*****Instituto Politécnico Nacional***

***Escuela Superior de Cómputo***

*Compiladores*

***Practica 2: AFN-E***

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

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

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

***Fecha de entrega:*** *22 de septiembre del 2021*

1. **Introducción:**
   1. ***Expresiones Regulares***

Las siguientes son las reglas que definen las expresiones regulares del alfabeto Σ.

1. ε es una expresión regular designada por {ε}.
2. Si a ∈ Σ ⇒ a es una expresión regular designada por {a}.
3. Si r y s son expresiones regulares representadas por L(r) y L(s) ⇒
4. (r)|(s) es una expresión regular representada por L(r) ∪ L(s).
5. (r)(s) es una expresión regular representada por L(r)L(s).
6. (r)\* es una expresión regular representada por (L(r))\*.
7. (r) es una expresión regular representada por L(r).

Un lenguaje designado por una expresión regular es un conjunto regular. Se compila una expresión regular en un reconocedor construyendo un diagrama de transiciones generalizado llamado autómata finito.

* 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. ***Construcción de un AFN a partir de una expresión regular por medio de la nomenclatura de Thompson***

La construcción de Thompson construye a partir de una expresión regular r un AFN que reconoce el lenguaje definido por r, esto se realiza con el objetivo de que en un algoritmo siguiente se pueda generar un AFD mínimo equivalente.

Las notaciones estándar para producir un AFN a partir de una expresión regular por medio método de Thompson son las siguientes:

* Diagrama

  Descripción generada automáticamentePara la representación de una cadena vacía se utiliza el símbolo ε.

Figura 1. Cadena Vacía

* Diagrama

  Descripción generada automáticamente con confianza bajaDiagrama

  Descripción generada automáticamente con confianza bajaPara representar un símbolo, se utilizan dos estados y una transición para el movimiento con el símbolo.

Figura 2. Representación de un símbolo

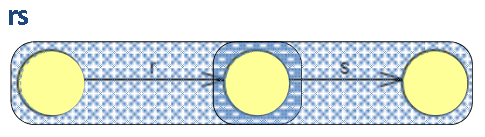
* Para la concatenación de dos símbolos únicamente se unen.

Figura 3. Concatenación de símbolos

* Diagrama

  Descripción generada automáticamente con confianza mediaPara la elección de alternativas, crear transiciones ε para la unión de las transiciones.

Figura 4. Unión de símbolos

* Un dibujo de una persona

  Descripción generada automáticamente con confianza bajaPara la cerradura positiva, se agregan transiciones ε para retornar al estado previo, permitiendo agregar una o más veces el símbolo

Figura 5. Cerradura Positiva

* Un dibujo de una persona

  Descripción generada automáticamente con confianza mediaPara la cerradura de Kleene, se agregan transiciones ε para retornar a estado previo. Y otra transición ε para saltar la transición con r.

Figura 6. Cerradura de Kleene

* 1. ***Ejemplo del Método de Thompson***
* Diagrama del AFN representado por la expresión regular **(b|(b\*a)\*)a:**

1. Se parte desde la cerradura de Kleene, que se encuentra dentro del paréntesis anidado. Es decir, por la expresión **b\*.**

Diagrama

Descripción generada automáticamente

Figura 7. AFN de la expresión b\*

1. Se completa la expresión regular dentro de dicho paréntesis concatenando la expresión regular anterior con a, es decir **b\*a.**

Diagrama

Descripción generada automáticamente

Figura 8. AFN de la expresión b\*a

1. Se aplica cerradura de Kleene a la expresión regular contenida por el paréntesis anidado, es decir **(b\*a)\*.**

Diagrama

Descripción generada automáticamente

Figura 9. AFN de la expresión (b\*a)\*

1. Se realiza la unión o elección de alternativas entre la expresión regular realizada previamente y b, es decir **b|(b\*a)\*.**

Gráfico, Gráfico de burbujas

Descripción generada automáticamente

Figura 10. AFN de la expresión b|(b\*a)\*

1. Como paso final, se realiza la concatenación de la expresión regular desarrollada previamente en el AFN con a, es decir **(b|(b\*a)\*)a**. Adicionalmente se enumeran los estados u se indica el estado final e inicial en el AFN.

