REPORTE DE PRÁCTICA NO. 0

NOMBRE DE LA PRÁCTICA

ALUMNO:

Daniel Martínez Escamilla

Dr. Eduardo Cornejo-Velázquez

Introducción

En esta práctica digital exploraremos los conceptos fundamentales de los **Autómatas Finitos Deterministas (AFD)** y **Autómatas Finitos No Deterministas (AFND)**, los cuales son modelos matemáticos esenciales en la teoría de la computación y el diseño de lenguajes formales.
Para comprender su funcionamiento y diferencias, analizaremos una serie de **videos explicativos** que presentarán sus estructuras, reglas de transición y aplicaciones. Estos videos nos permitirán visualizar cómo los autómatas procesan cadenas de entrada, establecen estados de aceptación y definen lenguajes reconocibles.

Herramientas empleadas

En esta práctica, se han utilizado diversas herramientas para el análisis y comprensión de los **Autómatas Finitos Deterministas (AFD)** y **Autómatas Finitos No Deterministas (AFND)**. Las fuentes de estudio incluyen tanto recursos digitales como físicos, los cuales han permitido profundizar en la teoría y aplicación de estos modelos computacionales.
Entre las herramientas principales se encuentran:

- **Videos de YouTube** del autor **CodeMath**, los cuales proporcionan explicaciones visuales y ejemplos prácticos sobre el tema.
- **Libros extraídos de la Biblioteca Central** de la **Universidad Autónoma del Estado de Hidalgo**, utilizados como referencia para el estudio teórico de autómatas y lenguajes formales.
- **Libros digitales** disponibles en la **Biblioteca Digital**, los cuales complementan el aprendizaje con enfoques académicos y técnicos adicionales.

4. Desarrollo

Resumen de los videos

RESUMEN DEL PRIMER VIDEO:——————————————————————————————-

Operaciones con palabras:
Concatenación: Es la unión de una palabra con otra palabra, sumando la longitud de cada variable: $m = 5 + n = 7 = 12$ —Holaa + amigoss— $= 12$ Si se concatena una palabra con la cadena vacía, es igual a la misma palabra: $x * \lambda = \lambda * x = x$ $hola * \lambda = \lambda * hola = hola$
Potencia: La potencia n-ésima de una palabra es el resultado de concatenar la palabra N veces consigo misma: $code^5 = codecodecodecodecode$ Pero si se eleva a 0, el resultado es la cadena vacía: $code^0 = \lambda$
Prefijo, sufijos y segmentos: Son conjuntos no vacíos que nos ayudan a visualizar lo que encontramos en nuestra cadena. Es un segmento de X si existen U y V tales que: $X = u * y * v$ Si $u = \lambda$, entonces y es un prefijo de x. Si $y = \lambda$, entonces y es un sufijo de x.

Ejemplo con la cadena: `1001`
Prefijos: $\{\lambda, 1, 10, 100, 1001\}$
Sufijos: $\{\lambda, 1, 01, 001, 1001\}$
Segmentos: $\{\lambda, 0, 00, 001, 1, 10, 100, 1001\}$

Ejemplo con la cadena: `seca`
Prefijos: $\{\lambda, s, se, sec, seca\}$
Sufijos: $\{\lambda, a, ca, eca, seca\}$
Segmentos: $\{\lambda, s, se, sec, seca, e, ec, eca, c, ca, a\}$

Reverso: Es la misma palabra leída de derecha a izquierda. x^r
Ejemplo con la cadena: `vaso` $vaso^r = osav$ $(xa)^r = ax^r$ $vaso^r = o * vas^r = os * va^r = osa^r = osav$
Ejemplo con la cadena: `pez` $pez^r = z * pe^r = ze * p^r = zep$
Lenguaje: Un lenguaje L es un subconjunto de la clausura de Klen sobre un alfabeto, pudiendo ser finito o infinito.
Ejemplo de lenguaje infinito: $L = \{ax \mid x \in \Sigma^* \text{ es español}\}$ = abeja, águila, ala, aaaaa ...
Ejemplo de lenguaje finito: $L = \{x \in \Sigma^* \text{ es español} : |x| < 6\}$ = bueno, hola, pez, águila ...

RESUMEN DEL SEGUNDO VIDEO——————————————————————:

Operaciones con lenguaje y aplicaciones:
Operaciones booleanas:
Unión: Todas las palabras existentes dentro de un lenguaje se unen con todas las palabras existentes de otro lenguaje.

$$L_1 = \{ax \mid x \in \Sigma^* \text{ (español)}\}$$
$$L_2 = \{xa \mid x \in \Sigma^* \text{ (español)}\}$$
$$L_1 \cup L_2 = \{x \mid x \in \Sigma^* \text{ (español)}, (a \in \text{Pref}(x) \vee a \in \text{Suf}(x))\}$$

Producto: Es la concatenación de cada una de las palabras del primer lenguaje con cada una de las palabras del segundo lenguaje.

