

Manual técnico

REQUERIMIENTOS TÉCNICOS:

1. REQUERIMIENTOS MÍNIMOS DE HARDWARE

- Procesador: Intel (en cualquiera de sus procesadores), AMD (Cualquiera de sus procesadores).
- Memoria RAM: Mínimo: 1 Gigabytes (GB)
- Disco Duro: 500Gb.

2. REQUERIMIENTOS MÍNIMOS DE SOFTWARE

- Privilegios de administrador
- Sistema Operativo: Mínimo: Windows vista, 7, 8,10, en cualquiera de sus versiones.

HERRAMIENTAS UTILIZADAS PARA SU DESARROLLO:

Visual Estudio (2017):

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc.

Visual Studio permite a los desarrolladores crear aplicaciones de escritorio, aplicaciones de sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y consolas, entre otros.

DESARROLLO:

Creación del Autómata Finito Determinista:

Método del Árbol:

Es un método que se utiliza para crear un AFD (Autómata Finito Determinista) mínimo. La técnica es una alternativa efectiva al método de Thomson ya que el método de Thomson no brinda el AFD mínimo.

- Partiendo de una expresión regular
- Permite la reducción de estados repetitivos.
- Produce autómatas más eficientes
- Permite la eliminación de nulos Creando un DFA óptimo.

Ya que la aplicación consiste en gran parte en un analizador léxico, para analizar el texto del código que se ingresa en un editor de texto se necesita definir un conjunto de símbolos o caracteres validos del lenguaje establecido:

Para el desarrollo esta aplicación será válidos estos caracteres:

L=letras, D=dígitos, S= Símbolos, **así como ‘:’, ‘=’, ‘“’, ‘_’ y #.**

T= cualquier tipo de carácter menos ‘“’

L= {A-Z a-z}

D= {0-9}

S= {‘;’, ‘{’, ‘}’, ‘,’}

Así mismo es una secuencia de caracteres que forma un patrón de búsqueda, principalmente utilizada para la búsqueda de patrones de cadenas de caracteres u operaciones de sustituciones. Por ejemplo, el grupo formado por las cadenas *Handel*, *Händel* y *Haendel* se describe con el patrón "H(a|ä|ae)ndel". Una expresión regular es una forma de representar los lenguajes regulares (finitos o infinitos) y se construye utilizando caracteres del alfabeto, llamados Expresiones Regulares.

Toda expresión regular tiene algún autómata finito asociado. Específicamente, las expresiones regulares se construyen utilizando los operadores:

Alternación '|':

Una barra vertical separa las alternativas. Por ejemplo, "marrón|castaño" se corresponde con *marrón* o *castaño*.

Cuantificación:

Un cuantificador tras un carácter especifica la frecuencia con la que éste puede ocurrir. Los cuantificadores más comunes son "?", "+" y "*":

?:

El signo de interrogación indica que el carácter que le precede puede aparecer como mucho una vez. Por ejemplo, "ob?scuro" se corresponde con *oscuro* y *obscur*.

+:

El signo más indica que el carácter que le precede debe aparecer al menos una vez. Por ejemplo, "ho+la" describe el conjunto infinito *hola*, *hoola*, *hoola*, *hoola*, etcétera.

*:

El asterisco indica que el carácter que le precede puede aparecer cero, una, o más veces. Por ejemplo, "0*42" se corresponde con *42*, *042*, *0042*, *00042*, etcétera.

Agrupación '()':

Los paréntesis pueden usarse para definir el ámbito y precedencia de los demás operadores. Por ejemplo, "(p|m)adre" es lo mismo que "padre|madre", y "(des)?amor" se corresponde con *amor* y con *desamor*.

Para el desarrollo del método la expresión regular a usar será:

Expression: (L(L|_|D) *| D+ | S | :(: =)? | "T*")

Contrición del Árbol:

Paso 1:

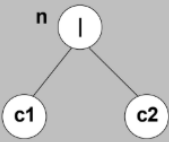
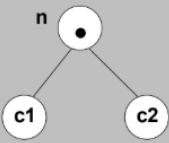

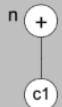

Aumentar la ER con #

ER: $(L(L|_|D)^* | D^+ | S | :(:=)? | "T^*") \#$

Paso 2:

Crear el árbol asociado a la ER y enumerar las hojas

Donde todas las ultimas hojas del árbol son no Anulables F

| Nodo n | Anulable(n) | Primerapos(n) | Últimapos(n) |
|---|----------------------------|---|---|
| n es una hoja etiquetada con e | True | vacío | Vacío |
| n es una hoja etiquetada con la posición i | False | {i} | {i} |
|  | $A(c1) \text{ or } A(c2)$ | $PP(c1) \cup PP(c2)$ | $UP(c1) \cup UP(c2)$ |
|  | $A(c1) \text{ and } A(c2)$ | if $A(c1)$ then $PP(c1) \cup PP(c2)$ else $PP(c1)$ | if $A(c2)$ then $UP(c1) \cup UP(c2)$ Else $UP(c2)$ |
|  | True | $PP(c1)$ | $UP(c1)$ |
|  | $A(c1)$ | $PP(c1)$ | $UP(c1)$ |
|  | True | $PP(c1)$ | $UP(c1)$ |

