Manual De Usuario

“Constructor de análisis léxico y sintáctico”

Proyecto 1, [OLC1]

Facultad de Ingeniería

Marzo, 2022

Proyecto 1 [OLC1]

Constructor de análisis léxico y sintáctico

Autor:

Sergie Daniel Arizandieta Yol

202000119

Facultad de ingeniería

Organización de Lenguajes y Compiladores 1

Universidad de San Carlos de Guatemala

Guatemala, marzo 2022

I. Objetivos

1. Objeto del documento

El documento tiene como finalidad proporcionar una guía del software con la cual puede conocer el manejo adecuado y correcto que la aplicación necesita para el funcionamiento de dicho programa.

2. Objetivos

- Otorgar al usuario una explicación grafica simple y concisa de entender todas las funcionalidades que el software posee de manera lógica y del mismo modo que simula los procesos requeridos.

- Entregar al usuario las indicaciones y pasos requeridos a seguir (un algoritmo) para que la simulación de la gestión de datos sea la adecuada y cumpla su función.

- Que todo usuario que utilice el software sea capaz de cumplirse con las finalidades de este.

II. Introducción

Este manual de usuario tiene como finalidad dar a conocer a los usuarios que utilicen el software las operaciones que brinda el software junto los pasos a seguir para el uso eficaz, para que dicho software “Constructor de análisis léxico y sintáctico” cumpla su objetivo. Por lo cual se dará breves explicaciones del manejo de la UI (Interfaz de Usuario) y sus funcionalidades, así como el uso de archivos, lectura y resultados para el mismo, junto guía gráfica para su mayor compresión.

Este software está orientado a la carga y gestión de archivos de entrada de expresiones regulares para su análisis, creación de autómatas determinista y no determinista, así mismo la creación de reportes en caso de errores.

III. Descripción del programa

“Constructor de análisis léxico y sintáctico” como el nombre lo indica recibe un archivo entrada para gestiona miento léxico y sintáctico del mismo, en la cual se tendrá la posibilidad de generar autómatas determinista y no determinista mediante el método del árbol y Thomson correspondientemente partiendo del archivo de entrada, teniendo estos generados se podrá validar cadenas de entrada para definir si estas son aceptadas por las expresiones regulares definidas.

Así mismo de brindará los pasos del método del árbol para poder visualizar el procedimiento y como salida final se tendrá un archivo JSON con los resultados de la cadena de una forma clara y concisa, junto la misma información en consola.

I. Requerimientos del sistema

Software mínimo

- 2 GB de RAM
- Windows vista
- Arquitectura 32 bits
- Espacio en sistema 128 MB
- Java 8

Java

Tener instalado java ya que fue el lenguaje con el que se ejecuta “Constructor de análisis léxico y sintáctico”

II. Requerimientos archivo de entrada

Para el funcionamiento de este los archivos de entrada debe cumplir con la estructura presentada en la figura 1 con extensión ".exp".

