requiere "optar", pues existe una —y solo una— opción posible. Un *autómata determinista* es aquel autómata finito que para cada entrada tiene asignado un único estado de transición posible, mientras que un *autómata no determinista*, es aquel autómata finito que por cada entrada, tiene asignadas dos o más estados de transición entre los cuales decidir.

**DIAGRAMAS DE ESTADO.** Un diagrama de estado es un grafo empleado para representar los diferentes estados de un autómata.

---

2 Se refiere a los circuitos secuenciales abarcados en *«Informe II: Teoría de conmutación de circuitos y diseño lógico»* (Bahit, E., 2020): https://tauniversity.org/sites/default/files/informe_2_ok_pdf_eugenia_bahit.pdf

## Definiciones formales

**AUTÓMATA FINITO. AUTÓMATA FINITO DETERMINISTA (AFD).** Es una quíntupla $(Q, \Sigma, \delta, q_{0,} F)$ donde:

$Q$ es un conjunto finito de estados.

$\Sigma$ es un alfabeto.

$\delta$ es la función de transición $\delta: Q \times \Sigma \rightarrow Q$ .

$q_0$ es el estado de inicio donde $q_0 \in Q$ .

$F$ es el conjunto de estados finales (o estados de aceptación) donde $F \subset Q$ .

**AUTÓMATA FINITO NO DETERMINISTA (AFN).** Es una quíntupla $(Q, \Sigma, \delta, q_{0,} F)$ donde respecto al AFD solo cambia la función de transición por:

$\delta$ es la función de transición $\delta: Q \times \Sigma_\epsilon \rightarrow P(Q)$ donde $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ .

***Tabla 3:** Resumen de definiciones formales y denotación de elementos de la Teoría de Autónomas*

| | SÍMBOLO | DEFINICIÓN FORMAL O DENOTACIÓN |
|---|---|---|
| **AUTÓMATA FINITO DETERMINISTA** | $M$ | $M = \{Q, \Sigma, \delta, q_{0,} F\}$ <br> $\delta: Q \times \Sigma \rightarrow Q$ |
| **AUTÓMATA FINITO DETERMINISTA** | $M$ | $M = \{Q, \Sigma, \delta, q_{0,} F\}$ <br> $\delta: Q \times \Sigma_\epsilon \rightarrow P(Q)$ |

En la siguiente etapa, se esclarecerá la forma en la que los conceptos precedentes se relacionan, explicando el funcionamiento de los mismos y empleando ejemplos gráficos que faciliten su comprensión.

## Funcionamiento de los autómatas y su relación con los elementos de la teoría de los lenguajes formales

La relación entre *alfabeto, símbolos, cadenas, lenguajes,* y *gramáticas* con los *autómatas* se explica de la siguiente forma:

- Los símbolos conforman un alfabeto.
- Las diversas combinaciones de estos símbolos son empleadas para crear cadenas, que en su conjunto conforman los lenguajes.
- Estos lenguajes son los aceptados por los autómatas.
- Y el procesamiento de los lenguajes, depende de las reglas definidas en las gramáticas de los mismos.

Un alfabeto $\Sigma_1$ puede definirse como: $\Sigma_1=\{a, b\}$. Mientras que $\Sigma_1$ es el alfabeto, $a$ y $b$ son sus símbolos. Una cadena $x$ se conformará entonces, mediante la combinación de los símbolos del alfabeto $\Sigma_1$. Así $x=aabb$ es una cadena sobre el alfabeto $\Sigma_1$ e $y=bbbb$ es otra. El conjunto de todas las cadenas, conformará entonces el lenguaje $L$ que se denotará por $\Sigma^*$. Así, $\Sigma^*=\{a, b, aa, bb, ab, aabb, aaabbb, aaaabbbb, aaaaa, bbbbb, \ldots\}$.

