Universidad de San Carlos de Guatemala
Laboratorio Lenguajes Formales y de Programación
Gramáticas Regulares
Vacaciones 1er semestre

Manual Técnico

Ivan de Jesus Pisquiy Escobar
201901815

Descripción del Proyecto

El proyecto consiste en desarrollar un programa en Python que permita trabajar con gramáticas regulares, autómatas finitos deterministas (AFD) y autómatas finitos no deterministas (AFN). El objetivo es brindar una herramienta visual que facilite la definición de gramáticas, la generación de AFD y AFN, y la evaluación de cadenas válidas para estas gramáticas, utilizando diferentes paradigmas de programación.

El programa se desarrollará utilizando la librería Tkinter para crear una interfaz gráfica con diversas ventanas, a las cuales se podrá acceder según las acciones deseadas por el usuario. Además, se implementará una sección de reportes que mostrará detalles sobre las gramáticas generadas. Para mejorar la comprensión, el programa también permitirá la creación de gráficos utilizando la herramienta Graphviz.

Una gramática regular es una herramienta fundamental para validar cadenas y se podrá transformar en un AFD o un AFN. Estos procesos se describirán detalladamente en el programa.

El usuario podrá crear las gramáticas directamente en la aplicación o cargarlas desde un archivo de texto con extensión ".grm". Este archivo contendrá las producciones, los terminales, los no terminales y la producción inicial. La estructura del archivo se explicará en la sección de carga masiva. No será necesario utilizar un analizador para leer el archivo, ya que se asumirá que viene sin errores de sintaxis y con un patrón definido.

Se realizará una revisión del código para verificar los paradigmas utilizados, y se requerirá que el desarrollador pueda explicarlos para comprobar la autoría de la aplicación.

El programa permitirá iniciar tanto desde la creación de una gramática para generar su respectivo AFD/AFN, como desde la creación de un AFD/AFN para generar su gramática correspondiente. Se detallará la notación a utilizar para AFD/AFN y gramáticas regulares, y se asegurará que los reportes incluyan todos los elementos necesarios de cada componente.

Guía de Uso

Importes:

```
from tkinter import *
from tkinter import filedialog
from tkinter import ttk
from tkinter import messagebox
import os
import graphviz
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
from reportlab.lib.units import inch
from PIL import ImageTk, Image
```

Se utilizó la biblioteca Tkinter para desarrollar el programa de forma visual y crear las distintas ventanas necesarias para la interacción con el usuario. Tkinter es una biblioteca estándar de Python que proporciona herramientas para construir interfaces gráficas de usuario (GUI, por sus siglas en inglés) de manera sencilla.

Al utilizar Tkinter, se puede diseñar una interfaz de usuario amigable y accesible, lo que mejora la experiencia del usuario al utilizar el programa. Además, Tkinter es una biblioteca ampliamente utilizada y bien documentada, lo que proporciona recursos y ejemplos útiles para el desarrollo de la interfaz gráfica.

Se utilizó la biblioteca Graphviz para generar gráficos que representen visualmente las estructuras y relaciones de las gramáticas, autómatas y producciones definidas en el programa.

Graphviz es una poderosa herramienta de visualización de gráficos que permite representar datos y estructuras complejas de forma clara y comprensible. Proporciona una amplia gama de opciones para crear diagramas y grafos, incluyendo autómatas finitos deterministas (AFD) y autómatas finitos no deterministas (AFN).

Se utilizó la biblioteca ReportLab para generar informes detallados que muestren todos los detalles de las gramáticas generadas en el programa.

ReportLab es una biblioteca de Python que permite crear documentos PDF de manera programática. Proporciona herramientas para generar informes estructurados y personalizados, lo que resulta útil para presentar la información de manera clara y profesional.

En el proyecto, la utilización de ReportLab permite generar informes que incluyan todos los elementos relevantes de las gramáticas, como producciones, terminales, no terminales y producción inicial. Estos informes pueden incluir tablas, gráficos y otros elementos visuales para representar la información de manera efectiva.

Funciones:

Main_window:

```
def main_window():
    main = Tk()
    main.title("Gramáticas Regulares")

    window = ttk.Frame(main, padding=50)
    window.grid()

    ttk.Label(window, text="Lenguajes formales y de Programación").grid(column=0, row=0)
    ttk.Label(window, text="Sección: A").grid(column=1, row=0)
    ttk.Label(window, text="Carné: 201901815").grid(column=0, row=1)
    ttk.Label(window, text="Ivan de Jesus Pisquiy Escobar").grid(column=1, row=1)

    ttk.Button(window, text="AFN", command=menu_afn).grid(column=0, row=4, padx=20, pady=10)
    ttk.Button(window, text="AFD", command=menu_afd).grid(column=1, row=4, padx=20, pady=10)
    ttk.Button(window, text="OE", command=menu_oe).grid(column=0, row=5, padx=20, pady=10)
    ttk.Button(window, text="Carga Masiva", command=carga_masiva).grid(column=1, row=5, padx=20,
    pady=10)
    ttk.Button(window, text="Cerrar", command=main.quit).grid(column=0, row=6, padx=20, pady=10)
    main.mainloop()
```

La función main_window() crea la ventana principal del programa utilizando la biblioteca Tkinter.

