EXREGAN

MANUAL DE USUARIO

Proyecto 1 del curso:

"Organización de lenguajes y compiladores 1"

Sección:

"C"

Catedrático:

Kevin Lajpop

Auxiliar:

Maynor Octavio Piló Tuy

Nombre del estudiante:

Josue Alejandro Perez Benito

Carne del estudiante:

201712602

Introducción

En el presente manual se indicará los pasos a seguir para la comprensión de las funcionalidades del programa EXREGAN; dichas funcionalidades constan del análisis de sistema de conjuntos-expresiones regulares-cadenas de prueba.

Dicho programa tiene como objetivo analizar e interpretar gráficamente la información a través del reconocimiento de patrones de tokens comenzando por el análisis de cada lexema de la cadena de entrada seguido de la revisión de patrones de tokens en el analizador sintáctico.

El programa se creo con el fin de facilitar la comprensión de las primeras fases del compilador.

Objetivos

Objetivo general

Desarrollar un programa capaz de implementar un analizador léxico y sintáctico para el análisis de cadenas de caracteres.

Objetivos específicos

- Implementar el método del árbol para el análisis de soluciones con el fin de poder desarrollar un autómata finito determinista.
- Implementar el análisis del método de Thompson para desarrollar autómatas finitos no deterministas.
- Trazar soluciones por medio del autómata finito determinista.

Requerimientos mínimos del sistema

Windows

Windows 10 (8u51 y superiores)

Windows 8.x (escritorio)

Windows 7 SP1

Windows Vista SP2

Windows Server 2008 R2 SP1 (64 bits)

Windows Server 2012 y 2012 R2 (64 bits)

RAM: 128 MB

Espacio en disco: 124 MB para JRE; 2 MB para Java Update

Procesador: Mínimo Pentium 2 a 266 MHz

Exploradores: Internet Explorer 9 y superior, Firefox

Mac OS X

Mac con Intel que ejecuta Mac OS X 10.8.3+, 10.9+

Privilegios de administrador para la instalación

Explorador de 64 bits

Se requiere un explorador de 64 bits (Safari, por ejemplo) para ejecutar Oracle Java en Mac.

Linux

Oracle Linux 5.5+1

Oracle Linux 6.x (32 bits), 6.x (64 bits)2

Oracle Linux 7.x (64 bits)2 (8u20 y superiores)

Red Hat Enterprise Linux 5.5+1 6.x (32 bits), 6.x (64 bits)2

Red Hat Enterprise Linux 7.x (64 bits)2 (8u20 y superiores)

Suse Linux Enterprise Server 10 SP2+, 11.x

Suse Linux Enterprise Server 12.x (64 bits)2 (8u31 y superiores)

Ubuntu Linux 12.04 LTS, 13.x

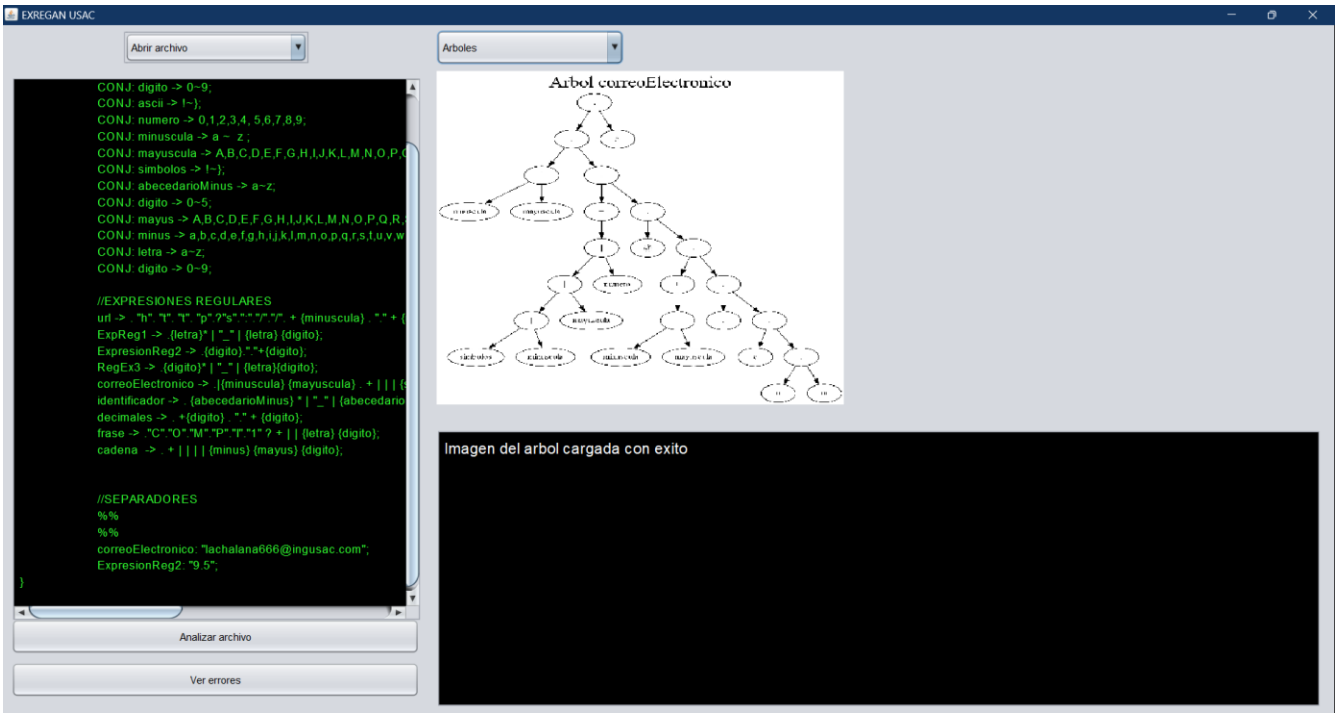
Ubuntu Linux 14.x (8u25 y superiores)

