Manual De Usuario

Proyecto 1

Kevin Steve Martinez Lemus
[OLC1] Universidad De San Carlos De Guatemala

Índice

I. Introducción

Objetivos

Objetivo General

- Aplicar los conocimientos sobre la fase de análisis léxico y sintáctico de un compilador para construcción de una solución de software que permita generar análisis por medio del método del árbol.

Objetivos Específicos

- Reforzar el concepto del método de Árbol de expresiones regulares en Autómatas Finitos Deterministas (AFD).
- Reforzar el concepto del método de Thompson de expresiones regulares en Autómatas Finitos No Deterministas (AFND).
- Identificar y programar el proceso de reconocimiento de lexemas mediante el uso de Autómatas Finitos Determinista

Requerimientos

- Tener Java (versión 8 o superiores) instalado.
- Se recomienda tener un editor de código (NetBeans 8.2 u otro) para ver y editar el código de este programa.

II. Opciones Del Sistema

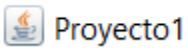
El presente Manual está organizado de acuerdo con la secuencia de ingreso a las pantallas del sistema de la siguiente manera:

1. Ingreso al Sistema.
2. Lenguaje.
3. Archivo.
4. Generar Autómatas.
5. Analizar Entradas.
6. Errores.

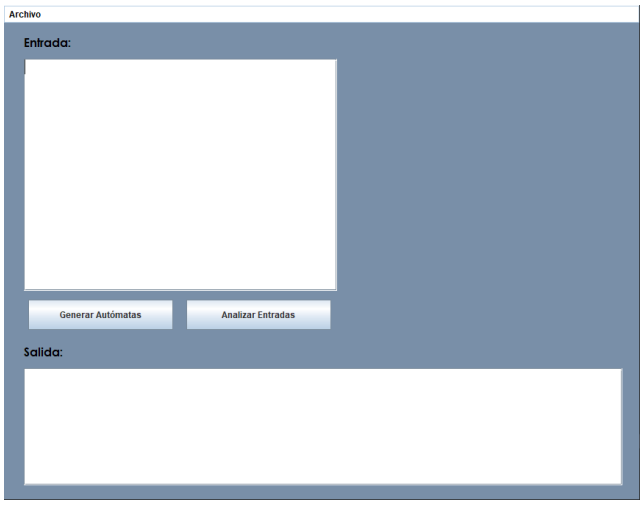
1. Ingreso al Sistema

Para poder ejecutar el programa necesitamos abrir Netbeans 8.2, abrir el proyecto y ejecutarlo:

O simplemente abrir el ejecutable JAR creado:



Al ejecutar el programa aparecerá la siguiente interfaz gráfica:



2. Lenguaje

Todo el código para ejecutar debe de estar entre llaves, el primer componente que se denomina conjunto se definirá de la siguiente forma:

```
CONJ: nombre_conjunto -> notacion;
```

En donde el id deberá iniciar con alguna letra o guion bajo seguido de números letras o guiones bajos, para la notación se puede definir los siguientes conjuntos:

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.
0,2,4,6,8	Conjunto {0, 2, 4, 6, 8}

La siguiente parte corresponde a las expresiones regulares, en donde:

```
tld -> Expresión_regular_en_prefijo;
```

En donde tld representa el nombre de la expresión y la expresión tendrá la siguiente estructura:

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

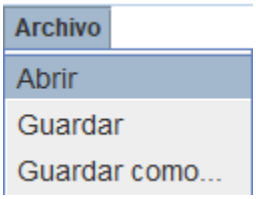
Para finalizar la parte de las entradas estará separada con el símbolo (%%) y se recibirán de la siguiente manera:

```
tld: "Lexema de entrada";
```

Cabe recalcar que todas las sentencias deberán terminar en punto y coma (;).

3. Archivo

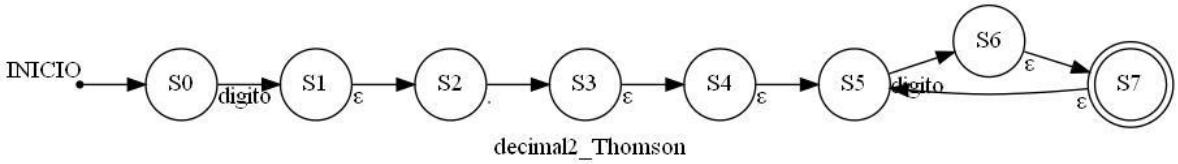
En la opción de archivo que se encuentra en la parte superior a la izquierda, se podrán encontrar las siguientes opciones:



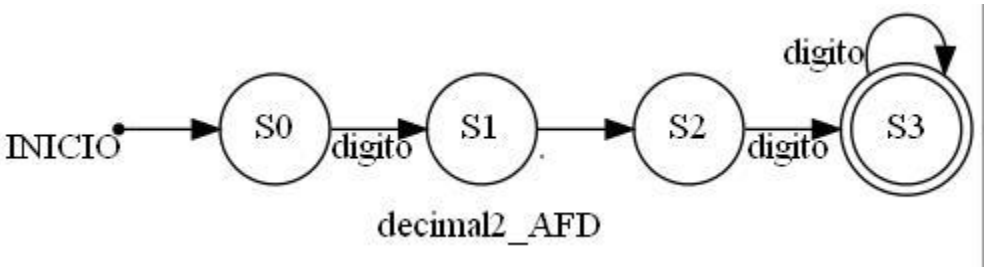
Acá se podrá abrir un archivo existente con extensión .exp que contenga el código que se desea ejecutar, asimismo, se puede guardar los cambios de este, o guardar como un archivo nuevo.

4. Generar Autómatas

El botón de generar autómatas permite la creación de una AFND (Autómata Finito No Determinista) creado por el método de Thomson, así como un AFD (Autómata Finito Determinista) creado por el método del árbol y los pasos necesarios para crearlos como la tabla de siguientes y tabla de transiciones. Estos se podrán visualizar en su carpeta correspondiente dentro del proyecto. El ejemplo de los autómatas es el siguiente:



decimal2_Thomson



decimal2_AFD

5. Analizar Entradas

El botón de analizar entradas realiza la verificación adecuada a las entradas propuestas, verificándolas con la expresión regular respectiva, imprimiendo en consola si es válida o no. Asimismo, se creará un archivo de salida .JSON en donde se escribirá los resultados de la misma. El archivo generado tendrá la siguiente estructura:

```
[
   {
     "Valor":"aa788787",
     "ExpresionRegular":"expre2",
     "Resultado":"Cadena Válida"
   },
   {
     "Valor":"aa",
     "ExpresionRegular":"expre2",
     "Resultado":"Cadena No Válida"
   },
   {
     "Valor":"aazbbb",
     "ExpresionRegular":"expre2",
     "Resultado":"Cadena No Válida"
   }
]
```

6. Errores

Si en cualquier parte de la ejecución del código llegara a ocurrir un error ya sea léxico o sintáctico, el programa automáticamente generará un archivo .html con el detalle de los mismos con el siguiente formato:

Lista De Errores

#	Tipo	Descripción	Linea	Columna
1	Sintáctico	Se detectó un error sintáctico ("\"")	1	20
2	Sintáctico	Se detectó un error sintáctico ("\"")	1	20