*****Instituto Politécnico Nacional***

***Escuela Superior de Cómputo***

*Compiladores*

***Practica 1: AFN***

***Nombre:*** *Mauro Sampayo Hernández*

***Grupo:*** *3CV17*

***Profesor:*** *Hernández Olvera Luis Enrique*

***Fecha de entrega:*** *15 de septiembre del 2021*

1. **Introducción:**
   1. ***Autómata***

Un Autómata es un modelo matemático para una máquina de estado finito, en el que, dada una entrada de símbolos, se realizan saltos mediante una serie de estados de acuerdo con una función de transición (que puede ser expresada como una tabla). Esta función de transición indica a qué estado cambiar dados el estado actual y el símbolo leído.

Un Autómata Finito puede ser determinista o no determinista. Para este reporte se explicarán los autómatas no deterministas.

* 1. ***Autómata Finito No Determinista (AFN)***

Un Autómata Finito No Determinista (AFN) es un autómata que posee al menos un estado, en el que, para un símbolo del alfabeto, existe más de una transición posible.

Los Autómatas Finitos No Deterministas están definidos por la siguiente tupla:

AFN = (Q, ∑ , δ, , F )

Donde:

**Q:** Conjunto finito de estados.

**∑:**Alfabeto de símbolos de entrada.

**δ:** Función de estados de transición definida como: , donde es el conjunto que se puede armar con todos los subconjuntos de Q.

**:** Estado inicial

**F:** Conjunto de estados finales de aceptación

Un AFN se puede representar diagramáticamente mediante un grafo dirigido etiquetado, llamado grafo de transiciones, en el que los nodos son los estados y las aristas etiquetadas representan la función de transición.

* 1. ***Ejemplo de Autómata Finito No Determinista (AFN)***

Sea A un AFN tal que A = (Q, ∑ , δ, , F ) donde:

**Q:** {0, 1, 2, 3}

**∑:**{a, b}

**δ:** {{0, a, {0,1}}, {0, b, {0}}, {1, b, {2}}, {2, b, {3}}}

**:** {0}

**F:** {3}

Define el siguiente grafo de transiciones:

Grafo de Transiciones del AFN A


Figura 1. Grafo de transiciones del AFN A.

Cuando se describe un AFN, se utiliza la representación de grafo de transiciones. En un computador, puede aplicarse la función de transición de un AFN de varias formas. La implantación más sencilla es una tabla de transiciones en donde hay una fila por cada estado y una columna por cada símbolo de entrada y ε, si es necesario. Para el AFN anterior la tabla quedaría de la siguiente forma:

|  |  |  |
| --- | --- | --- |
| Estado | Símbolo de entrada | |
| **a** | **b** |
| 0 | **{0, 1}** | **0** |
| 1 | **-** | **1** |
| 2 | **-** | **2** |

Un AFN acepta una cadena de entrada x si, y sólo sí, hay algún camino en el grafo de transiciones desde el estado de inicio a algún estado de aceptación. Así mismo puede haber más de una secuencia de movimientos que conduzca a un estado de aceptación.

1. **Problemas enfrentados al programar la práctica:**

Uno de los principales problemas enfrentados durante la programación de la practica fue la de llevar a cabo la implementación de la tabla de transiciones del AFN debido a que el lenguaje de programación Phyton no soporta el uso de arreglos mayores a una dimensión, lo cual no posibilita la creación de matrices. Como alternativa se opto por usar una lista tridimensional.

Lo anterior fue implementado de la siguiente manera: Se implemento una lista principal llamada Transiciones, la cual almacena listas las cuales representan cada una de las filas de la tabla de transiciones, y las cuales cuentan con el mismo índice que el de los estados que representa cada fila, dentro de la lista de Estados. De forma similar, dentro de estas listas hay otra serie de listas, dentro de las cuales esta almacenados los estados destino de cada transición; estas listas representan cada una de las columnas de la tabla de transiciones, y cuentan con el mismo índice que el de cada uno de los símbolos almacenados en la lista Alfabeto. De esta manera cuando se requiera saber los estados a los cuales se puede llegar en alguna transición, simplemente se utilizan los índices del estado origen y el símbolo dentro de sus respectivas listas, para así llegar a la lista donde se encuentran los estados destino.

Otro de los problemas enfrentados fue la implementación de un algoritmo que validará una cadena. Después de analizar varias posibles opciones se llegó a la conclusión de que la mejor alternativa es el uso de un árbol, donde el nodo raíz es el estado inicial, y los descendientes de dicho nodo son todos los posibles recorridos que se pueden hacer dentro del autómata usando la cadena proporcionada por el usuario. De esta manera se pude saber si una cadena es valida si es que esta cuenta con uno o más nodos hoja que representen un estado final.

Sin embargo, el problema más grande que tuvo que ser enfrentado, fue el de aprender el lenguaje Phyton desde 0 en pocos días, ya que personalmente es la primera vez que hago uso de este lenguaje, por lo que este ha sido mi primer proyecto desarrollado en Phyton.

