Laboratorio 1: Construcción Directa de AFD y ecosistema de reconocimiento de expresiones regulares.

Gerardo Pineda 22880

Video: https://youtu.be/_a_Uvlk9lvA

Repo: https://github.com/Gerax5/COM-LB1.git

Antes de empezar voy a recalcar que todo lo hecho en teoría, fueron autómatas completos con estrado de atrapamiento. El nuevo no lo tiene, de ser requerido se quita pero es solo un estado más.

Construcción directa

Para el apartado de la construcción directa, se realizo una clase con cada método que representa cada uno de los pasos de la construcción directa. Primero tengo definidos mis modelos definidos, estos fueron creados con anterioridad en el proyecto 1 de teoría de la computación. Por error mio, cree diferentes modelos para casi lo mismo pero en esta implementación el que se usa es el modelo Node

El constructor de esta clase recibe como parámetro la seed del árbol sintáctico y el alfabeto. Después se mandan a llamar distintas funciones en el siguiente orden (cabe recalcar que todas las funciones están basadas en recursión, donde el caso base es encontrar a las hojas):

1. enumerateLeaves: Este método lo que hace es que busca las hojas y las enumera en orden, mientras agrega a un set la hoja encontrada, para en futuros pasos realizar la siguiente posición.
2. setNullables: Este método busca las hojas y asigna si son nullales o no, luego de terminar con las hojas escala para arriba para determinar si los demás nodos son nullables o no.
3. setFirstPosition: Este método busca las hojas y asigna su ID a primera posición, luego de terminar con las hojas escala para arriba para determinar la primera posición de los demás nodos.
4. setLasPosition: Este método busca las hojas y asigna su ID a la ultima posición, luego de terminar con las hojas escala para arriba para determinar la ultima posición de los demás nodos.
5. setFollowPosition: Este método busca las hojas pero no hace nada solo las deja pasar y escala parra buscando concatenación o klean. Cuando las encuentra asigna a la siguiente posición a el set creado en el primer método.
6. Luego se crean variables de apoya para construir la tabla de transiciones y el grafico.

7. generateTransitionTable: Este método asigna como primer estado a la primera posición de la seed del árbol sintactico procesasdo. Luego busca va letra por letra buscando si existen transiciones y creando estados, mediante un stack.

Posteriormente de esto, se crea otro método el cual agrega los estados de aceptación y ya estaría listo para graficar y procesar palabras.

Construcción de NDFA, DFA, Minimizado DFA

Para este apartado me base completamente en mi implementación de estos algoritmos en Teoría de la computación (link en referencia), solo cambie el nombre de las carpetas para ser un poco más entendibles y estructure un poco mejor los modelos.

shunting yard

De igual forma este apartado es completamente basado en mi implementación anterior (link en referencia). Este procesa los símbolos como +, ? y [] (interpretado como or).

Referencias:

- Proyecto **1** Teoria de la computación: https://github.com/Gerax5/TCLAB04/tree/Proyecto
- Graphviz: https://graphviz.gitlab.io/
- Algoritmo de Construción directa AFD basado en la explicacion de Luis Urbina: https://www.youtube.com/watch?v=1elii9xzYlc