Un autómata finito determinista (AFD) $M_1$ que reconozca dicho lenguaje, podría quedar definido de la siguiente forma: $M_1=(\{q_1, q_2, q_3\}, \{a, b\}, \delta, q_1, \{q_2\})$ . Es decir que $Q=\{q_1, q_2, q_3\}$ (es el conjunto de todos los estados $Q$ de $M_1$ ); $\Sigma=\{a, b\}$ (es el alfabeto $\Sigma$ reconocido por $M_1$ ), la función de transición será $\delta: Q\times\Sigma\rightarrow Q$ ; de todos los estados $Q$ , $q_1$ sería el estado de inicio (es decir $q_0=q_1$ ) y $F=\{q_2\}$ (el conjunto de todos los estados finales, es decir, de los estados de aceptación del autómata).

El autómata, entonces, recibirá diferentes cadenas sobre el alfabeto reconocido. Cada símbolo de la cadena, será procesado por la función de transición, la cual, dependiendo del estado actual en el que se encuentre el autómata, y del símbolo recibido, pasará al siguiente estado.

El conjunto de las cadenas aceptadas por el autómata $M_1$ conforman el lenguaje $L$ del autómata, $L(M_1)$ .

> *Es importante remarcar que mientras las cadenas son aceptadas por un autómatas, los lenguajes son reconocidos por ese autómata.*

Un autómata puede no aceptar ninguna cadena y sin embargo, reconocer un lenguaje. Por ello, no es conveniente habla de *aceptación* de un lenguaje, sino, de *reconocimiento*.

Antes de definir el comportamiento de la función $\delta$ , se debe definir bajo qué condiciones una cadena es aceptada. Es decir, se debe definir $L(M_1)$ . Para este ejemplo, se dirá que $L(M_1)=\{w|w\, tiene\, al\, menos\, una\, 'b'\, seguida\, de\, dos\, 'a'\}$ (por ejemplo, aceptará $baa$ pero no aceptará $aab$ ).

Conociendo el lenguaje, se puede definir $\delta$ en una tabla como la siguiente:

| | $a$ | $b$ |
|---|---|---|
| $q_1$ | $q_1$ | $q_2$ |
| $q_2$ | $q_3$ | $q_2$ |
| $q_3$ | $q_2$ | $q_2$ |

La tabla anterior, comienza diciendo que *"si ingresa una 'a' en el estado inicial, el siguiente estado seguirá siendo el estado inicial; pero que si ingresa una 'b', el siguiente estado será un estado de aceptación..."* (ya que $q_2$ se definió previamente como un estado de aceptación, según la definición formal del autómata). Tanto este comportamiento como la definición formal del autómata, pueden verse reflejados en un *diagrama de estado*.

Las imágenes siguientes (Gráficos 3 y 4, respectivamente) muestran el diagrama de estado del ejemplo y la explicación de sus componentes (señalizada en rojo).

