Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Lenguajes formales y de programación

Primer semestre, 2022.

MANUAL TÉCNICO

Proyecto 1

Diego Andrés Huite Alvarez

202003585

17/03/2022

Contenido

INTRODUCCIÓN

El presente manual tiene como el explicar como funciona el analizador léxico elaborado para el proyecto 1 de LFP.

REQUERIMIENTOS

Para el correcto funcionamiento del software usted necesita del siguiente hardware:

- Mouse
- Teclado
- Memoria RAM (4gb mínimo)

Y en cuanto a software usted necesitará:

- Navegador
- Cualquier versión de Python superior a la 3.0.0
- Cualquier IDE para python.
- Sistema operativo tipo linux

LÓGICA DEL PROGRAMA

- **Análisis previo del lenguaje**

Para comenzar el análisis del siguiente lenguaje:

```
formulario ~>> [

        <
                tipo: "etiqueta",
                valor: "Nombre:"


        >,

        <
                tipo: "texto",
                valor: "Nombre:",
                fondo: "ingrese ####nombre"
        >,

        <
                tipo: "grupo-radio",
                nombre: "sexo",
                valores: ['Masculino', 'Femenino']
        >,

        <
                tipo: "grupo-option",
                nombre: "pais",
                valores: ['Guatemala', 'El salvador', 'Honduras']
        >,

        <
                tipo: "boton",
                valor: "Valor",
                evento: "entrada"
        >
]
```

debemos de llevar a cabo un proceso que se llama “tokenización” que consiste en hallar los tokens del lenguaje. Para este caso en específico se llegó a la siguiente tabla de tokens:

Descripcion	Patron	Regex
identificador	Una letra seguida de otra letra	[A-Za-z]+
corchete izquierdo	Caracter [	[
corchete derecho	Caracter]	]
signo menor	Caracter <	<
signo mayor	Caracter >	>
dos puntos	Caracter :	:
virgulilla	Caracter ~	~
coma	Caracter ,	,
valor_string	Cualquier cadena encerrada en comillas dobles o simples	“[^”]” | ‘[^’]’
palabra reservada nombre	nombre	nombre
palabra reservada valor	valor	valor
palabra reservada valores	valores	valores
palabra reservada tipo	tipo	tipo
palabra reservada evento	evento	evento
palabra reservada fondo	fondo	fondo

Una vez teniendo la tabla de tokens podemos generar la siguiente expresión regular: \ “[^”]” | ‘[^’]’ | < | > | , | ~ | : | [A-Za-z]+

Esta expresión puede ser simplificada generalizando los símbolos

Sea S = {< , > , , , ~ , :}

Sea L = {[A-Za-z]}

La expresión regular transformada luciría así:

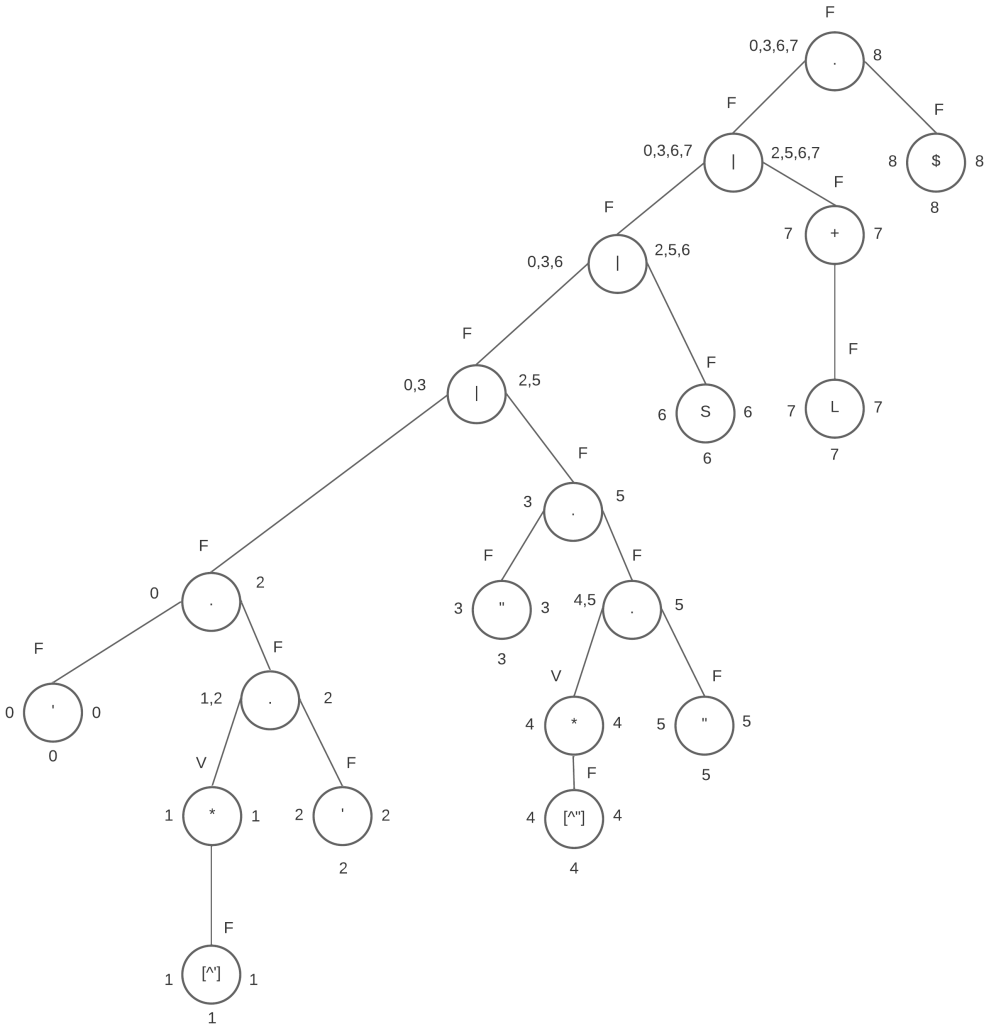
\ “[^”]” | ‘[^’]’ | S | L+

Generación de AFD

Para generar el AFD se debe de implementar un símbolo de finalización a la expresión regular. Siendo este símbolo “$”. La nueva expresión regular luce así:

\ (“[^”]” | ‘[^’]’ | S | L+) $

Se procede a generar el árbol binario para esta expresión:



Teniendo el árbol podemos generar la tabla de siguientes.

