REPORTE DE PRÁCTICA CERO ?

PRÁCTICA CERO

ALUMNO:

Martinez Angeles Mario Rodrigo

1. Introducción

La computación moderna se sustenta en principios teóricos profundos que determinan qué problemas pueden ser resueltos por una máquina y cómo pueden ser resueltos de manera eficiente. En el corazón de esta teoría se encuentra el estudio de los lenguajes formales y los autómatas. Un lenguaje formal, a diferencia de un lenguaje natural, es un conjunto de palabras construidas sobre un alfabeto finito siguiendo reglas sintácticas precisas y matemáticamente definibles. Estos lenguajes son la base de todas las estructuras de programación: desde una simple expresión regular hasta la sintaxis completa de un lenguaje de programación.
Para poder procesar, reconocer o validar si una cadena de símbolos pertenece a uno de estos lenguajes, se utilizan modelos abstractos de computación denominados autómatas. Un autómata es, en esencia, una máquina de estados finitos que procesa una entrada secuencialmente y decide si la acepta o rechaza.
Este artículo tiene como objetivo servir como una guía comprehensiva, desglosando esta compleja teoría en conceptos digeribles. Comenzaremos definiendo los pilares de los lenguajes formales, procederemos a describir los diferentes tipos de autómatas finitos y culminaremos demostrando cómo modelos aparentemente más complejos (AFND y AF-) pueden transformarse en un modelo determinista equivalente (AFD), un proceso fundamental en la implementación de analizadores léxicos para compiladores.

2. Marco teórico

2.1. Fundamentos de los Lenguajes Formales

Un lenguaje formal es un conjunto de palabras sobre un alfabeto finito (). Una cadena o palabra es una secuencia finita de símbolos de . La cadena vacía () tiene longitud cero. La clausura de Kleene (*) es el conjunto de todas las palabras posibles sobre , incluida .
Las operaciones básicas incluyen:
Concatenación: Unión de dos cadenas.
Potencia: Repetición de una cadena *n* veces.
Reflexión (w): Inversión del orden de los símbolos.
Operaciones de conjunto: Unión, intersección y diferencia aplicadas a lenguajes.

2.2. Teoría de Autómatas Finitos

Un autómata finito es una máquina abstracta definida por una 5-tupla (Q, , , q, F), donde:
Q: Conjunto finito de estados.
: Alfabeto de entrada.
: Función de transición.
q: Estado inicial.
F: Estados de aceptación.
Existen tres tipos principales, todos equivalentes en poder computacional:
AFD (Determinista): Para cada estado y símbolo, hay exactamente una transición definida.
AFND (No Determinista): Puede tener cero, una o múltiples transiciones para un mismo par (estado, símbolo).
AF-: Extiende el AFND permitiendo transiciones que no consumen un símbolo de entrada (transiciones-).
La conversión entre modelos es un proceso fundamental:
AFND a AFD: Se utiliza el algoritmo de construcción de subconjuntos, donde los estados del AFD son subconjuntos de estados del AFND original.
AF- a AFND: Se eliminan las transiciones- calculando la -clausura de cada estado (todos los estados alcanzables mediante transiciones-) y redefiniendo la función de transición para que incluya estos caminos.

4. Desarrollo

VIDEO 1: Introducción a los Lenguajes Formales

Se explora la intersección entre matemáticas y lengua, presentando los lenguajes formales como la herramienta fundamental para estudiar propiedades de los números (como los capicúas) desde una perspectiva lingüística. Se define un alfabeto () como un conjunto finito y no vacío de símbolos (ej: letras, 0,1). Una palabra o cadena es cualquier secuencia finita de símbolos de ese alfabeto. La cadena vacía (o) es una palabra de longitud 0 y actúa como el elemento neutro en operaciones. La longitud —w— es el número de símbolos de una palabra. La operación más importante es la clausura de Kleene (*), que representa el conjunto infinito de TODAS las palabras posibles (de cualquier longitud, incluida) que se pueden formar con el alfabeto. + es igual pero excluyendo la cadena vacía.

