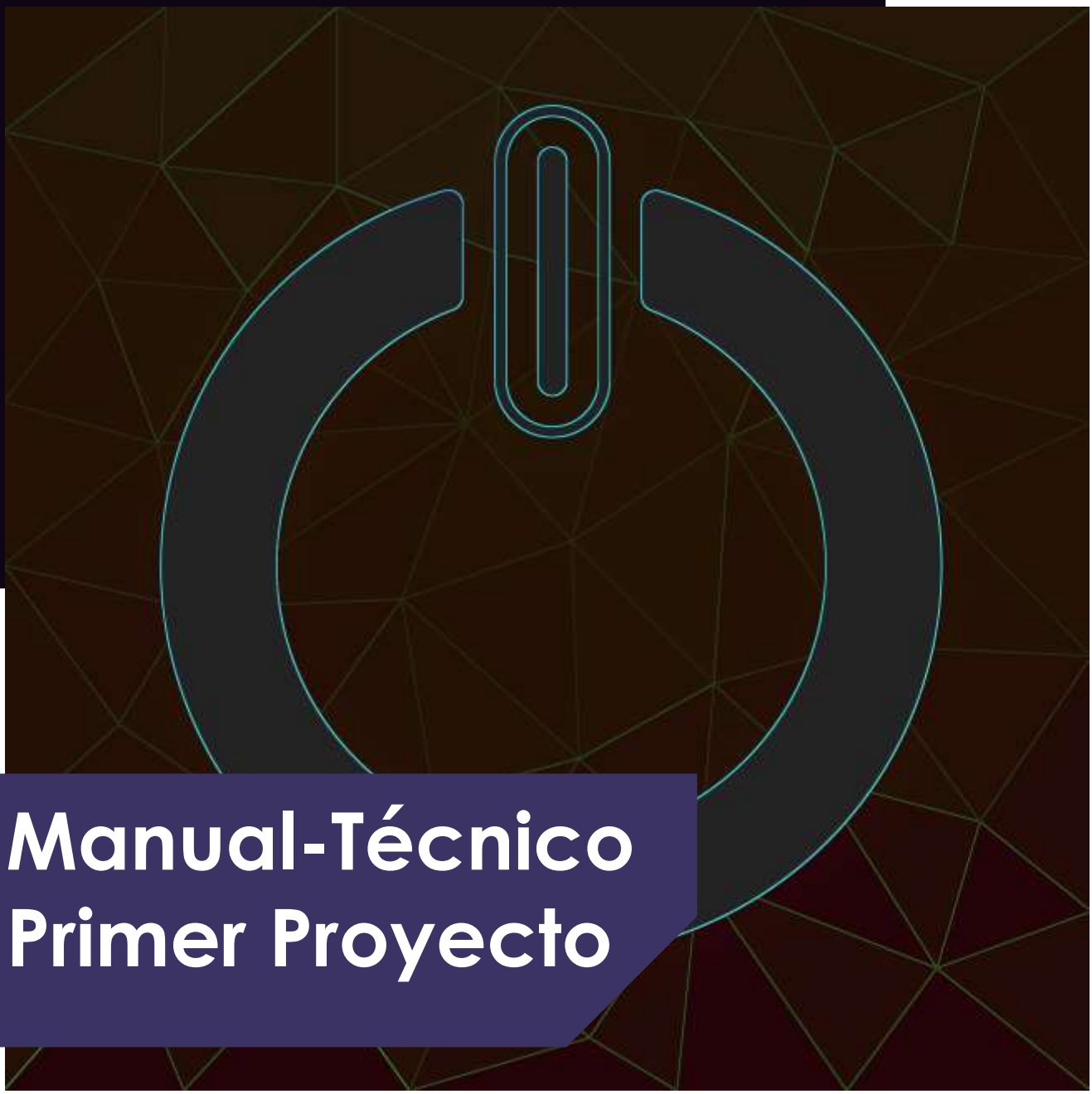


Manual-Técnico
Primer Proyecto

Juan Pablo García Monzón

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Organización de Lenguajes y Compiladores 1

Agosto 2020

TABLA DE CONTENIDO

DESCRIPCIÓN

Proyecto

La idea principal del proyecto es que se genere un análisis léxico de cada uno de los distintos archivos de entrada los cuales pueden ser de tipo Marcado como en el caso de HTML, de tipo diseño como CSS o de tipo interpretado como JavaScript.

