Universidad Rafael Landívar
Facultad de Ingeniería
Lenguajes Formales y Autómatas, sec 01
Mgtr. Moises Alonso

Proyecto Práctico Fase: 3

José Alejandro Montenegro Monzón

1229918

Guatemala 6 de mayo de 2020

ÍNDICE

I. Entradas, Procesos, Salidas y PseudoCódigo

1. **Entrada:** La única entrada de este proyecto es la lectura del archivo de texto. En ésta fase también podría contarse la cadena a validar como una entrada, con el fin de averiguar si cumple con la gramática establecida.
2. **Salida:** La salida será la validación de la cadena que se ingresó. Además, se mostrará en una lista que caracteres perteneces a que token.
3. **Proceso:**
   - Ingresar la cadena que se desea validar que cumpla con la expresión que se ingresó
   - Recorrer los estados carácter por carácter si se para validar en que estado se encuentra dicho carácter.
   - Si el carácter no se encuentra en el estado actual, el recorrido terminará con una negación, indicando que la cadena no es válida para dicha expresión.
   - Si la cadena que ingresó se termina de validar entre los estados correspondientes, la cadena es válida y se mostrarán en pantalla los caracteres de la cadena y a qué token pertenecen.

II. Pseudocódigo del método main()

Proceso

1. Verificar si existen valores en ListSETS para obtener las cadenas de caracteres aceptados
   a. Si contiene valores recorrer la lista
      i. Buscar la palabra previa al "="
      ii. Buscar la cadena posterior al "="
      iii. Obtener todos los caracteres ASCII que pertenecen a dicho SET
      iv. Retornar la lista con las cadenas pertenecientes a cada uno de los SETS
2. Crear Cada nodo de estados
   a. Recorrer la tablaE para obtener cada uno de los estados que aparecieron
   b. Obtener el estado
   c. Obtener los estados a los que dicho estado apunta
   d. Verificar si el estado es terminal
      i. Si el estado contiene el carácter de aceptación
         - Es terminal
      ii. Si el estado no contiene el carácter de aceptación
         - No es Terminal
   e. Obtener los símbolos que son aceptados en dicho estado
   f. Crear Nodo
3. Una vez Obtenidos cada uno de los Nodos estados, Obtener la cadena que se desea evaluar
4. Empezar el recorrido en el estado "S0". El estado actual será "S0".
   a. Mientras la cadena posea contenido
      i. Si la cadena [0] es igual al carácter de aceptación del estado siguiente
         - Estado actual igual a estado siguiente
      ii. Sino validar con el siguiente estado al que apunta el estado actual
      iii. Si el estado Actual no cambió
         - La cadena que ingresó no es válida en este AFD.
         - Terminar el recorrido.

iv. Sino continuar con los recorridos de los nodos

b. Verificar si la cadena no contiene caracteres y el estado actual es terminal
   i. Notificar que la cadena que ingresó fue válida.
c. Sino, Si la cadena contiene caracteres y el estado actual no es terminal
   i. Notificar que la cadena que ingresó no fue válida.
d. Sino, si la cadena contiene caracteres
   i. Notificar que la cadena que ingresó no fue válida

5. Terminar validación

PseudoCódigo(En frases)

Main

Dic<string,string[]> TablaE , List<string> Simbol, int ultimoV, List<string> ListSETS, string cadena;
Verificar si la ListaSETS.Count() es igual a 0.
Si no lo es
    Verificar las variables que serán válidas para cada uno de los SETS por ejemplo: Dígito, Letra, etc.
Si no lo es
    Continuar con el proceso
Recorrer (TablaE) para crear los Nodo del Grafo, en este caso los llamaremos NGrafos
    De cada uno de los estados se debe de obtener
        Su estado que sería Estado
        Los otros grafos a los que está conectado que serían los apuntadores[]
        Si el estado.Contains(ultimoV)
            Entonces el nodo es terminal, terminal = true
        Sino
            Terminal = false;
        Ingresar los símbolos que serán aceptados por dicho nodo.
        Ingresar los nodos al Nodo, Grafo = new Grafo(estado,apuntadores[],terminal, Símbolos)
Una vez teniendo creados los nodos del grafo, Recorrer el Grafo, Empezando por el nodo S0
    Validar el estado en el que se está,
    Mientras Cadea.lenght!=0
    Si (Estado Actual == grafo.Estado)
        Comparar carácter con símbolo de validación
            Si (símbolo[i]==cadena[0])
                Estado Actual = apuntador[i]
                Terminal = grafo.terminal
            Sino
            Break;
    Si la cadena.lenght ¡=0
        Return false;
    Si la cadena.length == 0 y terminal
        Return true;
    Sino la cadena.lenght == 0 y ¡ terminal
        Return false;
    No, entonces continuar al estado el recorrido buscando el estado actual
    Si no lo encuentra if(estado Actual ¡= encontrado)