VIDEO 2: Operaciones con Palabras y Definición de Lenguaje

Se profundiza en operaciones básicas con palabras. La concatenación une dos palabras; su elemento neutro es (ej: "monta" · "ña" = "montaña"). La potencia (w) concatena una palabra 'n' veces consigo misma (w =). Se definen los prefijos (inicio), sufijos (final) y segmentos (cualquier parte) de una palabra. El reverso (w) es la palabra leída al revés; una palabra capicúa o palíndroma cumple w = w. Finalmente, un lenguaje (L) se define formalmente como cualquier subconjunto de * (ej: x * — x empieza por 'a'), por lo que puede ser finito o infinito, y hasta vacío.

VIDEO 3: Operaciones con Lenguajes

Se explican operaciones entre lenguajes, extendiendo el concepto de conjuntos. El producto (L·L) concatena cada palabra de un lenguaje con todas las del otro. La potencia (L) es el producto de un lenguaje consigo mismo 'n' veces (L=). El cierre de Kleene (L*) es la unión de todas las potencias de L. El cociente (LŁ) elimina un prefijo o sufijo de las palabras de un lenguaje. Un homomorfismo es una función que transforma cada símbolo de un alfabeto en una cadena de otro, permitiendo codificar mensajes. Sus aplicaciones prácticas incluyen la encriptación y el uso de expresiones regulares para buscar archivos.

VIDEO 4: ¿Qué es un Autómata?

Un autómata es una máquina abstracta que modela procesos de cálculo mediante un número finito de estados y transiciones entre ellos, procesando una entrada secuencialmente. Se representa con un grafo (nodos=estados, arcos=transiciones). Existen varios tipos con distinto poder computacional: los autómatas finitos (los más simples, procesan lenguajes regulares), los autómatas de pila (más potentes, usan una memoria de pila para análisis sintáctico) y las máquinas de Turing (modelo teórico máximo de computación). Son la base teórica de compiladores, procesadores de texto y muchas herramientas digitales.

VIDEO 5: Autómatas Finitos Deterministas (AFD)

Un AFD se define formalmente como una tupla (Q, , , q, F). Q es un conjunto finito de estados. es el alfabeto de entrada. es la función de transición determinista: para cada estado y símbolo, hay exactamente UN estado de destino (o ninguno). q es el estado inicial. F es el conjunto de estados finales de aceptación. Se representa con una tabla de transiciones (filas=estados, columnas=símbolos) o un grafo. El lenguaje que acepta está formado por todas las cadenas que, al ser procesadas símbolo a símbolo, llevan al autómata desde q a un estado final. Se incluye un ejemplo para interpretar el lenguaje de un AFD a partir de su grafo.

VIDEO 6: Autómatas Finitos No Deterministas (AFND)

Un AFND comparte la estructura de tupla (Q, , , q, F) con el AFD. La diferencia radical está en la función : en un AFND, desde un estado y un símbolo, puede haber cero, una o MÚLTIPLES transiciones posibles. Esta falta de unicidad es el no determinismo. Se representa en el grafo con varios arcos con la misma etiqueta

saliendo de un mismo estado. Todo AFD es un caso particular de AFND. La importancia de los AFND radica en que suelen ser más fáciles y intuitivos de diseñar para ciertos lenguajes que sus equivalentes AFD.

VIDEO 7: Conversión de AFND a AFD

Este vídeo es práctico y muestra el algoritmo de construcción de subconjuntos para transformar un AFND en un AFD equivalente (que acepte el mismo lenguaje). El proceso consiste en crear una tabla para el nuevo AFD donde cada estado del AFD es un subconjunto de estados del AFND original. El estado inicial del AFD es la -clausura del estado inicial del AFND. Las transiciones del AFD se calculan viendo a qué conjunto de estados se lleza desde el conjunto actual con un símbolo. Un estado del AFD será final si contiene AL MENOS UN estado final del AFND. El vídeo guía paso a paso con un ejemplo concreto.