Tabla de siguientes:

Hoja	Terminal	Siguientes
0	‘	1,2
1	‘[^’]’	1,2
2	‘	8
3	“	4,5
4	“[^”]”	4,5
5	“	8
6	S	8
7	L	7,8
8	$	-

Generamos las transiciones posibles:

q0 = {0,3,6,7}

sig(0) = sig(‘) = {1,2} → q1

sig(3) = sig(“) = {4,5} → q2

sig(6) = sig(S) = {8} → q3

sig(7) = sig(L) = {7,8} → q4

q1 = {1,2}

sig(1) = sig([^’]) = {1,2} → q1

sig(2) = sig(‘) = {8} → q3

q2 = {4,5}

sig(4) = sig([^”]) = {4,5} → q2

sig(5) = sig(”) = {8} → q3

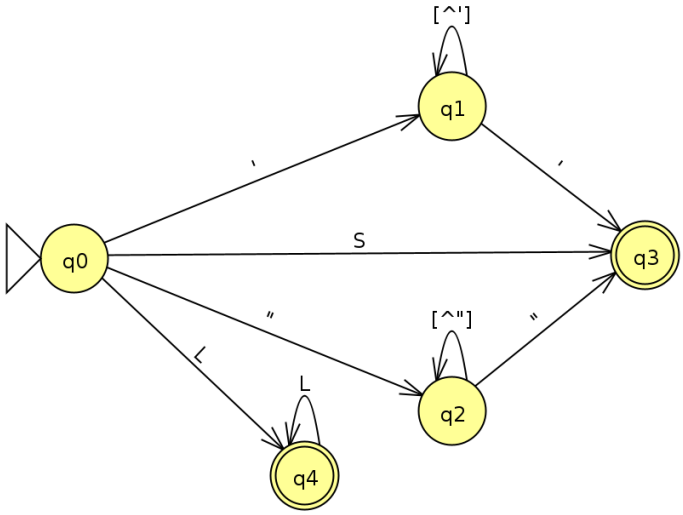
q4 = {7,8}

sig(7) = sig(L) = {7,8}→ q4

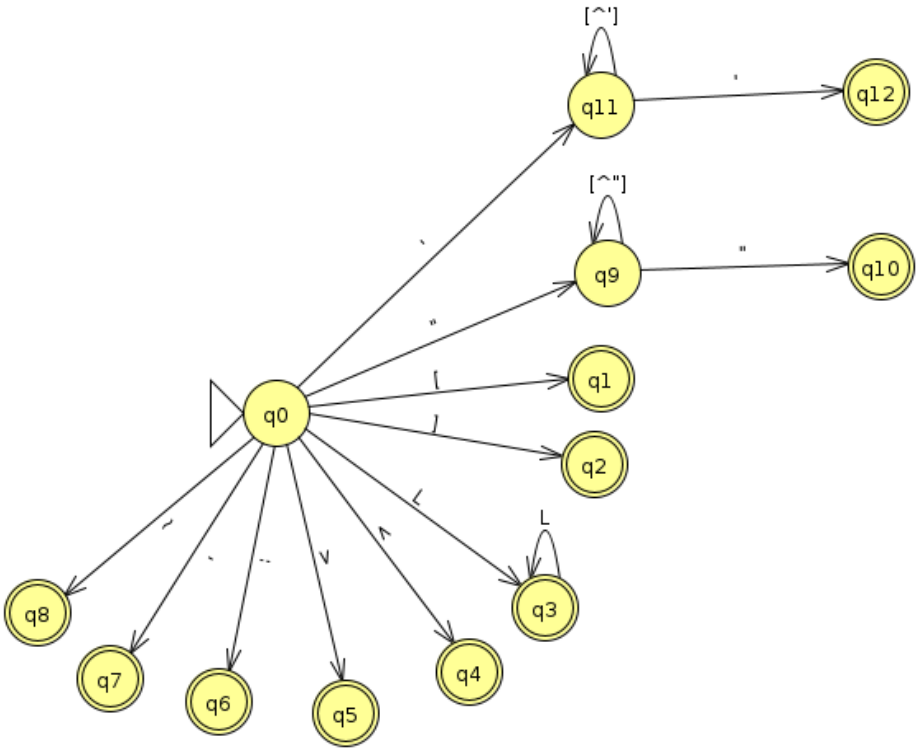
Tabla de transiciones

Estados/ terminales	‘	[^’]	“	[^”]	S	L
q0	q1	-	q2	-	q3	q4
q1	q3	q1	-	-	-	-
q2	-	-	-	-	-	-
*q3	-	-	-	-	-	-
*q4	-	-	-	-	-	q4

Se procede a dibujar el AFD



Por motivos de aprendizaje se ampliaron los dividió el conjunto S y se añadieron más estados. Por lo que el AFD ampliado luce así:



Este fue el automata utilizado para la codificación del programación.

CLASES UTILIZADAS

Diagrama UML de las clases que se usaron para la elaboración del software.