Diagrama

Descripción generada automáticamente

Figura 11. AFN de la expresión (b|(b\*a)\*)a

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

En general, el problema enfrentado a lo largo del desarrollo de la practica fue el de adaptar el algoritmo desarrollado en la Practica 1 (que consiste en el desarrollo de un programa que pudiese generar un autómata a partir de los datos proporcionados en un archivo de texto, en el cual se enlistaban los estados, alfabeto, estados iniciales y finales, y transiciones del AFN; y una vez generado el autómata evaluar cadenas insertadas por el usuario para verificar si son aceptados por el AFN o no, con un manejo de errores) para que esta pusiese leer AFNs que tuviesen transiciones épsilon y poder validar cadenas en ellos.

Se realizaron dos modificaciones principales en las funciones del programa.

La primera modificación fue en la función que se encarga de generar la tabla de transiciones, pues dado a que en los archivos de texto que fueron brindados por el profesor se nos da a entender que el símbolo épsilon no es incluido en los datos de entrada como parte del Alfabeto, más si se encuentra considerado en las transiciones; se debió incluir un caso especial durante el proceso de lectura de las transiciones en el archivo de texto, para que en caso de que se lea una “E” (que para el programa será un carácter reservado para representare épsilon) esta sea añadida al Alfabeto del AFN, y así el autómata pueda ser generado correctamente.

La segunda modificación se realizó en la función donde se genera el árbol de resultados. Debido a que el programa realizaba originalmente la validación de una transición tomando como símbolo cada uno de los caracteres presentes en la cadena, y en caso de no ser validos en alguna transición inevitablemente iban hacia un estado de error; se incluyo un algoritmo que se ejecuta antes de la validación del carácter, de tal forma que dicho algoritmo trazara en el árbol todas las transiciones épsilon posible, antes de llegar a un estado que forzosamente necesite de algún símbolo que no sea épsilon. Finalizado este proceso el algoritmo toma cada uno de los estados que el algoritmo detecto que requieren un símbolo que no sea épsilon para que su transición sea válida, y procede a realizar la comprobación del carácter de la cadena a evaluar de manera normal.

Una vez se hayan finalizado de evaluar todos los caracteres, el algoritmo para evaluar transiciones épsilon mencionado en el párrafo anterior se realiza una vez más, para que de esta manera se puedan considerar todas las posibles rutas entre los estados del AFN que se puedan realizar con transiciones épsilon una vez se haya finalizado de evaluar el ultimo carácter de la cadena, dentro del árbol de resultados. Esto último resulta bastante útil para los casos en donde haya un AFN, en el cuál para llegar al estado final después de haberse evaluado el último carácter de una cadena válida, se necesite de recorrer un o más transiciones épsilon; como sucede en el caso del AFN generado en el archivo de texto Thompson2.

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 de los AFNs con transiciones épsilon 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 Thompson 1***

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

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 12. Archivo de texto de entrada para la generación del primer AFN con transiciones épsilon

Los datos enlistados en este archivo de texto representan un autómata cuyo grafo de transiciones y tabla de transiciones se muestran a continuación:

|  |  |  |  |
| --- | --- | --- | --- |
| Estado | Símbolo entrada | | |
| **a** | **b** | **ε** |
| 0 | - | - | {1, 8} |
| 1 | - | - | {2, 7} |
| 2 | - | - | {3, 5} |
| 3 | - | 4 | - |
| 4 | - | - | {3, 5} |
| 5 | 6 | - | - |
| 6 | - | - | {2, 7} |
| 7 | - | - | 10 |
| 8 | - | 9 | - |
| 9 | - | - | 10 |
| 10 | 11 | - | - |
| 11 | - | - | - |

Diagrama

Descripción generada automáticamente

Figura 13. Autómata resultante del archivo de texto de la Figura 12

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

**Q:** {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

**∑:**{a, b, ε}

**:** {0}

**F:** {11}

* **Generación del autómata en el programa**

Al ejecutarse, el programa tomará como datos de entrada el archivo de texto de la Figura 12 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 14. Generación del primer AFN con transiciones épsilon en el programa de Phyton.

