Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 0

ALUMNO: Roberto Estrada Tovar

Dr. Eduardo Cornejo Velázquez

1. Introducción

Los **lenguajes formales y los autómatas** son fundamentales en la teoría de la computación, ya que permiten modelar y analizar el funcionamiento de sistemas computacionales. Los conceptos de **alfabetos, palabras y lenguajes** proporcionan la base para la construcción de autómatas, que a su vez pueden utilizarse para reconocer patrones y procesar información de manera estructurada.
En este documento, exploraremos los principales conceptos relacionados con los lenguajes formales y los autómatas, incluyendo:

- **Operaciones con palabras**: Definiciones y propiedades básicas.
- **Lenguajes regulares y no regulares**: Caracterización y métodos de demostración.
- **Operaciones booleanas sobre lenguajes**: Unión, intersección, complemento, entre otras.
- **Autómatas finitos**: Deterministas y no deterministas, así como su transformación y minimización.
- **Máquinas de Turing**: Su papel en la computación y su capacidad de resolver problemas complejos.

A través de estos temas, se proporciona una base teórica sólida para el análisis y diseño de autómatas, fundamentales en la construcción de compiladores, el análisis de texto, la robótica y diversas aplicaciones en la informática teórica y práctica.

Un **alfabeto** es un conjunto finito de símbolos. Los números naturales no pueden formar un alfabeto, ya que son infinitos y esto generaría una contradicción. Sin embargo, sí es posible definir un alfabeto con los dígitos del 0 al 9.
A partir de un alfabeto se forman **palabras** o **cadenas**, que son secuencias finitas y ordenadas de símbolos pertenecientes a dicho alfabeto. La cadena o palabra vacía (λ) es aquella cuya longitud es 0.

Propiedades de las palabras

- **Longitud**: Es la cantidad de símbolos que contiene una palabra y se denota con el valor absoluto de un número, es decir, $|X|$.
- **Apariciones**: Representa la cantidad de veces que un símbolo específico aparece en la palabra, expresado como $|X|_a$.
- **Ordenación**: Una palabra puede considerarse mayor o menor en función de su longitud y del orden definido para los símbolos en el alfabeto (por ejemplo, el orden alfabético).
- **Combinaciones**: La cantidad de palabras posibles varía según la longitud de la palabra y el número de símbolos en el alfabeto.

La **clausura de Kleene** (Σ^*) representa el conjunto de todas las palabras posibles, de cualquier longitud, formadas con los símbolos de un alfabeto, incluyendo la cadena vacía. Se define también $\Sigma^+ = \Sigma^* - \{\lambda\}$, es decir, el conjunto de todas las palabras posibles excepto la cadena vacía.

Operaciones con palabras

- **Concatenación**: La concatenación de m símbolos de la palabra x_1 con n símbolos de la palabra x_2 consiste en unir ambas palabras, generando una nueva palabra de longitud $m + n$. La cadena vacía actúa como el elemento neutro en esta operación.
- **Potencia**: La potencia n-ésima de una palabra se obtiene concatenándola consigo misma n veces. Por definición, cualquier palabra elevada a la potencia 0 es igual a la cadena vacía.
- **Prefijos, sufijos y segmentos**: Dadas dos palabras x e y pertenecientes a Σ^*, se dice que y es un segmento de x si existen u y v tales que:

$$x = u \cdot y \cdot v$$

  - Si $u = \lambda$, entonces y es un **prefijo** de x.
  - Si $v = \lambda$, entonces y es un **sufijo** de x.

- **Reverso**: El reverso de una palabra es aquella leída de derecha a izquierda. Se denota como:

$$(xa)^r = ax^r$$

- **Lenguaje**: Un **lenguaje** es un subconjunto (finito o infinito) de la clausura de Kleene sobre un alfabeto.

