Manual de Técnico Interprete Gramáticas

Objetivo: La solución de software solicitada, brinda al usuario por medio de un archivo cargado al sistema la interpretación de Gramáticas de Tipo 2 y proporciona, la gramática, el autómata y el procedimiento de solución mediante un autómata de pila, en un ambiente web, en el cual el usuario podrá verificar todas las imágenes generadas por el software, la tabla de resumen y el nombre de la gramática.

Presentación: en el siguiente manual encontrara la descripción de los métodos y partes mas importantes del programa.

1. Librerías y Variables Principales:
   Aquí se presenta las librerías y las listas utilizadas para llevar acabo el programa, recordemos que las listas son un TDA muy fácil de utilizar para estructurar la información:

```
import os
import copy
from copy import deepcopy
from tkinter.filedialog import askopenfilename

listaLineasArchivo=[]
listadeListasGramatica=[]
listaGramaticasT=[]
listaGramaticasTc=[]
listaGramaticasProducciones=[]
listaInfoGramaticas=[]
listaGramaticaAutoPila=[]
listaNombresGramaticasAP=[]
listaTerminalesGramaticasAP=[]
listaAlfabetoGramaticasAP=[]
listaProduccionesGramaticasAp=[]
listaProduccionesAPila=[]
contadorGraficas=0
listaIndiceGraficas=[]
nombreAutomata="AP_"
terminales=""
alfabetoPila=""
estados="Estados = i,p,q,f"
estadoInicio="Estado Inicio = { i }"
estadoFin="Estado Fin = { f }"
produccionCadTemp=""
listaProducciones=[]
listaCadenaIngresada=[]
pilaAutomata=[]
listaEstadoPila=[]
```

2. Método Lectura:
   Método encargado de hacer la lectura y la primera descomposición del archivo para luego ser interpretados.

```
def cargarArchivo():
    global listaGramaticasT
    archivo = askopenfilename()#Abre la interfaz para escoger el archivo a cargar
    print(archivo)#se obtiene el URL

    #Se abre el archivo
    archivoLectura = open('' + archivo + '', 'r')

    for linea in archivoLectura:
        lineaGramatica=linea.split()
        listaLineasArchivo.append(lineaGramatica)

        if lineaGramatica == ['*']:
            listaTemp = list(listaLineasArchivo)
            listadeListasGramatica.append(listaTemp)
            listaLineasArchivo.clear()

    indexPrueba = len(listadeListasGramatica)

    for x in range (indexPrueba):
        indexUltimo = len(listadeListasGramatica[x])
        listadeListasGramatica[x].pop(int(indexUltimo)-1)

    for j in range (indexPrueba):
        lTemp=[]
        lTempProd=[]
        indexRecorido = len(listadeListasGramatica[j])
```

```
    indexPrueba = len(listadeListasGramatica)

    for x in range (indexPrueba):
        indexUltimo = len(listadeListasGramatica[x])
        listadeListasGramatica[x].pop(int(indexUltimo)-1)

    for j in range (indexPrueba):
        lTemp=[]
        lTempProd=[]
        indexRecorido = len(listadeListasGramatica[j])
        for y in range(indexRecorido):
            if(y==0):
                lTemp.append(listadeListasGramatica[j][y])
            elif(y==1):
                cadena=listadeListasGramatica[j][y]
                cadena2=str(cadena)
                Temporal=cadena2.split(";")
                lTemp.append(Temporal)
            else:
                lTempProd.append(listadeListasGramatica[j][y])
        lTemp.append(list(lTempProd))
        listaGramaticasT.append(list(lTemp))
        lTemp.clear()
        lTempProd.clear()
    #---------------------------  Impresion de Todas las Gramaticas bien Separadas ----------------------------------------------
    indexPrueba = len(listaGramaticasT)

    for obj in listaGramaticasT:
        print(obj)
```

3. Producción Gramáticas:
   En este método se eliminan las gramáticas que no son de Tipo 2, se gurda en nuevas listas la información ya interpretada y se muestra el resultado en consola.

