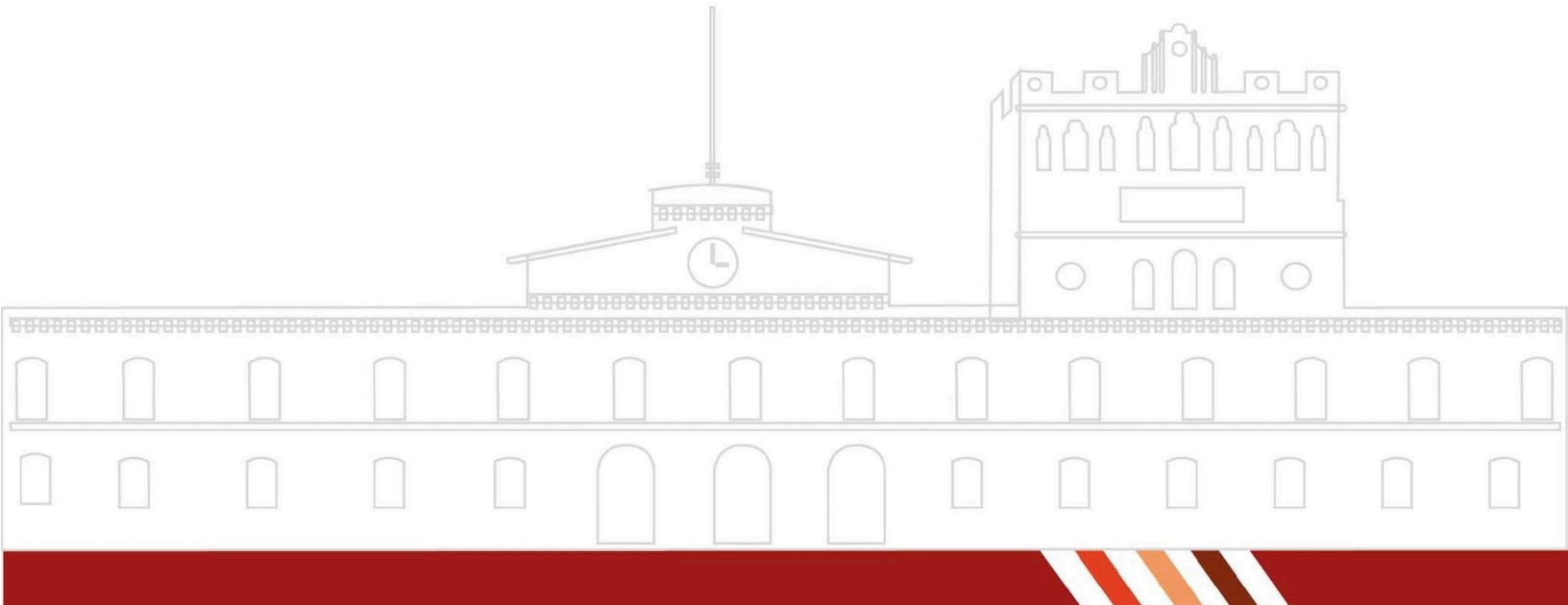
REPORTE DE PRÁCTICA NO. 1

LENGUAJES FORMALES Y AUTÓMATAS

POR: GÓMEZ ALAMILLA BRYAN TONINHO



Introducción

Los lenguajes formales representan un punto clave de convergencia entre las matemáticas y la lingüística, ya que ofrecen mecanismos formales para estudiar los números como si fueran secuencias de símbolos o palabras.Esta área constituye el fundamento de la teoría de autómatas y permite analizar propiedades interesantes como los números capicúa, los cuales se leen igual de izquierda a derecha que al revés. Estas herramientas conceptuales permiten explorar conjuntos infinitos y modelar con precisión el tratamiento de la información. En este informe se recopilan los elementos centrales abordados en los diez videos revisados, abarcando desde los principios de los lenguajes formales hasta aplicaciones más complejas como la detección de patrones

Marco teórico

Alfabetos y cadenas

Un alfabeto se define como un conjunto limitado de símbolos que sirve como base para construir un lenguaje formal. Por ejemplo, el alfabeto español incluye las letras de la A a la Z, mientras que el alfabeto binario únicamente contiene los dígitos 0 y 1.

Una cadena (también llamada palabra) es una secuencia finita, ordenada y construida a partir de los símbolos de un alfabeto. Por ejemplo, "codem" es válida dentro del alfabeto español, pero no tendría sentido dentro del binario.