Operaciones Booleanas

- **Unión**: Dados dos lenguajes L_1 y L_2, su unión contiene todas las palabras presentes en L_1 o en L_2.
- **Intersección**: Contiene únicamente las palabras que pertenecen tanto a L_1 como a L_2.
- **Otras operaciones**: Se pueden aplicar otras operaciones como:
  - **Complemento**: Contiene todas las palabras que no pertenecen al lenguaje dado.
  - **Diferencia**: Contiene las palabras de un lenguaje que no están en el otro.
  - **Diferencia simétrica**: Contiene las palabras que pertenecen a uno solo de los lenguajes, pero no a ambos.
- **Producto**: El producto de dos lenguajes es la concatenación de todas las palabras del primer lenguaje con todas las del segundo.
- **Potencias sobre lenguajes**: Si $n > 0$, indica cuántas veces se realiza el producto del lenguaje consigo mismo.
- **Cierre**: Es la unión de todas las potencias de un lenguaje.
- **Cociente**:
  - **Por la izquierda**: Consiste en obtener las palabras resultantes tras eliminar, al inicio, la palabra o símbolo sobre la cual se aplica el cociente en el lenguaje.
- **Homomorfismo**: Es una aplicación que transforma un alfabeto en otro, estableciendo reglas de asignación, como en el caso del código ASCII.

Autómata

Un **autómata** es una máquina abstracta que modela procesos de cálculo. Se compone de **estados** y **transiciones**, que determinan cómo debe reaccionar al recibir una entrada. Suelen representarse mediante **grafos**, donde los nodos corresponden a los estados y los arcos a las transiciones. Los autómatas son herramientas fundamentales para la comprensión y resolución de problemas computacionales.

Tipos de autómatas

- **Autómata finito**: Posee un número finito de estados y procesa las entradas de manera secuencial. Se utiliza principalmente para reconocer **lenguajes regulares**.
- **Autómata de pila**: Incorpora una pila para gestionar información adicional, lo que le permite analizar la estructura de lenguajes más complejos. Se usa en el **análisis sintáctico** de lenguajes de programación y en la **validación de expresiones matemáticas**.
- **Máquina de Turing**: Es un modelo computacional más poderoso que los anteriores, ya que posee una **cinta infinita** que le permite manipular información de manera ilimitada. Puede resolver cualquier problema que pueda ser resuelto por un ordenador.

Aplicaciones

Los autómatas tienen diversas aplicaciones en áreas como:

- **Análisis de texto**: Procesamiento de lenguaje natural.
- **Robótica**: Control de sistemas autónomos.
- **Creación de compiladores**: Análisis y traducción de código fuente.

Autómata Finito Determinista (AFD)

Un **autómata finito determinista** (AFD) es una máquina abstracta compuesta por **cinco elementos**:

1. **Conjunto de estados** (Q).
2. **Alfabeto del autómata** (Σ).
3. **Función de transición** ($\delta : Q \times \Sigma \to Q$), que define el estado al que se transita al leer un símbolo.
4. **Estado inicial** ($q_0 \in Q$).
5. **Conjunto de estados finales** ($F \subseteq Q$).

El AFD posee un **conjunto finito de estados** y se denomina **determinista** porque, para cada estado, **existe como máximo una transición por símbolo**, evitando así cualquier ambigüedad en su ejecución.
Los AFD pueden representarse mediante **tablas de transiciones** o **grafos**, donde se visualizan claramente sus elementos y el comportamiento del autómata.

Autómata Finito No Determinista (AFND)

Un **autómata finito no determinista** (AFND) es un tipo de autómata en el que un **estado puede tener múltiples transiciones definidas para un mismo símbolo**, a diferencia de un **autómata finito determinista** (AFD), donde solo puede existir una única transición por símbolo.
Al igual que el AFD, un AFND está compuesto por **cinco elementos**:

1. **Conjunto de estados** (Q).
2. **Alfabeto del autómata** (Σ).
3. **Función de transición** ($\delta : Q \times \Sigma \to \mathcal{P}(Q)$), que ahora devuelve un **conjunto de estados** en lugar de un único estado.
4. **Estado inicial** ($q_0 \in Q$).
5. **Conjunto de estados finales** ($F \subseteq Q$).

