Trabajo práctico N° 2
SSL Autómatas

Nombre y Apellido	Legajo	Puntos
Stefano Alejandro Gassmann	208.380-2	1 y 2
Santiago Rodriguez	209.171-9	3

Índice:

Introducción:

Para comprender la metodología empleada en nuestro proyecto, es importante señalar que seguimos un enfoque estructurado. Inicialmente, se procedió con la conceptualización y diseño de una expresión regular que capturara el patrón de interés en un conjunto de cadenas de texto.

1. **Diseño de la Expresión Regular:** En una etapa inicial, se definió cuidadosamente una expresión regular (ER) que representará de manera precisa el patrón deseado. Una expresión regular se utiliza para describir un conjunto de cadenas de caracteres y es fundamental para la posterior implementación del proceso.

Hemos llegado a la conclusión de que la estrategia más eficaz es la creación de tres expresiones regulares distintas, cada una destinada a reconocer y capturar números en diferentes formatos: octales, decimales y hexadecimales.

1. **E**xpresión Regular para Números Octales:
    - **U**n número octal consiste en dígitos en el rango de 0 a 7.
    - **C**omienza con un cero seguido de uno o más dígitos octales.
    - **E**l valor 00 representa al nulo en octal.

***ER** utilizada:*

```
0.(0+[1,7].[0,7]*)
```

2. **E**xpresión Regular para Números Decimales:
    - **L**os números decimales se componen de dígitos en el rango de 0 a 9.
    - **P**ueden ser positivos(sin signo) o negativo(signo -) opcional al principio.
    - **E**l valor 0 representa al nulo en decimal.

***ER** utilizada:*

```
(0+(-.[1-9].[0-9]*)+([1-9].[0-9]))
```

***ER** utilizada:*

3. **E**xpresión Regular para Números Hexadecimales:
    - **L**os números hexadecimales consisten en dígitos hexadecimales (0-9 y A-F).
    - **E**stán precedidos por "0x" " como prefijo.
    - **E**l valor 0x0 representa al nulo en hexadecimal.

***ER** utilizada:*

0x.(0+([1-9 A-F].[0-9 A-F]*))

Esta estrategia de tres expresiones regulares independientes optimiza la detección y captura de los diferentes tipos de números, garantizando un procesamiento preciso y eficiente en todo el proyecto.

2. **Algoritmo de Clausura de Epsilon:** En el contexto de los autómatas finitos no deterministicos (AFN) obtenidos, se empleó el algoritmo de clausura de epsilon. Este algoritmo resulta esencial para determinar todos los estados a los que se puede llegar desde un estado dado mediante transiciones epsilon, las cuales no consumen ningún símbolo de entrada. La aplicación de este algoritmo nos permitió reducir el tamaño de código en nuestro proyecto.

TT para Números Octales :

TT	0	1-7
0--	1	5
1	2	3
2++	5	5
3++	4	4
4++	4	4
5	5	5

Nota: El estado 5 es un estado de rechazo.

TT para Números Decimales:

TT	0	1-9	-
0	1	2	3
1++	7	7	7
2++	4	4	7

3	7	5	7
4++	2	7	7
5++	6	6	7
6++	6	6	7
7	7	7	7

Nota: el estado 7 es el de rechazo.

TT para Números Hexaecimales:

TT	0	1-F	X
0--	1	6	6
1	6	6	2
2	3	4	6
3++	6	6	6
4++	5	5	6
5++	5	5	6
6	6	6	6

Nota: el estado 6 es de rechazo.