$$L_1 = \{ax \mid x \in \Sigma^* \text{ (español)}\}$$
$$L_2 = \{xa \mid x \in \Sigma^* \text{ (español)}\}$$
$$L_1 * L_2 = \{axa \mid x \in \Sigma^* \text{ (español)}\}$$

Potencias sobre lenguaje: Concatenar las palabras del lenguaje con cada una del mismo lenguaje.

$$L = \{a, bb\}$$
$$L^2 = L * L = \{aa, abb, bba, bbbb\}$$

Cierre: Unión de todas las potencias de un lenguaje.

$$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \cdots \cup L^n$$

Cociente: Por la izquierda, son todas aquellas palabras obtenidas tras eliminar lo indicado a la que le aplicamos el cociente:

$$L = \{\text{recortar, revivir, codemath}\}$$
$$(re)^{-1}L = \{\text{cortar, vivir}\}$$

$$L_1 = \{ax \mid x \in \Sigma^* \text{ (español)}\}$$
$$(a)^{-1}L_1 = \{x \mid x \in \Sigma^* \text{ (español)}\}$$

Homomorfismo: Aplicación del alfabeto a otro y creación de reglas de asignación. Sirve para encriptar.

$$\begin{array}{cc} \Sigma^*_{\text{español}} & \Sigma^*_{\text{figuras}} \\ h(h) = O, & h(o) = \text{cuadrado} \\ h(l) = \text{triángulo}, & h(a) = \text{trapecio} \end{array}$$

$$(\text{trapecio})(\text{triángulo})(\text{cuadrado})\text{O}(\text{trapecio}) = aloha$$

RESUMEN DEL TERCER VIDEO: ——————————————————————

Autómata: Máquina abstracta que modela o describe procesos de cálculo, tiene estados y transiciones. Para representarlos usamos grafos:
- **Nodos** = Estados
- **Aristas** = Transiciones

Tipos de autómatas:
- **Autómata Finito:** Sencillo, con número finito de estados y procesa las entradas secuencialmente.
- **Autómata de Pila:** Usa una pila para procesar información, más poderoso que los anteriores.
- **Máquina de Turing:** Son las "supercomputadoras"de los autómatas, con una cinta infinita y capacidad de resolver cualquier problema computable.

Aplicaciones:
- Análisis de texto - Creación de compiladores - Robótica

RESUMEN DEL CUARTO VIDEO:——————————————————————
Autómata Finito Determinista (AFD): Máquina abstracta formada por una tupla de 5 elementos:

- Q = Conjunto de estados.
- Σ = Alfabeto de entrada.
- f = Función de transición.
- q_0 = Estado inicial.
- F = Subconjunto de Q, estado final.

Determinista: Cada estado tiene una única transición definida para cada símbolo del alfabeto.
Puede ser representado por:
- Tablas de transición. - Diagramas de transición.

Autómata Finito No Determinista (AFND): Compuesto por la misma tupla de 5 elementos que el AFD. Se diferencia en que un estado puede tener **múltiples transiciones** definidas para un mismo símbolo del alfabeto.

RESUMEN DEL QUINTO VIDEO:

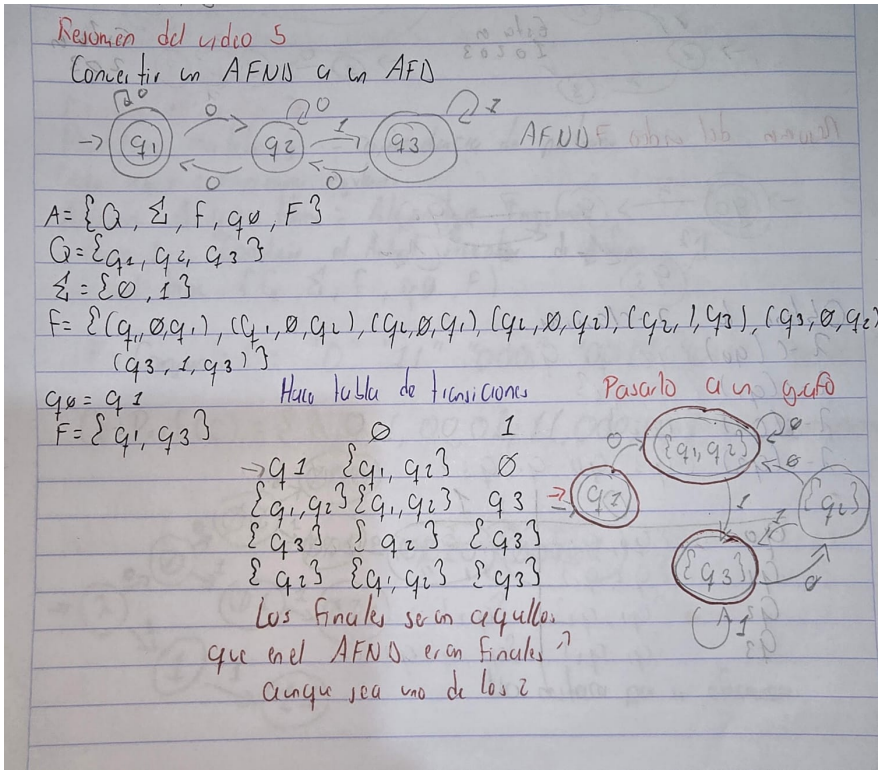

Figura 1: Conversión de un AFND a un AFD.

