Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 0

PRACTICA 0

ALUMNO: Carlos Alberto Sánchez Lara
Dr. Eduardo Cornejo-Velázquez

1. Introducción

La biblioteca de la universidad es un espacio realmente util al que cualquier estudiante puede acceder de forma presencial o digital. Funciona como un centro de conocimiento donde las personas pueden acceder a una vista de colecciones de libros, revistas y recursos digitales. Esta instalación de la universidad es bastante recomendado para hacer la busqueda de cualquier información necesaria en cualquier licenciatura debido a su basto catalogo de libros físicos y aun mas en formato digital.

2. Objetivo

Conocer y visualizar cada uno de los servicios que ofrece la biblioteca central de la UAEH, además de conocer que servicios y estancias de libros sirven para la materia de Automatas y Compiladores con el fin de ampliar nuestro conocimiento en las bases de los lenguajes formales y sus diferentes operaciones.

3. Marco Teórico

Lenguajes formales

Un lenguaje formal es un conjunto de cadenas de símbolos definidas sobre un alfabeto y estructuradas de acuerdo con reglas sintácticas precisas. A diferencia del lenguaje natural, que puede ser ambiguo y flexible, los lenguajes formales tienen una definición estricta que permite su análisis matemático y su implementación en sistemas computacionales. Para describir un lenguaje formal, se pueden emplear gramáticas formales, las cuales establecen las reglas de formación de las cadenas válidas dentro del lenguaje. Estas reglas pueden expresarse mediante expresiones regulares, autómatas finitos o gramáticas de Chomsky, dependiendo del nivel de complejidad del lenguaje. Dado su carácter estrictamente definido, los lenguajes formales permiten realizar operaciones computacionales sobre ellos, como análisis sintáctico, verificación de propiedades y transformación de estructuras, lo que los convierte en herramientas esenciales en la computación y las matemáticas.

Automatas

Un autómata es un modelo matemático de una máquina abstracta que procesa una secuencia de símbolos de acuerdo con un conjunto de reglas predefinidas. Se utiliza para representar y analizar sistemas que responden a entradas de manera secuencial y determinista o no determinista.

Operaciones con Lenguajes

Las operaciones más comunes con leguajes son:

- Unión: Combina dos lenguajes.
- Concatenación: Forma palabras concatenando elementos de diferentes lenguajes.
- Cierre de Kleene: Incluye todas las posibles cadenas que se pueden formar mediante la concatenación repetida de un lenguaje, incluyendo la palabra vacia.

3. Herramientas empleadas

Youtube

Se utilizaron recursos educativos en Youtube para comprender y profundizar en temas clave de lenguajes formales y autómatas. Los videos proporcionan una forma accesible y visual de explicar conceptos complejos.

Libros

Se utilizaron libros de la biblioteca para expandir la información sobre algunos conceptos clave que se trataban en los videos, ademas de proporcionarnos una fuente confiable de información

Editor de texto Latex

Para la redacción del informe y la organización del marco teórico, se empleó Latex, una herramienta eficiente para la creación de documentos científicos, que permite organizar y estructurar contenido de manera clara y precisa.

4. Desarrollo

Fundamentos básicos de los lenguajes formales

Los lenguajes formales son fundamentales en la teoría de la computación y están definidos por conjuntos de reglas matemáticas. En su base, encontramos el alfabeto, que es un conjunto finito de símbolos utilizados para construir cadenas o secuencias finitas de símbolos del alfabeto. Dentro de estas, la cadena vacía (representada comúnmente por) es una secuencia especial que no contiene símbolos. Las cadenas pueden poseer diversas propiedades, como su longitud o la posibilidad de ser comparadas mediante operaciones formales. Además, las combinaciones de cadenas forman estructuras más complejas dentro de un lenguaje, permitiendo la construcción de palabras más largas mediante operaciones algebraicas. Un concepto clave en este contexto es la clausura de Kleene, que permite generar todas las posibles concatenaciones de las palabras de un lenguaje, incluyendo la cadena vacía, lo que es esencial en la definición de gramáticas y autómatas formales.