3. **Algoritmo de Reducción**: Posteriormente, se implementó un algoritmo de reducción. Este proceso fue fundamental ya que permitió simplificar el AFD, lo que facilitó su comprensión y su aplicación en la solución del problema

TT reducida para Números Octales :

TT	0	1-7
0--	1	4
1	2	3

2++	4	4
3++	3	3
4	4	4

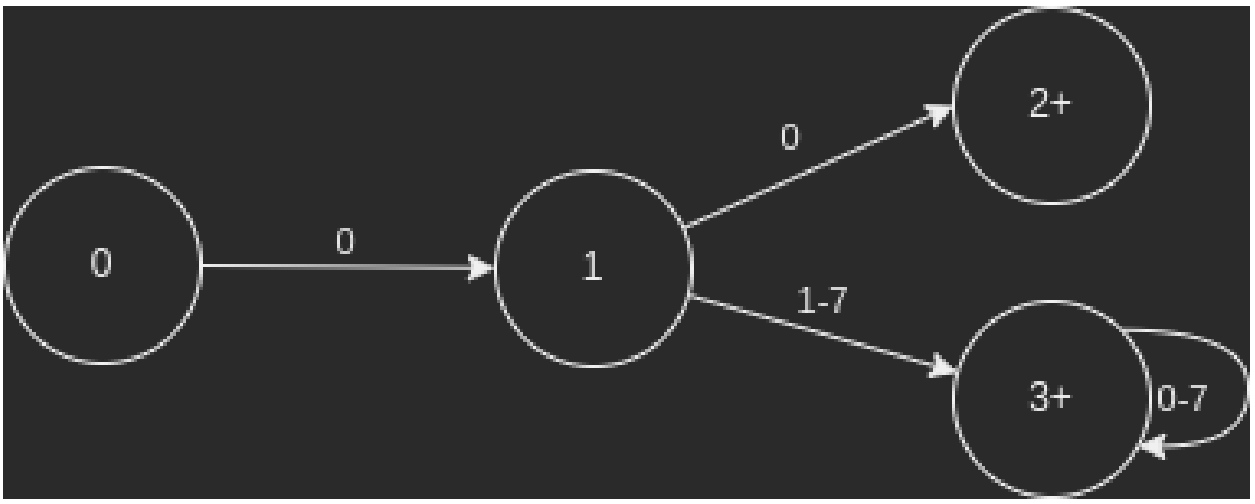
TT reducida para Números decimales :

TT	0	1-9	-
0-+	1	2	3
1++	5	5	5
2++	2	2	5
3	5	4	5
4++	4	4	5
5	5	5	5

TT reducida para Números hexadecimales :

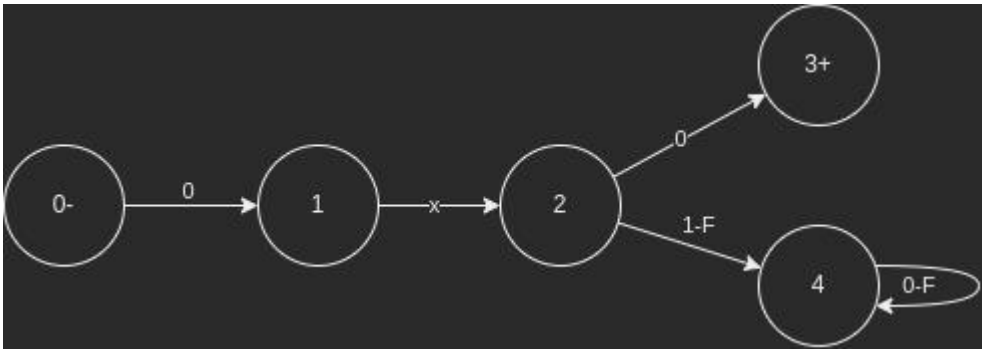
TT	0	1-F	X
0--	1	5	5
1	5	5	2
2	3	4	5
3++	5	5	5
4++	4	4	5
5	5	5	5

AFD octal:



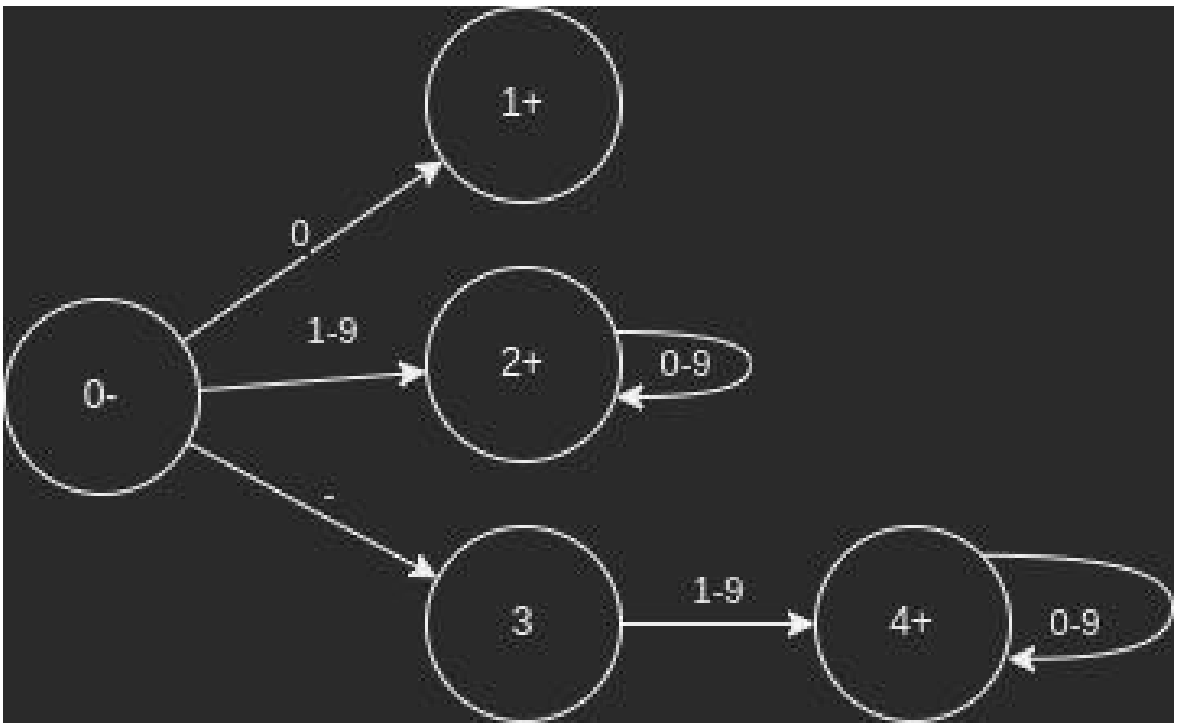
Nota: No está diagramado el estado de rechazo.

AFD hexa:



Nota: No está diagramado el estado de rechazo.

AFD Decimal:

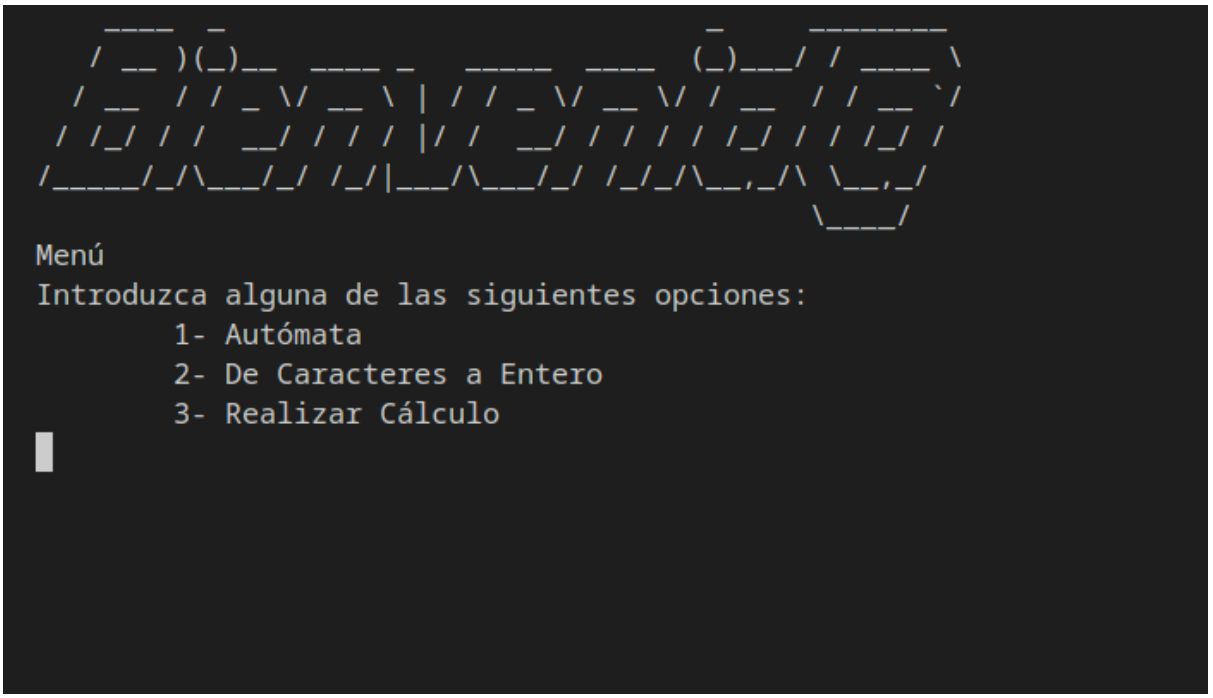


Nota: No está diagramado el estado de rechazo.

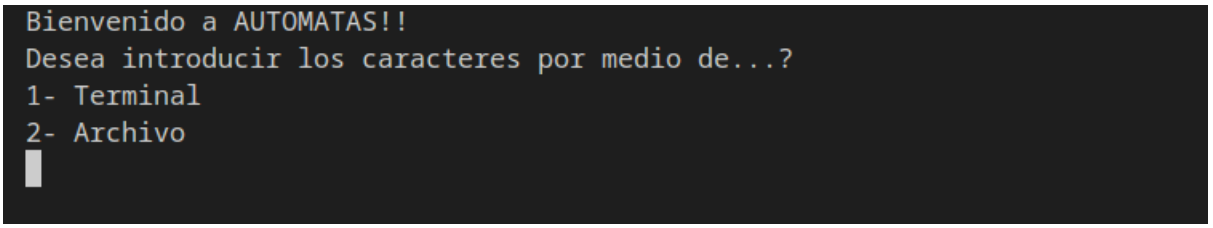
En resumen, este enfoque metodológico nos permitió desarrollar una solución robusta y eficiente para abordar el problema en cuestión, al utilizar expresiones regulares, el algoritmo de clausura de epsilon y el algoritmo de reducción para obtener un autómata finito determinista (AFD) adecuado para nuestro propósito.

Funcionamiento del programa:

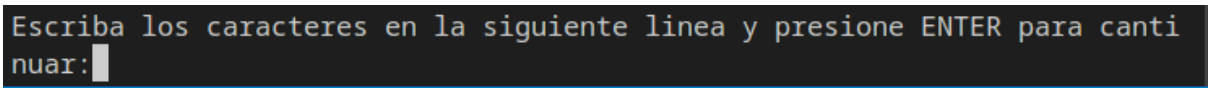
Al iniciar el programa se nos muestra una bonita pantalla de inicio desde la terminal.



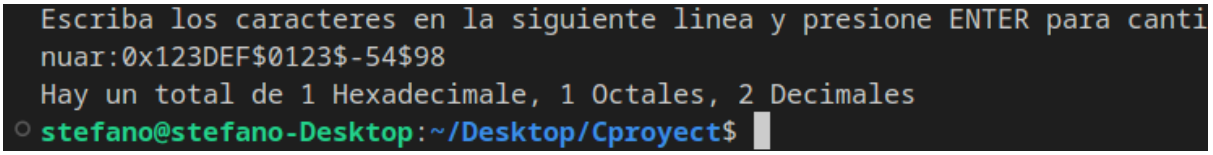
En este caso, seleccionamos la primera opción, introduciendo en valor **1** en la terminal y luego dando **enter**.