Entonces return false;

Si el resultado del recorrido es false, resultado == false
Write(“La cadena no es valida”);
Sino Write (“La cadena es Válida”);

PseudoCódigo

```
Main()
{
    // dichas variables ya poseen valores
    Dic<string,string[]> TablaE;
    List<string> Simbol;
    int ultimoV;
    List<string> ListSETS;
    string cadena;

    // Verificar que ListSETS contenga elementos
    Dictionary<string,string> dic;
    If(ListSETS!=null)
    {
       dic = grafos.GenerarCadenasDeValidación(ListSETS);
    }

    // Crear Los Nodos del Grafo
    List<Grafo> LNodo;
    Foreach(string llave in TablaE)
    {
        string estado = llave;
        string[] puntero = TablaE.Valor;
        bool terminal = false;
        If (estado.Contains(ultimoV))
        {
            terminal = true;
        }
        Grafo grafo = new Grafo(estado,terminal,puntero, Simbol);
        LNodo.Add(grafo);
    }
    //Empezar el recorrido del grafo
    NGrafo grafos;
    bool verdad = grafos.Validar(LNodo,cadena, dic)
    if(verdad)
    {
        Write("La frase es valida en esta lenguaje formal");
        Read();
    }
```

```
        Else
        {
                Write("La frase no es válida en esta lenguaje formal");
                Read();
        }
}
```