Operaciones

El tema a tratar después de lo básico es la concatenación, la cual es muy importante por sus usos en los lenguajes de programación, básicamente es la unión de una palabra después de otra, no tiene un límite en cuántas palabras se pueden juntar, y aquí también se puede usar la cadena vacía, la concatenar una palabra con la cadena vacía esta nos da la palabra sin modificaciones. La segunda operación es la potencia, la cual en resumidas cuentas es concatenar la misma palabra n veces consigo misma, si se eleva a cualquier palabra a cero nos da la cadena vacía. Después se tratan los conceptos de prefijos, sufijos y segmentos de una palabra, los prefijos se van consiguiendo desde la izquierda y van en aumento, y para los sufijos es lo mismo pero empezando desde la izquierda y respetando el orden, el primer prefijo y sufijo es la cadena vacía, y el segmento son las subcadenas dentro de la cadena principal. El reverso de una cadena es la operación que invierte el orden de los símbolos dentro de la cadena dada. Las palabras capicúas son las que se leen igual al derecho y al revés.
En esta parte nos hablan de las operaciones booleanas que nos permiten la manipulación y transformación de conjuntos de elementos mediante diversas reglas, facilitando su análisis y estructuración. En primer lugar, el producto combina elementos de dos conjuntos distintos para formar nuevas combinaciones, lo que resulta fundamental en la construcción de estructuras más complejas. A su vez, la potencia amplía esta idea al aplicar repetidamente una misma operación, generando así nuevas posibilidades dentro del conjunto original. Además, el cierre juega un papel clave, ya que garantiza la inclusión de todas las combinaciones posibles bajo ciertas reglas, asegurando que el sistema se mantenga estable y completo. Por otro lado, el cociente permite descomponer conjuntos al establecer relaciones específicas entre sus elementos, lo que facilita su organización y comprensión. Finalmente, el homomorfismo actúa como un puente entre diferentes estructuras, preservando propiedades esenciales y permitiendo una transformación sin pérdida de información. En conjunto, estas operaciones proporcionan herramientas fundamentales para modelar, analizar y optimizar sistemas en distintos campos del conocimiento.

Autómatas

Un autómata es una máquina abstracta que modela procesos de cálculo. Estos sistemas son fundamentales en la teoría de la computación y la lingüística formal, ya que permiten modelar procesos de toma de decisiones automáticos. Existen diferentes tipos de autómatas, los autómatas finitos son los más simples, capaces de reconocer lenguajes regulares mediante un número finito de estados. Por otro lado, los autómatas de pila y los autómatas de Turing tienen estructuras más complejas, permitiendo el reconocimiento de lenguajes contextuales y recursivos, respectivamente. Los autómatas tienen aplicaciones variadas, desde la construcción de compiladores y el procesamiento de cadenas de texto hasta la simulación de procesos físicos y la automatización de sistemas de control. Cada tipo de autómata es elegido según la complejidad del problema a resolver y la necesidad de capacidad de procesamiento.

Autómatas Finitos Deterministas

El Autómata Finito Determinista (AFD) es una herramienta esencial para modelar sistemas que procesan cadenas de símbolos de manera ordenada y predecible. Para construir un AFD, se debe identificar cómo cada entrada puede llevar al autómata a un estado determinado, y cómo este proceso se desarrolla a lo largo de toda la cadena. A medida que avanzamos en este proceso, es útil representar las transiciones de manera clara mediante una tabla de transiciones, la cual facilita visualizar cómo el autómata cambia de estado dependiendo de los símbolos que recibe. A su vez, esta tabla puede convertirse en un gráfico, lo que nos lleva a los autómatas con grafos, una representación visual que no solo mejora la comprensión del comportamiento del AFD, sino que también hace más sencillo el análisis de su funcionamiento. En conjunto, estas herramientas permiten estudiar lenguajes regulares y desarrollar aplicaciones en áreas tan diversas como los compiladores, el procesamiento de texto y el análisis de datos.