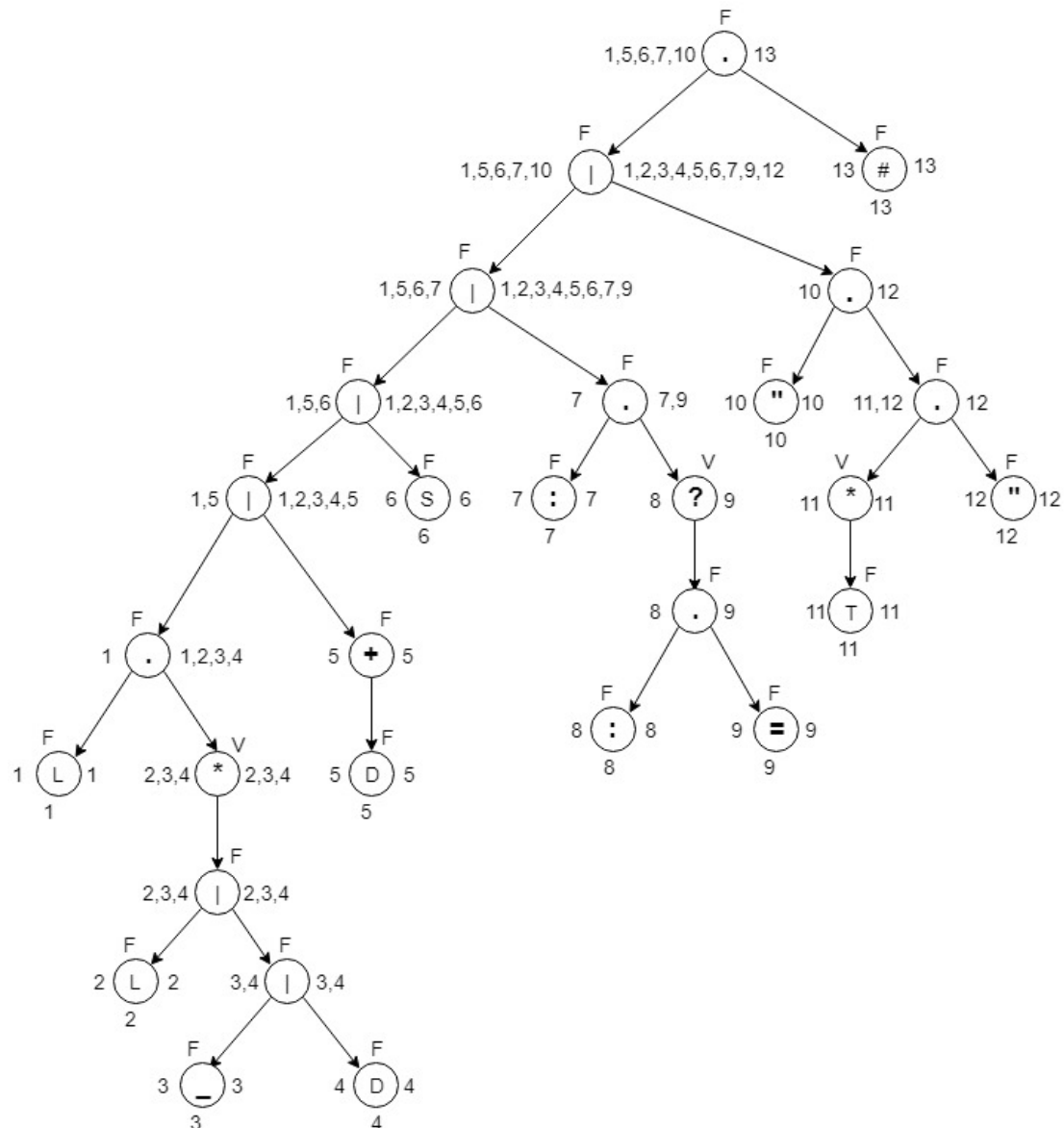
Calcular Anulable, first, last, follow (primeros, últimos, siguientes):

SiguientePos(l): esta función indica qué posiciones pueden seguir a la posición l en el árbol sintáctico.

Las reglas son:

1. Si n es un nodo de concatenación (.) con hijos izq c_1 y der c_2 , e i es una posición dentro de últimapos de c_1 , entonces todas las posiciones de primerapos de c_2 están en el siguientepos de i .