Es posible transformar un **AFND en un AFD** mediante un proceso llamado **construcción del conjunto de estados**, en el cual se agrupan estados en **conjuntos equivalentes** para eliminar la no determinación y convertirlo en un **autómata finito determinista**.

Transformación de un AFND a un AFD

Para realizar la transformación de un **AFND** a un **AFD**, se sugiere comenzar por **analizar los elementos del AFND** para comprender su funcionamiento. El análisis se debe enfocar principalmente en los siguientes aspectos:

1. **Función de transiciones**: En el AFND, la función de transición define cómo se transita entre estados al leer un símbolo. A diferencia del AFD, en el AFND esta función puede devolver un conjunto de estados, lo que da lugar a la no determinación.
2. **Tabla de transiciones**: Las tablas de transiciones son representaciones clave que muestran de manera explícita cómo se realizan las transiciones entre los estados del autómata para cada símbolo del alfabeto.

Una vez que se ha comprendido la estructura del AFND y sus transiciones, se puede proceder a aplicar el proceso de **construcción del conjunto de estados** para transformar el autómata en un AFD. Este proceso consiste en agrupar los estados del AFND en **conjuntos** y definir transiciones entre estos conjuntos, lo que elimina la no determinación y genera un autómata determinista.

Coincidencia de Patrones (Pattern Matching)

La **coincidencia de patrones** es una técnica utilizada en diversas aplicaciones, como el **análisis de textos** y la **búsqueda de secuencias en cadenas**. Para lograr esto, se emplean **expresiones regulares** (Regex), que permiten definir y buscar patrones dentro de un texto.
Existen dos algoritmos importantes para la coincidencia de patrones:

1. **Naïve Algorithm**: Este algoritmo se basa en la construcción del **Árbol Aceptor de Prefijos**, el cual modela los patrones de búsqueda mediante una estructura de **autómata finito**.
2. **Autómata Completo**:
   - En cada estado debe haber una **transición definida** para cada símbolo del alfabeto.
   - En este autómata, se establecen las **transiciones faltantes** del **Árbol Aceptor de Prefijos** para garantizar que cada estado tenga una transición bien definida para cada símbolo del alfabeto.

Lenguaje por la Derecha

El **lenguaje por la derecha** se define como la **cadena de caracteres que permite llegar desde un estado seleccionado hasta un estado final** dentro del autómata.
Este concepto es útil para **analizar y definir el comportamiento de los estados dentro del autómata**, permitiendo determinar **qué combinaciones de símbolos llevan a un estado de aceptación**.

Relación de Equivalencia

Una **relación** R sobre un **conjunto** A se considera una **relación de equivalencia** si cumple con las siguientes **tres propiedades fundamentales**:

1. **Reflexiva**: Para todo elemento $a \in A$, se cumple que $a\,R\,a$.
2. **Simétrica**: Si $a\,R\,b$, entonces $b\,R\,a$.
3. **Transitiva**: Si $a\,R\,b$ y $b\,R\,c$, entonces $a\,R\,c$.

Demostración de que un Lenguaje es Regular

Un lenguaje L es **regular** si y solo si R_L es de **índice finito**, es decir, si tiene un **número finito de clases de equivalencia**.
Matemáticamente, esto significa que la **relación de equivalencia** asociada a L induce un número **finito de particiones** en el conjunto de cadenas. Esto garantiza que L puede ser reconocido por un **Autómata Finito**.
Para demostrar que un lenguaje es **regular**, se pueden utilizar los siguientes métodos:

1. **Construcción de un Autómata Finito** que lo reconozca.
2. **Verificación de la definición de los lenguajes regulares**, asegurando que se cumpla la condición de número finito de clases de equivalencia.
3. **Expresión del lenguaje mediante expresiones regulares o gramáticas regulares**, ya que cualquier lenguaje representable por estos métodos es regular.

