Universidad Autónoma del Estado de Hidalgo

# REPORTE DE PRÁCTICA

## NOMBRE DE LA PRÁCTICA: Practica 0

**ALUMNO: Reyes Gutierrez Alan**
**Dr. Eduardo Cornejo-Velázquez**

# 1. Introducción

Los fundamentos de los Autómatas y Lenguajes Formales son una disciplina central en la Teoría de la Computación. Esta área se enfoca en el estudio de modelos abstractos de cómputo ( los autómatas) y las formas precisas de describir conjuntos de cadenas de símbolos los cuales se les llama lenguajes formales.
Este reporte abordara temas cómo:
1.-Cómo definir y operar con lenguajes.
2.-Cómo diseñar dispositivos teóricos (autómatas finitos) capaces de reconocer estos lenguajes.
3.-La relación de equivalencia entre los autómatas y las Expresiones Regulares, esenciales para tareas como el diseño de compiladores, el análisis sintáctico y la búsqueda de patrones (pattern matching).

# 2. Marco teórico

## Lenguajes Formales

**Conceptos Fundamentales:**
Alfabeto: Es el conjunto finito de símbolos permitidos (ej: $\Sigma=\{a,b\}$ o $\Sigma=\{0,1\}$).
Palabra: Cualquier secuencia finita de símbolos del alfabeto (ej: aabb, 10101). La palabra vacía ($\epsilon$ o $\lambda$) es una palabra de longitud cero.
**El Mundo de las Posibilidades ($\Sigma^*$):** La Clausura de Kleene ($\Sigma^*$) es el conjunto de *todas* las palabras posibles que se pueden formar con el alfabeto, incluyendo la palabra vacía. Un Lenguaje (L) es simplemente un subconjunto de $\Sigma^*$, es decir, un conjunto de palabras que cumplen una regla específica.
**Operaciones Fundamentales sobre Lenguajes:** Las reglas de formación de lenguajes se basan en tres operaciones clave:
Unión ($L1 \cup L2$): Palabras que están en L1 o en L2.
Concatenación (L1L2): Palabras formadas por una palabra de L1 seguida de una palabra de L2.
Clausura de Kleene ($L^*$): Palabras formadas por cero o más repeticiones de palabras de L.

## Autómatas Finitos

Un Autómata Finito (M) es un quinteto formal: $M=(Q,\Sigma,\delta,q\ 0,F)$, donde Q es el conjunto de estados, $\Sigma$ es el alfabeto, $\delta$ es la función de transición (cómo se mueve la máquina), q0 es el estado inicial y F es el conjunto de estados finales (o de aceptación).
**Autómata Finito Determinista (AFD):** Por cada estado y símbolo de entrada, existe una y solo una transición definida.
**Autómata Finito No Determinista (AFND):** Permite múltiples transiciones o ninguna para un estado y símbolo, y puede incluir transiciones epsilon ($\epsilon$), que se realizan sin consumir un símbolo de entrada.
**Equivalencia:** Se demuestra que todo AFND (incluidos los que tienen $\epsilon$-transiciones) puede convertirse en un AFD equivalente, lo que prueba que tienen el mismo poder de reconocimiento.

## Teorema de Myhill-Nerode

Este teorema proporciona la herramienta definitiva para clasificar los lenguajes:
Un lenguaje es Regular si y solo si el número de clases de equivalencia (o clases de Myhill-Nerode) inducidas por la relación de indistinguibilidad es finito. Este concepto es clave para:
Probar la regularidad de un lenguaje.
Probar la no regularidad de un lenguaje (demostrando infinitas clases de equivalencia).

## Cierre y Minimización

**Propiedades de Cierre:** La clase de Lenguajes Regulares es cerrada bajo las operaciones básicas (unión, concatenación y clausura). Esto significa que si combinas Lenguajes Regulares con estas operaciones, el resultado siempre será un Lenguaje Regular. Esto justifica el poder de las Expresiones Regulares.
**Minimización de Autómatas:** La Minimización garantiza la unicidad. El AFD Mínimo (el que tiene el menor número de estados) que acepta un lenguaje L es único (salvo por el nombre de los estados). Esto se logra identificando y fusionando estados que son indistinguibles (que se comportan de forma idéntica respecto al camino restante de la entrada).

# 3.- Herramientas empleadas

**1.-Overleaf:** Plataforma de edición colaborativa utilizada para redactar este reporte. Permite generar documentos con formato académico y científico de alta calidad, utilizando el sistema de composición LATEX para manejar las fórmulas matemáticas y los símbolos formales (Σ,Q,δ,ϵ,...).

