MANUAL TÉCNICO

Proyecto 2 | Lenguajes Formales y de Programación

Estudiante: Carlos Alfredo López de León
Carnet: 3355697810901

Índice

Introducción

El presente manual muestra las herramientas y la lógica utilizada para desarrollar una solución de software para una empresa la cual se dedica requiere de un programa que digitalice información de una manera muy específica.

La aplicación tiene como objetivo procesar la información de un archivo de texto con extensión ".lfp" para generar un archivo ".csv" con los datos contenidos y especificaciones dadas. El análisis del archivo consiste en el análisis léxico del mismo, separando los tokens y mostrando los errores contenidos; sintáctico, mostrando si hay errores y digitalización y exportación de los datos a un archivo el cual pueda trabajarse en excel.

Objetivos

General

- Proporcionar al lector una explicación de la lógica, clases y atributos implementados en el desarrollo de la aplicación programada para facilitar el mantenimiento y futuras modificaciones realizadas por terceros.

Específicos

- Exponer una descripción del SO, IDE y otros elementos utilizados durante el desarrollo de la aplicación.
- Otorgar al lector una explicación técnica de los métodos y atributos que componen la parte operativa de la aplicación.
- Proporcionar el conocimiento necesario para el mantenimiento sencillo del software.

Entorno de desarrollo

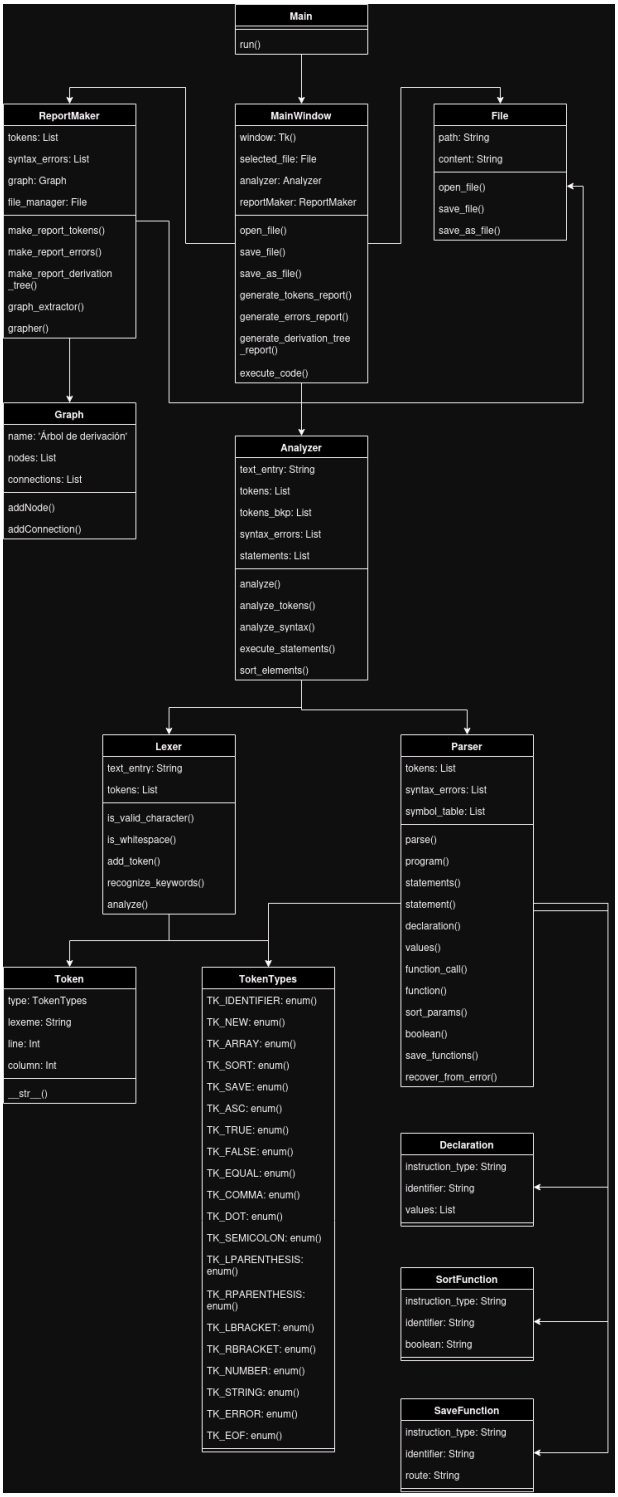
En el desarrollo de la aplicación fue utilizado el IDE PyCharm, debido a ser un entorno de trabajo completo y que nos proporciona una gran cantidad de funcionalidades, entre ellas la herramienta de Copilot, la cual agiliza el trabajo de desarrollo.