RESUMEN DEL SEXTO VIDEO:———————————————————————

Autómatas con transiciones Épsilon/Lambda: Este tipo de autómatas contiene los mismos 5 elementos en su tupla, pero su diferencia principal radica en sus funciones de transición.
A diferencia de los autómatas finitos deterministas y no deterministas, las transiciones **ϵ (épsilon) o λ (lambda)** permiten que el autómata cambie de estado sin necesidad de procesar una entrada.

Esto significa que un estado puede conducir a múltiples estados simultáneamente sin haber recibido alguna entrada del alfabeto, haciendo que el autómata pueda estar en diferentes estados a la vez.

RESUMEN DEL SÉPTIMO VIDEO:

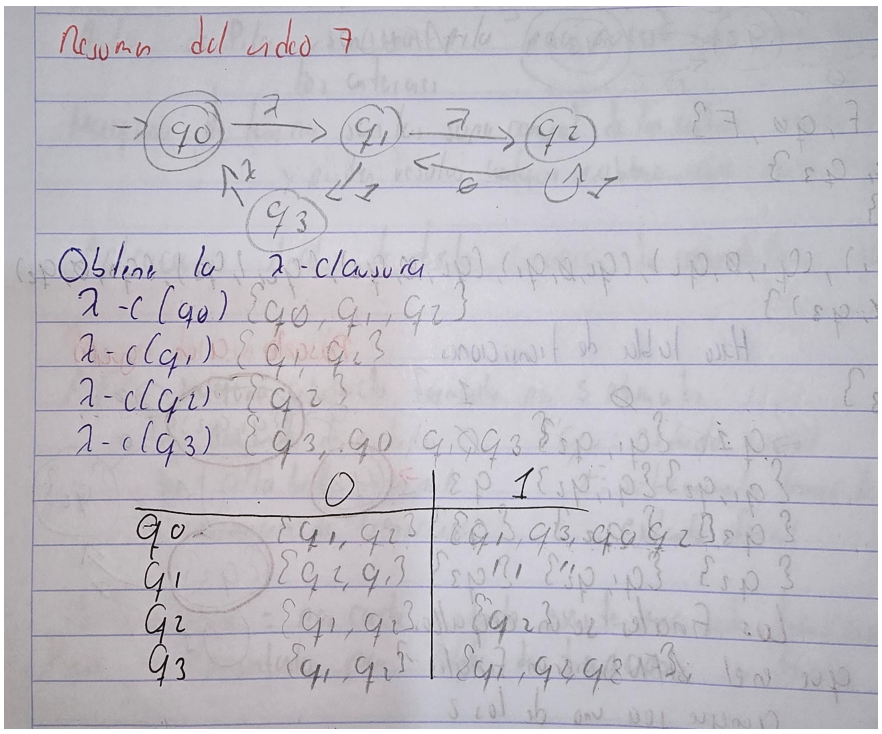

Figura 2: Obtener la λ-clausura y tabla de transición.

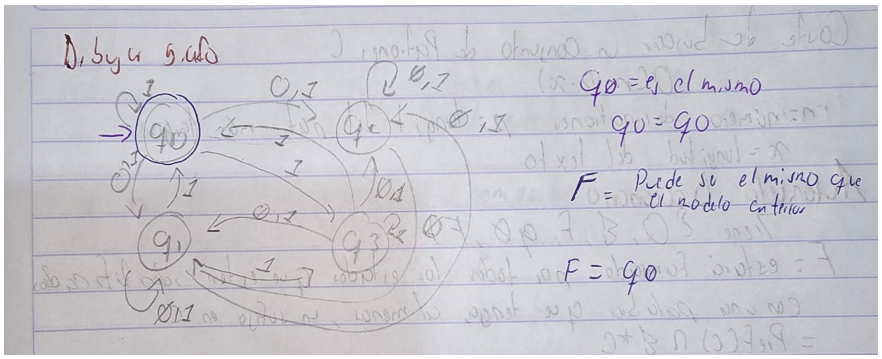

Figura 3: Construcción del autómata final.

RESUMEN DEL OCTAVO VIDEO:

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 utilizan expresiones regulares (Regex), que permiten definir y buscar patrones dentro de un texto.
Existen dos algoritmos importantes para la coincidencia de patrones:
Naive 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.
El autómata está compuesto por la tupla de 5 elementos:

$$(Q, \Sigma, f, q_0, F)$$

Donde: - Q: Conjunto de estados. - Σ: Alfabeto de entrada. - f: Función de transición. - q_0: Estado inicial. - F: Conjunto de estados finales.
En este caso, el conjunto de estados finales está dado por:

$$F = \{"0", "11", "000", "001", "011", "00101"\}$$

Y el conjunto de estados es el prefijo de C:

$$Q = \text{Prefijo}(C) = \{\lambda, 0, 00, 01, 11, 000, 001, 011, 0010, 00101\}$$