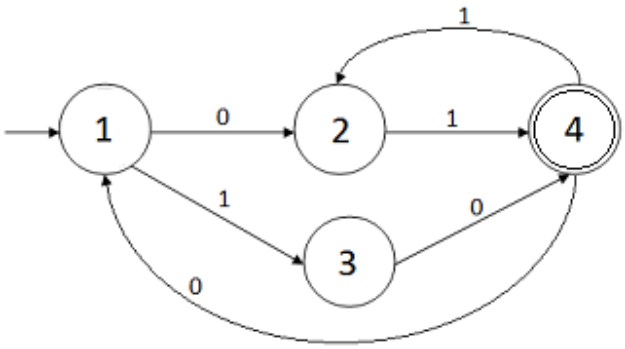

Figure 1: Representación de Automata Finito Determinista

Autómata Finito No Determinista

La construcción de un Autómata Finito No Determinista (AFND) implica un enfoque ligeramente diferente al de su contraparte determinista. A diferencia del AFD, el no determinismo permite que el autómata tenga múltiples transiciones posibles para un mismo símbolo de entrada desde un estado determinado, lo que significa que no hay un único camino predecible. Esta característica hace que el diseño de un AFND sea más flexible y adecuado para modelar lenguajes más complejos. En cuanto a la función de transición, en un AFND, no se limita a un único estado de destino por cada entrada, sino que puede haber varios posibles destinos, lo que genera una mayor libertad en su comportamiento. La representación de un AFND, a menudo a través de diagramas de transición, refleja esta característica al mostrar múltiples flechas saliendo de un mismo estado para un mismo símbolo. A pesar de estas diferencias, un AFND está estrechamente relacionado con un AFD, ya que ambos pueden reconocer los mismos lenguajes regulares. De hecho, cualquier AFND puede ser convertido en un AFD mediante un proceso conocido como determinización, lo que demuestra cómo ambos modelos, aunque distintos, tienen una relación fundamental en el estudio de la teoría de los autómatas.

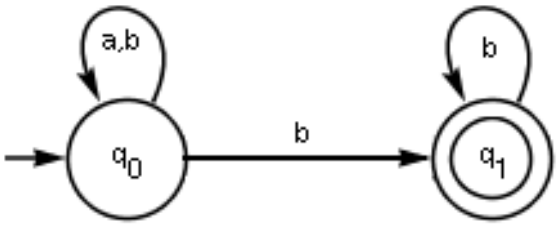

Figure 2: Representación de Automata Finito Determinista

Conversion de un AFND a un AFD

Convertir un Autómata Finito No Determinista (AFND) a Determinista (AFD), la principal diferencia entre ambos es que el AFND permite múltiples transiciones para un mismo símbolo desde un estado dado, mientras que el AFD requiere que haya una única transición definida para cada símbolo de entrada en cada estado. Para llevar a cabo esta conversión, se emplea un proceso conocido como determinización, que consiste en crear un AFD que, aunque puede tener más estados, es capaz de aceptar el mismo conjunto de cadenas que el AFND original. Este proceso se realiza identificando los "conjuntos de estados" del AFND, y luego asociando a cada conjunto un nuevo estado en el AFD. Así, el proceso asegura que el AFD resultante sea equivalente al AFND, pero con la diferencia crucial de que en el AFD, las transiciones son únicas y claramente definidas. Aunque la conversión puede aumentar el número de estados en el autómata, esta transformación facilita la implementación de los autómatas en aplicaciones prácticas, como el análisis léxico en compiladores y el procesamiento de cadenas en sistemas informáticos.

Autómata con transiciones epsilon

Al repasar los conceptos de Autómatas Finitos Deterministas (AFD) y No Deterministas (AFND), es crucial entender las equivalencias entre ambos tipos de autómatas, a pesar de sus diferencias en cuanto a transiciones. Mientras que el AFD sigue reglas estrictas donde cada estado tiene una única transición para cada símbolo, el AFND permite múltiples transiciones desde un mismo estado, lo que genera una mayor flexibilidad. Sin embargo, ambos pueden reconocer los mismos lenguajes regulares, y cualquier AFND puede ser convertido en un AFD. Un concepto clave que aparece en estos autómatas es la cadena vacía, que representa una secuencia de entrada que no requiere ningún símbolo para ser procesada, pero que aún así puede ser aceptada por el autómata. En el contexto de los AFND, se puede introducir la construcción de AF, un tipo de autómata que puede hacer transiciones sin consumir símbolos de entrada, es decir, utilizando transiciones epsilon (). Esto amplía las capacidades del AFND al permitir movimientos entre estados sin necesidad de procesar símbolos específicos. Finalmente, es importante considerar la conmutación de estados, que se refiere a la capacidad de un autómata de cambiar de un estado a otro en función de la entrada recibida. En los AFND, esta conmutación puede ser más compleja debido a las múltiples transiciones posibles, mientras que en los AFD, la conmutación está más controlada y definida, asegurando que siempre haya una transición única para cada símbolo de entrada.