Lenguaje de programación:
Python 3.12.3

Sistema operativo:
Ubuntu 24.04 LTS

Arquitectura del proyecto

Diagrama de clases



Autómata finito determinista

######### LANGUAGE EXAMPLE #########

// Editor de código fuente

// Comentario de una línea

/*
Comentario
multilínea
*/

```
Array miArray = new Array[15, 80, 68, 55, 48];
miArray.sort(asc=FALSE);
miArray.save("ruta/del/archivo.csv");
```

######### TOKENS #########

Note: // and /* */ are comments, so they are not tokens and are not included in the list.

Identified tokens

```
NAME              | REGEX
------------------|--------------------------------
TK_IDENTIFIER     | [a-zA-Z][a-zA-Z0-9_]*
TK_NEW            | new
TK_ARRAY          | Array
TK_SORT           | sort
TK_SAVE           | save
TK_ASC            | asc
```

TK_TRUE	TRUE
TK_FALSE	FALSE
TK_EQUAL	=
TK_COMMA	,
TK_DOT	\.
TK_SEMICOLON	;
TK_LPARENTHESIS	\(
TK_RPARENTHESIS	\)
TK_LBRACKET	\[
TK_RBRACKET	\]
TK_NUMBER	[0-9]+(\.[0-9]+)?
TK_STRING	"[^"\n]*"

Reserved words are case sensitive and must be written correctly.
The regex that will define them will be [a-zA-Z][a-zA-Z0-9_]* and then classified according to the reserved word.
For symbols, the regex will be the symbol itself, but will be recognized as a TK_SYMBOL first, the classification will be done later.

Simplefied tokens

NAME	REGEX
TK_IDENTIFIER	[a-zA-Z][a-zA-Z0-9_]*
TK_NEW	
TK_ARRAY	
TK_SORT	
TK_SAVE	
TK_ASC	
TK_TRUE	
TK_FALSE	

```
TK_EQUAL          | =
TK_COMMA          | ,
TK_DOT            | \.
TK_SEMICOLON      | ;
TK_LPARENTHESIS   | \(
TK_RPARENTHESIS   | \)
TK_LBRACKET       | \[
TK_RBRACKET       | \]
------------------|--------------------------------
TK_NUMBER         | [0-9]+(\.[0-9]+)?
TK_STRING         | "[^"\n]*?"
```

REGEX for the production of the tokens

```
[a-zA-Z][a-zA-Z0-9_]*|=|,|\.|\(|\)|\[|\]|"[^"\n]*"|[0-9]+(\.[0-9]+)?
```

The process of the construction of the AFD is done at the technical manual.