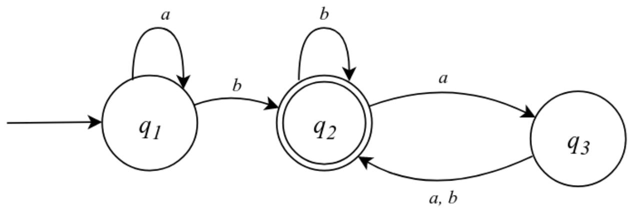


***Gráfico 3:*** *Diagrama de estado*

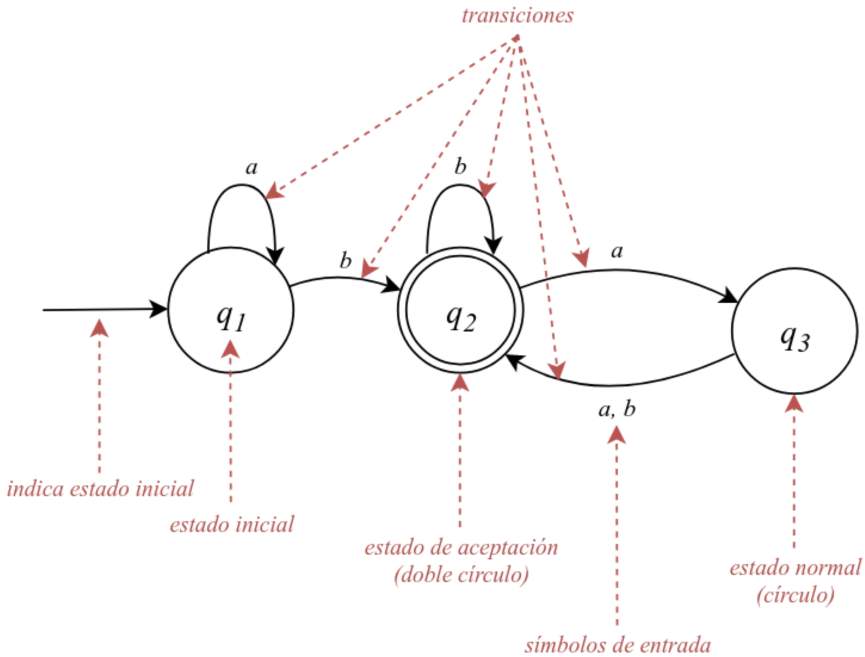


***Gráfico 4:*** *Diagrama de estado explicado*

## CAPÍTULO IV

# TEORÍA DE LA COMPUTABILIDAD

La teoría de la computabilidad tiene por objetivo responder a cuáles problemas pueden resolverse computacionalmente y cuáles no. Para ello, diferentes son las áreas de estudio en las que se enfoca, intentando dar una respuesta matemática a la pregunta. Se pretende con este capítulo hacer un breve resumen introductorio de los temas claves de esta teoría. Sin embargo, la teoría no es abarcada de forma más profundo puesto que demandaría un informe exclusivo a tal efecto.

## Marco contextual

Si se parte de la base de que la solución a un problema se consolida mediante un algoritmo que indique cómo dicho problema puede resolverse, podría ser posible determinar si un problema tiene o solución, abordando el análisis de los algoritmos. Si es posible crear un algoritmo que permita resolver el problema, entonces es posible resolver dicho problema. Y cuando dicho problema no sea posible de resolver, entonces es que no existe un algoritmo que permita resolverlo.

Sin embargo, entre los temas de estudio de la teoría de la computabilidad es posible encontrar a las máquinas de Turing, modelos computacionales considerados, quizás, la evolución de los autómatas.

La pregunta esperable es entonces ¿por qué? ¿por qué las máquinas de Turing son tema de estudio de la teoría de la computabilidad y no de la teoría de autómatas?

Y la respuesta demanda conocer el hecho anecdótico de que las máquinas de Turing fueron concebidas para poder establecer una definición formal —hasta ese momento inexistente— de algoritmo.

## Máquinas de Turing

Es habitual leer en la bibliografía que una máquina de Turing es similar a un autómata pero sin limitaciones de memoria. También es habitual leer que se vale de una "cinta" infinita para "grabar" símbolos, desde la cual un "cabezal" lee y escribe, moviéndose hacia un lado y hacia el otro.

Esta explicación —sugerida originalmente por su propio creador, Alan Turing[TC001]—, puede resultar ambigua si no se aclara que se trata de una "máquina imaginaria", de una "cinta imaginaria" y de un "cabezal imaginario". Es decir, que **no constituye una máquina de existencia real**.

Se trata entonces de una descripción que busca crear una analogía visual de un concepto matemáticamente abstracto, como lo es un modelo computacional. Por lo tanto, una máquina de Turing no constituye una máquina de existencia real, sino tan solo un modelo.

