Compiladores, Sem: 2021-1, 3CV2, Práctica 1, 26 de enero de 2021

Práctica 1: Clases AFN, AFD.

Iris Yonitzi Martínez González
Escuela Superior de Cómputo
Instituto Politécnico Nacional, México
irismartinezgonzalez.ipn@gmail.com

1. Objetivo

Utilizar los conocimientos sobre el paradigma orientado a objetos y los autómatas finitos, para diseñar e implementar las clases AFN y AFD.

2. Requerimientos

- Los autómatas se podrán almacenar en formato de archivo af.
- Las clases AFN y AFD deberán tener disponibles los siguientes métodos: (Nota: ninguno de los métodos usa la salida estándar.)
  - *cargar_desde(nombre:string)*
  - guardar_en(nombre:string)
  - agregar_transicion(inicio:int, fin:int, simbolo:char)
  - eliminar_transicion(inicio:int, fin:int, simbolo:char)
  - obtener_inicial():int
  - obtener_finales():vector<int>
  - establecer_inicial(estado:int)
  - establecer_final(estado:int)
  - esAFN(): bool
  - esAFD(): bool
  - acepta(cadena:string): bool
  - generar_cadena(): string

3. Experimentación y Resultados

Después de hacer el análisis de los requerimientos anteriores, aunque lo menciona explícitamente, los detalles que hay que tomar en cuenta son:

- Se tienen que generar cadenas aleatorias que sean aceptadas por el propio autómata, sin importar el tamaño de dichas cadenas.
- Se tiene que validar que el archivo que se esta cargando es realmente un archivo .af y que su contenido coincida con el ejemplo proporcionado por el profesor.
- Después de cargar el autómata este mismo puede ser modificado a través de los métodos agregar_transicion y eliminar_transicion.

- Los métodos anteriores no tienen limitantes, se pueden agregar transiciones cuales quiera con caracteres que pertenezcan o no al lenguaje original del autómata y si lo requiere es posible agregar estados al mismo, si se desea eliminar una transición se puede hacer, aunque los estados ya no tengan transiciones que los una y por ende se altera el altera el lenguaje.

Las instrucciones nos indican que debe de ser en dos clases, aunque ambas clases tendrán los mismos atributos por que pertenecen a una de nombre autómata, el objetivo que le veo a dichas clases es para aplicar el concepto de poliformismo, mi idea original era crear tres clases: automata, estado, transición, donde estas ultimas dos vayan formando una estructura grafo para cada una de sus transiciones con los estados.

El proceso del análisis aproximadamente me llevó 1 hora, en la media hora faltante no legré implementar de forma efectiva y eficaz las clases con sus métodos por una sola razón: no tengo buenos fundamentos de programación orientada objetos es por ello en esa media hora tuve que investigar sobre el tema. Después de eso me di cuenta que mi idea que tenia para implementar no era la más eficiente.

Después de un buen analisis y tener un mejor pannorama de lo que es orientado a objetos el duagrama de clases que obtuve es el siguiete:

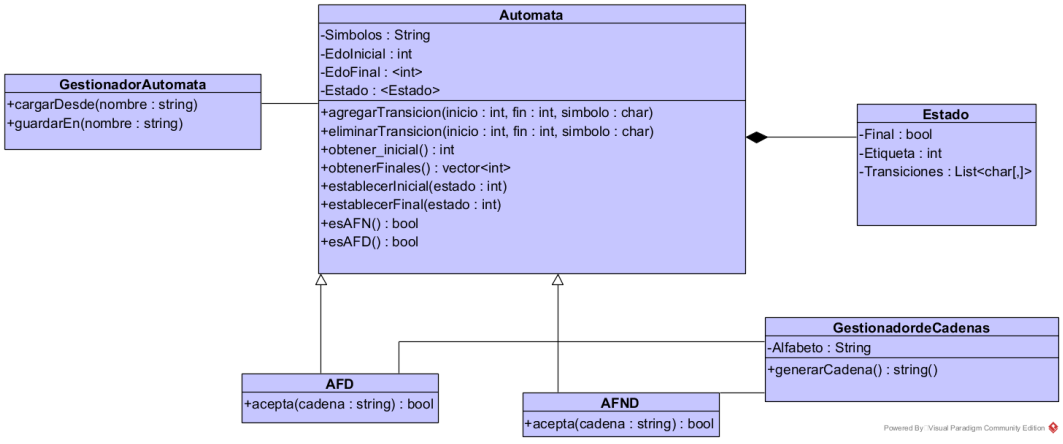

Figura 1: Diagrama de clases

4. Conclusiones

El no tener buenos conocimientos de programación orientada a objetos y a pesar de que manejo constantemente C++ impidió que realiza la con éxito esta práctica, lo que me da a entender es que si quiero ser un buen ingeniero en sistemas computacionales. Debo de mejorar mis conocimientos y habilidades y repasarlos constantemente, aunque no siempre me sean solicitados en la escuela y ser más curiosa con dichos temarios impartidos.

Conclusiones actualizadas: Despues de peliarme con C++ prferí cambiar de idioma a Java, ya que es mucho más facil entender y hacer las clases qeu en C++;