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

* **Validación de la cadena “ba”**

La primera cadena de texto a validar será “ba”.

Texto

Descripción generada automáticamente

Figura 15. Resultado de la validación de la cadena “ba"

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 con esta cadena (el cuál puede comprobarse usando como referencia el autómata de la Figura 13).

* **Validación de la cadena “ba1a”**

La segunda cadena de texto a validar será “ba1a”.

Texto

Descripción generada automáticamente

Figura 16. Resultado de la validación de la cadena “ba1a@"

En este caso la cadena insertada es válida, por lo que se nos muestra el recorrido de estados realizado en el autómata con esta cadena (el cuál puede comprobarse usando como referencia el autómata de la Figura 13). El manejo de errores del AFN, nos muestra la lista de símbolos no pertenecientes al Alfabeto presentes en la cadena validada y el estado en el que fueron detectados; y los cuales fueron omitidos para la validación. Para este caso el único carácter omitido fue 1.

* **Validación de la cadena “abbzbaa”**

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

Captura de pantalla de computadora

Descripción generada automáticamente

Figura 17. Resultado de la validación de la cadena “abbzbaa”

En este caso la cadena insertada es válida, por lo que se nos muestra el recorrido de estados realizado en el autómata con esta cadena (el cuál puede comprobarse usando como referencia el autómata de la Figura 13). El manejo de errores del AFN, nos muestra la lista de símbolos no pertenecientes al Alfabeto presentes en la cadena validada y el estado en el que fueron detectados; y los cuales fueron omitidos para la validación. Para este caso el único carácter omitido fue z.

* **Validación de la cadena “****ab@abaa”**

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

Texto

Descripción generada automáticamente

Figura 18. Resultado de la validación de la cadena “ab@abaa”

En este caso la cadena insertada es válida, por lo que se nos muestra el recorrido de estados realizado en el autómata con esta cadena (el cuál puede comprobarse usando como referencia el autómata de la Figura 13). El manejo de errores del AFN, nos muestra la lista de símbolos no pertenecientes al Alfabeto presentes en la cadena validada y el estado en el que fueron detectados; y los cuales fueron omitidos para la validación. Para este caso el único carácter omitido fue @.

* 1. ***Autómata Thompson 2***

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

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 19. Archivo de texto de entrada para la generación del segundo AFN con transiciones épsilon

Los datos enlistados en este archivo de texto representan un autómata cuyo grafo de transiciones y tabla de transiciones se muestran a continuación:

Diagrama

Descripción generada automáticamente

Figura 20. Autómata resultante del archivo de texto de la Figura 19

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Estado | Símbolo entrada | | | |
| **a** | **b** | **c** | **ε** |
| 0 | - | - | - | {1, 8} |
| 1 | - | - | 2 | - |
| 2 | - | - | - | {3, 5} |
| 3 | 4 | - | - | - |
| 4 | - | - | - | 7 |
| 5 | - | 6 | - | - |
| 6 | - | - | - | {5, 7} |
| 7 | - | - | - | 12 |
| 8 | - | - | - | {9, 11} |
| 9 | 10 | - | - | - |
| 10 | - | - | - | {9, 11} |
| 11 | - | - | - | 12 |
| 12 | - | - | - | - |

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

**Q:** {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

**∑:**{a, b, c, ε}

**:** {0}

**F:** {12}

* **Generación del autómata en el programa**

Al ejecutarse, el programa tomará como datos de entrada el archivo de texto de la Figura 19 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 21. Generación del segundo AFN con transiciones épsilon en el programa de Phyton.