Primero comenzaremos por seleccionar la primera opción (Terminal) introduciendo un **1** en la misma y posteriormente dándole al **enter**, mostrándonos el siguiente mensaje.



Para realizar una prueba introduciremos los caracteres *0x123DEF0123-54$98* lo que corresponde a un número hexadecimal, seguido de un octal y por último dos números decimales. Luego damos **enter**.



Nos imprime la cantidad de números que hay de los diferentes tipos.

Pasemos ahora a ver la opción de introducirlo por consola, para ello en vez de seleccionar la opción **1** en el menú de Autómatas, introducimos un **2** y luego damos al **enter**.

```
Bienvenido a AUTOMATAS!!
Desea introducir los caracteres por medio de...?
1- Terminal
2- Archivo
2
Introduzca los caracteres en el archivo "caracteres.txt"
Al terminar, guarde y pulse alguna letra
```

Se nos creará un archivo llamado "***Caracteres.txt***" en el cual debemos introducir la serie de caracteres, guardamos en archivo e introducimos cualquier otro caracter y damos enter.

Nota: en el archivo. introducimos los mismos caracteres que en el ejemplo anterior.

```
caraxteres.txt
1   0x123DEF$0123$-54$98
```

Una vez más, nos da la respuesta correcta.

```
Introduzca los caracteres en el archivo "caracteres.txt"
Al terminar, guarde y pulse alguna letra 1
Hay un total de 1 Hexadecimale, 1 Octales, 2 Decimales
stefano@stefano-Desktop:~/Desktop/Cproyect$
```

Aclaración: Cada vez que ejecutemos nuevamente el programa, el archivo perderá todo su contenido.

Punto2:

Para ejecutar, la segunda parte del trabajo nos dirigimos al menu principal en introducimos el caracter **2** en la Terminal y luego pulsamos enter

```
 ____  _                                _     ________
/ __ )(_)__  ____ _  _____  ____  (_)___/ / ____ \
/ __  / / _ \/ __ \ | / / _ \/ __ \/ / __  / / __ `/
/ /_/ / /  __/ / / / |/ /  __/ / / / / /_/ / / /_/ /
/_____/_/\___/_/ /_/|___/\___/_/ /_/_/\__,_/\ \__,_/
                                             \____/
Menú
Introduzca alguna de las siguientes opciones:
        1- Autómata
        2- De Caracteres a Entero
        3- Realizar Cálculo
```

A continuación se despliega el mensaje, en el cuál debemos introducir los caracteres que deben convertirse a enteros, en este caso introduciremos el valor **54390.**

```
Introduzca los caracteres a convertir en enteros :
```

```
Introduzca los caracteres a convertir en enteros : 54390

El caracter en entero es 54390
stefano@stefano-Desktop:~/Desktop/Cproyect$
```

y así se convierte **texto** en **enteros**.

punto 3:

Para ejecutar, la última parte del trabajo nos dirigimos al menu principal en el cual introducimos el caracter **3** en la Terminal y luego pulsamos enter

```
stefano@stefano-Desktop:~/Desktop/Cproyect$ ./a.out
    ____  _                              _     ________
   / __ )(_)__  ____ _   _____  ____  (_)___/ / ____ \
  / __  / / _ \/ __ \ | / / _ \/ __ \/ / __  / / __ `/
 / /_/ / /  __/ / / / |/ /  __/ / / / / /_/ / / /_/ /
/_____/_/\___/_/ /_/|___/\___/_/ /_/_/\__,_/\ \__,_/
                                             \____/
Menú
Introduzca alguna de las siguientes opciones:
        1- Autómata
        2- De Caracteres a Entero
        3- Realizar Cálculo
3
```

Se nos desplegará una opción para introducir una operación que deseemos realizar, en este caso “123+98”.

```
Ingrese una expresión aritmética: 123+98
El resultado de la expresión es: 221
stefano@stefano-Desktop:~/Desktop/Cproyect$
```

Y nos retorna su respuesta.