Un elemento clave en esta teoría es la cadena vacía, representada por ε, que es una cadena sin símbolos (longitud cero) y cumple el rol de identidad en operaciones como la concatenación.

Operaciones básicas sobre cadenas

Algunas de las operaciones fundamentales que se pueden realizar con cadenas son:

- **Concatenación:** Une dos cadenas para formar una nueva.
  Ejemplo: "monta" concatenado con "puercos" resulta en "montapuercos".

- **Potenciación:** Repite una cadena un número determinado de veces.
  Ejemplo: “code” elevado al cubo produce “codecodecode”.

- **Prefijos, sufijos y subcadenas:** Se refiere a partes o fragmentos de una cadena.
  Por ejemplo, para "100":

  - Prefijos: ε, 1, 10, 100

  - Sufijos: ε, 0, 00, 100

- **Reverso:** Invierte el orden de los símbolos de la cadena.
  Ejemplo: reverso("Hola") da como resultado "aloH".

Definición de lenguajes formales

Un **lenguaje formal** es un conjunto de cadenas derivadas de un alfabeto, usualmente representado como un subconjunto de su clausura de Kleene (Σ^*). Un lenguaje puede ser finito o contener una cantidad infinita de cadenas.

Ejemplos:

- Lenguaje de todas las palabras que inician con "a":
  $L = \{ax \mid x \in \Sigma^*\}$

- Lenguaje de cadenas cuya longitud es menor a 5:
  $L = \{x \in \Sigma^* \mid |x| < 5\}$

Operaciones sobre lenguajes

Al igual que los conjuntos, los lenguajes pueden ser manipulados mediante operaciones como la **unión**, la **intersección** y el **complemento**. También existen operaciones particulares, entre ellas:

- **Producto**: Combina palabras de dos lenguajes mediante concatenación.
- **Potencia y cierre**: Aplican el concepto de repetición sobre lenguajes completos.
- **Cociente**: Consiste en eliminar prefijos o sufijos comunes de las palabras de un lenguaje.
- **Homomorfismo**: Mapea símbolos de un lenguaje a símbolos de otro alfabeto, transformando las cadenas.

Autómatas finitos

Un **autómata** es una estructura teórica capaz de leer cadenas de entrada y decidir si estas cumplen ciertas reglas, devolviendo una respuesta de aceptación o rechazo

Los **autómatas finitos deterministas (AFD)** se representan como una tupla $(Q, \Sigma, \delta, q_0, F)$, donde:

- Q: Conjunto finito de estados
- Σ: Alfabeto de entrada
- δ: Función de transición, que asigna un estado a cada par (estado actual, símbolo)
- q_0: Estado inicial
- F: Conjunto de estados de aceptación

Autómatas no deterministas

Los **autómatas finitos no deterministas (AFND)** permiten múltiples rutas posibles desde un mismo estado para un mismo símbolo. En su definición, la función de transición entrega un conjunto de estados posibles: $\delta: Q \times \Sigma \rightarrow 2^Q$.

Todo AFD puede ser representado como un AFND equivalente, pero no todo AFND tiene una forma determinista simple.

Autómatas con transiciones vacías

Existen autómatas conocidos como **AF-ε** que pueden cambiar de estado sin consumir símbolos, es decir, mediante transiciones vacías (ε). Estas transiciones permiten una mayor flexibilidad, aunque se pueden eliminar al convertir el autómata en un AFND mediante el cálculo de la clausura-ε.

Detección de patrones (pattern matching)

El **pattern matching** es una técnica esencial en computación que permite localizar patrones específicos dentro de textos o secuencias de datos [10]. Algunos de los algoritmos utilizados incluyen:

- **Algoritmo básico o ingenuo**: Recorre todas las posiciones posibles, con un costo computacional elevado en algunos casos ($O(n \cdot m)$).
- **Autómata basado en diccionario**: Utiliza estructuras previas, como sufijos y prefijos, para agilizar la búsqueda, reduciendo el tiempo de ejecución a un costo lineal.

Herramientas utilizadas

1. **Teoría de conjuntos**: Proporciona el marco lógico para definir lenguajes, cadenas y símbolos.
2. **Grafos**: Facilitan la representación visual de los autómatas y sus transiciones.
3. **Notación matemática rigurosa**: Permite expresar con precisión los componentes y reglas de los autómatas.
4. **Algoritmos de conversión**: Se emplean para transformar un tipo de autómata en otro, garantizando equivalencias de comportamiento.

