Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 3

Autómatas y Compiladores

ALUMNO: López López Fernando
Dr. Eduardo Cornejo-Velázquez

1. Introducción

Los autómatas y lenguajes formales constituyen la base teórica de la computación moderna, proporcionando los fundamentos matemáticos para el diseño de compiladores, procesadores de texto y sistemas de verificación. Esta práctica explora los conceptos fundamentales de los autómatas finitos, lenguajes regulares y sus propiedades, demostrando la equivalencia entre diferentes representaciones y su aplicación en problemas del mundo real.

El estudio abarca desde los autómatas finitos deterministas (AFD) y no deterministas (AFND) hasta las expresiones regulares, analizando las operaciones de cierre, la minimización de estados y las técnicas para demostrar la regularidad de lenguajes. La comprensión de estos conceptos es esencial para cualquier profesional en ciencias de la computación, ya que establecen los límites de lo que puede ser computado eficientemente.

2. Marco teórico

Autómatas Finitos

Los autómatas finitos son modelos matemáticos de computación que procesan cadenas de símbolos y deciden si pertenecen o no a un lenguaje específico. Según [1], un autómata finito determinista (AFD) se define como una 5-tupla $(Q, \Sigma, \delta, q_0, F)$ donde Q es el conjunto de estados, Σ es el alfabeto, δ es la función de transición, q_0 es el estado inicial y F son los estados finales.

Lenguajes Regulares

Un lenguaje L es regular si existe algún autómata finito que lo reconoce. [2] establece que la clase de lenguajes regulares es cerrada bajo operaciones como unión, concatenación, clausura de Kleene, intersección y complemento.

Expresiones Regulares

Las expresiones regulares proporcionan una notación algebraica para describir lenguajes regulares. [3] define las expresiones regulares recursivamente a partir de casos base ($\emptyset$, λ, símbolos individuales) y operadores (unión, concatenación, clausura).

Equivalencia y Minimización

El teorema de Myhill-Nerode, según [4], establece que un lenguaje es regular si y solo si tiene un número finito de clases de equivalencia. La minimización de autómatas permite obtener el AFD mínimo único para un lenguaje regular dado.

3. Herramientas empleadas

Para el desarrollo de esta práctica se utilizaron las siguientes herramientas:

1. **Teoría de Autómatas**: Fundamentos matemáticos para la construcción y análisis de autómatas finitos, incluyendo las definiciones formales de AFD, AFND y autómatas con transiciones λ.
2. **Algoritmos de Conversión**: Técnicas sistemáticas para transformar entre diferentes representaciones de lenguajes regulares, incluyendo la conversión de AFND a AFD y de expresiones regulares a autómatas.
3. **Métodos de Demostración**: Técnicas formales como el lema de bombeo y el teorema de Myhill-Nerode para demostrar la regularidad o no regularidad de lenguajes.

4. Desarrollo

Autómatas Finitos y sus Variantes

Se estudiaron tres tipos principales de autómatas finitos:

- **AFD**: Autómatas con transiciones únicas por estado y símbolo
- **AFND**: Autómatas que permiten múltiples transiciones desde un estado con el mismo símbolo
- **AF-λ**: Autómatas que permiten transiciones con la cadena vacía

En la Tabla 1 se presenta la comparativa entre los diferentes tipos de autómatas finitos.

Cuadro 1: Comparativa de tipos de autómatas finitos.

Característica	AFD	AFND	AF-λ
Función δ	$\delta : Q \times \Sigma \to Q$	$\delta : Q \times \Sigma \to 2^Q$	$\delta : Q \times (\Sigma \cup \{\lambda\}) \to 2^Q$
Determinismo	Completo	Parcial	Parcial
Transiciones λ	No permitidas	No permitidas	Permitidas
Poder computacional	Lenguajes Regulares	Lenguajes Regulares	Lenguajes Regulares

Conversión entre Representaciones

Se implementó el algoritmo de conversión de AFND a AFD mediante el método de construcción de subconjuntos:

1. El estado inicial del AFD es el conjunto que contiene el estado inicial del AFND
2. Para cada estado S del AFD y símbolo a, calcular $\delta(S, a)$ como la unión de todos los estados alcanzables desde cualquier estado en S con a
3. Un estado S es final si contiene al menos un estado final del AFND

Expresiones Regulares

Se definieron las expresiones regulares mediante los casos base y operadores:

- Casos base: $\emptyset$, λ, a (para $a \in \Sigma$)
- Operadores: Unión $(+)$, Concatenación $(\cdot)$, Clausura $(*)$
- Propiedades: $A\lambda = \lambda A = A$, $\emptyset^* = \lambda$, $A^* = \lambda + AA^*$

Miniaturización de Autómatas

Se aplicó el algoritmo de particionamiento para minimizar autómatas:

Listing 1: Algoritmo de minimización de estados.

```
P0 = {F, Q-F}  // Partici n inicial: finales vs no finales
repeat
    P_{k+1} = refinar(P_k)
until P_{k+1} = P_k
```

El criterio de refinamiento: dos estados permanecen en la misma clase si para todo símbolo, sus transiciones van a estados de la misma clase.

Operaciones de Cierre

Se demostró que los lenguajes regulares son cerrados bajo:

- **Unión**: Mediante producto cartesiano de autómatas
- **Concatenación**: Conectando estados finales del primer autómata con el inicial del segundo
- **Clausura**: Añadiendo transiciones λ desde estados finales al inicial
- **Complemento**: Invirtiendo estados finales y no finales
- **Intersección**: Producto cartesiano con estados finales donde ambos son finales

Ejemplo de Construcción

Para la expresión regular $(a + b)^*ab$:

1. Linearización: $(a_1 + b_2)^*a_3b_4$
2. Construcción del autómata de posición
3. Eliminación de subíndices mediante autómata follow
4. Obtención del AFD mínimo equivalente

5. Conclusiones

El estudio de autómatas y lenguajes formales proporciona insights fundamentales sobre la naturaleza de la computación:

- Comprendí la profunda conexión entre diferentes representaciones de lenguajes regulares (autómatas, expresiones regulares, gramáticas) y cómo cada una ofrece perspectivas complementarias para el mismo concepto abstracto.
- El proceso de minimización de autómatas me reveló cómo la teoría de clases de equivalencia permite identificar y eliminar redundancias, obteniendo la representación más eficiente posible para un lenguaje dado.
- Las operaciones de cierre demostraron la robustez de la clase de lenguajes regulares, mostrando cómo se pueden construir lenguajes complejos a partir de componentes simples mediante operaciones bien definidas.
- El momento "WOW.ºcurrió cuando visualicé cómo el teorema de Myhill-Nerode conecta la noción abstracta de clases de equivalencia con la implementación concreta de autómatas finitos, revelando la elegante estructura matemática subyacente.

Esta práctica demostró que los autómatas finitos, aunque conceptualmente simples, encapsulan principios profundos sobre la computación y establecen los límites fundamentales de lo que puede ser reconocido con recursos finitos. La comprensión de estos límites es crucial para el diseño de algoritmos eficientes y sistemas computacionales robustos.

Referencias Bibliográficas

Referencias

[1] Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2007). *Introduction to automata theory, languages, and computation* (3rd ed.). Pearson Education.

[2] Sipser, M. (2012). *Introduction to the theory of computation* (3rd ed.). Cengage Learning.

[3] Linz, P. (2011). *An introduction to formal languages and automata* (5th ed.). Jones & Bartlett Learning.

[4] Kozen, D. C. (1997). *Automata and computability*. Springer-Verlag.