1. **Pruebas y Capturas de Pantalla:**

Se realizaron dos pruebas con el programa, utilizando como valores de entrada dos archivos .txt proporcionados por el profesor para la generación del autómata y algunas cadenas de texto a validar con dichos autómatas.

Cada renglón del archivo de texto representa los datos de cada elemento de la tupla del AFN a generar. El primer renglón representa la lista de estados del AFN, el segundo los símbolos del alfabeto, el tercero el estado inicial, el cuarto el o los estados finales, y el resto de los renglones representa las transiciones del AFN (enlistadas iniciando con el estado de origen, seguido del símbolo de transición y finalizando con el estado resultante de dicha transición.

A continuación, se describen los valores de entrada proporcionados y los resultados de cada prueba (la ejecución del programa se hizo usando el compilador de Phyton en el IDE PyCharm):

* 1. ***Autómata 1***

Para el primer autómata se nos proporciona el siguiente archivo de texto:

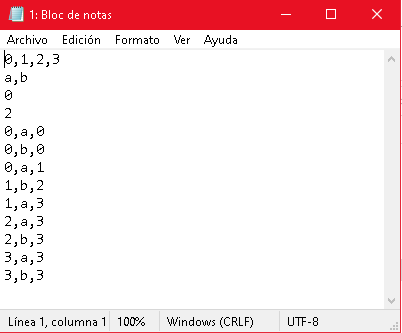


Figura 2. Archivo de texto de entrada para la generación del primer AFN

Los datos enlistados en este archivo de texto representan un autómata que puede recibir una cadena de texto que consista en una sucesión de a’s y b’s que puede estar vacía, seguidas de la cadena “ab”.

El grafo de transiciones del autómata mencionado anteriormente junto con su respectiva tabla de transiciones se muestra a continuación:

Imagen de la pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media

|  |  |  |
| --- | --- | --- |
| Estado | Símbolo entrada | |
| **a** | **b** |
| 0 | {0, 1} | 0 |
| 1 | 3 | 2 |
| 2 | 3 | 3 |
| 3 | 3 | 3 |

Figura 3. Autómata resultante del archivo de texto de la Figura 2

El resto de los elementos de la tupla quedan definidos de la siguiente manera:

**Q:** {0, 1, 2, 3}

**∑:**{a, b}

**:** {0}

**F:** {3}

* **Generación del autómata en el programa**

Al ejecutarse, el programa tomará como datos de entrada el archivo de texto de la Figura 2 para generar el autómata, y pasará a mostrarnos los datos contenidos en cada uno de los elementos de la tupla del AFN

Texto

Descripción generada automáticamente

Figura 4. Generación del primer AFN en el programa de Phyton. Este programa toma como entrada los datos contenidos en el archivo de texto de la Figura 2 para la generación del AFN

Una vez se haya generado el autómata, se requerirá al usuario que inserte una cadena de texto a validar.

* **Validación de la cadena “aa@cdab”**

La primera cadena de texto a validar será “aa@cdab”.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura 5. Resultado de la validación de la cadena “aa@cdab"

En este caso la cadena insertada es válida (es decir, la última transición llegó a un estado final), por lo que se nos muestra el recorrido de estados realizado en el autómata (el cuál puede comprobarse usando como referencia el autómata de la Figura 7), así como también el manejo de errores de símbolos no pertenecientes al alfabeto dentro de la cadena, los cuáles fueron omitidos para la validación de la cadena (en este caso los símbolos @, c y d).

* **Validación de la cadena “ababa@”**

La segunda cadena de texto a validar será “ababa@”.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura 6. Resultado de la validación de la cadena “ababa@"

En este caso la cadena insertada no es válida (es decir la última transición no llegó a un estado final), por lo que para este caso no se mostrará el recorrido de estados realizado en el autómata al no haberse llegado a un estado final, más sin embargo si se mostrará el manejo de errores de símbolos no pertenecientes al alfabeto dentro de la cadena (en este caso el único símbolo omitido fue @).

* 1. ***Autómata 2***

Para el segundo autómata se nos proporciona el siguiente archivo de texto:

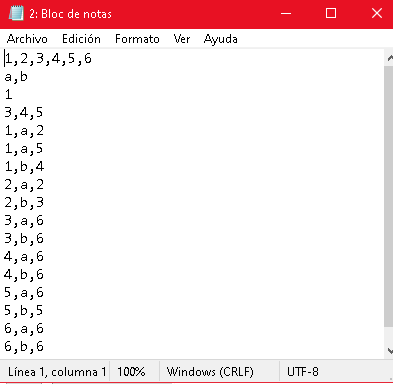


Figura 7. Archivo de texto de entrada para la generación del segundo AFN

Los datos enlistados en este archivo de texto representan un autómata que puede recibir una cadena de texto que contenga alguno de los siguientes casos:

* Una “a” que sea seguida de una sucesión de “a’s” que puede estar vacía, y sea finalizada por una “b”
* Una “a” seguida de una sucesión de “b’s” que puede estar vacía
* Una “b”

El grafo de transiciones del autómata mencionado anteriormente junto con su respectiva tabla de transiciones se muestra a continuación:

Diagrama

Descripción generada automáticamente

|  |  |  |
| --- | --- | --- |
| Estado | Símbolo entrada | |
| **a** | **b** |
| 1 | {2, 5} | 4 |
| 2 | 2 | 3 |
| 3 | 6 | 6 |
| 4 | 6 | 6 |
| 5 | 6 | 5 |
| 6 | 6 | 6 |

Figura 8. Autómata resultante del archivo de texto de la Figura 7

El resto de los elementos de la tupla quedan definidos de la siguiente manera:

**Q:** {0, 1, 2, 3, 4, 5, 6}

**∑:**{a, b}

**:** {1}

**F:** {3, 4, 5}

* **Generación del autómata en el programa**

Al ejecutarse, el programa tomará como datos de entrada el archivo de texto de la Figura 7 para generar el autómata, y pasará a mostrarnos los datos contenidos en cada uno de los elementos de la tupla del AFN.

Imagen que contiene Texto

Descripción generada automáticamente

Figura 9. Generación del segundo AFN en el programa de Phyton. Este programa toma como entrada los datos contenidos en el archivo de texto de la Figura 7 para la generación del AFN

Una vez se haya generado el autómata, se requerirá al usuario que inserte una cadena de texto a validar.

* **Validación de la cadena “ab@”**

La primera cadena de texto a validar será “ab@”.

Texto

Descripción generada automáticamente

Figura 10. Resultado de la validación de la cadena “ab@"

En este caso la cadena insertada es válida, por lo que se nos muestra el recorrido de estados realizado en el autómata (el cuál puede comprobarse usando como referencia el autómata de la Figura 8), así como también el manejo de errores de símbolos no pertenecientes al alfabeto dentro de la cadena, los cuáles fueron omitidos para la validación de la cadena (en este caso el único símbolo omitido fue @).

* **Validación de la cadena “aaab123”**

La segunda cadena de texto a validar será “aaab123”.

Texto

Descripción generada automáticamente

Figura 11. Resultado de la validación de la cadena “aaab123"

En este caso la cadena insertada es válida, por lo que se nos muestra el recorrido de estados realizado en el autómata (el cuál puede comprobarse usando como referencia el autómata de la Figura 7), así como también el manejo de errores de símbolos no pertenecientes al alfabeto dentro de la cadena, los cuáles fueron omitidos para la validación de la cadena (en este caso los símbolos 1, 2 y 3).

* **Validación de la cadena “a1b.a32b”**

La tercera cadena de texto a validar será “a1b.a32b”.

Texto

Descripción generada automáticamente

Figura 12. Resultado de la validación de la cadena “a1b.a32b"

En este caso la cadena insertada no es válida, por lo que para este caso no se mostrará el recorrido de estados realizado en el autómata al no haberse llegado a un estado final, más sin embargo si se mostrará el manejo de errores de símbolos no pertenecientes al alfabeto dentro de la cadena (en este los símbolos omitidos fueron 1, ., 3 y 2).

1. **Conclusión:**

A manera de Conclusión, puedo decir que a partir de esta práctica pude repasar conceptos relacionados a Autómatas, tanto deterministas y no deterministas, así como también de los elementos y características de estos, así mismo aprendí un nuevo lenguaje de programación cosa que considero resultará bastante útil en la vida laboral.

Finalmente puedo añadir que a partir de esta práctica pude comprender más a fondo lo útil que puede resultar el uso de AFNs en la informática, pues con ayuda de estos se pueden lograr cosas tales como la verificación de instrucciones o líneas de código en lenguajes de programación, para así determinar que estas estén bien escritas y estructuradas, y en caso de que no lo estén alertar al usuario sobre esto para que pueda revisarlas y corregirlas. De la misma manera pueden resultar útiles para los compiladores de estos lenguajes, pues estos pueden llevar a cabo un análisis de dichas líneas de código por medio del uso de AFNs, para así llevar a cabo su correcta interpretación.

1. **Referencias:**

* García, M., 2021. ¿Qué es un autómata? [online] Codingornot.com. Disponible en: <https://codingornot.com/que-es-un-automata> [Accedido el 15 de septiembre de 2021].
* GHD.com. 2021. MÁQUINAS DE ESTADOS: AUTÓMATAS FINITOS NO DETERMINISTAS (AFND. [online] Disponible en: <https://www.institucional.frc.utn.edu.ar/sistemas/ghd/T-M-AFND.htm> [Accedido el 15 de septiembre de 2021].
* Águila, J., 2021. AUTÓMATAS FINITOS NO DETERMINISTAS. [online] Kataix.umag.cl. Disponible en: <https://kataix.umag.cl/~jaguila/Compilers/T04\_AFN.pdf> [Accedido el 15 de septiembre de 2021].