# 4.- Desarrollo

## Construcción y Conversión de Autómatas (De AFND a AFD)

Esta es una habilidad fundamental: demuestra que la flexibilidad del AFND no es más poderosa que la estricta estructura del AFD.
El Desafío: Diseñar un autómata que acepte cadenas que terminen en "01". Diseñar un AFND es fácil, pero para la implementación real se necesita un AFD.
1. Diseño de AFND (Fácil y Flexible): Un AFND puede tener una transición $\epsilon$ o múltiples caminos, lo que simplifica el diseño. Para el lenguaje L={cadenas que terminan en 01}:

| Estado | Símbolo | Transición |
|---|---|---|
| q0 (Inicial) | 0 | q0,q1 (¡No determinismo!) |
| q0 | 1 | q0 |
| q1 | 1 | q2 (Final) |

Aquí, q0 se queda en q0 para cualquier carácter que **no** sea el final, pero también tiene la opción de "saltar" a q1 al ver un '0' para empezar a buscar el patrón final.

## Conversión a AFD (Método de Subconjuntos):

Tomamos el AFND y creamos estados en el AFD que representan un conjunto de estados del AFND.

| Estado de AFD | Conjunto de Estados AFND | Transición con **0** | Transición con **1** |
|---|---|---|---|
| **A** (Inicial) | {q0} | {q0,q1} (**B**) | {q0} (**A**) |
| **B** | {q0,q1} | {q0,q1} (**B**) | {q0,q2} (**C**) |
| **C** (Final) | {q0,q2} | {q0,q1} (**B**) | {q0} (**A**) |

Resultado: El AFD resultante (A → B → C) es más grande y complejo, pero es determinista. Esto prueba que ambos modelos tienen el mismo poder de reconocimiento.

## La Prueba Teórica: Teorema de Myhill-Nerode

Este es el procedimiento para demostrar, de manera irrefutable, si un lenguaje es regular o no, sin tener que construir un autómata. Se basa en encontrar las clases de indistinguibilidad (las "diferencias esenciales").
**Ejemplo de Lenguaje NO Regular:** L={anbn|n≥0} (Cadenas con igual número de a's y b's: $\epsilon$,ab,aabb,aaabbb).
**El Problema:** Un autómata finito no tiene memoria para "contar" cuántas a's ha visto antes de que empiecen las b's.
**Prueba con Myhill-Nerode:**
1.-Toma una secuencia infinita de prefijos: w0=$\epsilon$, w1=a, w2=aa, w3=aaa, ... wi=ai.
2.-Encuentra una cadena de prueba (z) para diferenciar cada prefijo. Usaremos zi=bi.
3.-Comprobación:
-Si tomamos w2=aa: le concatenamos z2=bb. →aabb$\in$L.
-Si tomamos w3=aaa: le concatenamos z2=bb. →aaabb$\in$/L.
4.-Como el prefijo wi requiere exactamente i letras b's para ser aceptado, ¡cada prefijo ai requiere un "conocimiento" distinto!
**Conclusión:** Se ha encontrado una cantidad infinita de clases de indistinguibilidad (a0,a1,a2,...). Por el Teorema de Myhill-Nerode, L no es un Lenguaje Regular. Necesitarías una máquina con memoria externa (un Autómata de Pila).