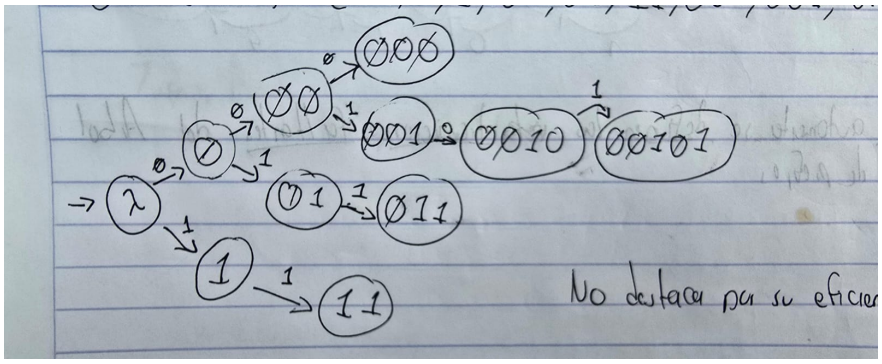

Figura 4: No destaca por su eficiencia.

Autómata Diccionario:
Este autómata está compuesto por la tupla de 5 elementos:

$$(Q, \Sigma, f, q_0, F)$$

Donde: - **Q**: Conjunto de estados. - **Σ**: Alfabeto de entrada. - **f**: Función de transición. - **q_0**: Estado inicial. - **F**: Conjunto de estados finales.
El conjunto de estados finales **F** estará formado por todos los estados que estén identificados con una palabra que tenga al menos un sufijo en:

$$C = \text{Pref}(C) \cap \Sigma^* C$$

La función de transición se define como:

$$f(x, a) = h(xa)$$

Donde, dada una palabra **u**, la función **$h(u)$** devuelve el sufijo más largo de **u** que pertenece a Pref(C).

Autómata Completo: En cada estado debe haber una transición definida para cada símbolo del alfabeto. En este autómata, se definen las **transiciones faltantes** del Árbol Aceptor de Prefijos para garantizar que cada estado tenga una transición bien definida para cada símbolo del alfabeto.

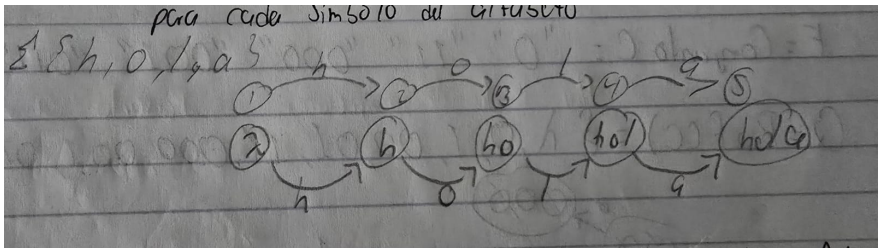

Figura 5: Autómata Completo.

RESUMEN DEL NOVENO VIDEO:

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:**

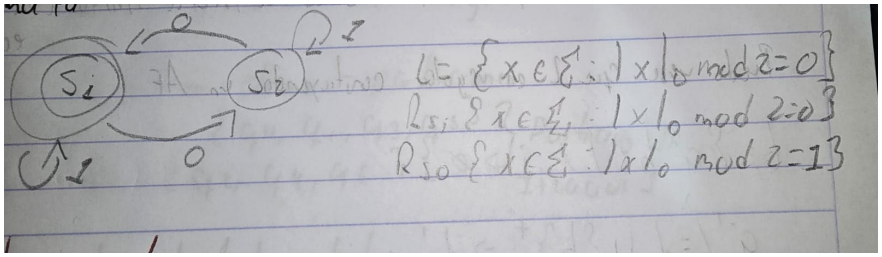

Figura 6: RL.

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

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

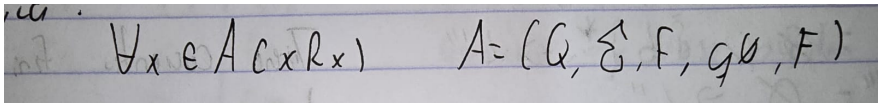

Figura 7: Reflexiva.

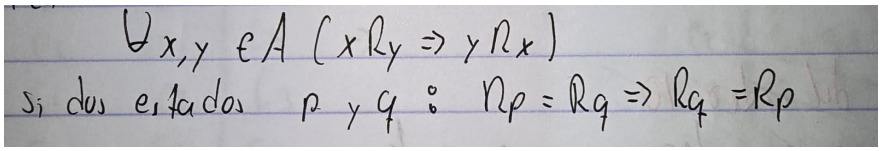

Figura 8: Simétrica.

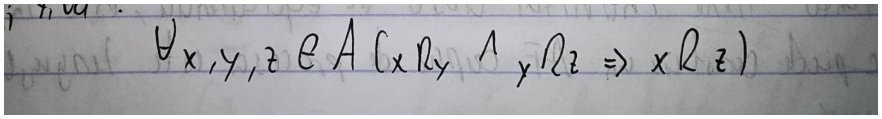

Figura 9: Transitiva.