VIDEO 8: Autómatas Finitos con Transiciones (AFD- o AFND-)

Se introduce un nuevo nivel de abstracción: los autómatas que permiten transiciones (o épsilon), que son arcos que se pueden recorrer sin consumir ningún símbolo de entrada. Esto permite al autómata "cambiar de estado gratis". La función de transición ahora incluye el símbolo . Para convertir un AFND- en un AFND normal, el primer paso es calcular la -clausura de cada estado: el conjunto de todos los estados alcanzables desde él mediante cero o más transiciones . Este concepto es clave para eliminar las transiciones vacías y preparar la conversión a un AFND standard.

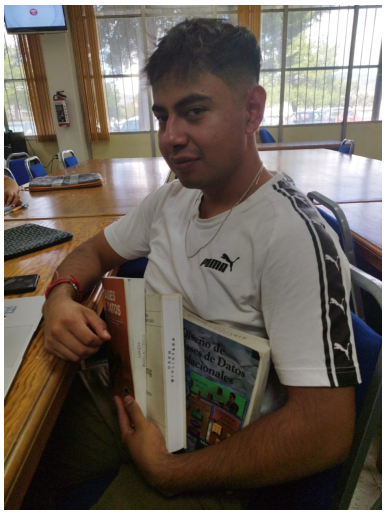
VIDEO 9: Conversión de AFND- a AFND

Este vídeo práctico completa el proceso de eliminación de transiciones . Partiendo del cálculo de la -clausura de cada estado (explicado en el vídeo 8), se construye una tabla de transiciones para el nuevo AFND (sin). La regla es: una transición con un símbolo 'a' desde un estado en el nuevo AFND debe llevar a todos los estados a los que se podría llegar desde cualquier estado de su -clausura, consumiendo 'a' y luego permitiendo cualquier número de transiciones posteriores. El estado inicial y los finales se ajustan considerando las -clausuras. El resultado es un AFND convencional que puede luego convertirse en un AFD usando el método del vídeo 7.

5. Conclusiones

El análisis de los lenguajes formales y los autómatas ofrece recursos esenciales para el manejo efectivo de datos. Las nociones de alfabetos, palabras y lenguajes sientan las bases para entender sistemas más elaborados, como los autómatas finitos . La diferencia entre autómatas deterministas y no deterministas muestra distintos mecanismos para abordar problemas de identificación de patrones.
Un momento wow fue comprender todo los elementos que involucran a un automata no me imaginaba todo los involucra.
El estudio de los autómatas y los lenguajes formales mantiene su vigencia en campos como el diseño de compiladores, el procesamiento del lenguaje natural y la verificación formal de software, evidenciando cómo las bases conceptuales sentadas hace años siguen impulsando el desarrollo tecnológico actual.

Fotos



Referencias Bibliográficas

References

[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). *Introducción 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). *Conversión 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). *Conversión 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). *Introducción al lenguaje ensamblador* [Video]. YouTube. https://youtu.be/JuTuMe8Q58c?si=bkXpOYdYjCHAsDmJ

[12] Canal de Ensamblador. (2023). *Instrucciones aritméticas y lógicas* [Video]. YouTube. https://youtu.be/gYOvlrjRBwg?si=-Ri5f5V222Bza2Wz

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

[14] Canal de Optimización. (2023). *Optimización y buenas prácticas* [Video]. YouTube. https://youtu.be/gd6uyNXsqcw?si=VWw_Nkl4Dh240Dzs

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

[16] Canal de Compiladores. (2023). *Análisis léxico y sintáctico* [Video]. YouTube. https://youtu.be/5dmyN5mWu1o?si=do9Sooj2TLnFQIMQ

[17] Canal de Compiladores. (2023). *Generación de código intermedio* [Video]. YouTube. https://youtu.be/qzVms1j23uE?si=HUblHGk0icLtUK9a

[18] Charte Ojeda, F., & Ruíz Calderón, V. M. (2020). *Lenguaje ensamblador*. RA-MA Editorial.

[19] Alfonseca Moreno, M. (1998). *Compiladores e intérpretes: teoría y práctica*. McGraw-Hill.

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