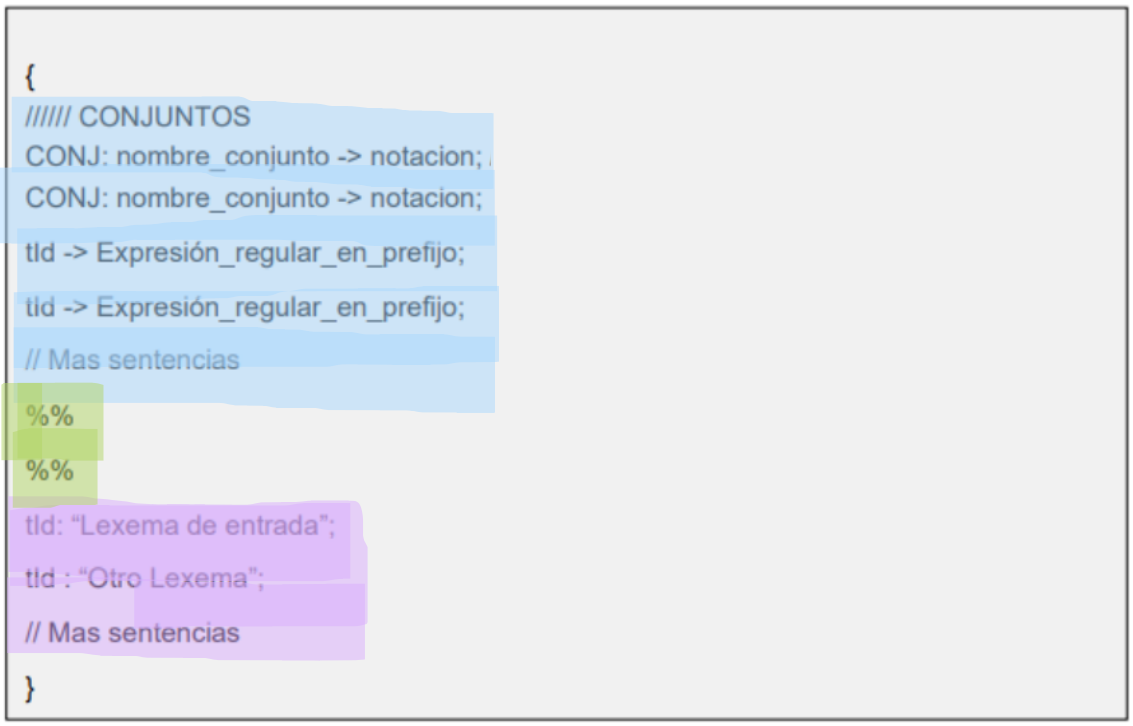


Figura 1.

El documento vendrá con llaves de inicio y al final de del archivo, donde dicho contenido se divide en 2 secciones delimitado por un par de "%%" los cuales hacen de división, las 2 secciones se dividen en:

- Primera parte, color: esta sección es para la parte de definición de **conjuntos** y **expresiones regulares**
- Segunda parte, color: esta sección es para la definición para las **cadenas** a validar mediante una expresión regular asignada.

En todo el documento pueden asignarse comentarios de la siguiente manera:

Comentario de una línea	Comentario multilínea
Se apertura con: "//" y abarca todo hasta el salto de linea	Se apertura con: "<!" y finaliza con "!>" lo cual puede puede abarcar cualquier cadena dentro de ellos
// soy un comentario	**<! Este es un comentario de una o muchas líneas !>**

Conjuntos

Estos se definen por la palabra reservada CONJ la cual puede ser mayúsculas o minúsculas seguido de ":", seguido del nombre del conjunto y su definición:

- Por separación de comas
- Por rango

Por separación de comas

Como el titilo lo indica son caracteres separados por comas ej: 1,2,3 - a,b,c – A,B,C – 1,b,C

Por rango

Este abarcara un rango de caracteres asignado siguiendo la siguiente tabla:

Notación	Definición
a~c	Conjunto {a, b, c}.
a~z	Conjunto de la a hasta la z en minúsculas.
A~Z	Conjunto de la A hasta la Z en mayúsculas.
0~7	Conjunto del 0 al 7.
!~&	Conjunto de signos entre ! (33 en código ascii) y & (38 en código ascii). Nota: el rango valido será desde el ascii 32 hasta 125 omitiendo los ascii de las letras y dígitos.

Expresiones regulares

Estas deben ser definidas en forma prefija siguiendo los siguientes operadores

Notación	Definición
. a b	Concatenación entre a y b
| a b	Disyunción entre a y b
* a	0 o más veces
+ a	1 o más veces
? a	0 o una vez

Al mismo tiempo se puede hacer referencia a los conjuntos anteriormente definidos de la siguiente forma:

{nombre del conjunto}

Un ejemplo de implementación seria:

. *{conj1}{conj2}* = concatenación entre conj1 y conj2

La definición de las mismas se realiza con el nombre de la expresión regular seguida de "->":

Exp1 -> . {conj1}{conj2}

Caracteres especiales

Dentro del lenguaje pueden utilizarse estos caracteres especiales:

Notación	Definición
\n	Salto de línea
\’	Comilla Simple
\”	Comilla Doble

Cadenas

Estas se definen con el nombre de la expresión regular, dos puntos y consecuentemente la cadena como tal a validar:

Nombre: “cadena”

Ej.