Se debe realizar un analizador léxico por lenguaje ya que en cada uno de ellos existen distintos tipos de caracteres reconocidos. Y como salida se espera la construcción del proyecto en directorios los cuales estarán identificados por medio de paths las cuales serán tomadas y almacenadas por cada archivo dentro de la ejecución del programa, reportando todos los errores léxicos encontrados.

Cabe mencionar que al momento de reconocer un error léxico el análisis no debe detenerse por ningún motivo, sino que debe tener la capacidad de recuperarse y seguir analizando.

Requerimientos mínimos

- ❖ Windows 7 / / Linux/UNIX (Cualquier Versión)
- ❖ 30MB de espacio en el disco duro
- ❖ 1GB de RAM
- ❖ Python 3
- ❖ Graphviz
- ❖ Tkinter (Librería Python)

Flujo del proyecto

1. Se abre el archivo ejecutable (OLC1_Proyecto1_201222615.exe)
2. En el menú “Archivo” se elige la opción “Abrir”
3. Se busca y se abre un archivo con extensión “.js”, “.css”, “.html” o “.rmt”
4. En el menú “Ejecutar” se elige la opción “Ejecutar”
5. Se hace un reporte de los tokens y errores en la consola
6. Si dentro del archivo de entrada hay un comentario especial con una ruta “PATHW:” o “PATHL” se creara un archivo en esa ruta ya sin los errores léxicos.

(POR FAVOR SIEMPRE CORRER LA APLICACIÓN O EL EJECUTABLE COMO ADMINISTRADOR)

7. En el menú “Reportes” se puede ver el “Reporte de Errores Léxicos”, “Reporte de Errores Sintácticos”, “Reporte de Árbol” y “Reporte de Estados”.
8. En el menú “Ayuda” se puede visualizar el “Manual Técnico” o el “Manual de Usuario” eligiendo las opciones “Manual Técnico” o “Manual de Usuario”
9. Si ya no se desea analizar otro archivo de entrada se puede cerrar la aplicación dándole clic a el botón “X” o en el menú “Archivo” la opción “Salir”

CLASES

Interfaz

Clase principal que debe de compilarse para que todo el proyecto pueda correr, en ella se encuentra todo el desarrollo de la interfaz gráfica del proyecto. En este se encuentran varios métodos y llamadas a otras clases para poder enviar y recibir datos para procesarlos para reportes o presentación en la consola.

CustomText

Esta clase y sus métodos son utilizados para el enumerado de línea del área de texto, agrega los eventos necesarios, para que el numero aparezca cada vez que el texto cambia.

TextLineNumbers

Aquí se encuentran los métodos para obtener el numero de línea actual y agregarlo a la interfaz de usuario

lex_JS

Clase donde se encuentra todo el desarrollo del analizador léxico del lenguaje JavaScript (JS) en ella se encuentra toda una lógica de estados para poder reconocer varios caracteres del lenguaje y producir una lista de tokens.

lex_CSS

Clase donde se encuentra todo el desarrollo del analizador léxico del lenguaje CSS (CSS) en ella se encuentra toda una lógica de estados para poder reconocer varios caracteres del lenguaje y producir una lista de tokens.

lex_HTML

Clase donde se encuentra todo el desarrollo del analizador léxico del lenguaje HTML (HTML) en ella se encuentra toda una lógica de estados para poder reconocer varios caracteres del lenguaje y producir una lista de tokens.

lex_RMT

Clase donde se encuentra todo el desarrollo del analizador léxico del lenguaje RMT en ella se encuentra toda una lógica de estados para poder reconocer varios caracteres del lenguaje y producir una lista de tokens. Este es el único lenguaje que luego tendrá un análisis sintáctico.

