Universidad San Carlos de Guatemala

Facultad de Ingeniería

Organización de Lenguajes y Compiladores 1

Proyecto 1

Juan Pablo Rojas Chinchilla

201900289

Primer Semestre 2021

Analizadores

Analizador Léxico:

* Descripción: El analizador léxico verifica que todos los caracteres de entrada estén dentro de los caracteres posibles a utilizar en el lenguaje.

|  |  |  |
| --- | --- | --- |
| Método | Variables | Descripción |
| Creación de analizador léxico | Tokens,  Lexemas,  Líneas,  Columnas | El analizador léxico debe cumplir con la tarea que tiene que es el de comprobar carácter por carácter si este pertenece al lenguaje, este proceso se realiza con JFLEX. |

Analizador Sintáctico:

* Descripción: El analizador sintáctico verifica que las cadenas ingresadas en el programa tengan un sentido, y que por ejemplo que no vengan elementos sin sentido, como un identificador asignando a otro identificador.

|  |  |  |
| --- | --- | --- |
| Método | Variables | Descripción |
| Creación del analizador sintáctico | Terminales,  No Terminales,  Líneas,  Columnas,  Gramática (Se especificará más adelante) | El analizador sintáctico debe cumplir con la tarea de verificar el orden de la entrada de tokens, este proceso se realiza con JCUP. |

Método del Árbol

Nodo Inicial:

* El nodo inicial es el nodo que hace el analizador sintáctico, este nodo solo contiene el valor de este mismo y quiénes son sus hijos.

|  |  |  |
| --- | --- | --- |
| Método | Variables | Descripción |
| Constructor | Value,  Izquierdo,  Derecho | El constructor se encarga de crear un objeto e indicar los valores que tendrán sus nodos hijos, esto en caso de ser una operación. |
| Recorridos | Izquierdo,  Derecho | Estos recorridos se hacen para obtener los valores de este árbol de una manera determinada, se utilizan para la graficación del método de Thompson y para obtener datos de toda la expresión regular. |

Nodo Árbol:

* El nodo que tiene el árbol a diferencia del nodo inicial contiene mas datos, como quienes son los siguientes si este nodo es anulable y demás interacciones necesarias, este es el nodo que se utilizara para procesos siguientes.

|  |  |  |
| --- | --- | --- |
| Método | Variables | Descripción |
| Constructor | Lexema,  Type,  Numero,  Izquierda,  Derecha,  Leaves,  Table | Este método genera un nodo con todos los datos que se tengan en ese momento, en caso de que no tengan algún atributo desde antes de su construcción este se mandara como null. |
| getNodo | Izquierdo,  Derecho | Retorna todos los nodos en un orden especifico (preorden). |
| Follow | Izquierdo,  Derecho | Hace un recorrido en todos los nodos (preorden), para después de esto llenar su lista de siguientes y anteriores, correspondientes de su valor, solamente los nodos con valor “.”, “Kleene”, “+” |

Siguientes:

* Esta parte del método es la encargada de establecer los siguientes de cada hoja y rama del árbol, esto es necesario para establecer las transiciones y con esto poder graficar y crear un AFD (Autómata Finito Determinista).

|  |  |  |
| --- | --- | --- |
| Método | Variables | Descripción |
| Append | Nodo,  Lexema,  Tabla | Este método agrega a la tabla de siguientes los siguientes de cada lexema, ya que cada lexema que sea una hoja debe tener siguientes “específicos” para los cuales después se puede hacer una transición |
| Next | Nodo,  Tabla | Este método solo consulta los siguientes, haciendo un recorrido, después de esto estos datos se agregan a la tabla haciendo un append. |

Transiciones:

* Esta parte del método es la encargada de establecer los nodos y las transiciones que estos tendrán además de mostrar como se conectaran los nodos, con que caracteres, conjuntos, cadenas, etc.

|  |  |  |
| --- | --- | --- |
| Método | Variables | Descripción |
| Transición | Inicial,  Transición  Final | Este método construye una transición con el estado de salida al estado de llegada con un valor en este objeto. |
| Comparación |  | Este método compara la transición que se genero al inicio para ver si esta transición es correcta, en caso este no es correcto retorna un false, en caso este sea correcto se pasa a una tabla para graficarlo después. |