En esta ventana se muestra el título "Gramáticas Regulares" y se presentan detalles del autor. Además, se incluyen botones que permiten al usuario acceder a diferentes secciones del programa, como la creación de autómatas finitos no deterministas (AFN), autómatas finitos deterministas (AFD), operaciones entre expresiones regulares y la carga masiva de gramáticas. También se incluye un botón "Cerrar" para salir del programa. La función utiliza la biblioteca ttk para crear y posicionar los elementos en la ventana principal. Una vez que se ha configurado todo, el programa entra en el bucle principal (mainloop()) para que la ventana sea visible y responda a las interacciones del usuario.

Carga Masiva:

```python
def carga_masiva():

    def abrir_archivo_afd():
        global contenido_texto
        # Abre el diálogo de buscar archivo y retorna la ruta del archivo seleccionado
        ruta_archivo = filedialog.askopenfilename(filetypes=[('Archivos AFD', '*.afd')])

        # Verifica si se seleccionó un archivo y si es así, lo lee y lo devuelve como un diccionario
        de Python
        if ruta_archivo:
            with open(ruta_archivo, 'r') as archivo_texto:
                contenido_texto = archivo_texto.read()
        main_masiva.destroy()

    def abrir_archivo_afn():
        global contenido_texto
        # Abre el diálogo de buscar archivo y retorna la ruta del archivo seleccionado
        ruta_archivo = filedialog.askopenfilename(filetypes=[('Archivos AFN', '*.afn')])

        # Verifica si se seleccionó un archivo y si es así, lo lee y lo devuelve como un diccionario
        de Python
        if ruta_archivo:
            with open(ruta_archivo, 'r') as archivo_texto:
                contenido_texto = archivo_texto.read()
        main_masiva.destroy()
```

La función carga_masiva() se encarga de mostrar una ventana para cargar archivos en el programa. Esta función contiene dos funciones internas: abrir_archivo_afd() y abrir_archivo_afn(), que se utilizan para seleccionar y leer archivos de tipo AFD y AFN, respectivamente.

La ventana de carga masiva se crea utilizando la biblioteca Tkinter. Se muestra el título "Carga Masiva" y se configura un marco para contener los elementos de la interfaz. Dentro de este marco, se coloca una etiqueta que indica al usuario que elija qué tipo de archivo desea cargar.

Se incluyen dos botones: uno para cargar archivos de tipo AFD y otro para cargar archivos de tipo AFN. Estos botones llaman a las funciones abrir_archivo_afd() y abrir_archivo_afn() respectivamente, cuando se les da clic. Estas funciones utilizan el diálogo de búsqueda de archivos para permitir al usuario seleccionar un archivo específico del tipo correspondiente. Luego, el contenido del archivo seleccionado se lee y se guarda en la variable contenido_texto (que se define como global). Finalmente, la ventana de carga masiva se cierra llamando a la función main_masiva.destroy().

Menu AFD/AFN:

```
main_afn = Tk()
main_afn.title("Menu AFN")

window = ttk.Frame(main_afn, padding=50)
window.grid()

ttk.Label(window, text="Sección: A").grid(column=0, row=0)
ttk.Label(window, text="Carné: 201901815").grid(column=1, row=0)
ttk.Label(window, text="Ivan de Jesus Pisquiy Escobar").grid(column=0, row=1)

ttk.Button(window, text="Crear AFN", command=crear_afn).grid(column=0, row=2, padx=20, pady=10)
ttk.Button(window, text="Evaluar Cadena", command=evaluar_afn).grid(column=1, row=2, padx=20,
pady=10)
ttk.Button(window, text="Generar Reporte AFN", command=reporte).grid(column=0, row=3, padx=20,
pady=10)
ttk.Button(window, text="Ayuda", command=ayuda_afn).grid(column=1, row=3, padx=20, pady=10)
ttk.Button(window, text="Cerrar", command=main_afn.destroy).grid(column=0, row=4, padx=20,
pady=10)
```

La función menu_afn() es parte de la funcionalidad del programa y se encarga de crear una ventana de menú para el manejo de un Autómata Finito No Determinista (AFN). Al igual que esta función, existe una función equivalente llamada menu_afd() que se utiliza para el manejo de Autómatas Finitos Deterministas (AFD).

Además, se presentan varios botones que ofrecen funcionalidades relacionadas con el AFN. Estas funcionalidades incluyen la creación de un AFN, la evaluación de cadenas, la generación de reportes del AFN, la opción de ayuda y la posibilidad de cerrar la ventana.