syn_RMT

Analizador sintáctico a partir de la lista de tokens del lenguaje RMT, en este se encuentra una lógica a partir de una gramática donde cada parte de la gramática (Terminales, No Terminales y Producciones) son estados. Este método para crear un análisis sintáctico es comúnmente llamado “Parea”.

error_Report

Clase donde se encuentra el machote para realizar un reporte de una tabla HTML a partir de los errores léxicos producidos en el análisis de uno de los lenguajes (JS, CSS, HTML o RMT)

css_Report

Clase donde se encuentra el machote para realizar un reporte de una tabla HTML a partir de los estados que se van transitando durante el análisis del lenguaje CSS.

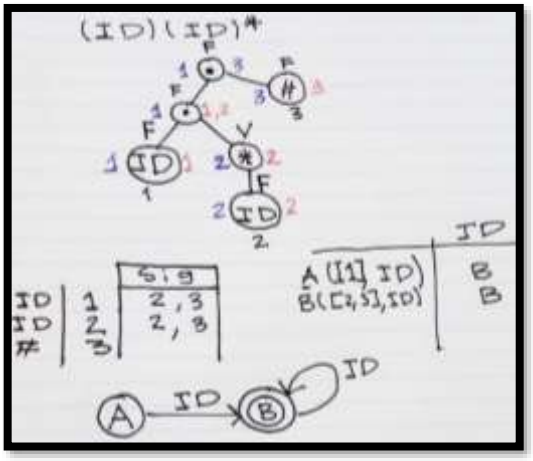
Grafos

Clase donde se encuentra los comandos para graficar con la herramienta “graphviz” los diagramas de estados que se usaron en “JS” específicamente “ID”, “String” y “Comentario Unilinea”.