RESUMEN DEL DÉCIMO VIDEO:——————————————————————————-

Demostrar que un lenguaje es regular:
Un lenguaje **L** es **regular** si y solo si **RL** 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, lo que garantiza que L pueda ser reconocido por un **Autómata Finito**.
Para demostrar que un lenguaje es regular, se puede construir un Autómata Finito que lo reconozca, verificando que cumple con la definición de los lenguajes regulares y que puede ser expresado mediante expresiones regulares o gramáticas regulares.

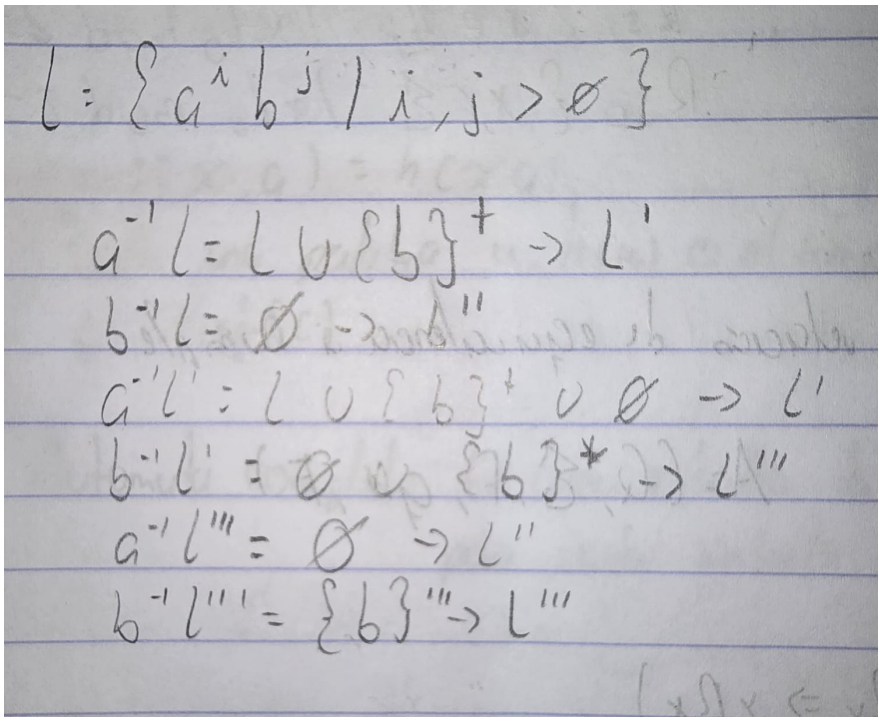

Figura 10: Tiene cocientes finitos

RESUMEN DEL VIDEO 11:

Demostrar 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 **RL**.
Esto implica que "no se puede construir un Autómata Finitoçapaz de reconocer este lenguaje, ya que los autómatas finitos solo pueden manejar un número finito de estados, mientras que un lenguaje con infinitas clases de equivalencia requeriría infinitos estados.
Un ejemplo de lenguaje que no es regular es:

$$L = \{0^n 1^n \mid n \geq 0\}$$

Este lenguaje requiere que el número de **0's** y **1's** esté balanceado, lo cual **no puede ser procesado por un Autómata Finito** porque necesitaría una memoria infinita para contar los **0's** y asegurarse de que haya la misma cantidad de **1's** después.
Un subconjunto importante de este lenguaje es:

$$A = \{0^i \mid i \geq 0\}$$

Este conjunto representa una secuencia arbitraria de ceros, lo que también contribuye a la imposibilidad de un reconocimiento finito.

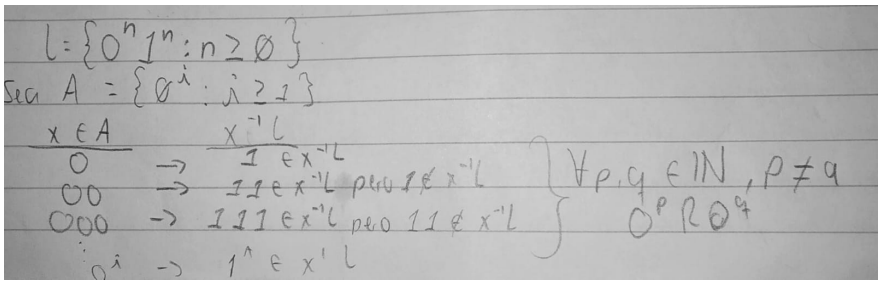

Figura 11: Lenguaje con procesos infinitos

RESUMEN DEL VIDEO 12:
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 importantes:

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

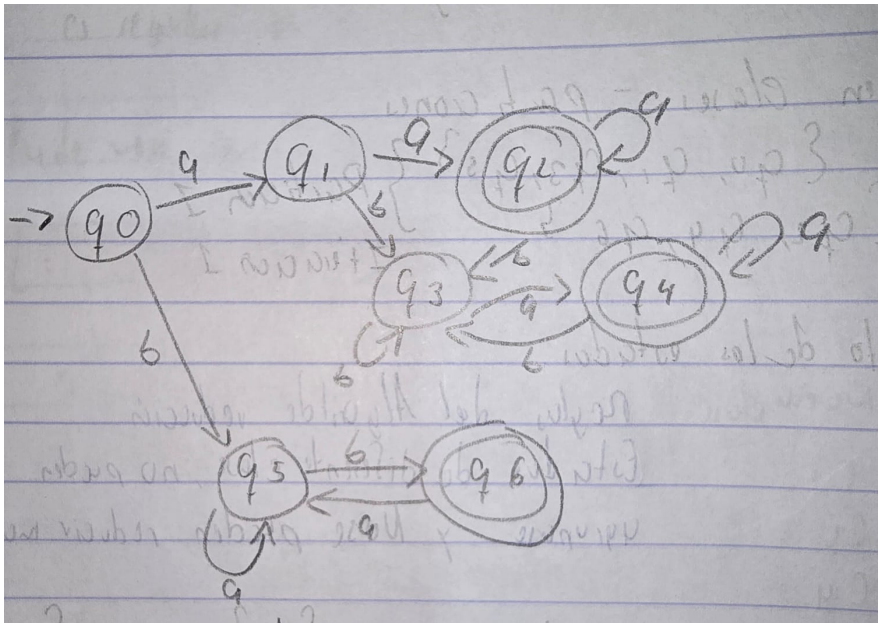

Figura 12: Autómata Completo

Primer paso: Agrupar los estados en una primera iteración o particiones

Se realiza la primera partición, dividiendo los estados en **dos clases**:

- **Estados No Finales:** $c_1 = \{q_0, q_1, q_3, q_5\}$
- **Estados Finales:** $c_2 = \{q_2, q_4, q_6\}$

Segundo paso: Estudiar el comportamiento de los estados
A continuación, se analiza cómo cada estado **transiciona** dentro de su clase y si existen estados que puedan fusionarse sin alterar el comportamiento del autómata.

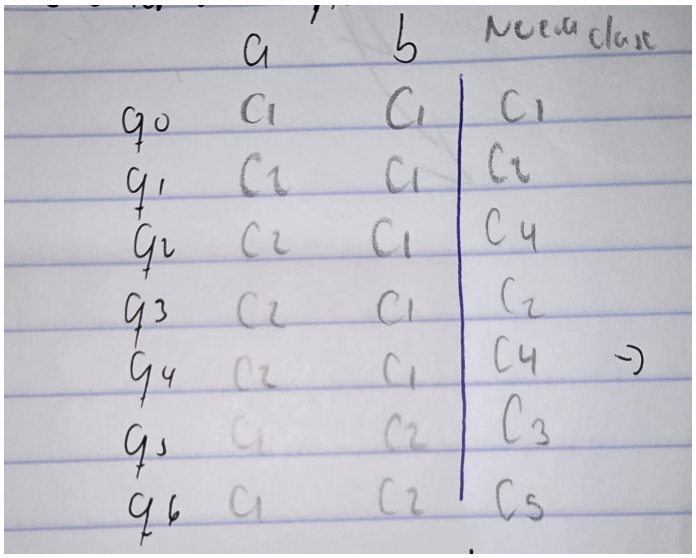

Figura 13: Comportamiento de los estados y la nueva clase

Paso 3: Agrupar los estados de mismas clases que tengan el mismo comportamiento
Se realiza una segunda partición refinando los conjuntos de estados en función de su comportamiento de transición.

Partición 2:

- $c_1 = \{q_0\}$
- $c_2 = \{q_1, q_3\}$
- $c_3 = \{q_5\}$
- $c_4 = \{q_2, q_4\}$

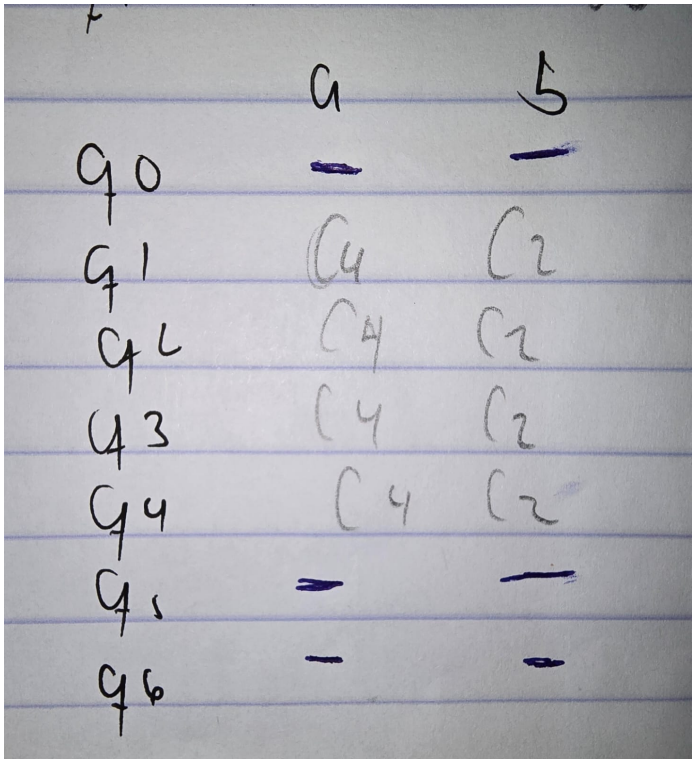

Figura 14: Comportamiento de la nueva clase

Paso 4: Construcción de la tabla de transición del nuevo autómata
Finalmente, construimos la tabla que define las transiciones del autómata reducido:

	a	**b**
c1	c2	c3
c2	c4	c2
c3	c3	c5
c4	c4	c2
c5	c3	c5

Cuadro 1: Tabla de transición del autómata reducido

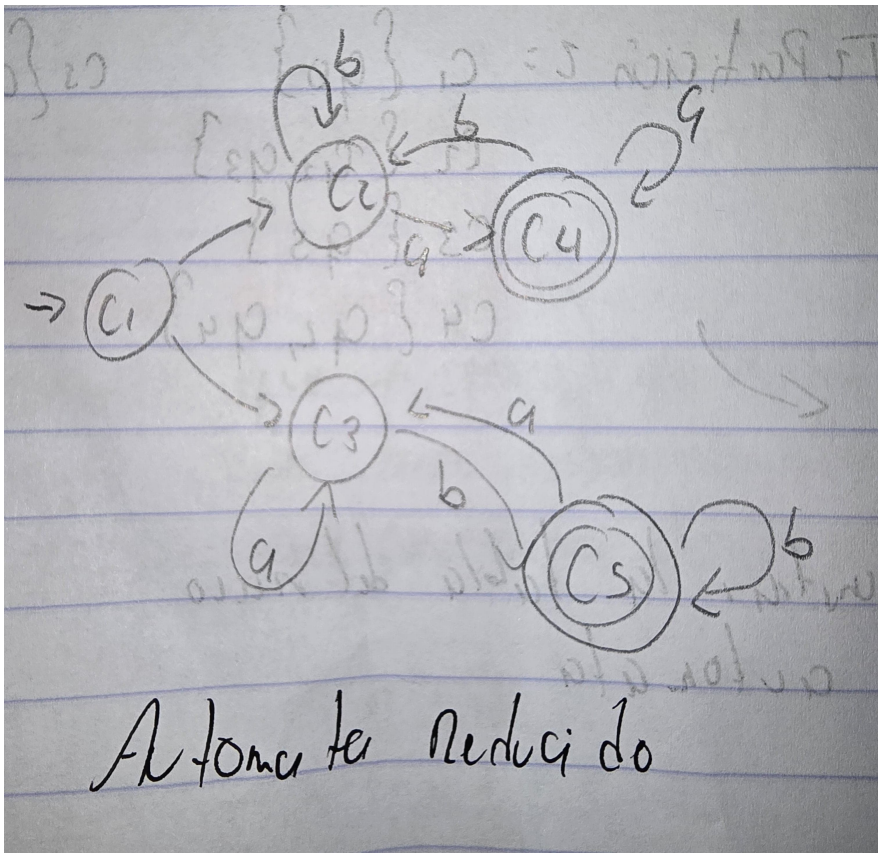

Figura 15: Nuevo Autómata Reducido

Este nuevo autómata es el resultado de aplicar el **algoritmo de reducción de estados** sin perjudicar ni alterar el comportamiento de los estados iniciales.
A través de este proceso, se han fusionado estados equivalentes, manteniendo la capacidad del autómata para reconocer el mismo lenguaje con una estructura más simplificada y eficiente.

5.Conclusiones

A lo largo de esta serie de videos, se ha desarrollado un conocimiento profundo sobre los **autómatas finitos, lenguajes formales y algoritmos esenciales** en el estudio de la teoría de la computación.
Uno de los aprendizajes más importantes fue comprender la diferencia entre un **Autómata Finito Determinista (AFD)** y un **Autómata Finito No Determinista (AFND)**, así como el proceso de **conversión entre ambos**.
El análisis del **Pattern Matching** y los **algoritmos de búsqueda** proporcionaron una base sólida para la comprensión de **expresiones regulares (Regex)** y su aplicación en la identificación de patrones en cadenas de texto.
Además la **reducción de estados en autómatas** resultó ser un proceso clave para la optimización de modelos computacionales, asegurando que los autómatas sean más eficientes sin perder su capacidad de reconocimiento de lenguajes.
Hubo momentos de **"WOW"** al notar cómo un autómata aparentemente complicado podía simplificarse con el **algoritmo de reducción de estados**, logrando una representación más clara y manejable. También surgieron algunos **"WTF"** al intentar demostrar que ciertos lenguajes **NO son regulares**, especialmente cuando se analizó la cantidad infinita de clases de equivalencia necesarias para ciertos lenguajes.

