UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

ESCUELA DE CIENCIAS Y SISTEMAS

LABORATORIO DE LENGUAJES FORMALES DE PROGRAMACION

PROYECTO #1

MANUAL DE TECNICO

DENNIS MAURICIO CORADO MUÑÓZ

CARNET: 202010406

CUI: 3032329780108

GUATEMALA, GUATEMALA, 23/09/2021

Objetivos y Alcances

Objetivos generales

Aplicar los conocimientos de clase y laboratorio para poder realizar la creación de un analizador léxico por medio de la teoría de autómatas utilizando el método del árbol y así dar solución al problema planteado.

Objetivos específicos

- Analizar un archivo de entrada por medio de un autómata.
- Poder obtener la información necesaria del archivo para generar una imagen.
- Brindar información detallada de la estructura del archivo por medio de un reporte que contiene la lista de los tokens analizados.

Especificación Técnica

Requisitos de Hardware (Mínimos)

- **Microsoft Windows Vista**
  - **Procesador:** Intel Pentium III o equivalente
  - **Memoria:** 512 MB
  - **Espacio en disco:** 750 MB libres

- **Ubuntu 9.10**
  - **Procesador:** Intel Pentium III o equivalente
  - **Memoria:** 512 MB
  - **Espacio en disco:** 650 MB libres

- **Macintosh OS X 10.7 Intel**
  - **Procesador:** Intel Pentium III o equivalente
  - **Memoria:** 512 MB
  - **Espacio en disco:** 750 MB libres

Cualquier hardware superior al establecido es capaz de ejecutar la aplicación y poder ver sus archivos por medio de un editor de código.

Requisitos de Software

- **Sistema Operativo:** Windows, masOS, Linux
- **Lenguaje de Programación:** Python 3.9 o superior
- **IDE:** Visual Studio Code, Intellij, Pycharm

Lógica del Programa

- **Clases y Métodos Utilizados**

Para el desarrollo del programa se utilizó una programación modular separando el código en diferentes secciones y diferentes archivos, en el archivo main se cuenta con la clase main la cual contiene métodos propios de su clase para inicializar la interfaz y poder dirigir desde los botones a las demás funciones del programa, la clase Admin contiene métodos para cargar el archivo, obtener la informacion de la imágenes generando objetos de tipo imagen y guardándolos en una lista posteriormente se llama a una clase en otro archivo para generar el archivo html de cada imagen, la función analizar verifica que al ser procesado el archivo este no contenga errores y si este no contiene errores se puede utilizar la función obtener imágenes anteriormente descrita, luego se tienen métodos un poco mas auxiliares dentro de esta clase como el método reportes que se encarga de realizar el reporte de tokens, método getNames que devuelve los nombres que contiene el archivo, metodo getFiltros al igual que el anterior devuelve los filtros por cada imagen dentro del programa y el método imagen que se encarga de generar la imagen original.

La clase reportes crea un archivo html con la informacion de los tokens, la clase pintar como su nombre lo indica se encarga de pintar una tabla de html para poder generar la imagen, para que esto suceda se hace uso de 2 métodos de ordenamiento ambos funcionan por el método bubblesort y el primero ordena las filas de menor a mayor y el siguiente en base a las filas ordena las columnas de menor a mayor, el método takeimage hace uso de una librería para tomar una captura de pantalla del archivo html generado.

La clase analizador trabaja en conjunto con la clase token acá es donde se encuentra programado el autómata obtenido por medio del método del árbol y de esta forma es que se implementa haciendo uso de lexemas y funciones propias de Python.

Obtener el DFA

Tabla de Tokens y Expresiones Regulares

Letras = L = [a-z, A-Z]+ = L+
Números = N = [0-9]+ = N+
Resto = R = [Cualquier Símbolo]
Cadenas = “(L|N|R)*”
Signos = S = [igual, llave izquierda, llave derecha, coma, punto y coma, corchete izquierdo, corchete derecho]
Hexadecimal = H = #(L|N)*

*****Debido a que el carácter ‘#’ estaría dentro de la definición dentro de la expresión de hexadecimal se utilizo como estado de aceptación el signo ‘$’

Expresión Regular:

Expresión regular obtenida en base a la tabla de tokens y la definición de las expresiones regulares que contaría el programa.