Ubuntu Linux 15.04 (8u45 y superiores)

Ubuntu Linux 15.10 (8u65 y superiores)

Exploradores: Firefox

Interfaz grafica



La presente interfaz consta con 2 **combo-box**:

- **La primera** para "*cargar, crear, actualizar o guardar*" el archivo;
- **La segunda** para "*seleccionar que reporte visualizar*".

Consta de 2 botones:

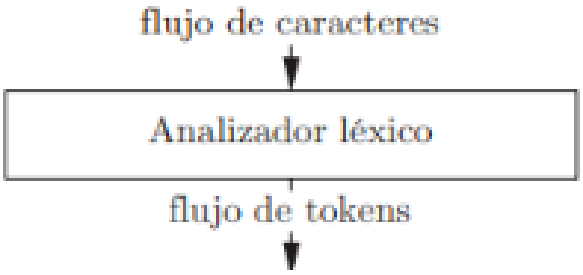
1. **Analizar archivo**: buscara los conjuntos, expresiones regulares y cadenas de entrada presentes en el archivo, además de los errores léxicos y sintácticos.
2. **Ver errores**: mostrara los errores léxicos y sintácticos encontrados en el archivo a evaluar.

Consta de 2 paneles:

1. Visualizador de archivos
2. Terminal

Analizador léxico

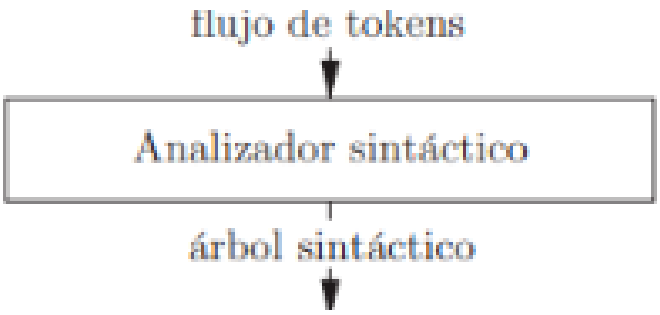
Dicho analizador se encargará de verificar que todas las entradas sean con caracteres validos para que el programa pueda entender sin dificultad las operaciones solicitadas.



Como se muestra en la imagen el analizador léxico retornara un flujo de tokens para que el analizador sintáctico pueda entrar en acción.

Analizador sintáctico

Dicho analizador se encargará de analizar que el flujo de tokens correspondiente cumpla con todas las leyes impuestas por la gramática definida.