2. Si n es un nodo asterisco(*) ó un nodo más (+), e l es una posición dentro de última pos de n , entonces todas las posiciones de primerapos de n están en siguientespos de l .



| I | Siguiente |
|-----------|-----------|
| 1 - L | 2,3,4,13 |
| 2 - L | 2,3,4,13 |
| 3 - ' _ ' | 2,3,4,13 |
| 4 - D | 2,3,4,13 |
| 5 - D | 5,13 |
| 6 - S | 13 |
| 7 - : | 8,13 |
| 8 - : | 9 |
| 9 - = | 13 |
| 10 - " | 11,12 |
| 11 - T | 11,12 |
| 12 - " | 13 |
| 13 - # | _____ |

Paso 4:

Construir los subconjuntos

So: {1,5,6,7,10} donde tu estado inicial siempre será c1 del tu nodo raíz,

S1: {2,3,4,13}

S2: {5,13}

S3: {8,13}

S4: {11,12}

S5: {9}

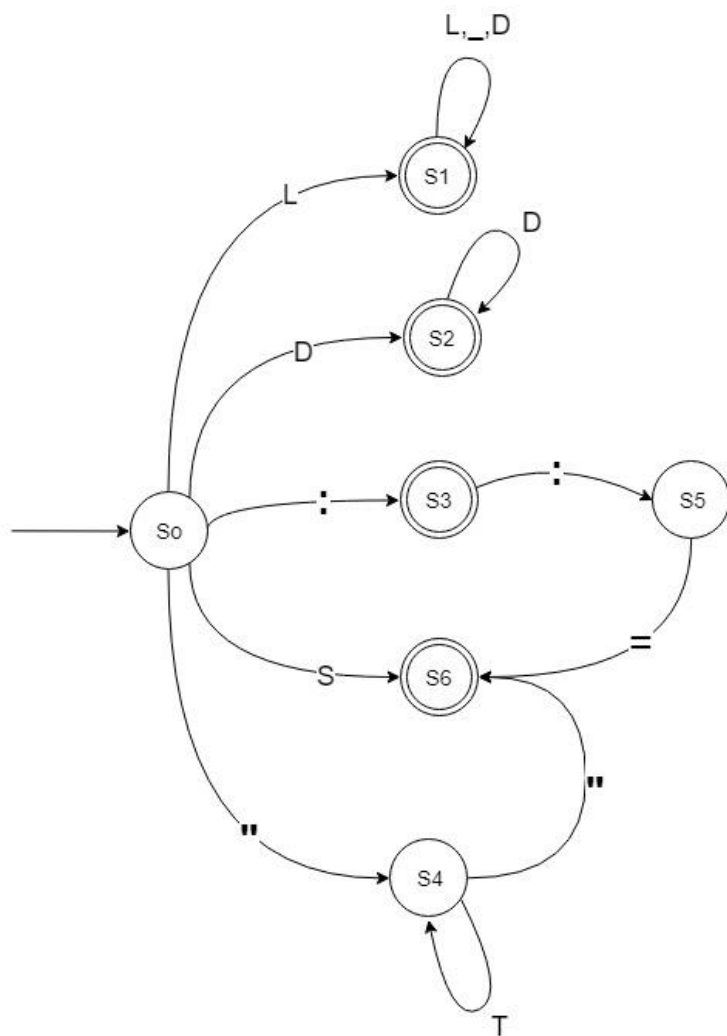
S6: {13}

Paso 5:

Construir la tabla de transiciones

| | L | _ | D | S | : | = | " | T |
|----|----|----|----|----|----|----|----|----|
| So | S1 | E | S2 | S6 | S3 | E | S4 | E |
| S1 | S1 | S1 | S1 | E | E | E | E | E |
| S2 | E | E | S2 | E | E | E | E | E |
| S3 | E | E | E | E | S5 | E | E | E |
| S4 | E | E | E | E | E | E | S6 | S4 |
| S5 | E | E | E | E | E | S6 | E | E |
| S6 | E | E | E | E | E | E | E | E |

Paso 6:
Construir el AFD.



Ya que esta aplicación es una aplicación de escritorio desarrollada en visual estudio, el lenguaje de C#, ya que este IDE no cuenta con la opción de ser multiplataforma necesariamente tenemos que trabajar en un entorno de Windows como sistema operativo.

Glosario de Clases:

Glosario de métodos

Program.cs

Main();

Año

Mes

Documento

Errores

Token

TablaErrores:

AceptarBtn_Click

AñadirdatosTablaErrores

TablaTokens

atrasbtn_Click

AñadirdatosTablaTokens

AdminDoc

buscarbtn_Click

inicializarTreeView

treeView1_NodeMouseClicked

guardarToolStripMenuItem_Click

Form1

AbrirToolStripMenuItem_Click

guardarToolStripMenuItem_Click

guardarComoToolStripMenuItem_Click

acercaDeToolStripMenuItem_Click

salirToolStripMenuItem_Click

analizarToolStripMenuItem_Click

GuardarToken

guardarErrores

validarPalabraReservada

tablaDeSimbolosToolStripMenuItem_Click

CambiarColor

ObtenerDatosTreeView

