Procesadores de Lenguajes.

Práctica 4ª: Analizador de un lenguaje.

Esta práctica ocupará dos sesiones. Se trata de realizar el diseño y la implementación de un reconocedor del léxico de un lenguaje sencillo. Se propone el uso de Java como lenguaje de implementación.

Corresponde a la parte esencial de la programación del analizador léxico en un compilador de un lenguaje de programación, salvo por una cuestión de escala, ya que nuestro lenguaje es de "juguete" frente a un verdadero lenguaje de programación. Sin embargo, nos va a permitir entender las dificultades a resolver a la hora de construir un analizador léxico real.

Partimos de la definición del léxico. Se trata de un subconjunto del lenguaje de las palabras sobre el alfabeto {a,b,c}. Consideramos únicamente cinco categorías léxicas, que vienen dadas por los patrones (expresiones regulares):

a	cero
b	uno
aabb	dos
((a)*)((b)+)	tres
(c)+	cuatro

El objetivo es, dada una secuencia que sólo contenga letras del alfabeto {a,b,c} (por simplificar obviamos el asunto del tratamiento de errores léxicos), desarrollar una aplicación en Java que "proporcione" la correspondiente secuencia de tokens, siguiendo las mismas condiciones que se imponen en el análisis léxico:

búsqueda del lexema más largo

en caso de ambigüedad, primer token según el orden de definición.

El primero de los pasos, que no vamos a “automatizar”, es el paso desde las expresiones regulares que definen los patrones léxicos hasta la construcción de un autómata determinista que reconozca los tokens. Debido a la sencillez del lenguaje, no se requiere aplicar métodos de simplificación de autómatas: reducción del número de estados, eliminación de estados inalcanzables, etc.
Lo realizado en este primer paso se plasmará en la parte final del diseño, cuando queramos construir un analizador léxico para este lenguaje concreto.

La programación incluye:

Clase AutomataFinito

Clase AutómataFinitoMatriz

Sin entrar en detalles, la primera de ellas será tratada como clase abstracta (la función de transición de un autómata queda como método abstracto) aunque implemente métodos para determinar: pertenencia de cadenas al lenguaje, decidir si un estado es final, etc.
La segunda de ellas será una implementación de un autómata finito, usando una matriz para implementar la función de transición.

Clase AnalizadorLex

Desde un autómata y una cadena (por simplificar String) proporciona los tokens conforme son solicitados, e informa del final del tratamiento (revisar la clase Java StringTokenizer). El método más importante de esta clase es “nextToken”, que proporcionará el siguiente token de la secuencia. Se recomienda hacer también una clase Token, pensando que un token consta de un identificador (los propios nombres dados a los tokens en el enunciado pueden servir como identificadores) y un lexema.

De este modo, queda para “la clase main” todo lo directamente relacionado con el lenguaje concreto que queramos analizar.