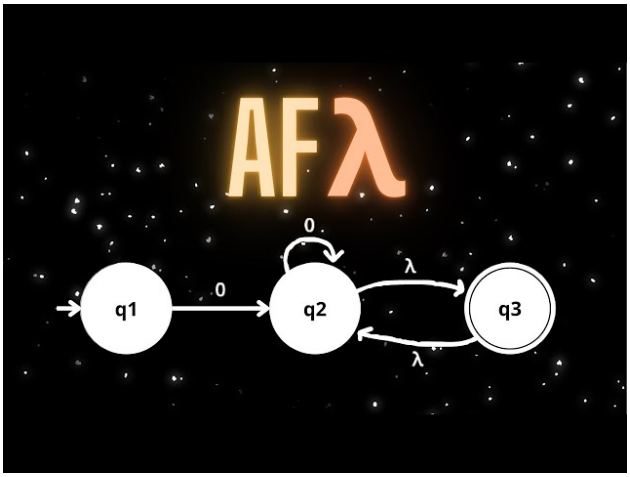

Figure 3: Representación de un Automata con transiciones epsilon

Convertir un AFND con transiciones epsilon a otro AFND

El proceso comienza identificando todos los estados que pueden ser alcanzados por transiciones desde cualquier estado dado, lo que implica que, al realizar una transición, el autómata no consume ningún símbolo y puede llegar a nuevos estados incluso sin procesar ninguna entrada. Este conjunto de estados alcanzables por se denomina cierre , y es fundamental para la conversión. Al realizar la conversión, se modifica el autómata para que las transiciones sean eliminadas, creando nuevas transiciones que correspondan a las posibles transiciones de entrada, pero considerando los estados que se alcanzan a través de . Como resultado, el AFND convertido se ajusta para que las transiciones sean completamente dependientes de los símbolos de entrada, eliminando la necesidad de transiciones , pero manteniendo la capacidad del autómata para reconocer el mismo lenguaje. Esta conversión simplifica el autómata sin perder su poder expresivo, y es útil cuando se busca obtener una representación más precisa de las operaciones de un AFND sin las transiciones vacías.

Pattern Matching

El Pattern Matching es una técnica esencial en la computación y el procesamiento de textos que permite identificar la ocurrencia de un patrón dentro de una secuencia de caracteres o datos. Uno de los métodos más básicos para llevarlo a cabo es el Naive Algorithm, el cual compara de manera secuencial cada posible posición del patrón dentro del texto, sin realizar optimizaciones avanzadas, aunque su implementación es sencilla, puede ser ineficiente en textos largos. Por otro lado, el Autómata Diccionario es un enfoque más avanzado que utiliza estructuras de datos como los autómatas finitos para mejorar la búsqueda de múltiples patrones simultáneamente. Este método permite una mayor eficiencia al procesar grandes volúmenes de información, como en la detección de palabras clave en documentos extensos o la búsqueda rápida en bases de datos.

Clases de equivalencia

El concepto de lenguaje por la derecha se refiere a una forma de estudiar los lenguajes a través de la estructura de sus cadenas, observando las propiedades de las secuencias que pueden ser extendidas de manera coherente a partir de ciertos patrones. Una herramienta fundamental para comprender cómo se agrupan las cadenas dentro de un lenguaje es la relación de equivalencia, que establece un criterio bajo el cual dos cadenas son consideradas equivalentes si pueden ser extendidas de manera similar, sin alterar el lenguaje al que pertenecen. Esta relación ayuda a definir cómo se dividen las cadenas en clases de equivalencia, donde las cadenas dentro de cada clase comparten una propiedad común en cuanto a su relación con el resto del lenguaje. Este enfoque es particularmente relevante para los lenguajes regulares, los cuales pueden ser reconocidos por autómatas finitos. Los lenguajes regulares tienen una estructura que puede ser comprendida de manera precisa a través de la relación de equivalencia, permitiendo que se clasifiquen en grupos de cadenas equivalentes que siguen las mismas reglas de transición. Un concepto clave que se deriva de esta relación de equivalencia es la equivalencia de Nerode, que proporciona una forma formal de caracterizar los lenguajes regulares mediante la identificación de los conjuntos de cadenas que pueden ser extendidas de manera similar, facilitando así su análisis y clasificación. Esta equivalencia de Nerode no solo ayuda a comprender los lenguajes regulares de una manera más estructurada, sino que también juega un papel crucial en la construcción de autómatas que los reconocen, ofreciendo una herramienta esencial para la teoría de autómatas y lenguajes formales.