Graficadores

Método Árbol:

* Para este método se cuentan con dos graficadores, estos son los del AFD y el árbol en si demás de las tablas que se abordaran más adelante.

|  |  |  |
| --- | --- | --- |
| Método | Variables | Descripción |
| Crear cuerpo | Cuerpo | Este método genera un .dot, el cual tiene la estructura para hacer un árbol, este método gestiona lo necesario para crear un árbol en graphviz, mostrando todos los atributos del árbol, como los siguientes, anulabilidad etc. |
| Crear imagen |  | Este método solo crea una imagen partiendo del archivo .dot creado antes, esto lo hace por medio de un runtime.exe() |

Tablas:

* En este apartado se encontrarán las tablas, estas son la tabla de errores, la tabla de siguientes y la tabla de transiciones.

|  |  |  |
| --- | --- | --- |
| Método | Variables | Descripción |
| Crear cuerpo |  | Aunque en el caso de las tablas se utiliza una variable estática para generar el cuerpo, así solo se va a agregando fila por fila, este tipo de grafico se hace por medio de HTML. |
| Crear imagen |  | Este procedimiento se utiliza igual que el anterior, este se hace por medio de un runtime.exec |

Método Thompson:

* Para el método de Thompson no se requiere prácticamente ningún procesamiento, por lo cual solo se graficará de cierta manera según los nodos que traiga.

|  |  |  |
| --- | --- | --- |
| Método | Variables | Descripción |
| Crear cuerpo |  | Este método no es necesario procesamiento entonces, este solo necesita las instrucciones para crear un grafo. |
| Crear imagen |  | Este procedimiento se utiliza igual que los anteriores, se hacen por medio de runtime.exec. |

Interfaz Grafica

Elementos Utilizados:

1. Botones
2. Tree View
3. Menu Button
4. Text Field
5. Image View

Analizador de REGEX

Conjuntos:

* Para los conjuntos deben recolectarse datos desde el archivo .cup y también deben de procesarse de cierta manera, para que así sea más fácil encontrar los caracteres que componen este conjunto.

Nodos:

* Los nodos del analizador de expresiones regulares, son diferentes a los demás, ya que estos pueden tener una cantidad diferente de siguientes entre ellos, además de tener diferentes atributos, los cuales favorecen el desplazamiento entre ellos, estos nodos se forman con el AFD generado previamente con el método del árbol.

|  |  |  |
| --- | --- | --- |
| Método | Variables | Descripción |
| Arreglar mapa | Nombres,  Nodos,  Conjuntos | Este método hace un HashMap para poder buscar los conjuntos por nombre, este retorna un arraylist |
| Find By Name | Nombre,  Nodo | Encuentra un nodo entre todos los nodos ya creados, este método es utilizado para encontrar algún nodo, principalmente para encontrar el nodo siguiente |
| Evaluar Expresiones | MapaEntero (conjuntos y cadenas) | Método que reúne todas los demás métodos para poder llamarlos en otra clase |
| Escribir JSON | Cuerpo (cadena del JSON) | Este escribe un cuerpo la estructura que debe llevar el JSON, este lleva una estructura en la cual se muestra el nombre del regex, la cadena evaluada y si es aceptada o no |
| Evaluación | Variables estáticas  (Nombres, Nodos, Conjuntos, Cadenas) | Este método solo evalua las expresiones, buscando en cada uno de los caracteres en los conjuntos y las cadenas posibles |

Creación de JSON:

* La creación del JSON utilizado es un procedimiento parecido a la creación de imágenes que se hacen en el graficador.

Métodos separados

Separación:

* Este método separa los diferentes elementos de una expresión regular (cadenas, conjuntos, caracteres especiales).

Comprobación palabra:

* Este método comprueba si la palabra buscada se encuentra en el arreglo creado por el método de separación.

Conteo tamaño:

* Este método devuelve el numero de elementos que contiene el arreglo creado por el método de separación.