EXPRESIONES REGULARES UTILIZADAS MAS IMPORTANTES EN EL ANÁLISIS LÉXICO DE LOS 4 LENGUAJES

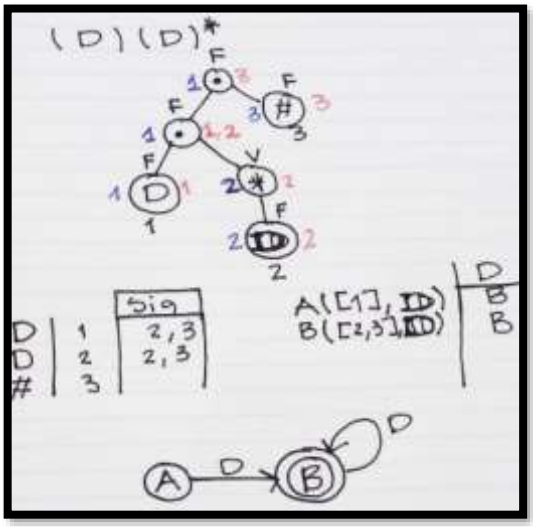
Identificador

ID(ID)*



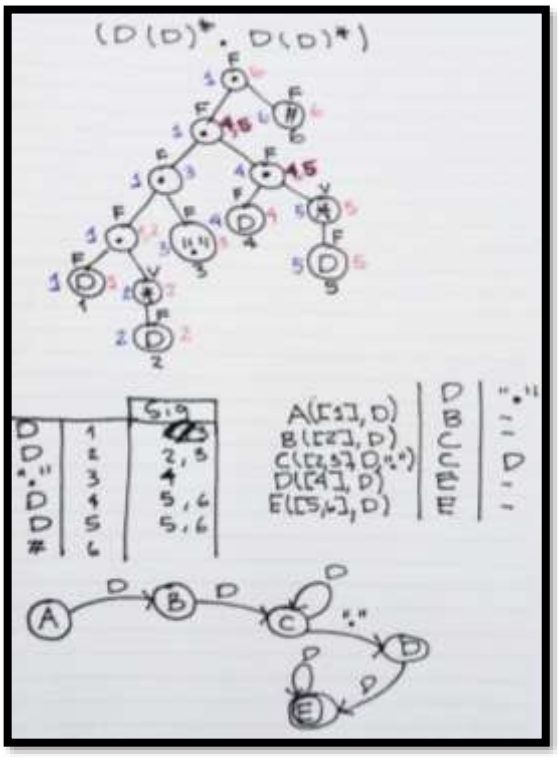
Entero

D(D)*



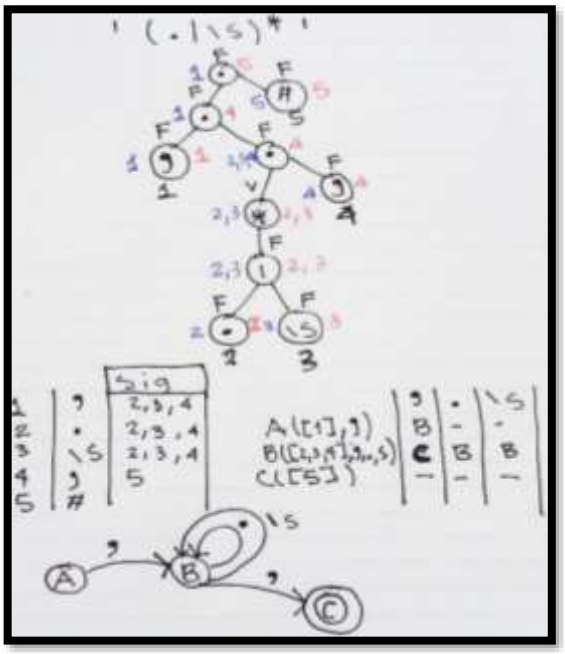
Decimal

D(D)* . D(D)*



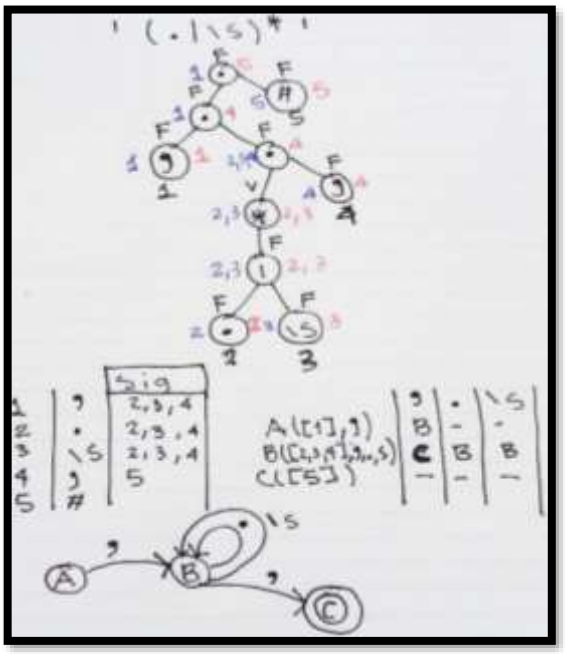
Cadena Simple

‘(. |\S) * ‘



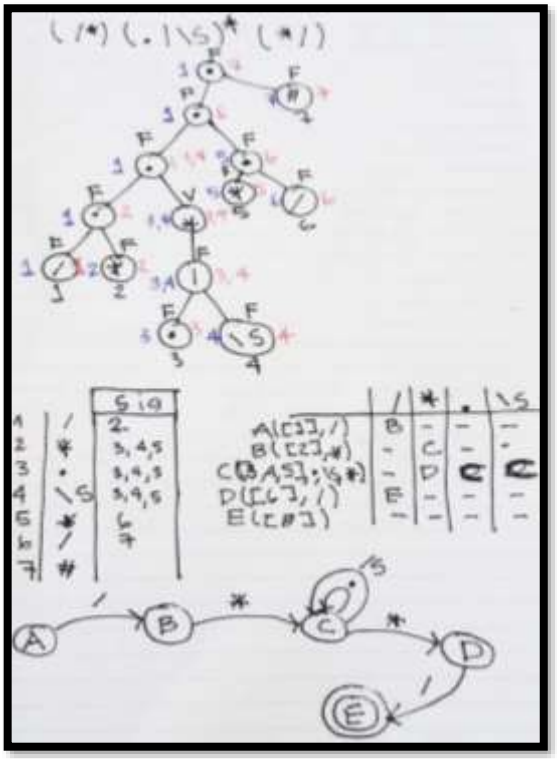