6.Preguntas y respuestas

Cuestionario sobre Autómatas y Lenguajes Formales:

1 ¿Cuál es la diferencia principal entre un Autómata Finito Determinista (AFD) y un Autómata Finito No Determinista (AFND)?
Respuesta:
Un AFD tiene una única transición definida para cada estado y símbolo del alfabeto, mientras que un AFND puede tener múltiples transiciones para un mismo símbolo, permitiendo que el autómata esté en varios estados simultáneamente.

2 ¿Qué representa el lenguaje por la derecha en un autómata?
Respuesta:
El lenguaje por la derecha de un estado en un autómata es el conjunto de cadenas que llevan desde ese estado hasta un estado de aceptación. Se usa para definir el comportamiento de los estados y analizar cómo llegan a la aceptación.

3 ¿Cuáles son las tres propiedades que debe cumplir una relación para ser una relación de equivalencia?
Respuesta:
Una relación es una relación de equivalencia si cumple:
Reflexiva: Todo elemento se relaciona consigo mismo.
Simétrica: Si un elemento se relaciona con otro, el otro también se relaciona con el primero.
Transitiva: Si un elemento se relaciona con otro y este con un tercero, entonces el primero se relaciona con el tercero.

4¿Cómo se puede demostrar que un lenguaje NO es regular?
Respuesta:
Un lenguaje no es regular si tiene infinitas clases de equivalencia en su relación de equivalencia RL. Esto significa que no se puede construir un autómata finito que lo reconozca, ya que los autómatas finitos solo pueden manejar un número finito de estados. Un ejemplo de lenguaje no regular es L = 01 — n 0, ya que requiere memoria infinita para verificar que el número de 0's y 1's sea igual.

5¿Qué es el algoritmo de reducción de estados en un autómata y qué condiciones deben cumplirse antes de aplicarlo?
Respuesta:
El algoritmo de reducción de estados es un proceso de partición que permite simplificar un autómata fusionando estados equivalentes sin alterar su comportamiento.
Antes de aplicarlo, el autómata debe ser:
Accesible: Todos los estados deben ser alcanzables desde el estado inicial.
Completo: Para cada estado y cada símbolo del alfabeto, debe haber una transición definida.

Referencias

[1] Helo G., J. E. (1999). Redes neuronales y autómatas finitos. Tecnología en Marcha, 13(2), 96-103.

[2] Gutiérrez Giraldi, O., Martínez Moreno, Martha, autora, Pérez Hernández, Pedro Jesús, autor. (2024). Algoritmo de conversión de una gramática libre de contexto a un autómata de pila = Conversion algorithm from a context-free grammar to a push down automata. Universidad Autónoma del Estado de Hidalgo.

[3] García, Pedro, colab. (2001). Teoría de automatas y lenguajes formales. Alfaomega.

[4] Codemath. (2023, December 4). Operaciones con palabras — Lenguajes Formales II [Video]. YouTube. https://www.youtube.com/watch?v=MXDl4Ts$_E$Z0

[5] Codemath. (2023, December 15). Operaciones con lenguajes y aplicaciones — Lenguajes Formales III [Video]. YouTube. https://www.youtube.com/watch?v=uU-fNuwbmZg

[6] Codemath. (2024, January 29). Descubre los autómatas: El corazón de la computación [Video]. YouTube. https://www.youtube.com/watch?v=pMIwci0kMv0

[7] Codemath. (2024, February 4). Qué es un autómata finito determinista (AFD) [Video]. YouTube. https://www.youtube.com/watch?v=d9aEE-uLmNE

[8] Codemath. (2024, April 23). Qué es un autómata finito no determinista (AFND) [Video]. YouTube. https://www.youtube.com/watch?v=dIgKBNuaglE

[9] Codemath. (2024, April 29). Convertir un autómata NO determinista (AFND) a determinista (AFD) [Video]. YouTube. https://www.youtube.com/watch?v=hzJ8CNdPElc

[10] Codemath. (2024, May 5). Qué es un autómata con transiciones épsilon [Video]. YouTube. https://www.youtube.com/watch?v=71P3daDZWlQ

[11] Codemath. (2024, May 11). Convertir un AFND con transiciones a un AFND [Video]. YouTube. https://www.youtube.com/watch?v=1yKBT8gWN-Y

[12] Codemath. (2024, May 27). Pattern matching con autómatas: Mejora tus algoritmos [Video]. YouTube. https://www.youtube.com/watch?v=22XqyZLhKPg

[13] Codemath. (2024, June 22). Clases de equivalencia en autómatas y lenguajes formales [Video]. YouTube. https://www.youtube.com/watch?v=JuTuMe8Q58c

[14] Codemath. (2024, July 1). Demostrar que un lenguaje es regular - Teorema de Myhill-Nerode [Video]. YouTube. https://www.youtube.com/watch?v=gYOvlrjRBwg

[15] Codemath. (2024, July 8). Demostrar que un lenguaje NO es regular - Teorema de Myhill-Nerode [Video]. YouTube. https://www.youtube.com/watch?v=FPWpCq20g0o

[16] Codemath. (2025, February 13). Minimización de estados de un autómata explicada desde cero [Video]. YouTube. https://www.youtube.com/watch?v=gd6uyNXsqcw