[(L)+ | (N)+ | “(L|N|R)*” | (S)+ | #(L|N)*]$

Desarrollo del método del árbol

Árbol de la expresión regular:

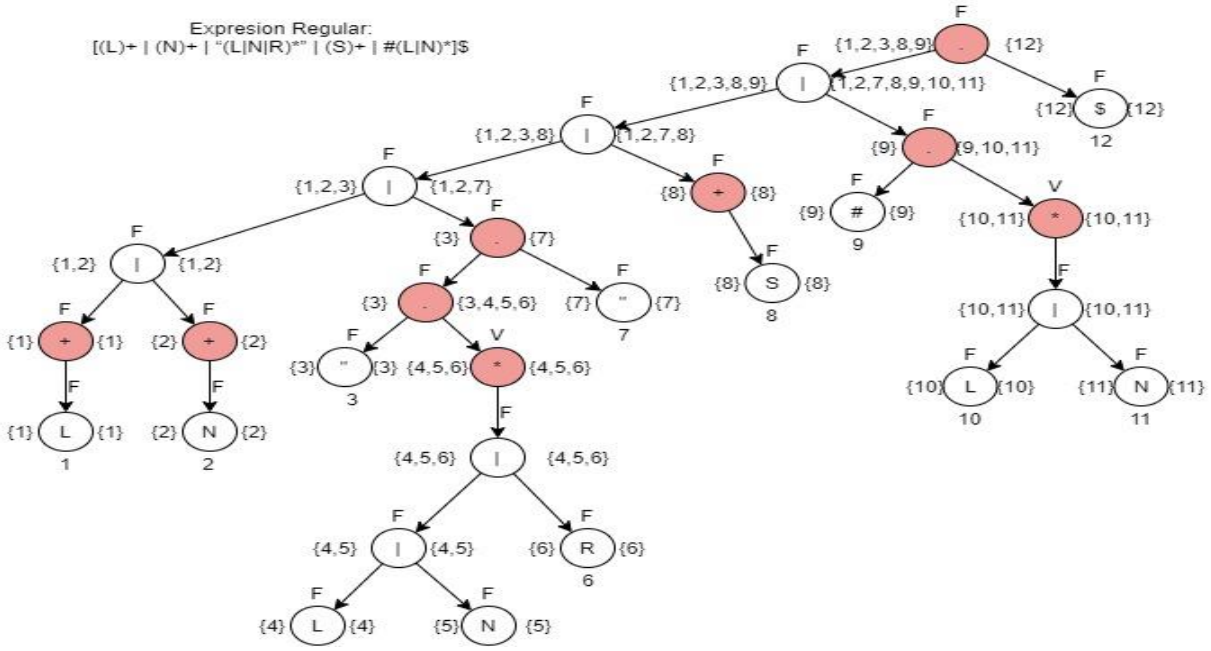

Tabla de siguientes y gramática regular:

No	Terminales	Siguiente
1	L	1,12
2	N	2,12
3	"	4,5,6,7
4	L	4,5,6,7
5	N	4,5,6,7
6	R	4,5,6,7
7	"	12
8	S	8,12
9	#	10,11,12
10	L	10,11,12
11	N	10,11,12
12	$	

So = {1,2,3,8,9}; L N " S #
Sig(1) = Sig(L) = {1,12} = S1
Sig(2) = Sig(N) = {2,12} = S2
Sig(3) = Sig(") = {4,5,6,7} = S3
Sig(8) = Sig(S) = {8,12} = S4
Sig(9) = Sig(#) = {10,11,12} = S5
S1 = {1,12}; L $
Sig(1) = Sig(L) = {1,12} = S1
S2 = {2,12}; N $
Sig(2) = Sig(N) = {2,12} = S2
S3 = {4,5,6,7}; L N R "
Sig(4) = Sig(L) = {4,5,6,7} = S3
Sig(5) = Sig(N) = {4,5,6,7} = S3
Sig(6) = Sig(R) = {4,5,6,7} = S3
Sig(7) = Sig(") = {12} = S6
S4 = {8,12}; S $
Sig(8) = Sig(S) = {8,12} = S4
S5 = {10,11,12}; L N $
Sig(10) = Sig(L) = {10,11,12} = S5
Sig(11) = Sig(N) = {10,11,12} = S5
S6 = {12}; $
Sig(12) = Sig($) = ACEPTACION

Autómata Finito Determinista: