Implementación de métodos computacionales

TC2037.2

Actividad Integradora 3.4:

Resaltador Léxico

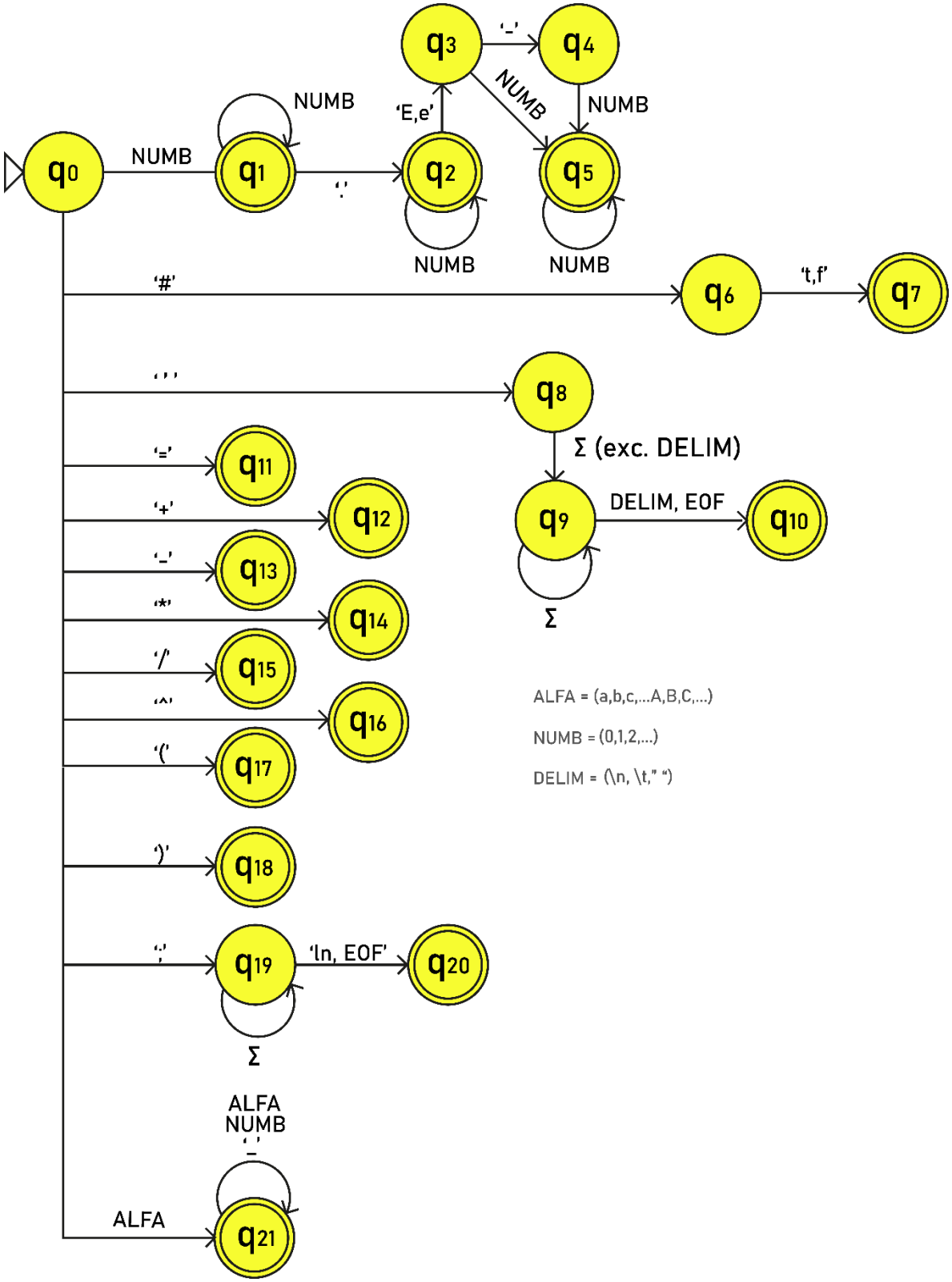
Profesor(a):

Alejandro de Gante

Abraham Mendoza Pérez	A01274857
Luis Alonso Martínez García	A01636255
Aldo Alejandro Degollado Padilla	A01638391

Domingo 25 de abril de 2021

1. **Mediante un diagrama (Autómata Finito), representar todas las categorías léxicas.**



2. **Reflexiona sobre la solución planteada, los algoritmos implementados y sobre el tiempo de ejecución de estos.**

Desde la entrega pasada, el equipo al estar analizando de qué manera podríamos resolver la situación planteada, fue primero a través de la creación del diseño de nuestro autómata finito, ya que de aquí partimos de una idea más clara sobre cómo es que podríamos implementar esto en el código a desarrollar en nuestro lenguaje de preferencia y la idea general que hubo entre los integrantes fue que el autómata, desde nuestro punto de vista, actuaba como un switch statement, ya que dependiendo del carácter que se detectará, es que este iba determinando el proceso en el cual debía entrar, a cual cambiar o incluso volver a llamarse a sí mismo para poder realizar de esa manera el funcionamiento.

De esta forma fue que se empezó a realizar la programación de la solución, en donde por medio de una variable que almacena el char actual, este va progresando por todo el código comparando entre los distintos procesos (cases), para así al final regresar un output en el que se muestre a qué tipo de token pertenecía siendo la entrada un archivo de texto tal y como se requería para la realización de la primera parte integradora de esta situación problema, pero claramente el funcionamiento del programa no se queda hasta aquí, ya que de alguna manera se tiene que ir leyendo tanto para las lectura de las líneas del archivo de entrada y la de los caracteres. La forma en la que desarrollamos esto fue con el uso de ciclos, específicamente con el uso de un while, encargado de la lectura de cada una de las líneas dentro del archivo de texto, dentro de un for, en el cual se va leyendo cada carácter de la línea, y este último tiene la función de revalidar a qué tipo de categoría léxica pertenece el char.

Ahora, aunque en nuestro algoritmo se repite varias veces el ciclo for donde leemos cada caracter, pues lo tenemos dentro de un while donde leemos

cada línea del texto de entrada, en realidad el número de iteraciones totales termina siendo el número de caracteres que tenemos en total por lo que el tiempo de ejecución sería T(n) = n.

3. **Calcula la complejidad de tu algoritmo basada en el número de iteraciones y contrastada con el tiempo obtenido en el punto 2.**

   Si calculamos entonces la complejidad en base a nuestro tiempo de ejecución obtendremos que: { O(1) } T(n) = n => O(n), o sea que, nuestro algoritmo implementado sería de complejidad lineal.

4. **Plasma en un breve reporte de una página las conclusiones de tu reflexión en los puntos 2 y 3.**

   El trabajo realizado en este proyecto era realizar un resaltador de sintaxis en el cuál podríamos identificar por medio de colores las diferentes categorías léxicas, en este caso del lenguaje *Scheme*. Para ello utilizamos un analizador basado en el diseño de un autómata finito, cuyo algoritmo fue implementado en el lenguaje C++, utilizando un switch statement para aplicar el concepto de los estados del autómata finito.

   La solución de este proyecto pudo realizarse de diversas formas, y debido a su complejidad intrínseca, teníamos que ser muy cuidadosos con el tiempo de ejecución del algoritmo implementado. Para ello entonces nos valimos de algunas funciones como getline() para obtener cada línea del archivo de texto de entrada para después recorrer cada carácter que se encuentre dentro de la misma, dónde evitamos que se pueda escapar algún caracter y utilizando posteriormente el switch mencionado para el proceso de cambio de estados.

   Con este algoritmo, en el ciclo for que utilizamos para el proceso de lectura de cada carácter, el número de iteraciones depende del número de

caracteres totales de entrada, siendo entonces, el número total de caracteres, el mismo número de iteraciones que realizará el algoritmo, por lo que, si calculamos el tiempo de ejecución, como ya lo mencionamos en puntos anteriores, tenemos: T(n) = n, lo que nos da una complejidad lineal, es decir, de O(n).

Con ello obtenemos una forma eficiente de realizar el análisis léxico, pues pasando por cada carácter se va realizando el cambio de estados, para determinar, en su caso, el estado final correspondiente y así mismo mandar la información de la categoría a la que corresponde cada parte del contenido del archivo de texto de entrada al archivo html, dónde se muestra el resultado final.

5. Video (explicación de los puntos 1 al 8).

Link de video: https://www.youtube.com/watch?v=AfbZvRw1I10