TPE LEX

Analizador de gramáticas

Autómatas, Teoría de Lenguajes y Compiladores

Integrantes: Augusto Nizzo Mc Intosh
Gonzalo M. Rey

Indice

Introducción

Para este trabajo, se realizó un analizador léxico que permitiera validar gramáticas, determinando si son regulares, y si están alineadas a izquierda o a derecha.
Para dicho analizador, se utilizó el analizador sintáctico LEX sobre el lenguaje C en un sistema tipo Unix.

Consideraciones realizadas

Para el desarrollo de este Trabajo Práctico, se tuvieron en cuenta las ventajas suministradas por el analizador sintáctico LEX, el cual nos permitió tomar un archivo y determinar si el mismo se adapataba en un principio, a las características especificadas en el trabajo, facilitándonos identificar cada uno de los elementos de la gramática (símbolos no terminales, símbolos terminales, etc...).

Descripción del desarrollo del TP

En primer lugar, se buscó poder analizar casos que se ajustaran a la sintaxis estipulada en el enunciado, sin tener en cuenta las gramáticas incorrectas, espacios, tabulaciones o fines de línea, con el fin de generar unas expresiones gramaticales claras y sencillas.
Nuestra estrategia fue lograr interpretar la descripción de una gramática completa sin errores, para luego analizarla de manera segura.

<INITIAL>{GRAMMAR_NAME}/={OPENING_GRAMMAR}"{"{SNTs}"},{"{STs}"},"{SI}",
{"{PRODs}"}"{CLOSING_GRAMMAR}

Donde GRAMMAR_NAME se corresponde con el nombre de la gramática, SNTs se corresponde con el conjunto de los símbolos no terminales, STs se corresponde con el conjunto de los símbolos terminales, SI símbolo inicial, y PRODs con el conjunto de las producciones.

Una vez obtenida una gramática léxicamente correcta, para la identificación de cada uno de los componentes de la gramática, se terminó utilizando una máquina de estados, que nos permitiera estar al tanto de que variables estábamos analizando.

Luego, como solución al tema de los espacios, tabulaciones y fines de línea, se agregó al desarrollo del trabajo, un preprocesador que se encargara de adaptar la entrada a la que el parser que se tuvo en cuenta en primera instancia aceptara. Este preprocesador acepta la entrada de comentarios de la forma /* comentario */ para describir y comentar las gramáticas.

Una vez realizados estos analizadores, se agregó al trabajo la lógica que permitiera analizar la validez de la gramática, teniendo en cuenta sus componentes (símbolos iniciales, terminales, etc...).
En primera instancia, se validaba la gramática verificando la existencia de las ocurrencias de los símbolos de las producciones dentro de los conjuntos de símbolos iniciales o terminales. De la mísma forma, se verificaba que el símbolo inicial estuviera dentro de las producciones.
Una vez validadas estas instancias, se analizaba la regularidad a izquierda o a derecha de la gramática, evaluando en cada producción que el antecesor tenga un solo símbolo, y el sucesor cero (interpretado como lambda), uno que debería ser exclusivamente un símbolo no terminal, o dos símbolos de los cuales deberían respetar todos una regularidad a izquierda o a derecha; ya que en caso de no hacerlo, no sería una gramática regular.

Dificultades encontradas en el desarrollo del TP

Debido a la ausencia de estructuras provistas por el lenguaje C, se tuvo que recurrir a la implementación bajo demanda de una lista linealmente encadenada, con un iterador sencillo para ir almacenando las gramáticas encontradas y una biblioteca de strings alocados en el heap para facilitar el uso de la memoria.

Futuras y posibles extensiones

En primer lugar, el analizador podría aceptar gramáticas regulares, válidas, pero desnormalizadas, y retornar las mismas normalizadas ya sea aliadas a izquierda o a derecha.
De la misma forma, se podría pasar de una gramática regular izquierda a una derecha, o viceversa.
Y también, dado que la construcción de la gramática regular es algorítmica, se podría generar un autómata que represente gráficamente a la gramática, y a su vez, también permitir su exportación a formatos como png, PDF, etc.