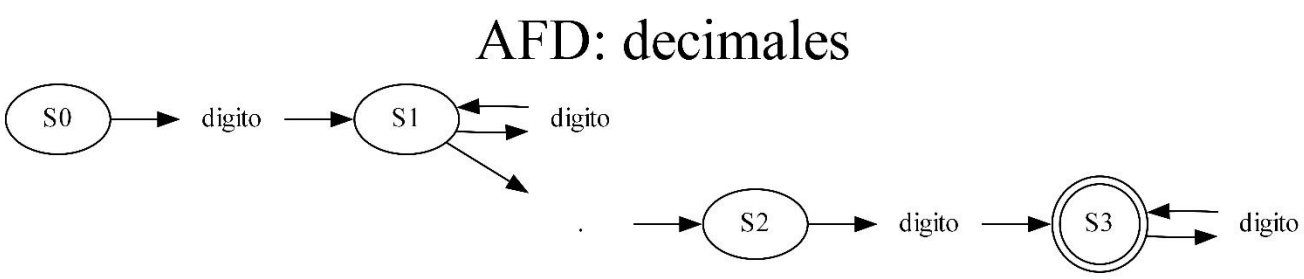


Como se muestra en la imagen retornará un árbol sintáctico que permitirá el cumplimiento de su siguiente fase.

Reportes

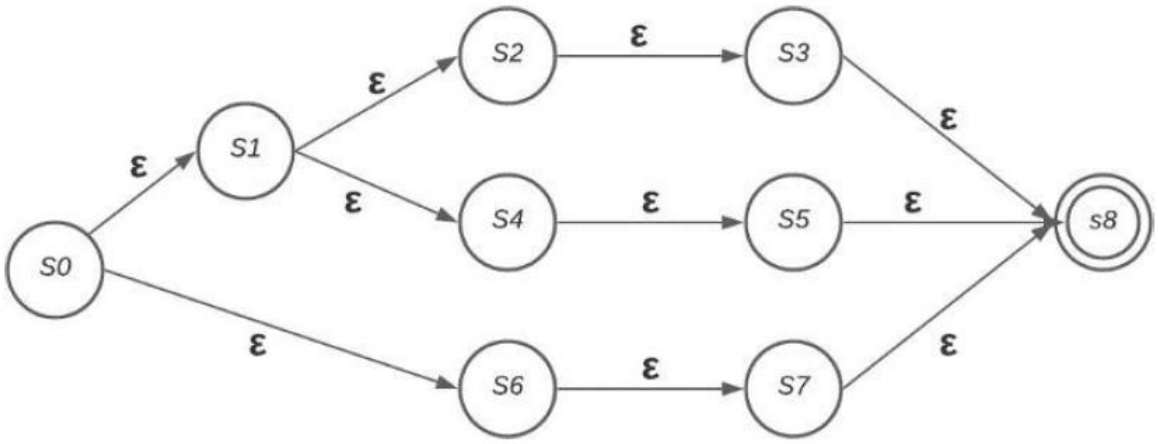
AFD

Los autómatas finitos deterministas no sufren de ambigüedad ni transiciones con épsilon.