Creación del autómata finito determinista

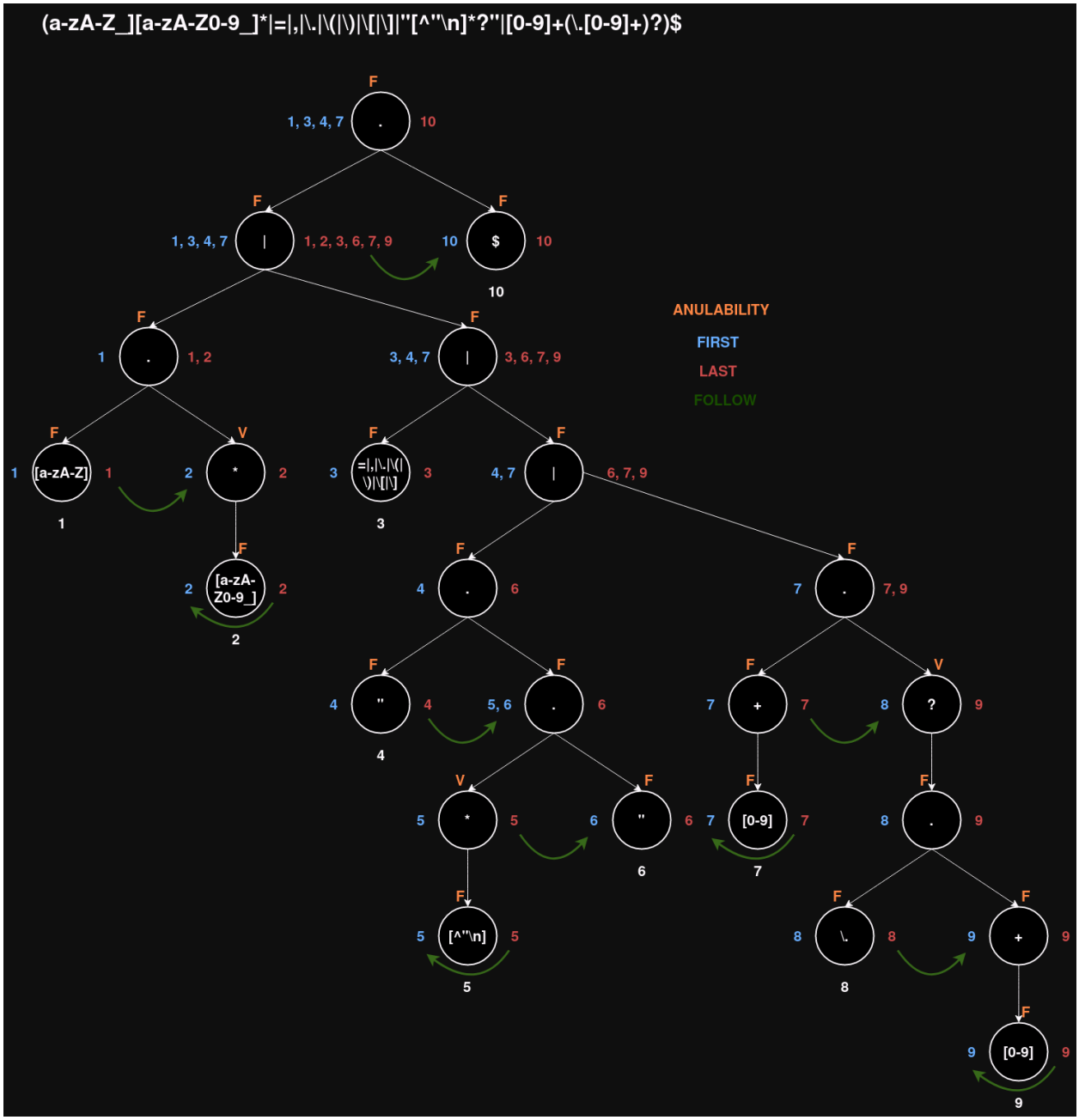

(a-zA-Z_][a-zA-Z0-9_]*|=|,|\.|\(|\)|\[|\]|"[^"\n]*?"|[0-9]+(\.[0-9]+)?)$
ANULABILITY
FIRST
LAST
FOLLOW
[a-zA-Z]
[a-zA-Z0-9_]
=|,|\.|\(|\)|\[|\]
[^"\n]
[0-9]
[0-9]

