UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

ESCUELA DE CIENCIAS Y SISTEMAS

LABORATORIO DE ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1

SECCION B

PROYECTO 1

MANUAL TECNICO

DENNIS MAURICIO CORADO MUÑÓZ

CARNET: 202010406

CUI: 3032329780108

GUATEMALA, GUATEMALA, 08/03/2021

Introducción

Los procesadores de lenguajes o también conocidos como compiladores son programas que transforman programas escritos en un lenguaje a otro. Un tipo muy común de procesador de lenguaje es el intérprete, este procesador en vez de producir un programa destino, el intérprete da la apariencia de ejecutar de forma directa las operaciones especificadas.

Para el desarrollo del presente proyecto se realizó un intérprete utilizando el lenguaje de programación Java, haciendo uso de las herramientas de JFlex y Cup para poder generar lo que son las expresiones y gramática, para así poder dar solución al problema planteado.

Índice

Objetivos

Objetivos Generales

- Aplicar los conocimientos adquiridos durante el curso de organización de lenguajes y compiladores 1, así como del laboratorio impartido, para poder desarrollar una solución óptima al problema planteado, permitiendo generar análisis por medio del método del árbol.

Objetivos Específicos

- Reforzar conocimientos posteriores adquiridos del método del árbol para generar un AFD.
- Poner en práctica y reforzar el concepto del método de Thompson de expresiones regulares y generar un AFN.
- Realizar un reconocimiento de lexemas mediando el uso de AFD’s

Requisitos del Sistema

Para el correcto funcionamiento de la aplicación se deben tener en cuenta algunos requisitos mínimos bastante específicos, dentro de los cuales tenemos:

- Procesador – Intel Pentium o superior.
- RAM – 1GB de RAM o superior.
- Espacio en Disco – 1GB libres en disco.
- Sistema Operativo – Windows, MacOS o Linux.
- Tener instalado el software de Java JDK versión 16 o superior.
- Tener instalado el software de Graphviz versión 2.48 o superior.

Estos requisitos garantizan el poder utilizar la aplicación de una manera correcta.

Lógica del Programa

Paquete de Analizadores

En este paquete como su nombre lo indica se realiza todo lo pertinente a los analizadores, tanto léxico y sintáctico, dentro de esta carpeta se encuentran más archivo de extensión '.java' que son los generados al ejecutar los comandos contenidos dentro del archivo compilar.

Lexico

Dentro de este archivo sin extensión se realiza lo que es el analizador léxico bajo la sintaxis de JFlex que es una herramienta que nos ayuda a realizar un analizador léxico optimo para las funciones que se pretendan. Dentro de este archivo se definen todos los patrones que serán aceptados por nuestro lenguaje, como números, cadenas, en otros.

Sintactico

En este archivo se realiza toda la definición de las reglas sintácticas que debe respetar nuestro lenguaje, por medio de una gramática libre de contexto bajo la sintaxis de la herramienta de analizadores sintácticos Cup, se definen todas estas reglas y en este mismo archivo se extrae la información que posteriormente es utilizada en el programa y se procesa según se desee.

Compilar.bat

Dentro de este archivo se dan simplemente 3 comandos, uno para poder activar lo que es JFlex, el siguiente para ejecutar el archivo jar de jflex y que al ejecutarlo realice el analizador léxico descrito en este mismo paquete, el ultimo comando es para ejecutar el parser de Cup.

Paquete 'objects'

BinaryTree

Como su nombre lo indica es una clase que contiene la estructura de datos de un árbol binario, el objetivo de este árbol binario es almacenar la información obtenida del analizador sintáctico y ordenarla a manera de crear un árbol parecido al que se realiza con el método del árbol para construir un AFD, la función es prácticamente la misma, dentro de este árbol se ejecutan funciones propias que nos permiten convertir el árbol binario a una imagen por medio de graphviz así como también nos permite realizar el método del árbol aplicando anulables, primera posición y última posición para posteriormente realizar lo que es el llenado de un tabla de siguientes.

Error_

Dentro de esta clase se manejan lo que son los errores, este objeto es un objeto que se utilizara de forma estática mas adelante para que su información no desaparezca de manera inmediata, acá se almacena toda la información pertinente a los errores tanto léxicos como sintácticos.

Intervalo
Al igual que el error este objeto se guardará de manera estática en una lista que será descrita posteriormente, el objetivo de esta clase es de guardar los intervalos descritos en la entrada y darles un formato más legible para posteriormente ser utilizados en distintas partes del código.

LinkedListCadena
En esta clase se crea lo que es una lista simplemente enlazada, o al menos un molde parecido a ello ya que para la creación de esta clase y las posteriores listas enlazadas se realiza herencia de todos los métodos contenidos dentro un ArrayList, la diferencia de esta lista es que guardara objetos de tipo cadena, cadenas que vienen en el archivo de entrada para ser analizadas y ver si cumplen con un autómata ya definido.

