Ejercicios Laboratorio 1

Lenguajes de programación y procesadores de lenguaje

26 de Septiembre de 2024

En el Campus Virtual de la asignatura en el apartado de "Laboratorio 1: Análisis Léxico" encontraréis un enlace para obtener JFlex como archivo ".jar" y otro para obtener el código Java para generar el analizador léxico en "alex.zip". También encontraréis un ejemplo de especificación del léxico usando JLex "ejemplo.l" y varios ejemplos de entrada para el procesamiento final en "input.txt", "input_1.txt" y "input_erroneo.txt". Descargad todos estos archivos en una carpeta y descomprimid "alex.zip".

Para más información o versiones más recientes de JLex visitad `https://www.jflex.de/`.

Podéis instalar JLex en vuestra cuenta usando el archivo ".jar" o lo podéis usar directamente en la línea de comandos. También podéis utilizar el entorno para desarrollo de código Java que prefiráis. Deberéis definir los caminos necesarios en el CLASSPATH o usar `-cp` al llamar a java.

Ejercicio 1: Primer ejemplo con JLex

En este primer ejercicio vamos a probar a generar un primer analizador usando JLex. Para ello vamos a usar la especificación dada como expresión regular en el ejemplo "ejemplo.l".

1. Para procesar el archivo `ejemplo.l` con JFlex, tenéis que ejecutar:

   ```
   $ java -cp jflex.jar jflex.Main ejemplo.l
   ```

   Esto generará un archivo `AnalizadorLexicoTiny.java` cuyo contenido debería coincidir con el archivo `alex/AnalizadorLexicoTiny.java` (si no es el caso, reemplazadlo por el nuevo generado). Echad un ojo a la salida de jflex y el fichero que generamos, podréis ver un poco de información sobre el autómata que estamos generando.

2. Compilad el código java de la carpeta "alex":

   ```
   $ javac alex/*.java
   ```

   y ya podéis usar el analizador.

3. En la versión del Main que se adjunta, se lee de la entrada de un fichero que se pasa por parámetro. El fichero `input.txt` contiene la siguiente entrada:

   ```
   evalua
   -456.12 * vacas + 1.34 * (ovejas + cabras)
   donde
   vacas = 567,
   ovejas = 456,
   cabras = 10
   ```

   Ejecutad:

   ```
   $ java alex.Main input.txt
   ```

   desde el directorio de trabajo y obtendréis los tokens de la entrada (puede ser necesario añadir `-cp .` si no tenéis "." en el CLASSPATH). Comprobad que los tokens y la unidad léxica a la que pertenecen son los esperados.

Probad también el resto de ejemplos input_1.txt e input_falla.txt. En el caso de este último, observad cómo se muestran los errores y pensad por qué nuestra sintaxis no acepta esas cadenas. Probad a meter otros errores y ver cómo se detectan.

Ejercicio 2: Variaciones

Vamos a probar a realizar pequeños cambios en la especificación JFlex y el código asociado. Por ejemplo, podemos ampliar la representación de números para admitir números en binario y hexadecimal con la notación `0b01101101` para binarios y `0x1faCd5` para hexadecimales. Otro posible cambio es permitir que los identificadores puedan comenzar por _ o que los números puedan ser 0 o ser decimales comenzando en 0.

Ejercicio 3: Lenguaje de Listas

Vamos a hacer un analizador léxico para el lenguaje `NumListLang`. Se trata de un lenguaje simple para definir y tratar listas heterogéneas de números. El siguiente ejemplo muestra el lenguaje (en la entrada podrá haber comentarios que van después de `//`):

```
L = [] // L contiene la lista vacía
L2 = [1,2,3] // L2 contiene una lista con 3 números
L3 = L#L2 // L3 es la concatenación de L y L2, es decir L2
m4 = [[[1,2],3],4] // m4 es una lista heterogénea
L5 = lreduce + m4 // L5 tiene un único elemento: la suma de los números de m4
```

```
L6 = lmap - 1 m4 // L6 se obtiene restando 1 a los números de m4
print L5 // Se imprime [10]
print L6 // Se imprime [[[0,1],2],3]
L7 = lfilter != 1 m4 // L7 es una copia de L4 eliminando los números iguales a 1
print L7 // Se imprime [[[2],3],4]
L8 = lfilter > 2 L7 // L8 contiene [[[],3],4]
L9 = (L#L2)#L3 // Podemos usar paréntesis
```