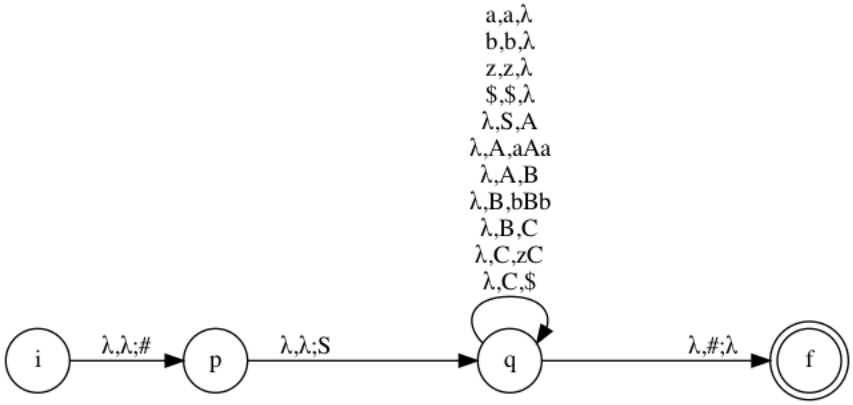
```
def produccionesGramaticas():
    listaGramaticasTc= deepcopy(listaGramaticasT)
    indexPrincipal = len (listaGramaticasTc)
    bandera = False
    for x in range(indexPrincipal):
        indexInterno = len(listaGramaticasTc[x])
        for y in range(indexInterno):
            if(y==2):
                tempProducciones = list(listaGramaticasTc[x][y])
                producciones=[]
                indexProducciones = len(tempProducciones)
                for z in range(indexProducciones):
                    cadenaProduccion = str(listaGramaticasTc[x][y][z]).split("->")
                    cadenaProduccion2=(str(cadenaProduccion).split(","))
                    producciones.append(cadenaProduccion2)
                listaGramaticasTc[x][y]=list(producciones)
                producciones.clear()

    print ("------------------ Aqui vere como quedan Gramticas Normales --------------------")
    print("--------------------------------------------------------------------------------------------------")
    for obj in listaGramaticasT:
        print(obj)
    print("\n")
    print ("------------------ Aqui vere como quedan Gramticas Expandidas --------------------")
    print("--------------------------------------------------------------------------------------------------")
    for obj in listaGramaticasTc:
        print(obj)
    print("------------------- Eliminacion ------------------------------------------------")
    for x in range(indexPrincipal):
```

```
    for obj in listaGramaticasTc:
        print(obj)
    print("------------------- Eliminacion ------------------------------------------------")
    for x in range(indexPrincipal):
        indexInterno = len(listaGramaticasTc[x])
        print(indexInterno)
        for y in range(indexInterno):
            #print("prueba"+str(y))
            if(y==2):
                indexBusquedaInterna = len(listaGramaticasTc[x][y])
                print(indexBusquedaInterna)
                for z in range(indexBusquedaInterna):
                    print(str(listaGramaticasTc[x][y][z]))
                    print(int(len(listaGramaticasTc[x][y][z])))
                    if(int(len(listaGramaticasTc[x][y][z])>=4)):
                        bandera=True
                print(bandera)
        if(bandera==False):
            del(listaGramaticasTc[x])
            del(listaGramaticasT[x])
        else:
            bandera=False
    print("--------------------------------------------------------------------------------------------------")
    for obj in listaGramaticasT:
        print(obj)
    print("\n")
    print ("------------------ Aqui vere como quedan Gramticas Expandidas --------------------")
    print("--------------------------------------------------------------------------------------------------")
    for obj in listaGramaticasTc:
```

4. Generador de Automata:

```
    listaProducciones.clear()
    #------------------------------ Datos para Grafo y HTML ---------------------------------------------------
    indexlistas=len(listaProduccionesGramaticasAp)
    for x in range(indexlistas):
        indexProducciones1 = len(listaProduccionesGramaticasAp[x])
        for y in range(indexProducciones1):
            print(listaProduccionesGramaticasAp[x][y])
            produccionCadTemp=produccionCadTemp+str(listaProduccionesGramaticasAp[x][y])+"\n"
        listaProduccionesAPila.append(produccionCadTemp)
        produccionCadTemp=""
        #-------------------- Creacion de Grafos ----------------------------------
        global contadorGraficas
        file = open("grafo"+str(contadorGraficas)+".dot","w")
        file.write("digraph G{\n")
        file.write("rankdir=LR;\n")
        file.write(crearNodo("A","i","circle","black"))
        file.write(crearNodo("B","p","circle","black"))
        file.write(crearNodo("C","q","circle","black"))
        file.write(crearNodo("D","f","doublecircle","black"))
        file.write(unionNodo("A","B")+"[label = \"λ,λ;#\"];")
        file.write(unionNodo("B","C")+"[label = \"λ,λ;"+str(listaGramaticasT[0][1][2]).replace("']","")+"\"];")
        file.write(unionNodo("C","C")+"[label = \""+str(listaProduccionesAPila[x])+"\"];")
        file.write(unionNodo("C","D")+"[label = \"λ,#;λ\"];")
        file.write("}")
        file.close()
        os.system('dot -Tpng grafo'+str(contadorGraficas)+'.dot -o grafo'+str(contadorGraficas)+'.png')
        #os.startfile("grafo.png")
        listaIndiceGraficas.append(contadorGraficas)
        contadorGraficas=contadorGraficas+1
```



5. Código Base para HTMls:

```
    file = open("index.html","w")
    file.write("<!DOCTYPE HTML>\n")
    file.write("<htm lang = \"es\">\n")
    file.write("<head>\n")
    file.write("<TITLE>GENERAR AUTOMATA DE PILA</TITLE>\n")
    #file.write("<link href=\"/Users/negrocorado/Desktop/Style.css\" rel=\"stylesheet\" type=\"text/css\">\n")
    file.write("</head>\n")
    file.write("<body>\n")
    file.write("<div id = \"titulo\">\n")
    file.write("<h1>RESULTADO DE AUTOMATA DE PILA GENERADO</h1>\n")
    file.write("</div>\n")
    file.write("<div id= \"cuerpo\">\n")
    file.write("<table id= \"TablaGramatica\">\n")
    file.write("<tr>\n")
    file.write("    <td>\n")
    indexHTML=len(listaNombresGramaticasAP)
    for x in range(indexHTML):
        file.write("<h2>Nombre:  "+str(listaNombresGramaticasAP[x])+" </h2>\n")
        file.write("<h2>Terminales: { "+str(listaTerminalesGramaticasAP[x])+" }</h2>\n")
        file.write("<h2>AlfabetoPila: { "+str(listaAlfabetoGramaticasAP[x])+" }</h2>\n")
        file.write("<h2>"+estados+"</h2>\n")
        file.write("<h2>"+estadoInicio+"</h2>\n")
        file.write("<h2>"+estadoFin+"</h2>\n")
    file.write("    </td>\n")
    file.write("    <td>\n")
    indexGrafos=len(listaIndiceGraficas)
    for y in range(indexGrafos):
        file.write("        <img src=\""+"grafo"+str(listaIndiceGraficas[y])+".png"+"\">\n")
    file.write("    </td>\n")
```

6.