1. Introducción

> 🖊 **Nota**
>
> Esta asignatura es una locura la puta mierda apuntes que tiene, menos mal que en prácticas nos salvan. Me molaría reestructurar todo para que tuviese un orden un poco más lógico, pero me da pereza.

1.1 Definición general

1.1 Definición General: ¿De qué va esto?

Definición Formal

La **Teoría de Autómatas y Lenguajes Formales** estudia modelos matemáticos para representar procesos de cálculo. Su objetivo es entender qué problemas pueden resolver los ordenadores y con qué eficiencia. Aborda dos preguntas clave:

- **Decidibilidad:** ¿Qué puede hacer un ordenador?
- **Complejidad:** ¿Con qué eficiencia puede hacerlo?

Nota del Tutor (El concepto real)

Piensa en esta asignatura como el **"Diseño de Máquinas que leen cosas"**. Todo se reduce a un esquema básico:

1. **Input:** Le das una cadena de texto a la máquina.
2. **Proceso:** La máquina (el autómata) procesa símbolo a símbolo.
3. **Output:** La máquina responde **SÍ** (Aceptado/Válido) o **NO** (Rechazado/Inválido).

1.2 Utilidad de los autómatas, gramáticas y expresiones regulares

Los conceptos abstractos que veremos son la base de herramientas que usas a diario:

Autómatas Finitos (DFA/NFA): El motor de búsqueda de patrones.

- *Uso real:* **Ctrl+F** en Word, validación de emails en formularios web.

Gramáticas (Sintaxis): Reglas de construcción.

- *Uso real:* **Compiladores**. Cuando VS Code te marca un error rojo, es porque tu código rompió las reglas de la gramática del lenguaje.

Expresiones Regulares: Búsqueda compacta.

- *Uso real:* Filtrado de datos y comandos `grep` en Linux.

1.3 El "Diccionario" de la Asignatura

Peligro de Examen: Confundir estos términos es la causa #1 de suspensos

Concepto	Símbolo	Definición Formal	💡 Analogía Práctica
Alfabeto	Σ	Conjunto finito y no vacío de símbolos. Ej: $\Sigma = \{0, 1\}$	Las **piezas de Lego** disponibles. (No puedes usar piezas que no estén en la caja).
Palabra / Cadena	w, x, y	Secuencia finita de símbolos de Σ.	Una **torre** construida con esas piezas.
Longitud	$	w	$
Lenguaje	L	Conjunto de cadenas ($L \subseteq \Sigma^*$).	El **manual de instrucciones** que dice qué torres son válidas.

La Cadena Vacía (ε o λ) vs. El Lenguaje Vacío ($\emptyset$):

Error conceptual más común

Cadena Vacía (ε): Es una palabra que **no tiene símbolos**.

- Longitud: $|\varepsilon| = 0$.
- *Analogía:* Es como un string vacío en programación `""`. Existe, pero no tiene nada dentro.

Lenguaje Vacío ($\emptyset$): Es un lenguaje que **no contiene ninguna palabra**.

- *Analogía:* Una carpeta vacía.

> **Regla de oro:**
>
> - $L = \{\varepsilon\} \rightarrow$ Un lenguaje que contiene una palabra (la vacía). **No está vacío.**
> - $L = \emptyset \rightarrow$ Un lenguaje sin palabras.

1.4 Operaciones básicas

Sobre Palabras

- **Concatenación (xy):** Pegar y detrás de x. **OJO:** No es conmutativa ($ab \neq ba$).

- **Potencia (x^i):** Repetir la cadena i veces. (Ej: $a^3 = aaa$).
- **Reflexión (x^{-1}):** Leerla al revés (de derecha a izquierda).

Sobre Lenguajes

Además de Unión ($\cup$), Intersección ($\cap$) y Diferencia ($-$), existen operaciones críticas en esta teoría:

1. **Concatenación ($L_1 \cdot L_2$):** Combina cada palabra de L_1 con cada palabra de L_2.
2. **Cierre de Kleene / Estrella (L^*):**
   - Representa **cero o más** repeticiones de palabras del lenguaje.
   - Siempre incluye la cadena vacía ε.
   - Σ^* = El conjunto de **todas** las palabras posibles que se pueden formar con el alfabeto.