II. PRUEBAS REALIZADAS ()

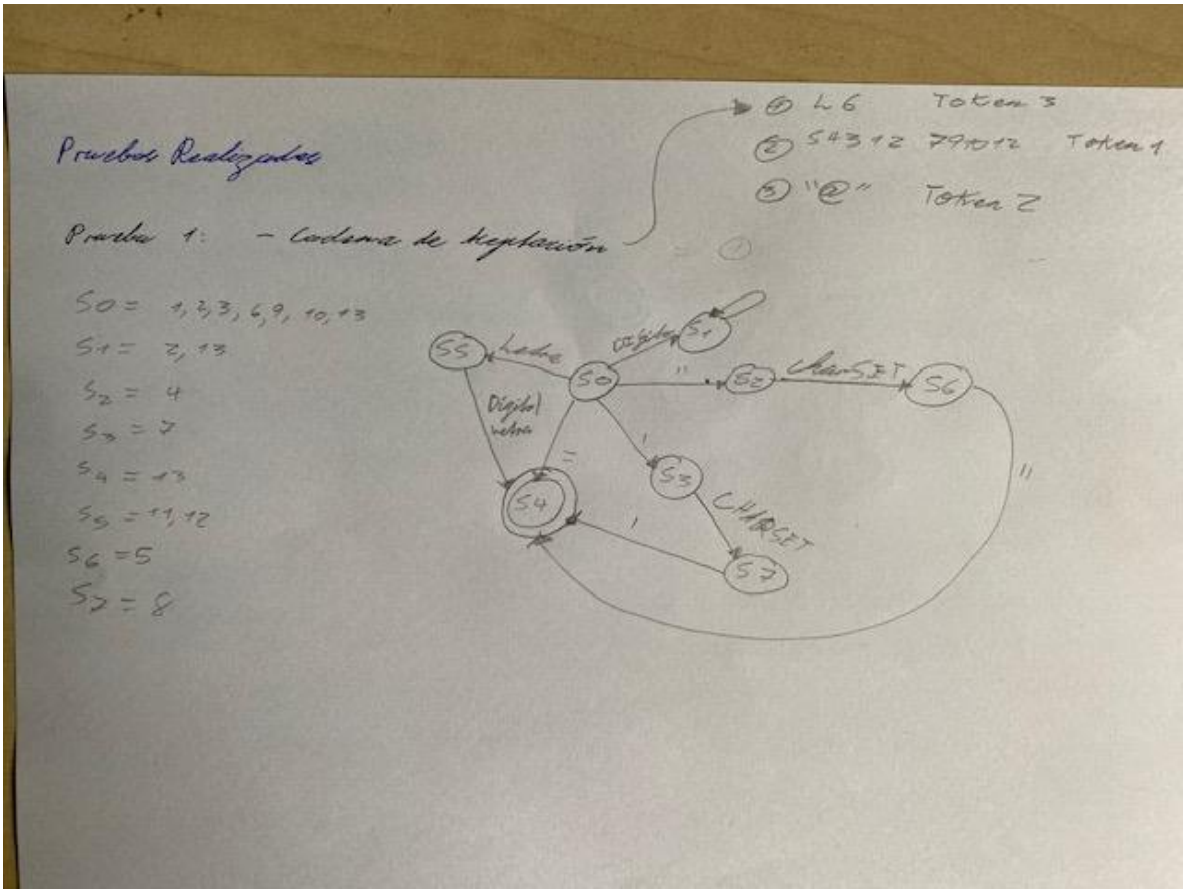


Imagen 1 : Propia

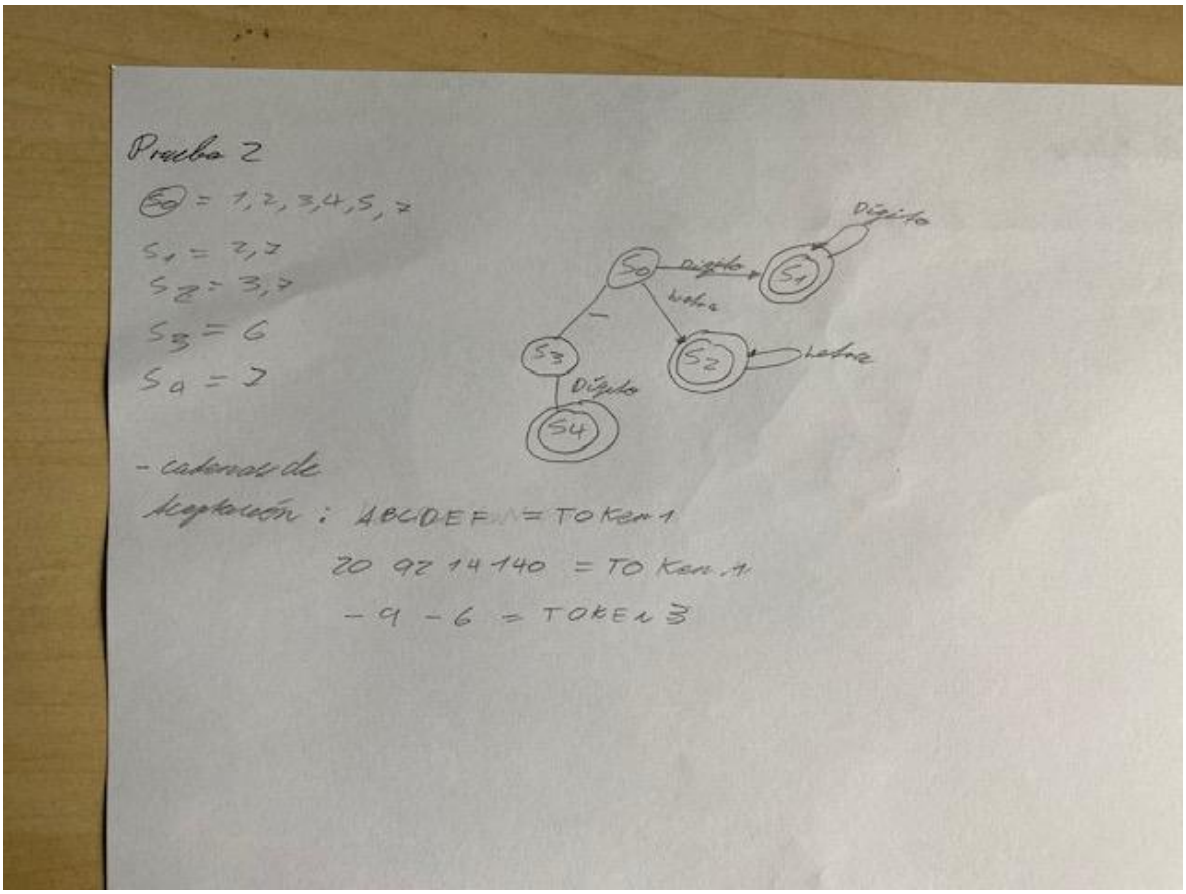
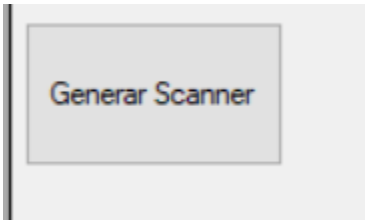


