Segundo Proyecto: Lenguajes Libres de Contexto

Diego Vázquez

Matricula: A00226803

En este documento, explicaré la arquitectura del software que he programado para obtener la solución del proyecto que elegí. En mi caso, elegí diseñar el convertidor de gramáticas a Forma Normal de Chomsky, Forma Normal de Greibach, y crear el Autómata de Pila No Determinista de la gramática otorgada.

Mi software consta de dos clases programadas en Java. La primera representa un símbolo no terminal en el cual guardamos el símbolo en cuestión y, en un ArrayList, tenemos todas las producciones generadas por el símbolo. Por otro lado, la segunda clase que se creó es el programa principal en el que se realizan todas las conversiones necesarias.

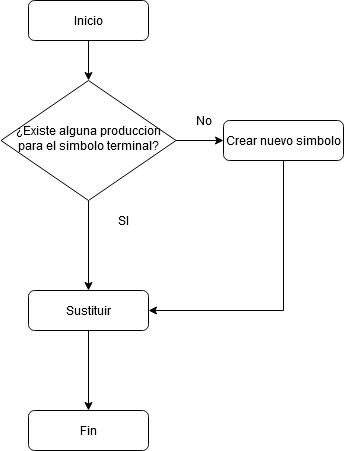
Después, de leer la entrada mediante un archivo de texto, el algoritmo que se realiza se divide en los siguientes pasos

Diagrama

Descripción generada automáticamente

En cada uno de los símbolos, se revisa si contienen alguna épsilon producción. Si es así, son guardados en una lista. Después, buscamos que símbolos no terminales producen alguno de los generadores que tiene alguna épsilon producción, incluso si ese símbolo es el mismo. Se agregan las producciones con la sustitución adecuada, y se borran las épsilon producciones.

A continuación, se busca saber cuáles símbolos no terminales tienen alguna producción unitaria. Si es así, se agregan todas las producciones del símbolo de la producción unitaria y se borra dicha producción



Para terminar con la Forma Normal de Chomsky, debemos revisar si hay algún símbolo que produzca cada uno de los símbolos terminales. Si no existe, crearemos un nuevo generador con la producción.

Continuando con la Forma Normal de Greibach, primero, guardamos el estado actual de la gramática para conocer cuántos símbolos no terminales existen. Después, verificamos que todas las producciones se encuentren en orden. Si no lo están, realizamos la sustitución, agregamos la nueva producción, borramos las “viejas” producciones y volvemos a verificar. Una vez que estén en orden, realizamos la sustitución en reversa y la sustitución en los nuevos símbolos generadores, para estos dos pasos hacemos la comparación entre como estaba la gramática antes y como está ahora. De esta manera, es posible discernir los símbolos nuevos de los que ya existían en la gramática

Diagrama

Descripción generada automáticamente

Finalmente, se construye el autómata de pila. En mi caso, utilice la librería JGrapht, para crear un grafo. El grafo está hecho por tres nodos que representan los tres estados base del autómata (estado inicial, estado de trabajo y estado final). La transición de q0 a q1 se realiza insertando el símbolo inicial a la pila sin consumir ningún carácter de una cadena. Las transiciones de q1 a si mismo funcionan de la siguiente manera: Si la producción es solamente un carácter, sabemos que solo es un símbolo terminal por lo que debemos consumir el carácter en la cadena y el símbolo generador respectivo y no insertar nada en la pila. De lo contrario, debemos insertar en la pila todos los símbolos no terminales de la producción.

Todo lo anterior, se despliega en la consola. Las formas normales se muestran como se suele expresar una gramática.

Símbolo -> producción | producción

En cuanto al autómata, la librería despliega el grafo de esta forma

([q0, q1, q2], [(q0,q1), (q1,q1), (q1,q2)])

El primer conjunto que vemos son los nodos, y el segundo es un conjunto de pares ordenados que representan las aristas. Además, se visualiza las transiciones que se realizan en los nodos como se suele ver en un autómata de pila:

Carácter por consumir, símbolo en la pila / símbolo a insertar en la pila

Con todo esto, se forma el convertidor de gramáticas, el cual realiza un proceso bastante largo, pero divido en varios pasos claros siguiendo los procedimientos impartidos en clase.

Librería JGrapht: <https://jgrapht.org/>

Para utilizarla se descarga el .zip que se encuentra en la página y se extrae. Después, en la carpeta que se cree (jgrapht-1.5.0) se entra en lib, se copia el archivo jgrapht-core-1.5.0.jar. Luego, en Eclipse, se crea una nueva carpeta (en mi caso la llame lib), y se pega el archivo y finalmente, lo seleccionamos, damos clic izquierdo y se eligen las opciones “Build Path” y después “Add to Path”.

Link Video: https://youtu.be/j7JivhU7HMY