Como demostrar que un lenguaje si es un lenguaje regular

El Teorema de Myhill-Nerode constituye una herramienta esencial en la teoría de lenguajes formales, que permite demostrar si un lenguaje es regular. Este teorema establece que un lenguaje es regular si y solo si el número de clases de equivalencia de Nerode es finito. Para aplicar esta idea, se examinan las cadenas del lenguaje y se analizan las posibles maneras en que estas pueden ser extendidas por otras secuencias sin alterar su pertenencia al lenguaje en cuestión. Si el número de clases de equivalencia es finito, significa que el lenguaje puede ser reconocido por un autómata finito, lo que confirma su regularidad. Este teorema no solo ofrece un marco teórico robusto para la clasificación de lenguajes, sino que también sirve como una clave decisiva para el diseño y análisis de autómatas en la teoría formal de lenguajes.

Como demostrar que un lenguaje no es un lenguaje regular

El Teorema de Myhill-Nerode también ofrece una vía para demostrar que un lenguaje no es regular. Según este teorema, si el número de clases de equivalencia de Nerode de un lenguaje es infinito, entonces dicho lenguaje no puede ser reconocido por un autómata finito, lo que implica que no es regular. En este contexto, se analiza la posibilidad de que existan infinitas clases de cadenas que no puedan ser diferenciadas por un número finito de estados en un autómata. Si se identifica que hay una cantidad infinita de tales clases, la regularidad del lenguaje se ve refutada, ya que un autómata finito no puede manejar un número infinito de clases de equivalencia. Este enfoque permite una demostración formal y rigurosa de la no regularidad de ciertos lenguajes, estableciendo límites claros sobre las capacidades de los autómatas finitos y las estructuras de los lenguajes que pueden reconocer.

5. Conclusiones

Los lenguajes formales son un tema muy extenso, el cual no solo abarca conceptos importantes para la computación sino tambien para la vida diaria, cabe mecionar su capacidad para definir estructuras sintácticas precisas permite la creación de compiladores eficientes, la verificación de programas y la automatización de procesos complejos. A medida que la informática avanza, el estudio de los lenguajes formales seguirá siendo esencial para mejorar la seguridad, la inteligencia artificial y la comprensión de los límites del cómputo.

Preguntas

1. ¿Qué es un alfabeto en los lenguajes formales?
   (a) Un conjunto de números enteros.
   (b) **Un conjunto finito de símbolos.** (Respuesta correcta)
   (c) Un lenguaje de programación.
   (d) Un autómata.
2. ¿Cuál de las siguientes opciones representa una palabra en un lenguaje formal?
   (a) {0, 1, 2, 3}
   (b) {a, b, c}
   (c) **abba** (Respuesta correcta)
   (d) {ε, a, aa, aba}
3. ¿Qué operador representa la "Clausura de Kleene" en los lenguajes formales?
   (a) +
   (b) * (Respuesta correcta)
   (c) -

(d) /

4. Si $\Sigma = \{a, b\}$, ¿cuál de las siguientes opciones NO es una palabra válida sobre Σ?

   (a) aba

   (b) bbab

   (c) **abc** (Respuesta correcta)

   (d) aab

5. ¿Qué define un lenguaje formal?

   (a) Un conjunto de reglas gramaticales de un idioma natural.

   (b) **Un conjunto de palabras formadas a partir de un alfabeto y siguiendo ciertas reglas.** (Respuesta correcta)

   (c) Un lenguaje de programación con estructura sintáctica.

   (d) Un conjunto de números primos.