1.5 Conceptos Fundamentales

El "Trío Sagrado": Lenguaje, Gramática y Máquina

Imagina que quieres preparar un plato de comida específico. Para que ese plato exista, necesitas tres elementos que están íntimamente conectados pero son distintos:

1. **El Lenguaje (El Plato Final):** Es el conjunto de cadenas (palabras) que queremos validar o generar. Es el concepto abstracto.
   - *Ejemplo:* "El conjunto de todos los emails válidos".
2. **La Gramática (La Receta):** Son las reglas generativas. Te dice cómo construir una cadena válida paso a paso. Es el "constructor".
   - *Ejemplo:* `Email -> Texto @ Texto . Dominio`
3. La Máquina / Autómata (El Crítico de Comida):
   Es el mecanismo que verifica. Le das una cadena y te dice "Sí, pertenece al lenguaje" o "No, rechazada". Es el "reconocedor".

> La Relación Fundamental:
>
> Para cada tipo de Lenguaje, existe una Gramática que lo genera y una Máquina que lo reconoce. Son tres caras de la misma moneda.

La Jerarquía: Las "Muñecas Rusas"

Aquí es donde entra el lío de los nombres. No son categorías aisladas, son **niveles de complejidad**.

- Cada nivel **incluye** al anterior.

- Todo lo que es Regular (Nivel 3) es TAMBIÉN Independiente del Contexto (Nivel 2), etc.
  Vamos del más simple (y restrictivo) al más potente (y libre).

Nivel 3: Lo Regular (Lo más simple)

Aquí no hace falta memoria compleja, solo saber "dónde estoy".

- **Lenguaje:** Regular.
- **Gramática:** Regular (Lineal por la derecha o izquierda). Reglas muy rígidas ($A \to aB$).
- **Máquina: Autómata Finito (AFD/AFN)**.
  - *¿Qué puede hacer?* Patrones simples, búsquedas de texto.
  - *¿Qué NO puede hacer?* Contar (no sabe si hay el mismo número de 'a' que de 'b').
  - *Ejemplo:* Validar un email o un número de teléfono.

Nivel 2: Independiente del Contexto (La estructura)

Aquí añadimos una memoria tipo "pila" (LIFO). Podemos recordar cosas para cerrarlas después.

- **Lenguaje:** Independiente del Contexto (LIC).
- **Gramática:** Independiente del Contexto (GIC). Reglas tipo $A \to \alpha$ (una variable cambia por cualquier cosa).
- **Máquina: Autómata con Pila (Pushdown)**.
  - *¿Qué puede hacer?* Anidar cosas. Paréntesis `(( ))`, estructuras `if-then-else`, contar pares ($a^n b^n$).
  - *¿Qué NO puede hacer?* Depender del contexto cruzado (ej: $a^n b^n c^n$).
  - *Ejemplo:* La sintaxis de lenguajes de programación (Java, C, Python).

Nivel 1: Sensible al Contexto (El contexto importa)

Aquí la memoria es una cinta, pero limitada al tamaño de la palabra. Podemos mirar alrededor.

- **Lenguaje:** Sensible al Contexto.
- **Gramática:** Sensible al Contexto (GSC). Reglas donde importa qué hay a los lados ($xAy \to xBy$).
- **Máquina: Autómata Linealmente Acotado**.
  - *¿Qué puede hacer?* Coordinar tres o más conteos ($a^n b^n c^n$) y verificar que una variable ha sido declarada antes de usarse (contexto real).
  - *Ejemplo:* El lenguaje natural (español, inglés) en muchos aspectos gramaticales complejos.

Nivel 0: Recursivamente Enumerable (El poder total)

Aquí no hay límites. Si se puede calcular, está aquí.

- **Lenguaje:** Recursivamente Enumerable.
- **Gramática:** Irrestricta (Sin restricciones).
- **Máquina: Máquina de Turing**.
  - *¿Qué puede hacer?* Cualquier algoritmo computable por un ordenador actual.
  - *El peligro:* La máquina podría quedarse pensando para siempre (bucle infinito) y nunca responder.

Tabla Resumen Definitiva (La "Chuleta")

Nivel (Chomsky)	LENGUAJE	GRAMÁTICA	MÁQUINA (Autómata)	Poder Principal
Tipo 3	Regular	Regular	**Autómata Finito** (AFD/AFN)	Sin memoria (solo estados).
Tipo 2	Indep. Contexto	GIC (Context-Free)	**Autómata con Pila**	Memoria LIFO (paréntesis, anidación).
Tipo 1	Sensible Contexto	GSC (Context-Sensitive)	**Autómata Linealmente Acotado**	Memoria acotada (relaciones complejas).
Tipo 0	Rec. Enumerable	Irrestricta	**Máquina de Turing**	Memoria infinita (Cómputo universal).

¿Cómo recordarlo?

1. **AFD:** Un interruptor de luz (encendido/apagado). Simple.
2. **Pila:** Una pila de platos (solo toco el de arriba). Estructura.
3. **Turing:** Un ordenador con memoria infinita. Dios.