IMAGEN 2 :PROPIO

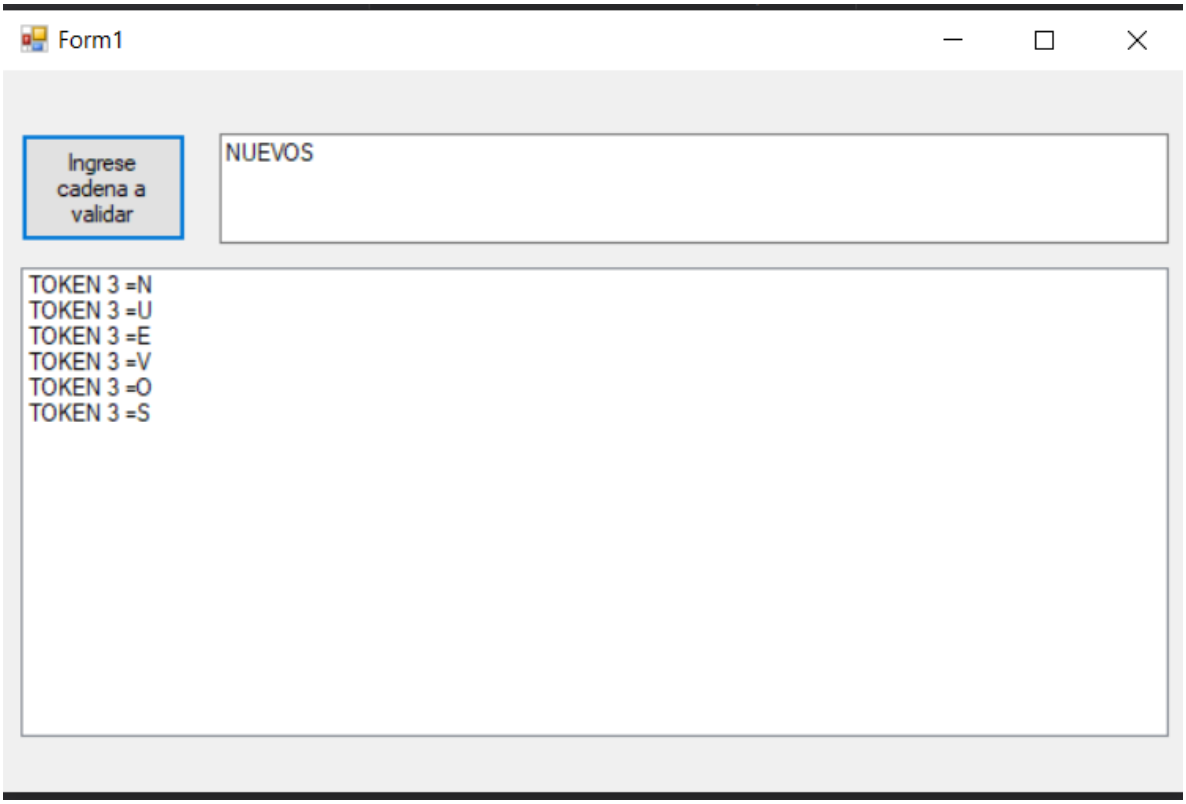
III. MANUAL DE USUARIO

Tras haber seleccionado el archivo de gramática, debe de presionar el botón Generar Scanner



Una vez presionado usted debe de colocar el nombre que desea al archivo, simplemente su nombre, no ninguna terminación o algún .txt, .png, etc.

Después se dirige a la carpeta a la que ingresó su archivo, corre su programa y lo ejecuta. A continuación se le mostrará la pantalla donde debe de ingresar la cadena que desea valida para dicha gramática.



Tras haber ingresado la cadena, si la cadena es válida, en la pantalla inferior, se desplegarán a que token pertenece cada carácter.