Demostración de que un Lenguaje NO es Regular

Para demostrar que un lenguaje **no es regular**, se debe comprobar que tiene **infinitas clases de equivalencia** en su relación de equivalencia R_L.
Esto implica que **no es posible construir un Autómata Finito** capaz de reconocer dicho lenguaje, ya que:

- Los **Autómatas Finitos** solo pueden manejar un **número finito de estados**.
- Un lenguaje con **infinitas clases de equivalencia** requeriría **infinitos estados**, lo que no es posible dentro de un Autómata Finito.

Reducción de Estados de un Autómata

Antes de reducir los estados de un autómata, es importante asegurarse de que el autómata sea:

- **Accesible**: Todos los estados deben ser alcanzables desde el estado inicial.
- **Completo**: Para cada estado y cada símbolo del alfabeto, debe existir una transición definida.

El **algoritmo de reducción de estados** es un **algoritmo de partición**, cuyo objetivo es **identificar y fusionar estados equivalentes** para **simplificar el autómata sin alterar su comportamiento**.

Proceso de Reducción de Estados

Para llevar a cabo la reducción de estados, debemos seguir dos reglas fundamentales:

1. **Las clases con un único estado no pueden reducirse más**.
2. **Los estados que pertenezcan a clases diferentes no pueden agruparse**.

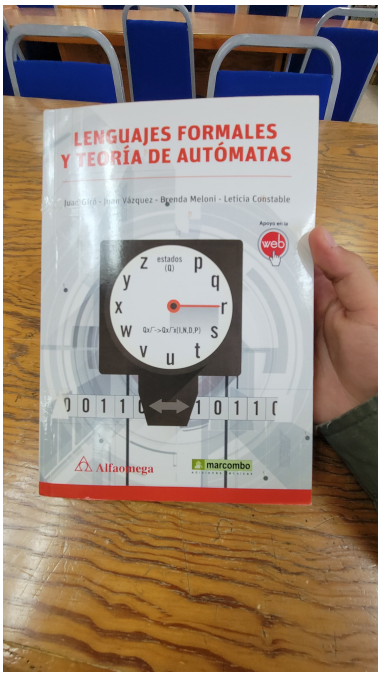
5. Conclusiones

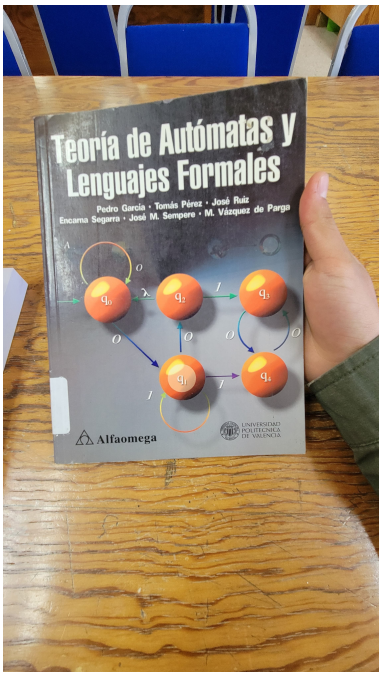
El estudio de los **lenguajes formales y los autómatas** es fundamental en la **teoría de la computación**, ya que permite modelar y comprender el funcionamiento de sistemas computacionales. A través del análisis de **alfabetos, palabras y lenguajes**, se establecen las bases para la construcción de autómatas capaces de reconocer patrones y procesar información de manera estructurada.