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

* **Validación de la cadena “aaaa”**

La primera cadena de texto a validar será “aaaa”.

Interfaz de usuario gráfica, Texto, Chat o mensaje de texto

Descripción generada automáticamente

Figura 22. Resultado de la validación de la cadena “aaaa"

En este caso la cadena insertada es válida, por lo que se nos muestra el recorrido de estados realizado en el autómata con esta cadena (el cuál puede comprobarse usando como referencia el autómata de la Figura 20).

* **Validación de la cadena “ab@ab”**

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

Texto

Descripción generada automáticamente

Figura 23. Resultado de la validación de la cadena “ab@ab"

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. El manejo de errores del AFN, nos muestra la lista de símbolos no pertenecientes al Alfabeto presentes en la cadena validada y el estado en el que fueron detectados; y los cuales fueron omitidos para la validación. Para este caso el único carácter omitido fue @.

* **Validación de la cadena “cbqbb”**

La tercera cadena de texto a validar será “cbqbb”.

Texto

Descripción generada automáticamente

Figura 24. Resultado de la validación de la cadena “cbqbb"

En este caso la cadena insertada es válida, por lo que se nos muestra el recorrido de estados realizado en el autómata con esta cadena (el cuál puede comprobarse usando como referencia el autómata de la Figura 20). El manejo de errores del AFN, nos muestra la lista de símbolos no pertenecientes al Alfabeto presentes en la cadena validada y el estado en el que fueron detectados; y los cuales fueron omitidos para la validación. Para este caso el único carácter omitido fue q.

* **Validación de la cadena “cb\_a”**

La tercera cadena de texto a validar será “cb\_a”.

Texto

Descripción generada automáticamente

Figura 25. Resultado de la validación de la cadena “cb\_a"

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. El manejo de errores del AFN, nos muestra la lista de símbolos no pertenecientes al Alfabeto presentes en la cadena validada y el estado en el que fueron detectados; y los cuales fueron omitidos para la validación. Para este caso el único carácter omitido fue \_.

1. **Conclusión:**

A manera de Conclusión, además de haber tenido la oportunidad de haber repasado los conceptos relacionados a la nomenclatura de Thompson y conocer la manera en la que se puede construir un AFN a partir de una expresión regular a partir de estas; puedo añadir que los AFN con transiciones épsilon resultan ser bastante útiles en el manejo de autómatas al brindarnos un método de poder representar expresiones regulares de una manera sencilla de entender, y fácil de interpretar para un compilador o interprete al proporcionarnos facilidades para programarlos.

Además, que a partir de el uso de nomenclaturas Thompson nos facilita el entender una expresión regular por medio de la construcción del AFN del mismo por medio del uso de Thompson, pues con estos se pueden evaluar cadenas o palabras de una manera sencilla para verificar si pueden ser aceptados por el AFN o no.

1. **Referencias:**

* Gaz.wiki. Construcción de Thompson. [online] Disponible en: <https://gaz.wiki/wiki/es/Thompson%27s\_construction> [Accedido el 22 de septiembre de 2021].
* M. en C. Edgardo Adrián Franco Martínez, 2013. Clase 09: AFN, AFD y Construcción de Thompson. [online] Docplayer.es. Disponible en: <https://docplayer.es/21780698-Clase-09-afn-afd-y-construccion-de-thompson.html> [Accedido el 22 de septiembre de 2021].
* Águila, J., 2004. PASO DE EXPRESIÓN REGULAR EN AFN. [online] Kataix.umag.cl. Disponible en: <https://kataix.umag.cl/~jaguila/Compilers/T05\_ER\_en\_AFN.pdf> [Accedido el 22 de septiembre de 2021].