Sin embargo, al utilizar la función menu_afd(), se muestra una ventana de menú similar pero diseñada específicamente para el manejo de Autómatas Finitos Deterministas (AFD). Esta ventana ofrece las mismas opciones que la función menu_afn(), como la creación del AFD, la evaluación de cadenas, la generación de reportes, la ayuda y el cierre de la ventana.

Crear AFD/AFN:

```
nombre = entry_nombre.get()
estados = entry_estados.get()
alfabeto = entry_alfabeto.get()
inicial = entry_inicial.get()
aceptados = entry_aceptados.get()
transiciones = entry_transiciones.get().split(' ')

contenido = f"{nombre}\n"
contenido += f"{estados}\n"
contenido += f"{alfabeto}\n"
contenido += f"{inicial}\n"
contenido += f"{aceptados}\n"
for transicion in transiciones:
    contenido += f"{transicion}\n"

nombre_archivo = f"{nombre}.afn"

file = open(nombre_archivo, 'w')
file.write(contenido)
file.close()

create_afn.destroy()

messagebox.showinfo("¡Éxito!", "Datos guardados correctamente")
```

La función crear_afn() se utiliza para crear un archivo de texto que representa un Autómata Finito Determinista (AFD) a partir de los datos ingresados por el usuario en una interfaz gráfica. Esta función es similar a la función crear_afn(), pero está diseñada específicamente para trabajar con AFD.

Al igual que en la función crear_afn(), la función crear_afd() crea una ventana llamada "Crear AFD" utilizando la biblioteca Tkinter. En esta ventana, el usuario puede ingresar información sobre el AFD que desea crear, como el nombre, la lista de estados, el alfabeto, el estado inicial, los estados de aceptación y las transiciones.

Una vez que el usuario ha proporcionado los datos requeridos, puede hacer clic en el botón "Aceptar" para guardar los datos y generar un archivo de texto que representa el AFD. La función guardar_datos() se encarga de recopilar los datos ingresados por el usuario, crear una cadena de texto que describe la estructura del AFD y guardarla en un archivo con la extensión ".afd".

Generar Reporte:

```
def reporte():
    global contenido_texto

    lineas = contenido_texto.splitlines()
    nombres = []

    grupos = []
    grupo_actual = []

    for elemento in lineas:
        if elemento == "%":
            grupos.append(grupo_actual)
            grupo_actual = []
        else:
            grupo_actual.append(elemento)

    for i in range(len(grupos)):
        nombres.append(grupos[i][0])
```

La función reporte() se encarga de generar un informe (reporte) a partir de los datos almacenados en el archivo de texto y visualizados en la interfaz gráfica. Este informe incluye información detallada sobre un autómata, como sus estados, alfabeto, estado inicial, estados de aceptación y transiciones.

Al seleccionar un autómata de la lista y hacer clic en el botón "Crear Reporte", se invoca la función crear_reporte(). Esta función recopila los datos relevantes del autómata seleccionado, como el nombre, los estados, el alfabeto, el estado inicial, los estados de aceptación y las transiciones.

A continuación, se crea un gráfico del autómata utilizando la biblioteca Graphviz. Se generan los nodos correspondientes a los estados y se crean las conexiones entre los estados mediante las transiciones definidas. Estas conexiones se agregan al gráfico, permitiendo visualizar la estructura del autómata.

Además del gráfico, se crea un archivo de imagen en formato PNG con el nombre del autómata. Luego, se crea un archivo de informe en formato PDF utilizando la biblioteca ReportLab.

Ayuda:

```
def ayuda_afn():

    info_afn = Tk()
    info_afn.title("Ayuda")

    window = ttk.Frame(info_afn, padding=50)
    window.grid()

    info = "Un autómata finito no determinista (AFN) es un modelo matemático utilizado en el
    campo de la teoría de autómatas y lenguajes formales."
    info2 = "Es una variante del autómata finito (AF) que permite transiciones no deterministas,"
    info3 = "lo que significa que en un estado dado puede haber múltiples transiciones posibles
    para un símbolo de entrada determinado."

    ttk.Label(window, text="¿Qué es un Autómata AFN?").grid()
    ttk.Label(window, text=info).grid()
    ttk.Label(window, text=info2).grid()
    ttk.Label(window, text=info3).grid()

    ttk.Button(window, text="Cerrar", command=info_afn.destroy).grid(pady=10)
```

La función ayuda_afn() ha sido modificada para proporcionar ayuda específica sobre los autómatas finitos deterministas (AFD). Ahora se llama ayuda_afd() y muestra información relevante sobre los AFD en lugar de los AFN.

Al ejecutar la función ayuda_afd(), se creará una nueva ventana con el título "Ayuda". Dentro de esta ventana, se utilizará el framework Tkinter para crear un marco (ttk.Frame) con un relleno de 50 píxeles, el cual se ajustará al tamaño de la ventana principal.

Se mostrará el título "¿Qué es un Autómata AFD?" utilizando un widget ttk.Label. A continuación, se incluirán tres etiquetas adicionales (ttk.Label) que proporcionarán información sobre los autómatas finitos deterministas.