> *Una máquina de Turing es un **modelo computacional abstracto**, que sirvió como base para establecer una definición formal de «algoritmo».*

Formalmente, una máquina de Turing es un modelo computacional definido como una séptupla formada por los elementos $(Q,\Sigma,\Gamma,\delta,q_{0,}\#,F)$ , los cuáles denotan, cada uno, a los siguientes componentes:

| | |
|---|---|
| $Q$ | Es un conjunto finito de todos los estados posibles, para los que $q_0$ es el estado inicial, y $F\subset Q$ . |
| $\Sigma$ | Es el conjunto finito de los símbolos de entrada. |
| $\Gamma$ | Es el conjunto finito de los símbolos de cinta de los cuales $I\subset\Gamma$ . |
| # | Símbolo vacío, para el que se cumple que $\#\in(\Gamma-I)$ . |
| $\delta$ | Es la función que define un conjunto finito de reglas de transición. |
| $q_0$ | El estado inicial que además $q_0\in Q$ . |
| $F$ | Conjunto finito de estados finales. |

## Funcionamiento de una máquina de Turing

Para entender el modelo de una máquina de Turing de una forma no tan abstracta, se puede imaginar una máquina construida a partir de un autómata y constituida de los siguientes COMPONENTES:

| | |
|---|---|
| CINTA | Es en realidad el flujo de entrada de los datos, al que imaginariamente se le asigna un extremo izquierdo pero se considera que hacia adelante (hacia la derecha) es infinita. |

La cinta puede emplearse tanto para realizar operaciones de cómputo como de almacenamiento auxiliar.

Esta cinta debe pensarse como longitud infinita de celdas en las que en cada una de ellas existirá un símbolo escrito.

| | |
|---|---|
| ESTADOS | Una máquina de Turing puede tener un número finito de estados pero nunca menos de dos estados, puesto que el estado inicial y el estado de aceptación, no pueden ser el mismo estado. |
| CABEZAL | Es un mecanismo de control que se mueve hacia delante (o derecha de la cinta imaginaria) y hacia atrás (o hacia la izquierda de la cinta imaginaria) y puede, en cualquier momento, encontrarse en uno de todos los estados posibles. |
| ALFABETO | Se trata de un conjunto finito de símbolos de entrada. |
| SÍMBOLOS DE CINTA | Una máquina de Turing tendrá unos símbolos de entrada (alfabeto) pero también unos símbolos propios de cinta que puede escribir para generar “marcas” imaginarias que le permitan identificar segmentos de cinta cuando esta se utilice como medio de almacenamiento auxiliar. |
| SÍMBOLO VACÍO | Es un símbolo de relleno que estará escrito en todas las celdas imaginarias de la cinta en las que no haya escrito un valor de entrada. |

Las ACCIONES que una máquina de Turing puede realizar, se limitan a operaciones de escritura, donde sustituye un símbolo por otro, y operaciones de movimiento de una celda a otra, hacia ambos lados.

El FUNCIONAMIENTO general de la máquina de Turing, se inicia con la entrada de los datos. La función de transición aplicará la regla que corresponda al símbolo actualmente visible por el cabezal.

Las reglas estarán definidas sobre la base de dos factores:

1. El símbolo actual de la celda actual (visible para el cabezal).
2. El estado actual de la máquina.

Y tendrán el siguiente formato $\delta(q_a, \gamma_a)=(q_n, A)$ donde:

| | |
|---|---|
| $q_a$ y $\gamma_a$ | son el estado actual y el símbolo actual, respectivamente; |
| $q_n$ | es el nuevo estado, y $A$ la acción a ejecutar. Dicha acción, puede ser de escritura, en la cual implique escribir un nuevo símbolo tal que o bien de movimiento, pudiendo moverse hacia una celda izquierda ( $L$ ) o derecha ( $R$ ) tal que $\delta(q_a, \gamma_a)=(q_n, \{L, R\})$ . |

Si se toma como base la función de transición definida anteriormente, es posible determinar que **la máquina de Turing descrita, siempre será determinista**, pues existe una única transición posible para cada símbolo y estado.

Cuando el nuevo estado ( $q_n$ ) de la máquina de Turing sea un *estado de parada* (o estado de aceptación), la máquina de Turing finalizará el cómputo, aunque esto no implique que siempre deba finalizar. En algunas ocasiones, podrá verse envuelta en un *bucle infinito*, y en otras, rebasar imaginariamente el extremo izquierdo, provocando una *detención anormal*.