Fotos

Figure 4: Foto en biblioteca

Figure 5: Foto en biblioteca

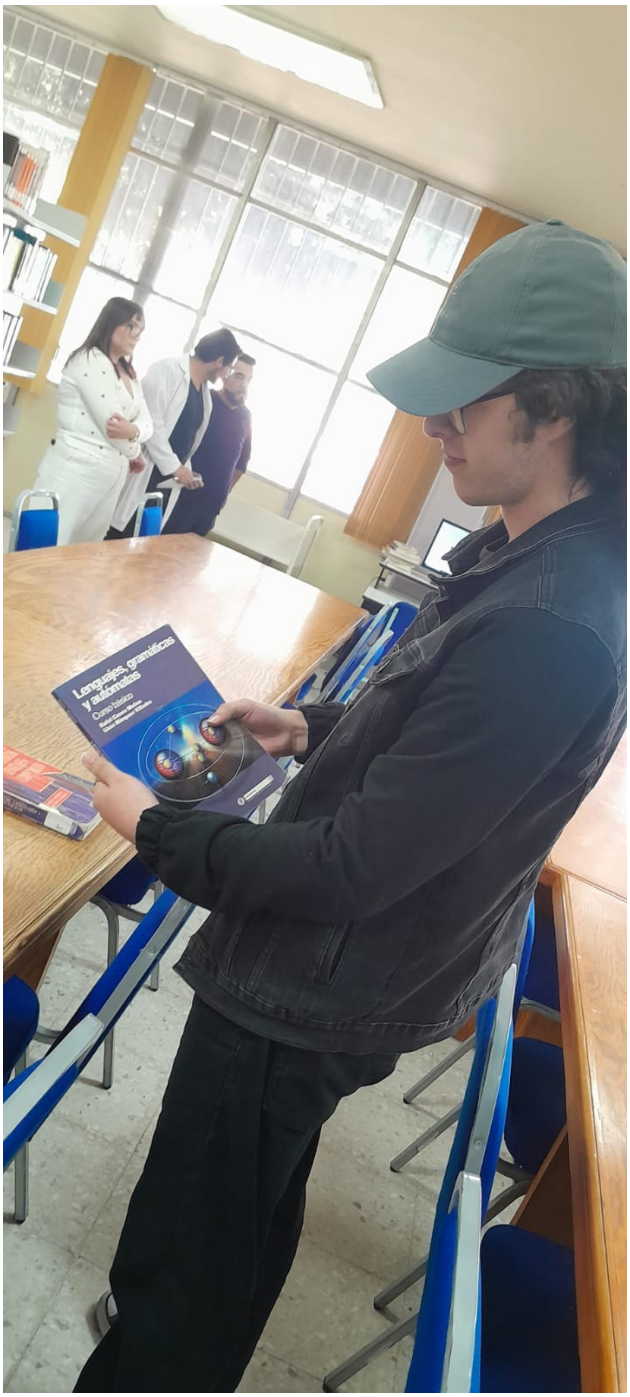

Figure 6: Foto en biblioteca

Referencias Bibliográficas

References

[1] Codemath. (28 de noviembre de 2023). *Lenguajes Formales desde CERO — Palabra, Alfabeto y Clausura de Kleene* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=$_U dVL - 84rXc$

[2] Codemath. (4 de diciembre de 2023). *Operaciones con Palabras — Lenguajes Formales II* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=MXDl4Ts$_E Z0$

[3] Codemath. (28 de noviembre de 2023). *Lenguajes Formales desde CERO — Palabra, Alfabeto y Clausura de Kleene* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=$_U dVL - 84rXc$

[4] Codemath. (4 de diciembre de 2023). *Operaciones con Palabras — Lenguajes Formales II* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=MXDl4Ts$_E Z0$

[5] Codemath. (15 de diciembre de 2023). *Operaciones con Lenguajes y Aplicaciones — Lenguajes Formales III* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=uU-fNuwbmZg

[6] Codemath. (29 de enero de 2024). *Descubre los autómatas: el corazón de la computación* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=pMIwci0kMv0

[7] Codemath. (4 de febrero de 2024). *Qué es un Autómata Finito Determinista (AFD)* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=d9aEE-uLmNE

[8] Codemath. (23 de abril de 2024). *Qué es un Autómata Finito No Determinista (AFND)* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=dIgKBNuaglE

[9] Codemath. (29 de abril de 2024). *Convertir un Autómata NO Determinista (AFND) a Determinista (AFD)* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=hzJ8CNdPElc

[10] Codemath. (5 de mayo de 2024). *Qué es un Autómata con Transiciones Epsilon* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=71P3daDZWlQ

[11] Codemath. (11 de mayo de 2024). *Convertir un AFND con Transiciones a un AFND* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=1yKBT8gWN-Y

[12] Codemath. (27 de mayo de 2024). *Pattern Matching con Autómatas: Mejora tus Algoritmos* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=22XqyZLhKPg

[13] Codemath. (22 de junio de 2024). *Clases de Equivalencia en Autómatas y Lenguajes Formales* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=JuTuMe8Q58c

[14] Codemath. (1 de julio de 2024). *Demostrar que un Lenguaje es Regular - Teorema de Myhill-Nerode* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=gYOvlrjRBwg

[15] Codemath. (8 de julio de 2024). *Demostrar que un Lenguaje NO es Regular - Teorema de Myhill-Nerode* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=FPWpCq20g0o

[16] Codemath. (13 de febrero de 2025). *Minimización de estados de un autómata explicada desde cero* [Vídeo]. YouTube. Disponible en: https://www.youtube.com/watch?v=gd6uyNXsqcw