Cadena Doble

“(. |\S) * “



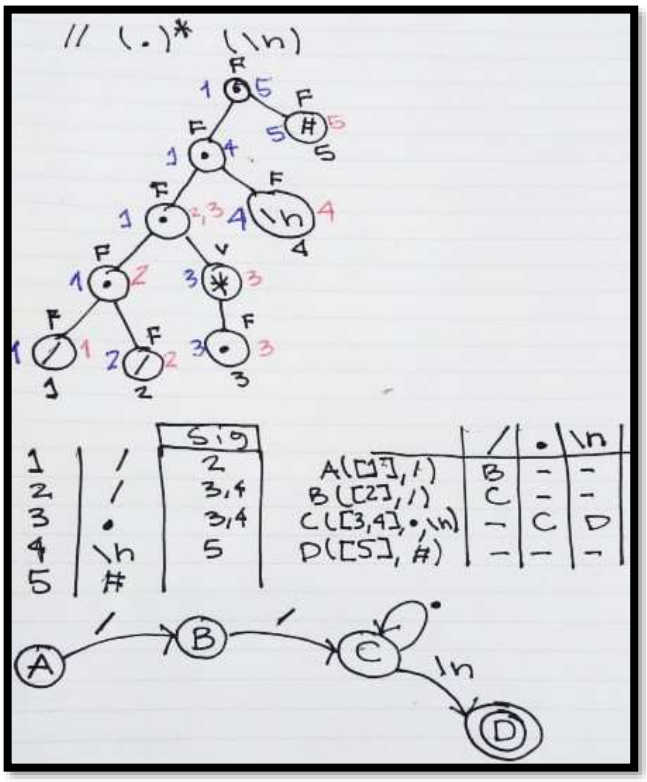
Comentarios Multilínea

(/*) (. |\S) * (*/)



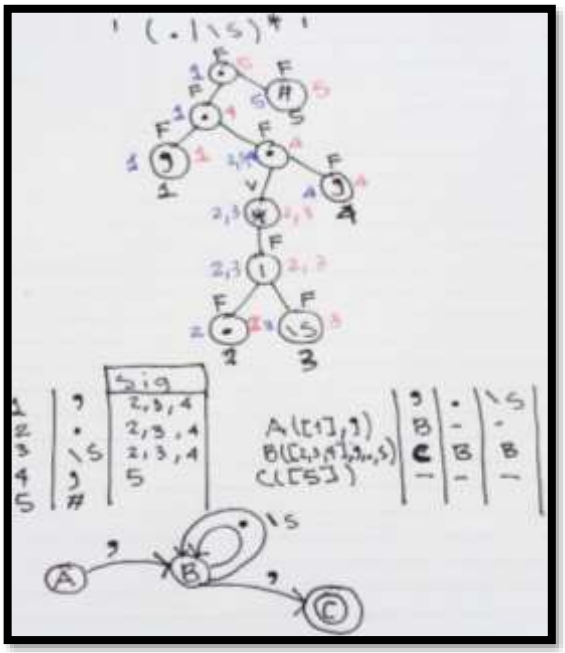
Comentario Unilinea

// (.)*(\n)



Tags

(>) (. |\S) * (<)



GRAMÁTICA

```
S->E R
R-> E R
    |EPSILON
E-> T EP
EP-> + T EP
   | - T EP
   | EPSILON
T->F TP
TP-> * F TP
   | / F TP
   | EPSILON
F->  (E)
   | NUMERO
   | ID
```