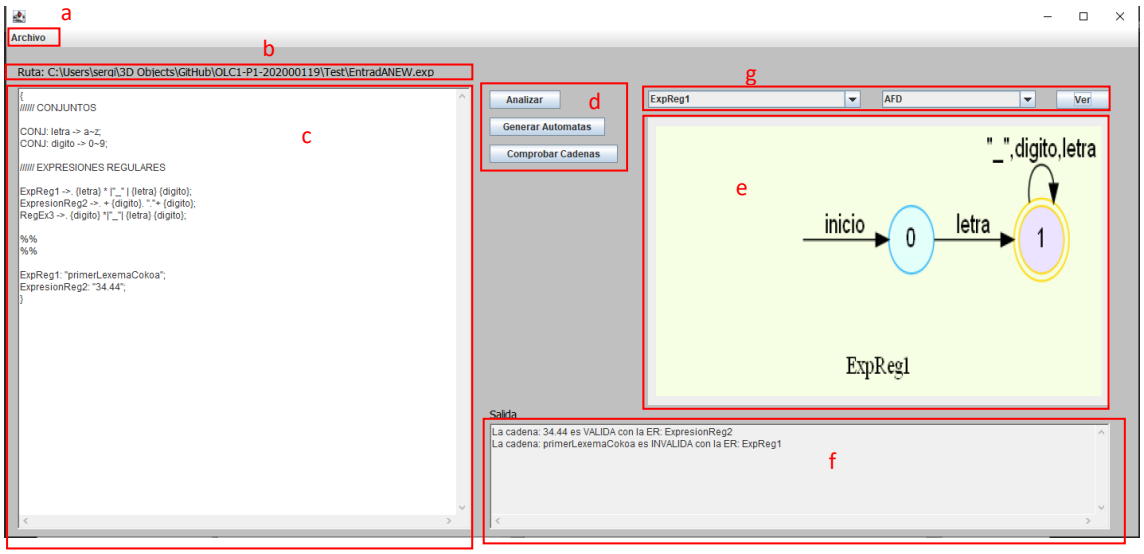
Expre1: “ejemplo de cadena a validar”;

Archivo de ejemplo:

```
{
////// CONJUNTOS

CONJ: letra -> a~z;
CONJ: digito -> 0~9;

/////// EXPRESIONES REGULARES

ExpReg1 -> . {letra} * | "_" | {letra} {digito};
ExpresionReg2 -> . {digito} . "." + {digito};
RegEx3 -> . {digito} * | "_" | {letra} {digito};

%%
%%

ExpReg1 : “primerLexemaCokoa”;
ExpresionReg2 : “34.44”;
}
```

III. Operaciones del sistema

1. Partes de la interfaz



a) Operaciones de archivos
Despliega un menú con las operaciones de archivos disponibles

b) Ruta de archivo
Muestra la ruta del archivo editado.

c) Editor de texto
Permite la edición del archivo cargado.

d) Operaciones de análisis
Opciones para el manejo del análisis.