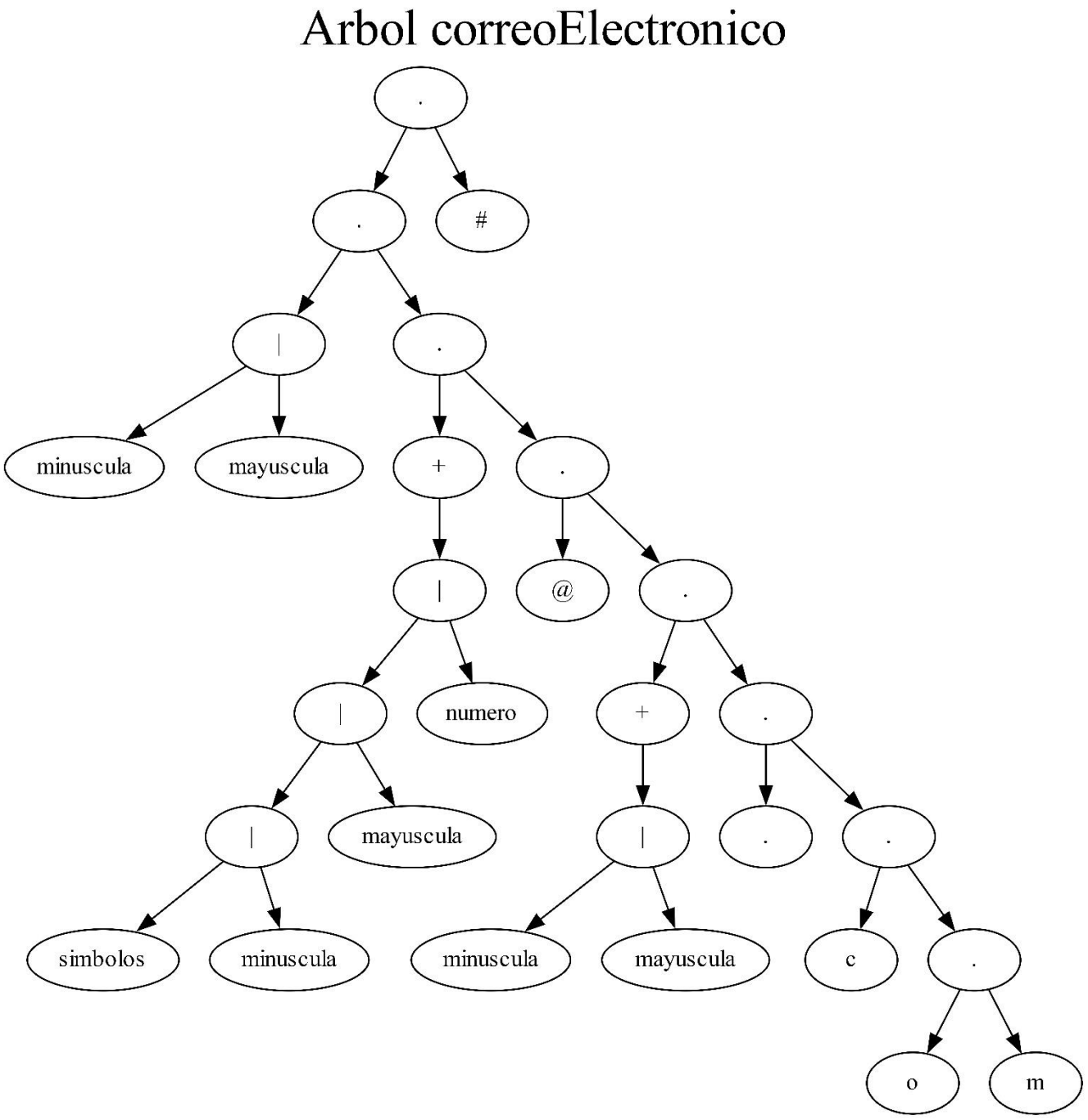
AFND

A diferencia de los AFD los AFND si presentan ambigüedad y transiciones con épsilon para el desarrollo de soluciones.



Método del árbol

El método del árbol es esencial para el desarrollo de la fase de análisis semántico, este provee una vista de las rutas para llegar a la solución de una cadena de entrada.



Errores

El presente programa mostrara los errores encontrados en 2 archivos HTML, uno para los errores léxicos y otro para los errores sintácticos.

LISTADO DE ERRORES LEXICOS ENCONTRADOS EN LA COMPILACIÓN

- Error léxico: / Linea: 18 Columna: 40
- Error léxico: / Linea: 18 Columna: 44

LISTADO DE ERRORES SINTACTICOS ENCONTRADOS EN LA COMPILACIÓN

- **Error sintáctico recuperable: . Linea: 18 Columna: 42**
- No se logro leer la expresion: frase
- No se logro leer la expresion: cadena

Salidas

EXREGAN al finalizar el análisis del archivo mostrara el resultado de las cadenas por medio de un archivo con extensión Json.