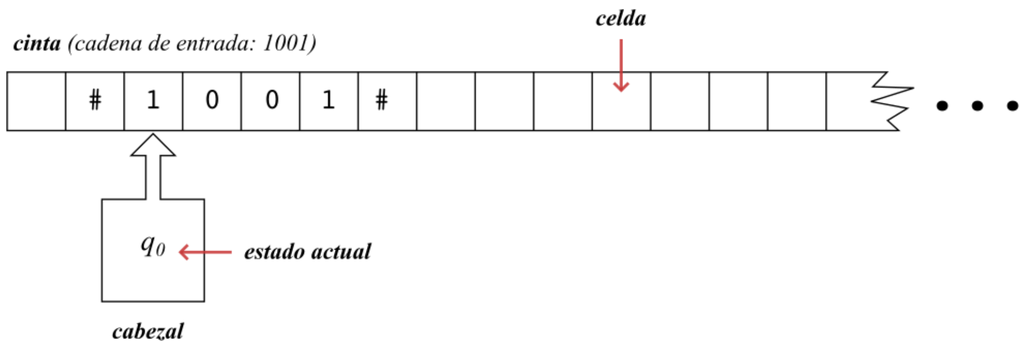


***Gráfico 5:*** *Ejemplo de modelo gráfico (imaginario) de una máquina de turing*

En el el ejemplo del gráfico anterior, el cabezal se encuentra en el símbolo de entrada $1$ y en el estado inicial $q_0$ . La regla a aplicar sería entonces la que corresponda a dicho estado y dicho símbolo, es decir, una función definida para $\delta(q_{0,}1)$ . Si dicha función fuese $\delta(q_{0,}1)=(q_{3,}R)$ , significaría que el cabezal pasaría al estado $q_3$ y se movería a la derecha ( $R$ ).

El gráfico 6 muestra cómo reaccionaría el cabezal si la máquina de Turing procesara la primera celda de la cinta del ejemplo del gráfico 5, basándose en la regla de la función $\delta(q_{0,}1)=(q_{3,}R)$ .

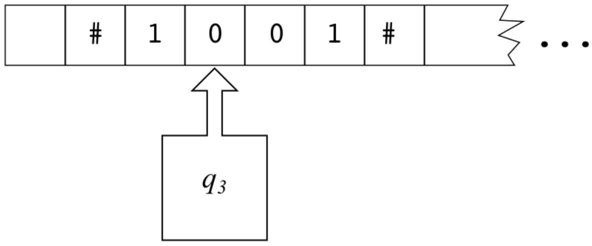


***Gráfico 6:*** *Cabezal de la máquina de Turing en movimiento y cambiando de estado.*

Desde el momento en el que toda entrada de una máquina de Turing (MT) es una cadena, se hace necesario convertir cualquier objeto $\langle O \rangle$ de entrada en una cadena. La MT puede ser programada para decodificar o decodificar cadenas y objetos que no lo sean. De esta forma, es posible incluso construir un modelo universal, es decir, una Máquina de Turing Universal (MTU) que permita simular el comportamiento de cualquier máquina de Turing (MT). Así, una máquina de Turing universal, podría capturar y procesar cualquier algoritmo.

## Decidibilidad[3]

Desde el momento en el que demostrar que un lenguaje es *decidible* necesariamente implica demostrar que un problema computacional lo es, el objetivo de la decidibilidad como mecanismo para determinar si un algoritmo tiene o no solución, es demostrar la decidibilidad de un lenguaje. Por lo tanto, se pueden utilizar lenguajes para representar problemas computacionales. Un ejemplo de ello, sería demostrar que un autómata finito determinista (AFD) acepta una determinada cadena, expresando dicho problema como un lenguaje $L_{ADF}$ .

3 **DECIDIBILIDAD.** Que tiene capacidad de decidir, que puede decidir, que puede tomar decisiones.