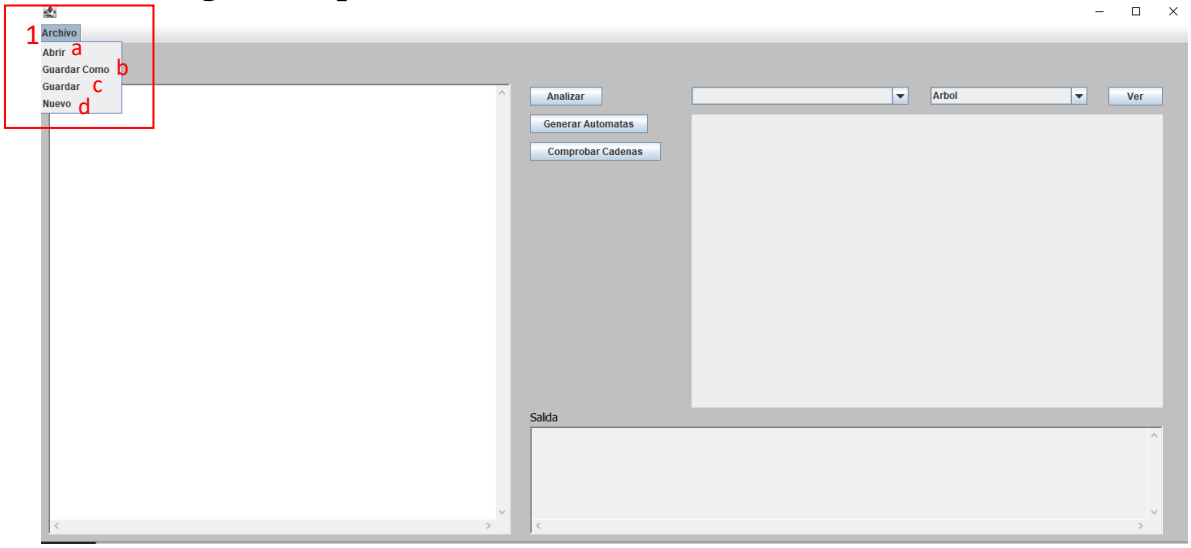
e) Visualizador de grafos
Permite visualizar los grafos.

f) Consola
Muestra de forma resumida las salidas de cadenas validadas.

g) Opciones de visualización
Permite seleccionar el grafo especifico a desplegar en el visualizador de grafos.

2. Operaciones de archivos

a. Carga de requisitos



h) Abrir

Permite abrir un archivo existente de extensión “exp” y desplegar la información en el cuadro de edición de texto

i) Guardar Como

Permite guardar el documento abierto para guardarlo con otro nombre distinto.

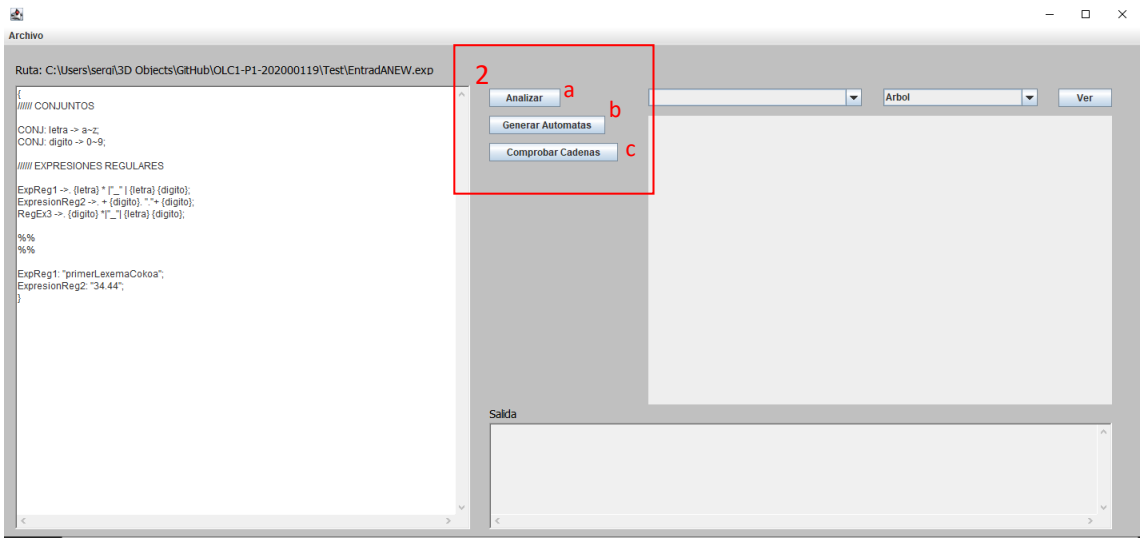
j) Guardar

Permite guardar el archivo actualmente editado.

k) Nuevo

Permite abrir un nuevo archivo en blanco.

