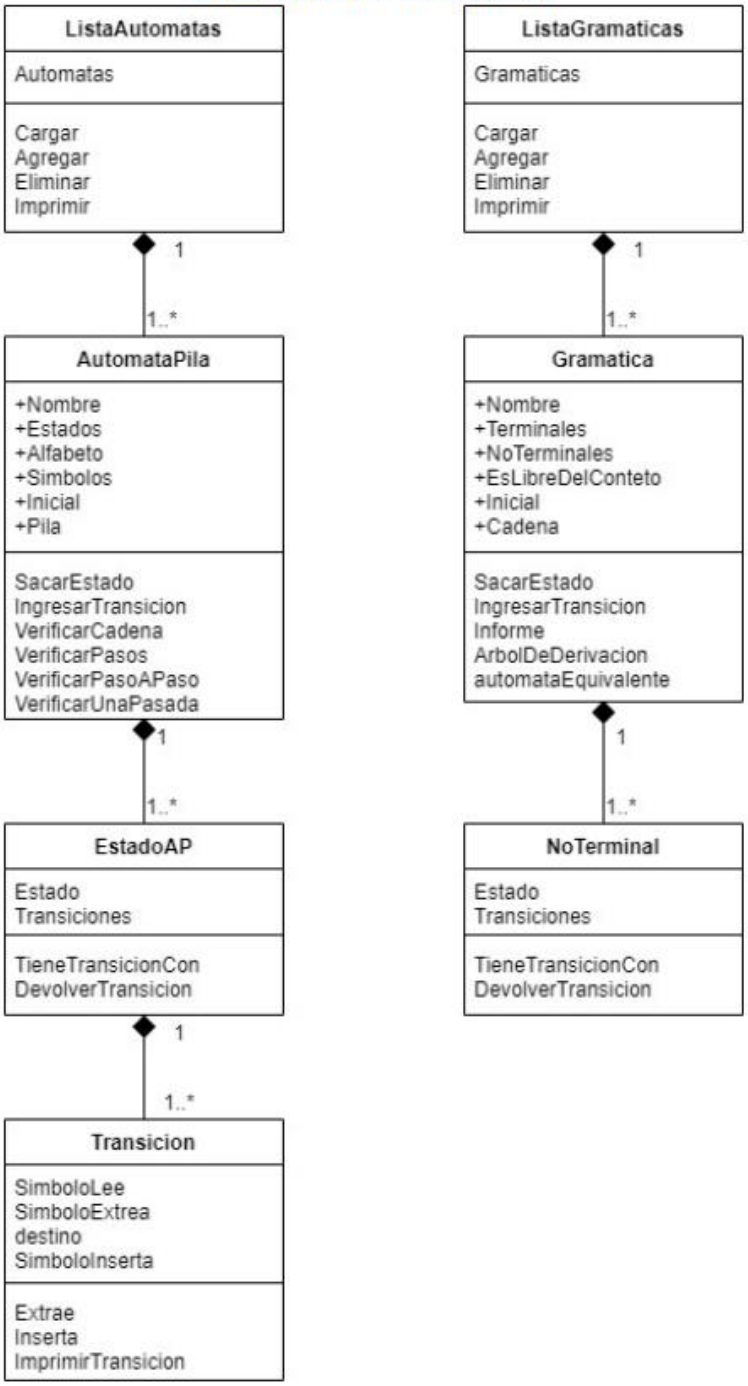
Diccionario de clases



Transición

Es una clase bastante sencilla que solo cuenta con 2 atributos, dicha clase pudiera ser perfectamente reemplazado por un pequeño vector de 2 espacios, sin embargo, me pareció más practico manejar esta clase para tener bien identificados los atributos y no manejar posiciones de un vector, de esta manera es más entendible

Estado

La clase estado crea un objeto por cada estado de un autómata, dicho objeto contiene sus transiciones, es decir desde el mismo objeto se puede saber con qué carácter y hacía que otro estado se puede mover el autómata, esto hace al código más dinámico, legible y con una estructura más segura, considerando también que el movimiento entre estados es efectivo, se puede pensar perfectamente en un código que elimine esta clase y la clase de transiciones, sin embargo sería un código que haría un uso excesivo de ciclos para encontrar las transiciones y movimientos entre estados, los atributos para este objeto son principalmente el símbolo que lo identifica y la lista de transiciones, por medio de la conexión entre estados se logra formar la cadena valida solicitada en los reportes.

No terminal

Es una clase similar a la clase Estado, sus funciones son las mismas pero las transiciones no emplean la clase Transición, y sencillamente sen cadenas que luego con la lógica de otras clases son utilizadas

AutomataPila

Por medio de esta clase se crean los objetos AP los cuales son los pilares del segundo modulo del programa, dichos objetos cuentan con los siguientes atributos, los cuales son los elementos de un autómata; nombre del autómata, lista de estados del autómata (esta lista guarda objetos Estado), lista del alfabeto del autómata, lista de símbolos de pila, estado inicial y estado de aceptación. Esta clase cuenta con sus respectivos setter para cada uno de sus atributos, a si mismo ella misma se sustenta con las validaciones necesarias para saber si un carácter existe ya en el alfabeto o los estados, si una transición a sido correctamente ingresada y tiene caracteres válidos para la gramática, verificación de cadenas, creación de reporte y generación de gramáticas a partir de si mismo. Todos estos métodos fueron hechos desde la misma clase para tener un mayor manejo de los datos debido a que todas estas consultas se hacen hacia sus propios atributos, realizar esto en clases externas implicaría encontrar primero el objeto correcto para luego hacer sus validaciones. Pensando siempre en la metodología Top-Down la programación de estos métodos dentro de la clase se hace perfecta para luego solo estructurar operaciones mas complejas llamando a sus propios métodos, esto se ve en gran medida en la creación de los autómatas, puesto que en la lista de autómatas sencillamente se va llamando a los métodos del propio autómata para crearse y en ultimo momento agregar el autómata totalmente creado, dejando un código bastante limpio, y entendible.

ListaAutomatass

La función principal de esta clase es almacenar los objetos AP que se vayan creando a lo largo de la ejecución del programa, pero, porque existe una clase para esto y no se hace sencillamente con una de las listas predefinidas de Python, dicha clase contiene métodos especiales que ayuden en el desarrollo del programa. Por ejemplo, la creación de los AP se realiza desde esta clase, para luego agregarlo a la lista interna que esta maneja, dicha creación como se mencionó anteriormente se basa en la llamada de métodos del autómata para poco a poco ir creándose hasta llegar a ser agregados en la lista, algo similar se realiza para la carga de archivos, pero masificado y sin esperar respuestas del usuario. Esta clase también tiene los métodos necesarios para listar los autómatas y mostrarlos uno a uno.

Gramática

Por medio de esta clase se crean los objetos de tipo gramática los cuales son el pilar del primer módulo. Los atributos también cambian su nombre, aunque en esencia siguen siendo los mismos, en el caso de las gramáticas tenemos Terminales, No terminales que es una lista de objetos NoTerminal, Aceptación, No Terminal inicial y el nombre de la gramática. Al igual que los AP tiene sus métodos para conformarse, hacer sus reportes, generar sus archivos, etc.

Lista Gramáticas

Al igual que la lista de APs esta clase se encarga del almacenamiento de las gramáticas con métodos extra que aportan a la funcionalidad del programa. Comparte en su gran mayoría los métodos que tiene la lista de autómatas, obviando métodos especiales para estos y agregando métodos que solo funcionan en la gramática, como por ejemplo el cargar un archivo con recursividad, en este se vale del método de detección que tienen las gramáticas, para poder detectar las producciones y a si tratarlas de manera especial para luego agregarlas.

Listas

```
def ExisteAFD(self, AFD):...

def EstaVacio(self):...

def EliminarAFD(self, nombre):...

def Espacios(self):...

def Agregar(self, AFD):...

def MostrarAPs(self):
    for afd in self.automatas:
        print(afd.nombre)

def CargarAFDs(self, ruta):...

def ImprimirAutomata(self, EnCreacion):...

def GenerarReportes(self):...
```

Existe AP/Gramática

Se le pasa como parámetro un nombre y verifica si ya existe un AP/Gramática con este nombre, devolviendo valores Booleanos Esta vacía Por medio de este método se verifica si la lista está vacía o ya tiene algún valor cargado

Eliminar AP/Gramática

Se le pasa como parámetro un nombre y por medio de este busca el objeto para eliminarlo Agregar Se inserta a la lista un nuevo AP/gramática

Mostrar APs/Gramáticas

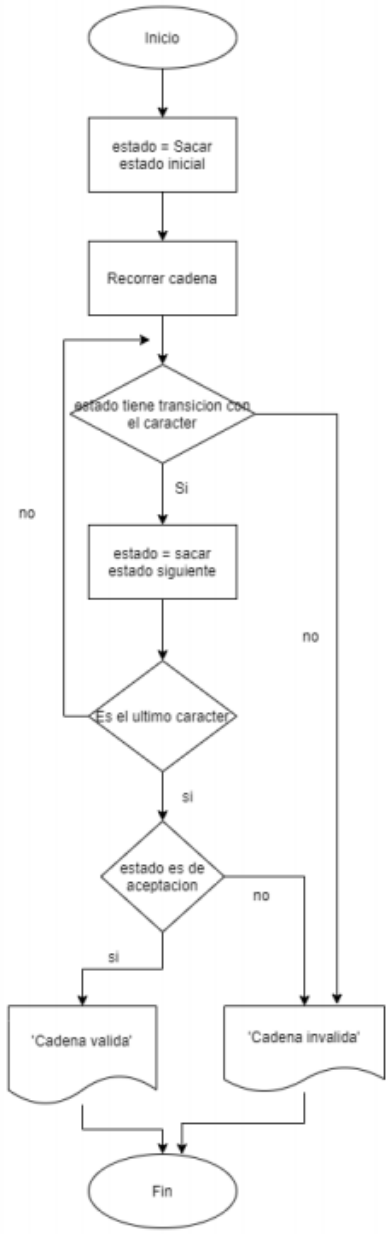
Por medio de este método se enlistan los nombres de las gramáticas/autómatas que estén cargados en ese momento en el sistema Imprimir autómata/gramática Imprime los datos del autómata o gramática que se le pase como parámetro

Cargar APs/Gramáticas

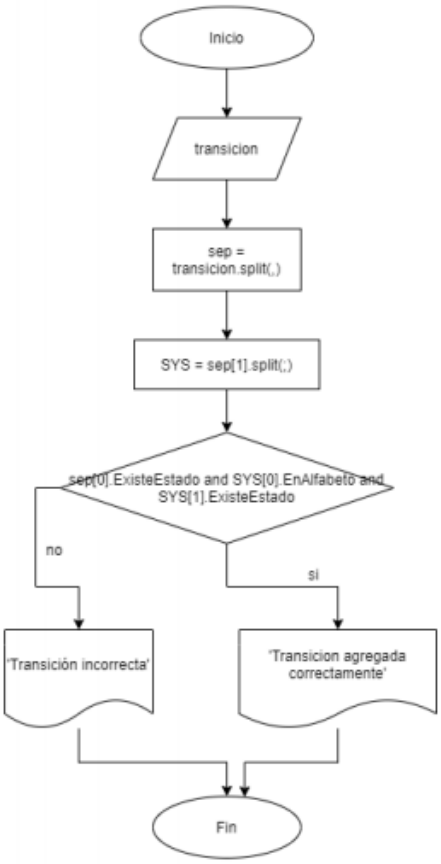
Por medio de este método se lee un archivo de texto plano con extensión. glc el cual contendrá toda la información necesaria de uno o varios autómatas/gramáticas las cuales serán cargadas al sistema Devolver AFD/gramática Por medio del nombre se busca un objeto y este se retorna.

Flujo de métodos principales

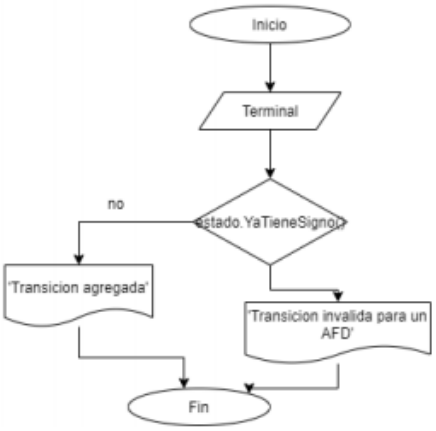
Validacion de cadenas

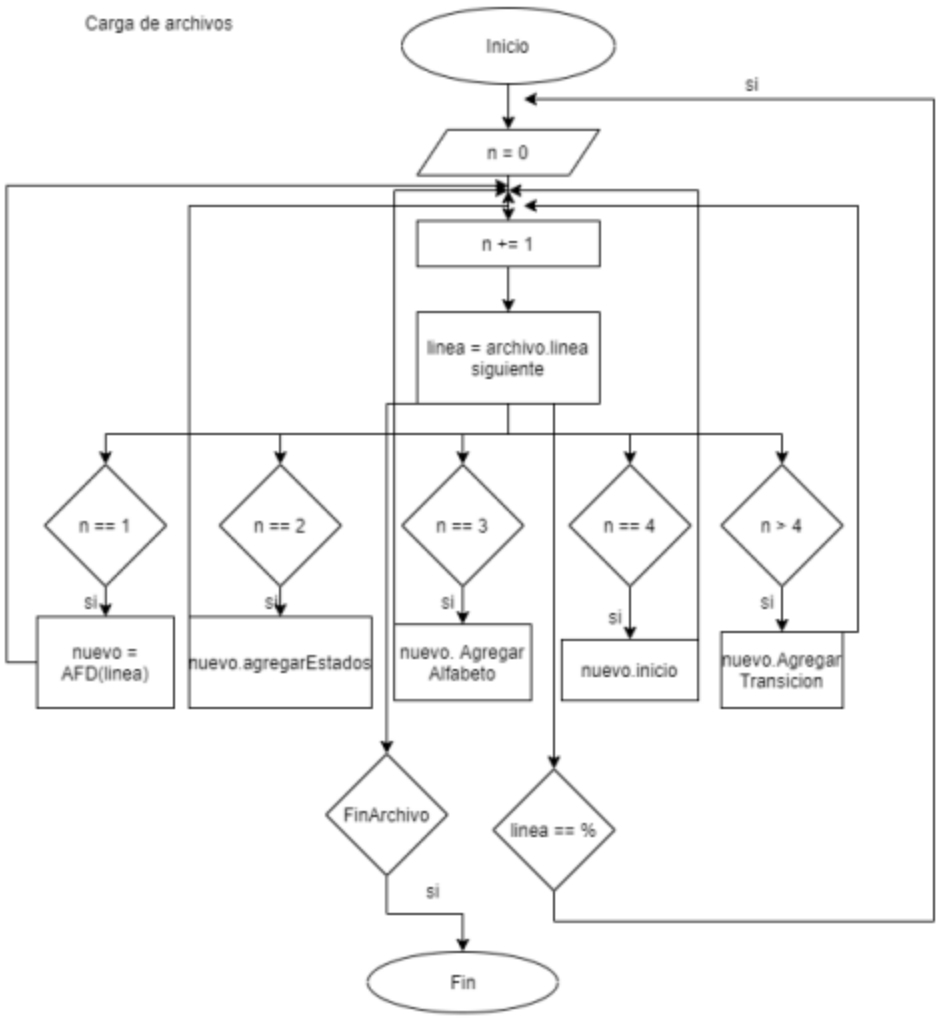


Verificacion transiciones validas



Verificar transiciones solo a un No Terminal




Carga de archivos
Inicio
si
n = 0
n += 1
linea = archivo.linea siguiente
n == 1
n == 2
n == 3
n == 4
n > 4
si
si
si
si
si
nuevo = AFD(linea)
nuevo.agregarEstados
nuevo. Agregar Alfabeto
nuevo.inicio
nuevo.Agregar Transicion
FinArchivo
linea == %
si
Fin