Para este lenguaje, se define que:

$$L_{ADF}=\{\langle \boldsymbol{M}_{1,}\boldsymbol{w}\rangle \mid \boldsymbol{M}_1 \; es\, un\, AFD\, que\, acepta \; \boldsymbol{w}\}$$

**TEOREMA 4.1.** $L_{AFD}$ es un lenguaje decidible.

**IDEA.** Crear una máquina de Turing $MT_1$ que decida $L_{ADF}$ de la siguiente forma:

$$\begin{array}{l} MT_1 = " \\ \quad \langle \boldsymbol{M}_{1,}\boldsymbol{w}\rangle \; es\, la\, entrada\, , siendo \; \boldsymbol{M}_1 \; un\, AFD\, y \; \boldsymbol{w} \; una\, cadena: \\ \quad \mathbf{1.}\, Simular\, \boldsymbol{M}_1 con\, \mathbf{w}\, como\, entrada\, . \\ \quad \mathbf{2.}\, Si\, la\, simulación\, termina\, en\, estado\, de\, acepción\, , \\ \qquad entonces\, \boldsymbol{aceptar}\, . \\ \qquad Sino\, , \boldsymbol{rechazar}\, . \\ \quad " \end{array}$$

**DEMOSTRACIÓN.**

En la tabla 3 se definió formalmente un AFD como la quíntupla $(Q,\Sigma,\delta,q_{0,}F)$ cuya función de transición tenía la forma $\delta : Q\times\Sigma\rightarrow Q$. La MT procesa entonces $w$ utilizando como reglas las del autómata (paso 1). Si al finalizar se encuentra en estado de aceptación, entonces $MT_1$ también finalizará en dicho estado. En caso contrario, rechazará la entrada.

## CAPÍTULO V

# TEORÍA DE LA COMPLEJIDAD

Cuando la teoría de la computabilidad ha logrado determinar a qué se considera (al menos hasta la fecha), un problema que puede resolverse computacionalmente, la teoría de la complejidad busca clasificar dichos problemas en fáciles o difíciles de resolver, y al igual que la teoría de la computabilidad, pretende dar una respuesta matemática a esta pregunta. El objetivo de este capítulo es hacer una breve introducción, a mero título orientativo pro sin un abordaje en profundidad, sobre los puntos claves que son objeto de estudio de esta teoría.

## Complejidad temporal y tiempo polinómico. Las clases P y NP.

**MEDIR LA COMPLEJIDAD DE UN PROBLEMA.** Se parte de la base de que el problema a resolver es un problema decidible. Esto significa que en principio, podría resolverse. Sin embargo, en la práctica, resolver el problema podría demandar más recursos de los existentes o insumir más tiempo del disponible. Por ello, en principio, la teoría de la complejidad, medirá la dificultad de un problema sobre la base del tiempo insumido por la solución del problema.

**DEFINICIÓN 5.1** La ***complejidad temporal*** de una máquina de Turing determinista —asumiendo que para en todas las entrada — estará determinada por una función $f : N \rightarrow N$ , para la que $f(n)$ representa el número máximo de pasos que la máquina emplea para el procesamiento de cualquier entrada de longitud $n$ .

**DEFINICIÓN 5.2** Sea $MT$ una máquina de Turing determinista; y $w$ cada una de las entradas de $MT$ para las que $f(w)$ está definida. Se dice que $MT$ calcula $f(w)$ en ***tiempo polinómico*** $p(x)$ si $MT$ calcula $f(w)$ en $p(|w|)$ pasos.

**DEFINICIÓN 5.3** Una clase $P$ es la clase de los lenguajes que una máquina de Turing determinista acepta en tiempo polinómico.

**DEFINICIÓN 5.4** Una clase $NP$ es la clase de los lenguajes que una máquina de Turing no determinista acepta en tiempo polinómico.

# CONCLUSIONES

El abordaje de las teorías de funciones, de conjuntos, y la inducción matemática del capítulo I, han permitido alcanzar el objetivo de sentar las bases matemáticas para el estudio y comprensión de las teorías de autómatas, de la computabilidad, y de la complejidad, dado que las tres teorías hacen un uso reiterado de estas bases.

Abordar la teoría de lenguajes formales desde sus elementos en el capítulo II, ha permitido alcanzar el objetivo de definir las bases formales que describen las teorías de autómatas, de la computabilidad, y de la complejidad, puesto que al definir con precisión sus componentes, se han eliminado ambigüedades.

