1.1 Definición general

La **Teoría de Autómatas y Lenguajes Formales** es una rama fundamental de la computación que estudia modelos matemáticos para representar y analizar procesos de cálculo. Su objetivo principal es entender **qué problemas pueden ser resueltos por ordenadores** y **cómo se pueden describir y procesar los lenguajes que usan**. Estos modelos incluyen máquinas abstractas, como los autómatas, y sistemas de reglas, como las gramáticas.

En esencia, la teoría aborda dos preguntas: - **¿Qué puede hacer un ordenador?** (capacidad de cómputo) - **¿Con qué eficiencia puede hacerlo?** (complejidad de los algoritmos)

1.2 Utilidad de los autómatas, gramáticas y expresiones regulares

Los conceptos estudiados aquí tienen aplicaciones prácticas en varias áreas de la informática:

- **Autómatas finitos:**
  Son modelos simples que reconocen patrones en cadenas de símbolos. Se utilizan en:
  - Diseño y verificación de circuitos digitales.
  - Analizadores léxicos de compiladores (reconocer palabras clave, identificadores, etc.).
  - Búsqueda de patrones en textos (por ejemplo, encontrar palabras en documentos).
  - Verificación de protocolos de comunicación (comprobación de secuencias válidas de mensajes).
- **Gramáticas:**
  Conjunto de reglas que definen cómo se pueden construir cadenas válidas en un lenguaje. Son fundamentales para:
  - Definir la sintaxis de lenguajes de programación.
  - Implementar analizadores sintácticos (*parsers*) que verifican si una expresión o programa está correctamente formulado.
- **Expresiones regulares:**
  Especifican patrones de cadenas usando una sintaxis compacta. Se emplean en:
  - Validación de entradas en formularios (por ejemplo, correos electrónicos).
  - Búsqueda y reemplazo en editores de texto.
- **Complejidad y decidibilidad:**
  La teoría también estudia:
  - **Decidibilidad:** Qué problemas pueden resolverse mediante algoritmos.
  - **Complejidad:** Qué tan eficiente (rápido) puede ser la solución de un problema.

1.3 Conceptos básicos

Para entender la teoría:

- **Conjuntos:**
  Agrupaciones de elementos. Se pueden combinar usando operaciones clásicas como unión ($\cup$), intersección ($\cap$), diferencia ($-$) y complemento.
- **Alfabeto (Σ):**
  Conjunto finito y no vacío de símbolos (por ejemplo, $\Sigma = \{0, 1\}$ para binario).
- **Palabra o cadena:**
  Secuencia finita de símbolos tomados del alfabeto (Σ).
  - **Cadena vacía:** Representada por ε o λ. No contiene símbolos.
  - **Longitud de una cadena ($|w|$):** Número de símbolos que contiene.
- **Lenguaje (L):**
  Cualquier conjunto de cadenas sobre el alfabeto ($L \subseteq \Sigma^*$). Ejemplos:
  - $L = \emptyset$ (lenguaje vacío: no contiene ninguna cadena).
  - $L = \{\varepsilon\}$ (solo contiene la cadena vacía).
  - $L = \{0^n 1^n \mid n \geq 1\}$ (todas las cadenas con n ceros seguidos de n unos, para n mayor o igual a 1).

1.4 Operaciones básicas

- **Sobre palabras (cadenas):**
  - **Concatenación (xy):** Unir dos cadenas, colocando una después de la otra.
  - **Potencia (x^i):** Concatenar la misma cadena x, i veces.
  - **Reflexión o reverso (x^{-1}):** Invertir el orden de los símbolos en la cadena.
- **Sobre lenguajes:**
  - **Unión ($L_1 \cup L_2$):** Conjunto de cadenas que están en L_1 o L_2.
  - **Intersección ($L_1 \cap L_2$):** Cadenas que están en ambos lenguajes.
  - **Diferencia ($L_1 - L_2$):** Cadenas que están en L_1 pero no en L_2.
  - **Concatenación ($L_1 \cdot L_2$):** Todas las cadenas obtenidas al concatenar una de L_1 con una de L_2.
  - **Potencia (L^i):** Concatenar cadenas del lenguaje L, i veces.
  - **Reflexión (L^{-1}):** Conjunto de las cadenas de L invertidas.

1.5 Gramáticas y clasificación de Chomsky

Una **gramática** es un conjunto de reglas que define cómo se pueden generar las cadenas de un lenguaje. No todas las gramáticas tienen el mismo poder para describir lenguajes; por eso, Chomsky propuso una jerarquía de tipos:

- **Tipo 0 (Sin restricciones):**
  Puede generar cualquier lenguaje que sea **recursivamente enumerable**.

Modelo: Máquina de Turing. Estos lenguajes pueden ser tan complejos que no siempre hay un algoritmo para decidir si una cadena pertenece al lenguaje (problemas indecidibles).

- **Tipo 1 (Sensibles al contexto):**
  Reglas dependen del contexto de los símbolos.
  Modelo: Autómata linealmente acotado.
  Complejidad: Exponencial.

- **Tipo 2 (Independientes del contexto):**
  Reglas no dependen del contexto; solo de los símbolos individuales.
  Modelo: Autómata con pila (pushdown automaton).
  Complejidad: Polinómica.
  Ejemplo: Lenguajes de programación.

- **Tipo 3 (Regulares):**
  Reglas muy simples, solo para cadenas que pueden ser reconocidas por **autómatas finitos** (sin memoria adicional).
  Complejidad: Lineal.
  Ejemplo: Expresiones regulares, búsqueda de patrones simples.

Cada tipo de gramática genera un tipo de lenguaje. Decimos que un lenguaje L es de tipo i ($i = 0, 1, 2, 3$) si existe una gramática de ese tipo capaz de generar todas sus cadenas.