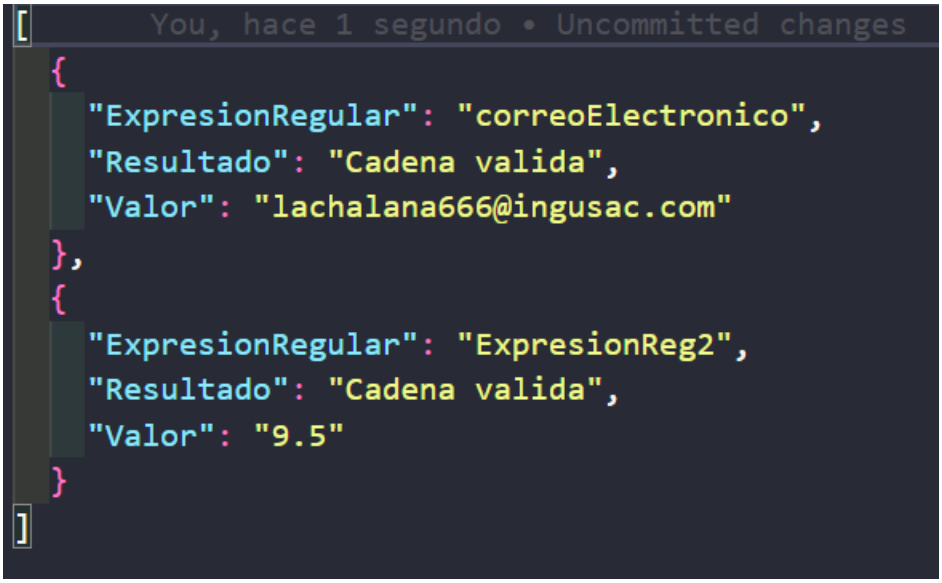


Tabla de siguientes

A partir del árbol sintáctico generado se obtiene la tabla de siguientes.

Hoja		Siguientes
a	1	2,3,4
a	2	2,3,4
b	3	2,3,4
b	4	5
#	5	--

Tabla de transiciones

A partir del árbol sintáctico y la tabla de siguientes logramos formar la tabla de transiciones.

Estado	Terminales	
	a	b
S0 {1}	S1	--
S1 {2,3,4}	S1	S2
S2 {2,3,4,5}	S1	S2

Sintaxis para el correcto funcionamiento

Definición de conjuntos

Para la definición de conjuntos deberás seguir la siguiente estructura:

CONJ: IDENTIFICADOR -> 1,2,3,4,5,6,7;

CONJ: IDENTIFICADOR -> A~Z;

CONJ: la palabra reservada CONJ será la pauta para indicar al programa que se está por declarar un conjunto, siempre debe ir seguido por ":" los dos puntos.

IDENTIFICADOR: después debemos ingresar un identificador único para llamar a nuestro conjunto.

->: EL presente símbolo representa la asignación de los elementos que tendrá nuestro conjunto.

1,2,3,4,5,6,7 ó **A~Z**: para definir los elementos que conformaran nuestro conjunto tenemos 2 opciones:

1. Definir nuestros elementos por medio de comas.
2. Definir un grupo especifico de elementos ayudándonos con el símbolo "~".

Siempre debemos terminar con el punto y coma "**;**".

Definición de expresiones regulares

Para establecer nuestras expresiones regulares usaremos la siguiente simbología:

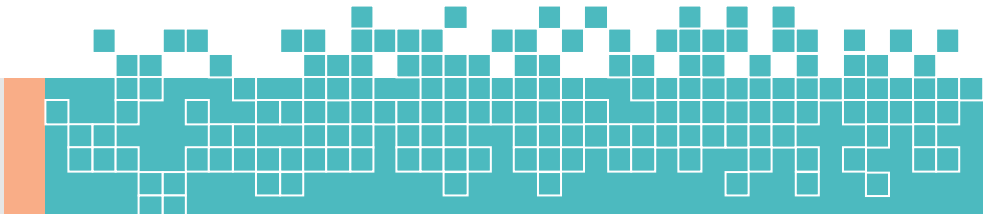
- . El punto representara la concatenación de elementos
- | el presente símbolo representara el OR o un elemento u otro.
- + el presente símbolo representara que el elemento puede venir 1 o más veces.
- * el presente símbolo representara que el elemento puede venir 0 o más veces.
- ? el presente símbolo representara que el elemento puede venir o no.
- “ ” las comillas adjuntaran cualquier elemento a nuestra expresión, dentro de ellas debe ir el elemento que se desea incluir.
- { } las llaves nos servirán para llamar a nuestros conjuntos, el identificador del conjunto debe ir dentro de ellas.

IDENTIFICADOR -> EXPRESION;

IDENTIFICADOR: Debemos ingresar un identificador único para llamar a nuestra expresión.

->: EL presente símbolo representa la asignación de los elementos que tendrá nuestro conjunto.

EXPRESION: la expresión deberá finalizarse con el punto y coma para ser valida “**;**”.



Definición de cadenas de entrada

IDENTIFICADOR: “CADENA”;

IDENTIFICADOR: Se deberá llamar al identificador de la expresion regular a valuar, después deberá añadirse los dos puntos “**:**”.

CADENA: Se deberá escribir la cadena a valuar dentro de comillas dobles **“”** y terminar con el punto y coma “**;**”.