Desarrollo

Ejemplo 1: Autómata finito determinista para cadenas binarias que contienen al menos un 0

Se diseñó un autómata finito determinista (AFD) cuyo objetivo es aceptar aquellas cadenas binarias que contienen **al menos un dígito 0**. El AFD se define de la siguiente manera:

- **Conjunto de estados:** $Q = \{q_0, q_1\}$
- **Alfabeto:** $\Sigma = \{0, 1\}$
- **Estado inicial:** q_0
- **Estados de aceptación:** $F = \{q_1\}$
- **Función de transición δ:**
    - $\delta(q_0, 0) = q_1$
    - $\delta(q_0, 1) = q_0$
    - $\delta(q_1, 0) = q_1$
    - $\delta(q_1, 1) = q_1$

Este autómata acepta cadenas como "0", "10", "010", "1010", ya que contienen al menos un 0. En cambio, rechaza entradas como "1", "11" o "111", pues no cumplen con el criterio.

La representación gráfica del autómata muestra que desde el estado inicial **q_0** (no final), se permanece en el mismo estado al leer un **1**, pero en cuanto se detecta un **0**, se transita al estado **q_1** (final), desde donde se acepta cualquier secuencia posterior de 0s o 1s.

Ejemplo 2: Conversión de un AFND a un AFD equivalente

En este segundo ejemplo, se aplicó el **algoritmo de conversión** por medio del método de subconjuntos para transformar un autómata finito no determinista (AFND) a su versión determinista (AFD). A continuación, se describe el AFND original:

- **Estados:** $Q = \{q_0, q_1, q_2\}$
- **Alfabeto:** $\Sigma = \{a, b\}$
- **Estado inicial:** q_0
- **Estado final:** $F = \{q_2\}$
- **Transiciones δ:**
    - $\delta(q_0, a) = \{q_1, q_2\}$

- $\delta(q_0, b) = \{q_1\}$
- $\delta(q_1, a) = \{q_2\}$
- $\delta(q_1, b) = \{q_0\}$
- $\delta(q_2, a) = \{q_2\}$
- $\delta(q_2, b) = \{q_1\}$

Aplicando el método de subconjuntos, se genera el AFD equivalente con los siguientes componentes:

- **Estados del AFD:** $Q' = \{\{q_0\}, \{q_1\}, \{q_2\}, \{q_0,q_1\}, \{q_1,q_2\}, \{q_0,q_1,q_2\}\}$
- **Estado inicial:** $\{q_0\}$
- **Estados de aceptación:** Todos los subconjuntos que contienen q_2
- **Transiciones δ′:**
  - $\delta'(\{q_0\}, a) = \{q_1, q_2\}$
  - $\delta'(\{q_0\}, b) = \{q_1\}$
  - $\delta'(\{q_1, q_2\}, a) = \{q_2\} \cup \{q_2\} = \{q_2\}$
  - $\delta'(\{q_1, q_2\}, b) = \{q_0\} \cup \{q_1\} = \{q_0, q_1\}$
  - (Y así sucesivamente para cada subconjunto generado)

Este procedimiento evidencia que un AFND con tres estados puede transformarse en un AFD equivalente que reconoce el mismo lenguaje, aunque con una cantidad mayor de estados (en este caso, hasta 6 posibles combinaciones de subconjuntos). Sin embargo, se conserva completamente el comportamiento del autómata original.

5. Conclusiones

El estudio de los lenguajes formales y los autómatas ofrece una base sólida para la comprensión y el diseño de sistemas capaces de procesar información de forma precisa y eficiente. Los conceptos iniciales de alfabetos, cadenas y lenguajes son esenciales para el desarrollo de estructuras más avanzadas como los autómatas finitos.

La diferenciación entre autómatas deterministas y no deterministas permite abordar un mismo problema desde distintas perspectivas, destacando las ventajas de la no determinación en términos de diseño, y del determinismo en términos de eficiencia computacional.

Uno de los momentos más reveladores fue descubrir cómo los autómatas que incluyen **transiciones vacías (ε)** pueden transformarse en versiones sin ε-transiciones, sin perder capacidad de reconocimiento. Esta equivalencia demuestra la potencia del modelo.

Finalmente, las aplicaciones en **pattern matching** evidencian cómo estos conceptos teóricos tienen aplicaciones prácticas concretas, especialmente en la mejora de algoritmos de búsqueda en textos, donde el uso de autómatas optimizados puede reducir el tiempo de ejecución de cuadrático a lineal.