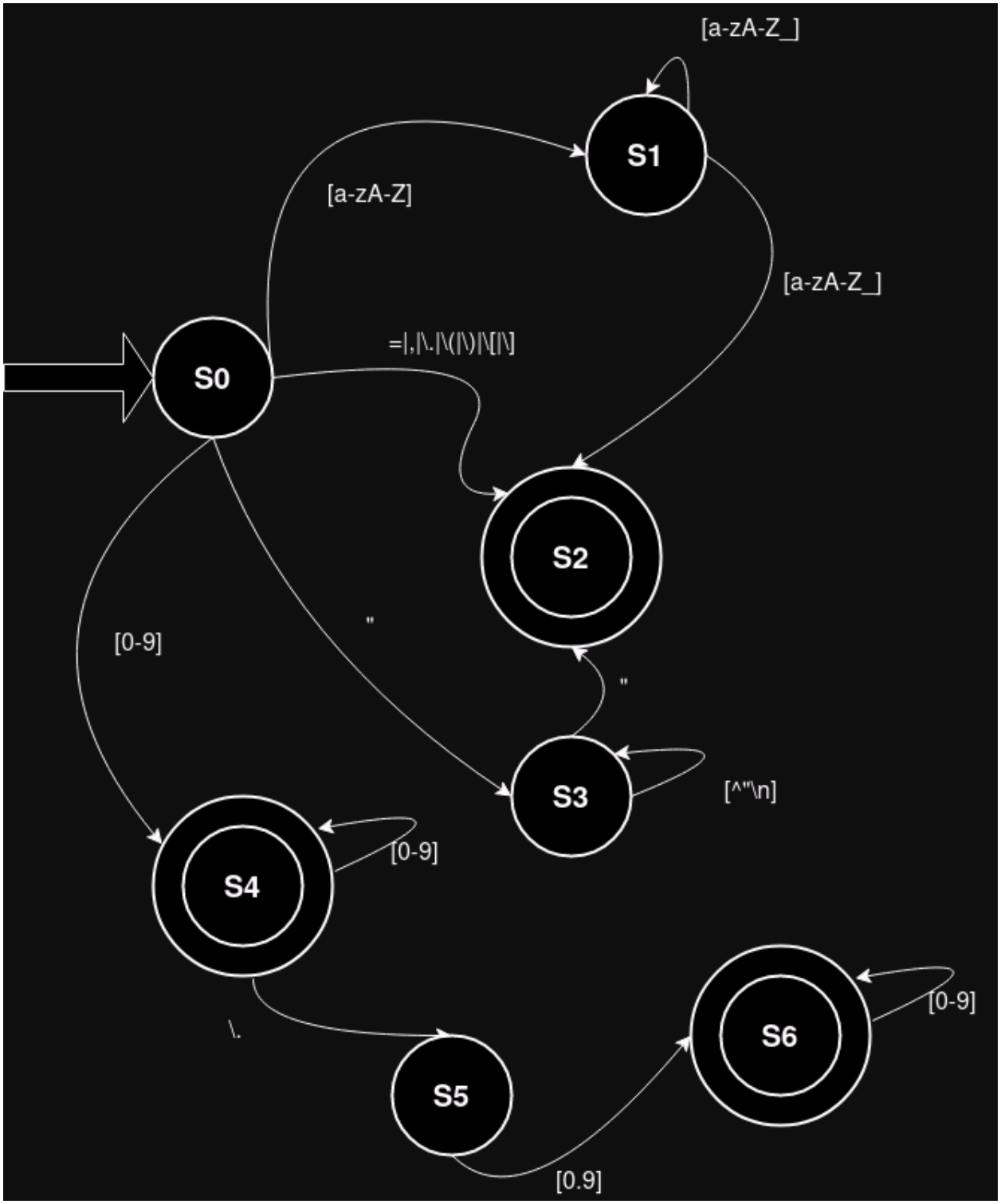
Tabla de follows

1	[a-zA-Z]	2, 10
2	[a-zA-Z_]	2, 10
3	=|,|\.|\(|\)|\[|\]	10
4	“	5, 6
5	[^"\n]	5, 6
6	“	10
7	[0-9]	7, 8, 10
8	\.	9
9	[0-9]	9, 10
10	$	-

Tabla de estados y transiciones

		[a-zA-Z]	[a-zA-Z_]	=|,|\.|\(|\)|\[|\]	“	[^"\n]	[0-9]	\.
S0	{1, 3, 4, 7}	S1		S2	S3		S4	
S1	{2, 10}		S1					
S2	{10}							
S3	{5, 6}				S2	S3		
S4	{7, 8, 10}						S4	S5
S5	{9}						S6	
S6	{9, 10}						S6	

Autómata Finito Determinista Resultante



Gramática

######### LANGUAGE EXAMPLE #########

// Editor de código fuente

// Comentario de una línea

/*
Comentario
multilínea
*/

```
Array miArray = new Array[15, 80, 68, 55, 48];
miArray.sort(asc=FALSE);
miArray.save("ruta/del/archivo.csv");
```