En el capítulo III, abarcar la teoría de autómatas desde las perspectivas descriptiva, formal y funcional, ha permitido alcanzar el objetivo de entender cómo la Teoría de Autómatas influye en el procesamiento de la información, mientras que el abordaje tanto formal como funcional de las máquinas de Turing del capítulo IV, ha permitido lograr conocer el objeto de estudio de la Teoría de la Computabilidad. Al abordar la teoría de complejidad desde la definición de los elementos que permiten medir la complejidad de un problema y clasificarlos, ha permitido alcanzar el objetivo de  conocer el objeto de estudio de la misma.

Los primeros dos capítulos han permitido abordar las teorías de autómatas, de la computabilidad, y de la complejidad desde diferentes perspectivas, logrando así obtener una visión general de los principales temas de estas tres teorías, de forma tal que su abordaje futuro pueda centrarse exclusivamente en la perspectiva teórica con mayor nivel de profundidad.

# BIBLIOGRAFÍA

Para mayor comodidad, se presenta la bibliografía organizada por tema, alfabéticamente.

**AUTÓMATAS, TEORÍA DE. LENGUAJES FORMALES, TEORÍA DE.**

[TA01] Brookshear, G. (1993). *Teoría de la computación. Lenguajes formales, autómatas y complejidad* (1st ed., pp. 21-49). Delaware: Addison-Wesley Iberoamericana.

[TA02] De Castro Korgi, R. (2004). *Teoría de la computación. Lenguajes, Autómatas, Gramáticas.* (1st de., pp. 25-62). Bogotá: Universidad Nacional de Colombia.

[TA03] Gómez, D., & Pardo, L. M. (2015, February 15). *Teoría de Autómatas y Lenguajes Formales (para Ingenieros Informáticos)*. Universidad de Cantabria. https://personales.unican.es/pardol/Docencia/TALF2012.pdf

[TA30] Gupta, M. (2020). *Discrete Mathematics* (11th ed., pp. 525-552). Meerut: Krishna Prakashan Media ℗ Ltd.

[TA04] Hopcrpft, J., Motwani, R. & Ullman, J. (2001). *Introduction to Automata Theory, Languages, and Computation* (2nd ed., pp. 37-54). Boston: Addison-Wesley.

[TA05] Kulkarni, V. (2013). *Theory of Computation* (1st ed., pp. 29-37). New Dheli: Oxford University Press.

Maheshwari, A., & Smid, M. (2019, April 17). *Introduction to Theory of Computation*. Computational Geometry Lab (Carleton University). https://cglab.ca/~michiel/TheoryOfComputation/TheoryOfComputation.pdf

[TA06] Sipser, M. (1997). *Introduction to the Theory of Computation* (1st ed., pp. 29-62. Boston: PWS Publishing Company.

[TA07] Wood, D. (1987). *Theory of Computation* (1st ed., pp. 95-154). New York: John Wiley & Son.

**COMPLEJIDAD, TEORÍA DE LA.**

[TDC01] Brookshear, G. (1993). *Teoría de la computación. Lenguajes formales, autómatas y complejidad* (1st ed., pp. 247-295). Delaware: Addison-Wesley Iberoamericana.

[TDC02] Dean, W. (2016). *Computational Complexity Theory*. Stanford Encyclopedia of Philosophy. https://plato.stanford.edu/archives/win2016/entries/computational-complexity/

**COMPUTABILIDAD, TEORÍA DE LA.**

[TC010] Brookshear, G. (1993). *Teoría de la computación. Lenguajes formales, autómatas y complejidad* (1st ed., pp. 142-158). Delaware: Addison-Wesley Iberoamericana.

[TC011] De Mol, L. (2018). *Turing Machines*. Stanford Encyclopedia of Philosophy. https://plato.stanford.edu/entries/turing-machine/

[TC012] Kulkarni, V. (2013). *Theory of Computatio*n (1st ed., pp. 142-151). New Dheli: Oxford University Press.