En resumen, la teoría de autómatas y lenguajes formales sigue siendo una herramienta esencial en áreas modernas como el diseño de compiladores, el procesamiento de lenguaje natural y la verificación automática de programas, consolidando su importancia incluso en la tecnología contemporánea.

Referencias Bibliográficas

[1] Canal de Lenguajes Formales. (2023). *Introducción a los lenguajes formales: Alfabetos y palabras* [Video]. YouTube. https://youtu.be/ UdVL-84rXc?si=Zs7SrEkDkI9Rz3MA

[2] Canal de Lenguajes Formales. (2023). *Operaciones con palabras y definición de lenguajes* [Video]. YouTube. https://youtu.be/MXDl4Ts EZ0?si=vm FcIcgdVgmOojY

[3] Canal de Lenguajes Formales. (2023). *Operaciones con lenguajes formales* [Video]. YouTube. https://youtu.be/uU-fNuwbmZg?si=pagfNDOZqySlOjjv

[4] Canal de Autómatas. (2023). *Introduccio´n a los autómatas* [Video]. YouTube. https://youtu.be/pMIwci0kMv0?si=mmHN-gSXu24OdnBp

[5] Canal de Autómatas. (2023). *Autómatas finitos deterministas (AFD)* [Video]. YouTube. https://youtu.be/d9aEE-uLmNE?si=JsItPkLMacFsJSd1

[6] Canal de Autómatas. (2023). *Autómatas finitos no deterministas (AFND)* [Video]. YouTube. https://youtu.be/dIgKBNuaglE?si=ABSDA6qFIvP1GYyl

[7] Canal de Autómatas. (2023). *Conversi´on de AFND a AFD* [Video]. YouTube. https://youtu.be/hzJ8CNdPElc?si=TutNvEEsa REey7B

[8] Canal de Autómatas. (2023). *Autómatas con transiciones vacías (AF-)* [Video]. YouTube. https://youtu.be/71P3daDZWlQ?si=XqakDHcsRlQ0-xO6

[9] Canal de Autómatas. (2023). *Conversi´on de AF- a AFND* [Video]. YouTube. https://youtu.be/1yKBT8gWN-Y?si=VsH9aHmwM7P2IyGK

[10] Canal de Pattern Matching. (2023). *Pattern matching con autómatas* [Video]. YouTube. https://youtu.be/1yKBT8gWN-Y?si=66SmNEaVkG1CdpxC

[11] Canal de Ensamblador. (2023). *Introducci´on al lenguaje ensamblador* [Video]. YouTube. https://youtu.be/JuTuMe8Q58c?si=bkXpOYdYjCHAsDmJ

[12] Canal de Ensamblador. (2023). *Instrucciones aritm´eticas y l´ogicas* [Video]. YouTube. https://youtu.be/gYOvlrjRBwg?si=-Ri5f5V222Bza2Wz

[13] Canal de Ensamblador. (2023). *Interrupciones y manejode E/S* [Video]. YouTube. https://youtu.be/FPWpCq20g0o?si=xfI7OkCD3MOuP2L6

[14] Canal de Optimización. (2023). *Optimizaci´on y buenaspr´acticas* [Video]. YouTube. https://youtu.be/gd6uyNXsqcw?si=VWw Nkl4Dh240Dzs

[15] Canal de Compiladores. (2023). *Introducci´on a compiladores* [Video]. YouTube. https://youtu.be/x4CugLUufTc?si=uZfBrxQhWqX-i0A9

[16] Canal de Compiladores. (2023). *Análisis l´exico y sint´actico* [Video]. YouTube. https://youtu.be/5dmyN5mWu1o?si=do9Sooj2TLnFQIMQ

[17] Canal de Compiladores. (2023). *Generaci´on de códigointermedio* [Video]. YouTube. https://youtu.be/qzVms1j23uE?si=HUblHGk0icLtUK9a

[18] Charte Ojeda, F., & Ru´ız Calder´on, V. M. (2020). *Lenguaje ensamblador.*

RA-MA Editorial.

[19] Alfonseca Moreno, M. (1998). *Compiladores e interpretes: teoria´ıa y práctica*. McGraw-Hill.

[20] Aho, A. V., Sethi, R., & Ullman, J. D. (2007). *Compiladores: principios, técnicas y herramientas* (2.a ed.). Pearson Educación.