######### GRAMMAR #########

Terminal: TK_IDENTIFIER, TK_NEW, TK_ARRAY, TK_SORT, TK_SAVE, TK_ASC, TK_TRUE, TK_FALSE, TK_EQUAL, TK_COMMA, TK_DOT, TK_SEMICOLON, TK_LPARENTHESIS, TK_RPARENTHESIS, TK_LBRACKET, TK_RBRACKET, TK_NUMBER, TK_STRING
NonTerminal: <program>, <statements>, <statement>, <declaration>, <values>, <value>, <function_call>, <function>, <sort_function>, <save_function>, <sort_params>, <boolean>

Note: // and /* */ are comments, so they are not included in the grammar and are ignored by the parser and lexer.

Productions:

<program> ::= <statements>
<statements> ::= <statement> <statements>
| epsilon
<statement> ::= <declaration>
| <function_call>

<declaration> ::= TK_ARRAY TK_IDENTIFIER TK_EQUAL TK_NEW TK_ARRAY TK_LBRACKET <values> TK_RBRACKET TK_SEMICOLON
<values> ::= <value> TK_COMMA <values>
| <value>
<value> ::= TK_NUMBER
| TK_STRING

<function_call> ::= TK_IDENTIFIER TK_DOT <function> TK_SEMICOLON
<function> ::= <sort_function>
| <save_function>

<sort_function> ::= TK_SORT TK_LPARENTHESIS <sort_params> TK_RPARENTHESIS
<sort_params> ::= TK_ASC TK_EQUAL <boolean>
| epsilon
<boolean> ::= TK_TRUE
| TK_FALSE

<save_function> ::= TK_SAVE TK_LPARENTHESIS TK_STRING TK_RPARENTHESIS

Librerías utilizadas

Las librerías utilizadas en el proyecto son las siguientes:

- tkinter | Para la creación de la interfaz gráfica
- graphviz | Para el manejo de grafos

Descripción de la solución

Para poder desarrollar este proyecto, se analizó lo solicitado por el cliente, sus restricciones, ambiente y forma de trabajo.

Entre estas, las principales son:

- El usuario carga un archivo de entrada, con extensión ".lfp", donde se le permitirá exportar a un editor de texto donde podrá modificar y visualizar de una manera intuitiva el texto dentro del mismo. El programa le permitirá guardar, sobreescribiendo el archivo, o guardar en un nuevo archivo los cambios realizados, manteniendo el formato ".lfp".
- Al ingresar un archivo, este debe ser exclusivamente con extensión ".lfp", donde se analizará el texto con un lexer y un parser, donde si no hay errores se procederá a la digitalización de información.
- En caso de que un archivo contenga errores, entonces se le notificará al usuario y no se ejecutará la digitalización.
- Al finalizar el análisis, se crearán los reportes de los tokens analizados, errores encontrados, siendo estos léxicos y/o sintácticos y el árbol de derivación. Estos en archivos diferentes de tipo "html", podrán ser almacenados en una ruta asignada por el usuario.
- La información será digitalizada en un archivo ".csv", el cual podrá trabajarse posteriormente en excel.

Lógica de la solución

La lógica utilizada para el desarrollo de la aplicación es la siguiente:

Al ejecutar el archivo, este es analizado por el AFD con el fin de identificar

1. Tokens
2. Errores léxicos

Luego, esta lista de tokens es enviada al parser, el cuál tiene el objetivo de analizar sintácticamente los tokens, quiere decir que este se encargará de analizar el orden en que los tokens son recibidos y que estos sigan una gramática establecida, identificando:

1. Gramáticas o instrucciones correctas
2. Errores sintácticos

Solo los archivos que no contengan errores podrán ser ejecutados, donde se hará el trabajo de digitalizar la información que fue ingresada, siendo esta declarando información, ordenando información y guardando información en un archivo ".csv".

Los reportes serán generados en archivos de tipo "html", donde uno contendrá los tokens válidos y otro contendrá los errores léxicos y sintácticos, y para el reporte de árbol de derivación se generará un archivo tipo ".svg" para su visualización.

Al cargar un nuevo archivo, cada opción se limpia y queda con la nueva información ingresada.