LinkedListError
Dentro de esta clase al igual que la anterior se guardan lo que son los errores encontrados en la entrada para posteriormente mostrarse en la interfaz y un archivo HTML que contendrá la información de estos errores.

LinkedListParams
Acá se guardan objetos de tipo intervalo, los intervalos se almacenan ya que con estos y el autómata se realiza el análisis para verificar si la cadena presentada es válida, de ser una entrada valida esta se muestra en un archivo JSON y en el área de texto salida que contiene la interfaz.

LinkedListSiguientes
Dentro de esta lista se realizan mas procesos que solo almacenar los siguientes, una vez llenada la lista con objetos de tipo Siguientes, se crean más métodos y variables globales, que permiten desde crear la tabla de siguientes utilizando la herramienta graphviz, hasta llenar lo que son las transiciones, para las transiciones se crea un algoritmo bastante complejo y hasta cierto punto enredado que requirió de bastante tiempo de análisis, en el momento que se crean las transiciones básicamente lo que se realiza es enviar la información del nodo raíz, crear los primeros estados y realzar verificaciones para ver si todos los estados ya cuentan con su propia cantidad de transiciones, una vez verificado esto se termina de crear las transiciones y se procede a llenar otra lista que contendrá a esta misma y el nombre de la expresión regular.

LinkedListTransiciones
Esta lista como su nombre lo indica guardara objetos de tipo transición, esta lista es llenada al momento de realizar las transiciones en la lista de siguientes, una vez llenada la lista se puede observar que la clase contiene más métodos, uno de estos métodos su función es crear la tabla que contiene las transiciones y poder generar una imagen donde se pueda observar esto, también se tiene un método que genera el autómata finito determinista.

LinkedListTrees
Acá se guardan los arboles binarios, el objetivo de guardar los arboles binarios es que si viene más de una expresión regular en la entrada, el almacenar este árbol en una lista hace que podamos tener acceso a todos ellos sin perder su información debida.

ObjCadena

Este es un objeto que como su nombre lo indica guarda la información de la cadena que va a ser analizada, este objeto es el que se guarda dentro de la lista simple 'LinkedListCadena'.

Siguientes

Este objeto contiene información bastante importante debido a que de este se parte para la creación de la tabla de siguientes, transiciones y autómatas, este objeto se almacena en la lista simple anteriormente descrita para que posteriormente todos los diferentes objetos creados se puedan utilizar.

TablaSiguientes

Este objeto tiene como función contener lo que es el nombre de la expresión que esta siendo analizada, la lista de los siguientes y el nodo raíz del árbol binario, esto con el objetivo de posteriormente cuando se creen las transiciones pueda ser utilizada esta información.

TablaTransiciones

En la tabla de transiciones su función es similar a la tabla anteriormente descrita, su objetivo es guardar la información de las transiciones para poder generar la tabla de transiciones y en base a lo recolectado también generar el autómata en las anteriores clases.

Transicion

En este objeto se guarda lo que es la información de una transición, es decir su estado inicial, por ejemplo S0 o S1, su transición destino, el carácter con el que va a moverse de una transición a otra, conocer si es estado de aceptación en el que estoy y también la lista de siguientes de cada transición.

Paquete 'proyecto1'

Administrador

Como su nombre lo indica la función de esta clase es fungir como un administrador para la interfaz y así evitar saturar la clase Gui con elementos extras que pueden ser manejados desde otra parte del código y brindar así más modularidad.

Gui

Acá como su nombre em ingles indica, es donde está la graphic user interface, la interfaz con la que el usuario va a interactuar.

Proyecto1

En esta clase se encuentra el main desde la cual únicamente se instancia la interfaz grafica para que esta inicie.

Paquete 'references'

Dentro de este paquete se busca terminar de consolidar el patron StandAlone cuyo objetivo es mantener la referencia de los objetos y no perderla mientras el código está en ejecución, ahorrando así variables y más líneas de código.

Cadena
Acá se usan métodos estáticos desde los cuales se instancian objetos como la lista enlazada de cadenas, los objetos serán instanciados una vez se llame al método analize

Conjunto
Al igual que el anterior se hacen usos de métodos estáticos para mantener las instancias y la diferencia acá es que se instancia una lista enlazada de conjuntos.

ErrorRef
Al igual que el anterior se hacen usos de métodos estáticos para mantener las instancias y la diferencia acá es que se instancia una lista enlazada de conjuntos.

SiguienteRef
Al igual que el anterior se hacen usos de métodos estáticos para mantener las instancias y la diferencia acá es que se instancia una lista enlazada de siguientes.

Transitions
Al igual que el anterior se hacen usos de métodos estáticos para mantener las instancias y la diferencia acá es que se instancia una lista enlazada de transiciones.

Tree
Al igual que el anterior se hacen usos de métodos estáticos para mantener las instancias y la diferencia acá es que se instancia una lista enlazada de árboles binarios.