Los operadores que aparecen en el lenguaje son

`#, lreduce, lmap, lfilter, >, <, ==, !=, +, -, *, /`

También aparecen la instrucción de asignación (=) y de mostrar por pantalla *print*. Además, otros símbolos que pertenecen al lenguaje son los paréntesis, corchetes y la coma. Los comentarios se indican utilizando // y llegan hasta el final de línea. Es importante que guardéis los saltos de linea como unidades léxicas ya que tienen significado sintáctico: cada instrucción aparece en una línea distinta.

Debéis crear una nueva definición léxica con JFlex y modificar convenientemente el código java en `ClaseLexica.java` y `ALexOperations.java`. Utilizad el ejemplo del Ejercicio 1 como referencia, como veréis los cambios que debéis realizar son sencillos. Eliminad las unidades léxicas que ya no utilizamos (números, evalúa, ...).

Ejercicio 4: Patrones

Suponemos que nos pueden entrar palabras (formadas solo por letras mayúsculas y minúsculas), patrones (formados por letras mayúsculas y minúsculas y un carácter `'*'`) y cualquier otra cadena de caracteres (sin separadores). Dada una entrada, debemos extraer las palabras, los patrones y el resto de cadenas como unidades léxicas distintas (`PALABRA`, `PATRON` y `OTRO`) y para cada patrón indicar el número de palabras que le encajan: palabras que coinciden con el patrón reemplazando el `'*'` por una cadena de caracteres.

Así, con la entrada:

```
hola
ho*a
laz+dfg
ho*
ho*+sd
pista
+fdsghf
hoja
a*a
hoyo
ala
a
```

la salida debería ser:

```
[clase:PALABRA,fila:1,col:1,lexema:hola]
[clase:PATRON,fila:2,col:1,lexema:ho*a]
[clase:OTRO,fila:3,col:1,lexema:laz+dfg]
[clase:PATRON,fila:4,col:1,lexema:ho*]
[clase:OTRO,fila:5,col:1,lexema:ho*+sd]
[clase:PALABRA,fila:6,col:1,lexema:pista]
[clase:OTRO,fila:7,col:1,lexema:+fdsghf]
[clase:PALABRA,fila:8,col:1,lexema:hoja]
[clase:PATRON,fila:9,col:1,lexema:a*a]
[clase:PALABRA,fila:10,col:1,lexema:hoyo]
[clase:PALABRA,fila:11,col:1,lexema:ala]
[clase:PALABRA,fila:12,col:1,lexema:a]
[clase:EOF,fila:13,col:1]
```

PALABRAS:

```
hola
pista
hoja
hoyo
ala
a
```

PATRONES:

```
ho*a: 2
ho*: 3
a*a: 1
```

Debéis crear una nueva definición léxica con JFlex y modificar convenientemente el código java en `ClaseLexica.java` y `ALexOperations.java`. Utilizad el ejemplo del Ejercicio 1 como referencia, como veréis los cambios que debéis realizar son sencillos. Eliminad las unidades léxicas que ya no utilizamos (números, evalúa, ...).

*** Opcional: Finalmente, tenéis que añadir código al `Main.java` para que haga el procesamiento final de las unidades léxicas obtenidas. Es decir, en primer lugar debéis recopilar y mostrar todas las palabras y después ver para cada una de ellas en qué patrones encaja.