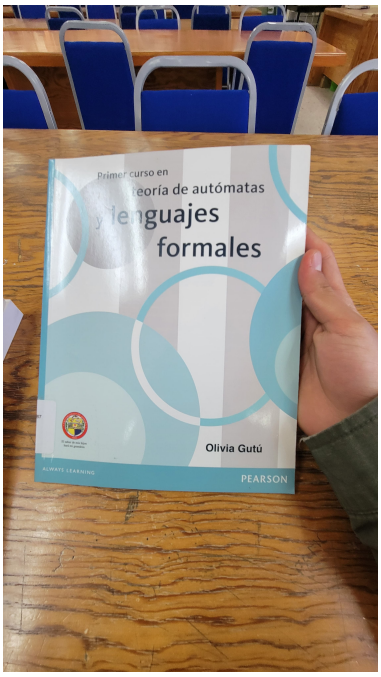
Hemos explorado las **operaciones con palabras**, los **lenguajes regulares y no regulares**, las **operaciones booleanas sobre lenguajes**, así como la estructura y funcionamiento de **autómatas finitos y máquinas de Turing**. Además, se han abordado **métodos de transformación y reducción de estados**, esenciales para la optimización de los autómatas.

En conclusión, la teoría de autómatas y lenguajes formales tiene aplicaciones clave en diversas áreas de la informática, incluyendo el **análisis léxico y sintáctico**, el **diseño de compiladores**, la **verificación de software** y la **inteligencia artificial**. Su estudio no solo proporciona herramientas teóricas, sino que también permite la resolución de problemas prácticos en el ámbito de la computación.

6. Evidencias de libros físicos







Preguntas y Respuestas

¿Cuál es la diferencia entre un autómata finito determinista (AFD) y un autómata finito no determinista (AFND)?

- **Autómata Finito Determinista (AFD):** Para cada estado y símbolo de entrada, existe una única transición posible.
- **Autómata Finito No Determinista (AFND):** Un mismo estado puede tener múltiples transiciones para un mismo símbolo, permitiendo múltiples caminos posibles en la computación.

¿Qué son las operaciones booleanas sobre lenguajes y cómo afectan a los lenguajes regulares?
Las operaciones booleanas sobre lenguajes incluyen:

- **Unión:** $L_1 \cup L_2 \rightarrow$ Palabras que están en L_1 o L_2.
- **Intersección:** $L_1 \cap L_2 \rightarrow$ Palabras comunes en L_1 y L_2.
- **Complemento:** $\sim L \rightarrow$ Palabras que no pertenecen al lenguaje.
- **Diferencia:** $L_1 - L_2 \rightarrow$ Palabras en L_1 que no están en L_2.
- **Producto:** $L_1 \cdot L_2 \rightarrow$ Concatenación de palabras de L_1 con palabras de L_2.

¿Cuál es la diferencia entre un lenguaje regular y un lenguaje no regular?

- Un **lenguaje regular** es aquel que puede ser reconocido por un autómata finito y puede expresarse mediante expresiones regulares o gramáticas regulares.
- Un **lenguaje no regular** requiere mecanismos más potentes, como autómatas de pila o máquinas de Turing, ya que no puede ser representado únicamente con un autómata finito.

¿Qué es una expresión regular y para qué se usa?
Una **expresión regular (ER)** es una representación compacta de un lenguaje formal que define patrones en cadenas de texto. Suelen emplearse en:

- La validación de datos (correos electrónicos, números de teléfono, etc.).
- La búsqueda de patrones en archivos de texto.

¿Qué representa un estado en un autómata?
Un estado en un autómata representa una situación en la que se encuentra la máquina en un momento dado:

- **Estado inicial:** Es el punto de partida del autómata.
- **Estados intermedios:** Representan transiciones dentro del autómata.
- **Estados finales:** Indican si una cadena es aceptada por el autómata.

Referencias Bibliográficas

- Codemath. (n.d.). *Lenguajes Formales desde CERO [Videos]*. In YouTube. Recuperado de https://www.youtube.com/playlist?list=PLyRNgg3I27WiOZqDamrZon3QDPZMlZ5P4
- Giró, J., Vázquez, J., Meloni, B., Constable, L. (2015). *Lenguajes formales y teoría de autómatas*. Colombia: Alpha Editorial.
- García, P., Pérez, T. (2001). *Teoría de autómatas y lenguajes formales*. México: Alfaomega.
- Gutú, O. (2013). *Primer curso en teoría de autómatas y lenguajes formales*. Pearson Educación.