3. Operaciones de análisis



a) Analizar

Procede a analizar el archivo que se esta editando el cual se puede visualizar en la ruta, así mismo con la detección de errores léxicos y semánticos.

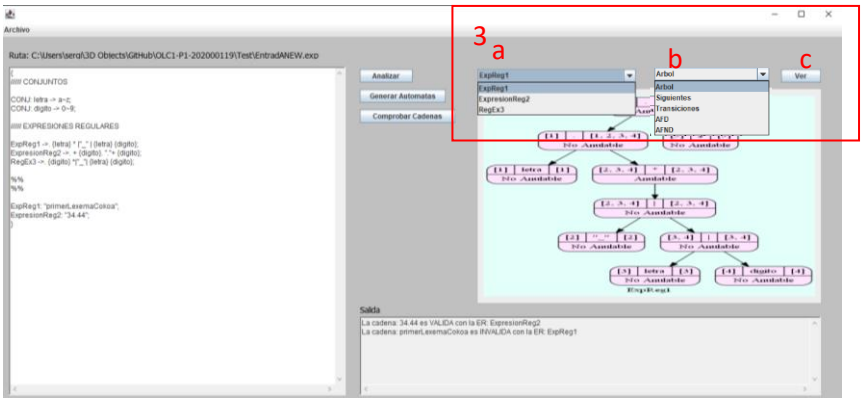
b) Generar Autómatas

Realiza el método del árbol para obtener el autómata determinista y también el método de Thomson para el autómata no determinista, donde se generarán los pasos del árbol para la visualización aparte de los autómatas.

c) Comprobar Cadenas

Realiza la comprobación de las cadenas con su expresión regular asignada arrojando como resultado si fue valido o invalido en caso de no encontrar la expresión regular asignada un no encontró la expresión regular.

4. Opciones de visualización



a. Lista de expresiones regulares

Permite seleccionar alguna de las expresiones regulares definidas en el archivo de entrada para la visualización de algún grafo.

b. Lista de grafos

Permite seleccionar alguna de las opciones de grafo que cada una de las expresiones regulares posee

c. Ver

Despliega en el área para grafos el grafo seleccionado de la expresión regular indicada

IV. Reporte de errores

1) Errores

Al intentar analizar el archivo de entrada y se genere un error este se le notificara y al mismo tiempo generando un reporte de errores el cual está hecho en HTML.

2) Ruta del reporte

Este se generará en una carpeta de nombre “ERRORES_202000119” la cual se encontrará en la carpeta donde el usuario haya ejecutado el software.

V. JSON salida

a. Archivo JSON

Este archivo de extensión “JSON” contiene el resultado del análisis de las cadenas establecidas de la siguiente manera en la Figura 2

b. Ruta

Este se generará en una carpeta de nombre “SALIDAS_202000119” la cual se encontrará en la carpeta donde el usuario haya ejecutado el software.


```
[
  {
    "Resultado": "Cadena Valida",
    "ExpresionRegular": "ExpresionReg2",
    "Valor": "34.44"
  },
  {
    "Resultado": "Cadena Invalida",
    "ExpresionRegular": "ExpReg1",
    "Valor": "primerLexemaCokoa"
  }
]
```


Figura 2.

VI. Grafos

1. Ruta de los grafos

Existen diferentes tipos de grafos generados por cada expresión regular, estos se generarán en la carpeta del proyecto donde el usuario este ejecutando el software, los tipos de grafos son:

a) Árbol
   a. Grafo del árbol binario de la expresión regular en “ARBOLES_202000119”.

b) Siguientes
   a. Grafo de los siguientes de las hojas de el árbol en “SIGUIENTES_202000119”.

c) Transiciones
   a. Grafo sobre las transiciones de estados para el autómata finito determinista en “TRANSICIONES_202000119”.

d) Autómata Finito Determinista
   a. Grafo del AFD en “AFD_202000119”.

e) Autómata Finito No Determinista
   a. Grafo del AFND en “AFND_202000119”.