Grammar Analyzer

Luis Alejandro Lara
Carlos Alejandro Arrieta
Grupo C-312

1 Instrucciones de ejecución

El entorno visual del proyecto está implementado en la plataforma *streamlit*. Por tanto, para ejecutarlo es necesario tener instalada esta herramienta. El archivo a ejecutar se llama $StreamlitVisual$, por tanto la ejecución en consola del mismo sería *streamlit run* $StreamlitVisual.py$.
Con esta ejecución en el buscador web se mostrarán una caja de texto y un selector de *parser*. Primero se selecciona el *parser* con el cual se quiere analizar la gramática y luego se escribe la gramática en la caja de texto.
A la hora de escribir la gramática hay que hacerlo de la siguiente forma:
Las producciones tienen la forma: $A --> a$;

- Toda producción debe finalizar en punto y coma (;).
- El símbolo de derivación ($-->$) consta de dos guiones($--$) seguidos por un símbolo de mayor ($>$).
- Cada *token*, sea un terminal (a), un no terminal (A), el símbolo de derivación ($-->$), o el punto y coma (;) , debe estar separado por uno o más espacios en blanco o por saltos de línea.
- Todo símbolo de la gramática que se encuentre en la cabecera de al menos una producción va a ser tomado como no-terminal, y todo símbolo de la gramática que se encuentre en el cuerpo de una producción (a la derecha del símbolo $-->$) y no se encuentre en la cabecera de ninguna producción, será tomado como terminal.

Luego de escrita la gramática, se debe pulsar $ctrl + enter$, o pulsar el botón *run* que aparece en el menú de la esquina superior derecha de la pagina para *parsear* la gramática.
Al ejecutar, debajo del selector de *parser* debe de salir un texto con el resultado de *parsear* la gramática escrita con el *parser* seleccionado. Este texto tiene la siguiente estructura:

- $Firsts$: El conjunto de los $firsts$ calculado. Este conjunto tiene la misma estructura de los $firsts$ implementados en clase práctica.
- $Follows$: El conjunto de $follows$. Tiene la misma estructura a la utilizada en clase práctica.
- Regular: Muestra si la gramática es regular o no.

- Gramática sin recursión izquierda inmediata ni prefijos comunes: Muestra los terminales, no terminales y producciones de la gramática después de haberle quitado los prefijos comunes y la recursión izquierda inmediata.
- Conflictos: En caso de haber encontrado conflictos, muestra que la gramática no es *parseable* por ese *parser*, seguido por los conflictos encontrados

2 Implementación

El único algoritmo implementado en el proyecto que no fue impartido en clases es el utilizado para encontrar cadenas de conflicto.
Para ello, el algoritmo se apoya en el autómata y en la *parsing* table construidos por el *parser*.
El objetivo es primeramente formar una cadena de terminales y no terminales que de conflicto en la gramática, para después a partir de ella suprimir los no terminales por terminales y lograr una cadena de terminales que funcione como cadena de conflicto. A esta cadena de conflicto se le llamará s.
Lo primero que se hace es buscar en la tabla el estado e1 del autómata en el cual se encontró el conflicto. Teniendo ese estado se busca un camino posible en el autómata que vaya desde el estado inicial e0 hasta e1, y se guardan todos los símbolos que se hayan requerido en las transiciones de estado para cruzar este camino. En esta búsqueda se verifica en todo momento que al estar formando un posible camino, este no pase dos veces por un mismo estado, asegurando de esta forma que no forme un ciclo infinito.
Luego del estado actual e1 se busca una posible forma de llegar hasta el estado inicial e0, pero esta vez realizando 2 posibles acciones: sumar un posible símbolo a la cadena o efectuar una acción Reduce. Destacar que se busca solamente llegar al estado e0 habiendo reducido el símbolo inicial de la gramática (S), ya que luego con este símbolo existe una transición hacia un estado final. En esta búsqueda también se verifica que no se revisen 2 estados en un mismo camino.
Luego de haber ejecutado este procedimiento en la cadena s se tiene una lista de terminales y no terminales, tal que a la hora de *parsearlos* el autómata pasa por un estado de conflicto.
El próximo paso es sustituir los no terminales por terminales, lo cual se hace en un recorrido por las posibles derivaciones de cada uno de esos no terminales, acumulando todos los terminales derivados.
Luego de esta sustitución, ya se tiene una cadena solo de terminales que constituye una cadena de conflicto, y esta es la que se devuelve.