[TC013] Levelt, W. J. M. (1974). *An Introduction to the Theory of Formal Languajes* (1st ed., pp. 102-114). The Netherlands: Mouton Publishers.

[TC014] Sipser, M. (1997). *Introduction to the Theory of Computation* (1st ed., pp. 125-222. Boston: PWS Publishing Company.

[TC001] Turing, A. M. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, *s2-42*(1), 230–265. https://doi.org/10.1112/plms/s2-42.1.230

**CONJUNTOS, TEORÍA DE.**

[TC01] Brookshear, G. (1993). *Teoría de la computación. Lenguajes formales, autómatas y complejidad* (1st ed., pp. 6-10). Delaware: Addison-Wesley Iberoamericana.

[TC02] Gupta, K. (2020). *Set Theory and Related Topics* (20th ed., pp. 39-49). Meerut: Krishna Prakashan Media ℗ Ltd.

[TC03] Gupta, M. (2020). *Discrete Mathematics* (11th ed., pp. 39-112). Meerut: Krishna Prakashan Media ℗ Ltd.

[TC04] Kulkarni, V. (2013). *Theory of Computation* (1st ed., pp. 3-11). New Dheli: Oxford University Press.

[TC05] Sipser, M. (1997). *Introduction to the Theory of Computation* (1st ed., pp. 3-7). Masachusets: PWS Publishing Company.

**FUNCIONES, TEORÍA DE.**

[F01] Carena, M. (2019). *Manual de matemática preuniversitaria* (1st ed., pp. 151-279). Santa Fe: Ediciones UNL.

[F02] Courant, R. & John, F. (2001). *Introducción al cálculo y al análisis matemático. Vol. 1* (1st ed., pp. 41-47). México D.F.: Editorial Limusa.

[F03] Gallardo López, D., Arques Corrales, P. & Lesta Pelayo, I. (2005). *Introducción a la teoría de la computabilidad* (2nd ed., pp. 19-21). San Vicente del Raspeig: Publicaciones de la Universidad Alicante.

[F04] Gupta, K. (2020). *Set Theory and Related Topics* (20th ed., pp. 57-86). Meerut: Krishna Prakashan Media ℗ Ltd.

[F05] Sipser, M. (1997). *Introduction to the Theory of Computation* (1st ed., pp. 7-9). Boston: PWS Publishing Company.

[F06] Wood, D. (1987). *Theory of Computation* (1st ed., pp. 17-20). New York: John Wiley & Son.

**INDUCCIÓN MATEMÁTICA.**

[IM01] Courant, R. & John, F. (2001). *Introducción al cálculo y al análisis matemático. Vol. 1* (1st ed., pp. 80-83). México D.F.: Editorial Limusa.

[IM02] Hopcrpft, J., Motwani, R. & Ullman, J. (2001). *Introduction to Automata Theory, Languages, and Computation* (2nd ed., pp. 13-27). Boston: Addison-Wesley.

[IM03] Kishan, H. (2020). *Number Theory* (13th ed., pp. 1-8). Meerut: Krishna Prakashan Media ℗ Ltd.

[IM04] Kulkarni, V. (2013). *Theory of Computatio*n (1st ed., pp. 14-16). New Dheli: Oxford University Press.

[IM05] Sipser, M. (1997). *Introduction to the Theory of Computation* (1st ed., pp. 21-24. Boston: PWS Publishing Company.

[IM06] Wood, D. (1987). *Theory of Computation* (1st ed., pp. 28-37). New York: John Wiley & Son.

**LENGUAJES FORMALES, TEORÍA DE.**

*(ver también: AUTÓMATAS, TEORÍA DE. LENGUAJES FORMALES, TEORÍA DE.)*

[LF03] Chomsky, N. (1956). Three models for the description of language. *IEEE Transactions on Information Theory*, *2*(3), 113–124. https://doi.org/10.1109/tit.1956.1056813

[LF01] Levelt, W. J. M. (1974). *An Introduction to the Theory of Formal Languajes* (1st ed., pp. 1-34). The Netherlands: Mouton Publishers.