## Evidencia Fotográfica


Figura 1: Evidencia de libro

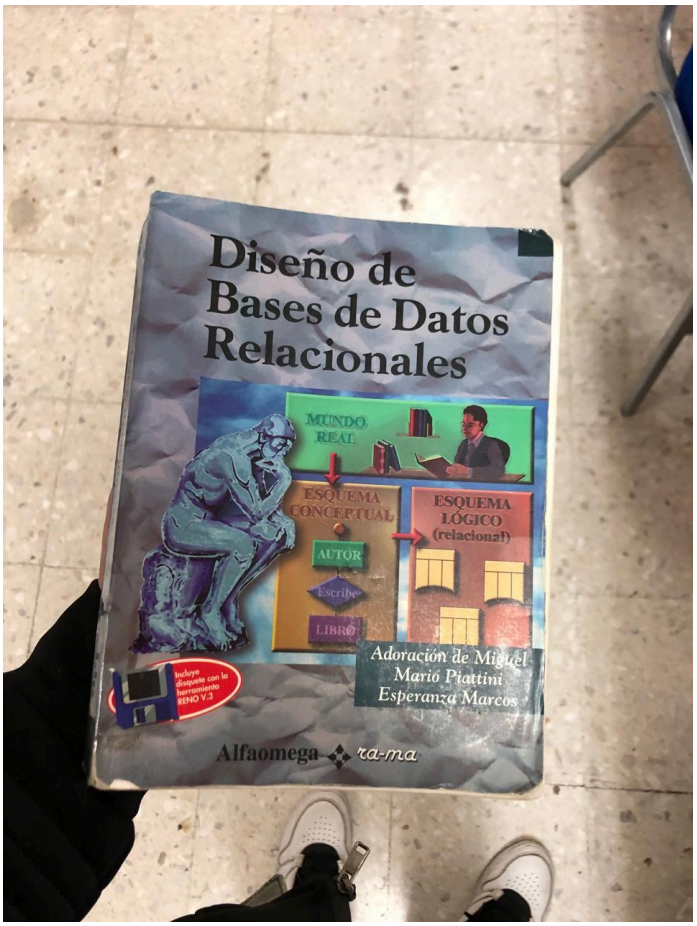


Figura 2: Evidencia de libro

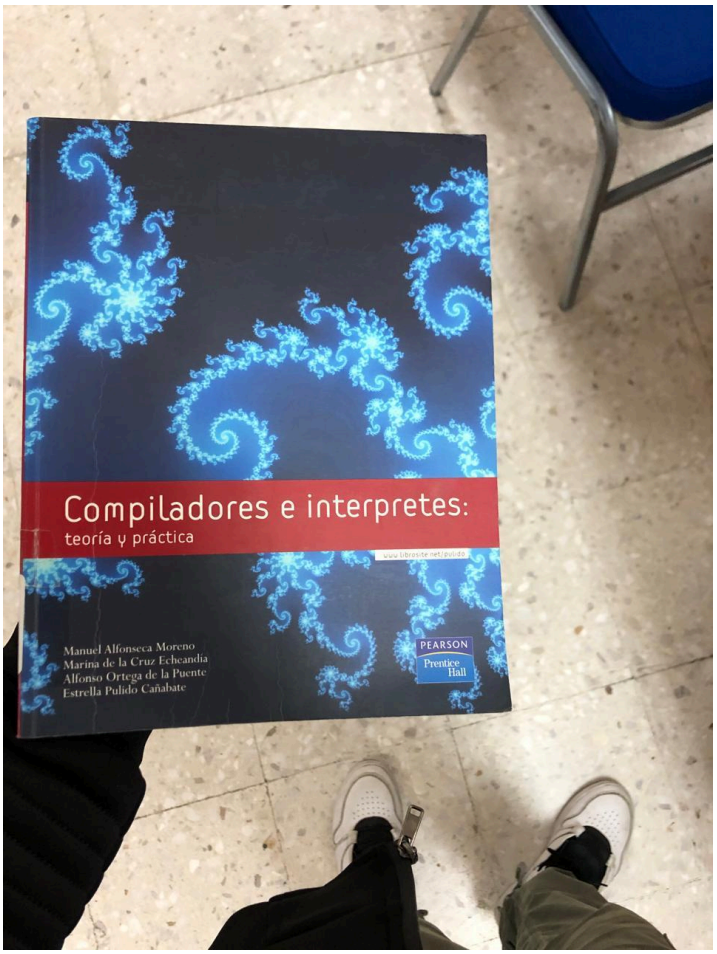


Figura 3: Evidencia de libro

# 4.- Conclusiones

Los Autómatas Finitos son modelos de cómputo con un poder limitado, capaces de reconocer únicamente Lenguajes Regulares. Su restricción principal es su memoria finita, lo que les impide manejar dependencias de longitud o estructuras balanceadas.
Se confirma la equivalencia formal entre los tres modelos de representación: AFD, AFND y Expresiones Regulares. Esta equivalencia permite a los ingenieros utilizar la herramienta más conveniente (ER para concisión, AFND para facilidad de diseño, AFD para implementación eficiente).

# Referencias Bibliográficas

## References


[2] De Miguel, A., Piattini, M., & Marcos, E. (2000). Diseño de bases de datos relacionales. Alfaomega/Ra-Ma.

[1] Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2008). *Compiladores: Principios, técnicas y herramientas* (2.ª ed.). Pearson Educación (Addison Wesley).

[3] Alfonseca Moreno, M., De la Cruz Echeandía, M., Ortega de la Puente, A., & Pulido Cañabate, E. (2006). *Compiladores e intérpretes: teoría y práctica*. Pearson Educación (Prentice Hall).