Ejercicios sesión 2 de laboratorio PLG

Albert Rubio

21 de febrero de 2020

Ejercicio 1: *Instalar CUP*. En el Campus Virtual de la asignatura en el apartado de "Código ejemplo y herramientas" encontraréis un enlace para obtener *CUP* y otro para obtener "Ejemplo de implementación con CUP".

Instalad CUP en vuestra cuenta usando el archivo ".jar". Podéis utilizar el entorno para desarrollo de código Java que prefiráis. Deberéis definir los caminos necesarios en el `CLASSPATH`.

Para procesar el archivo `Tiny.cup` que encontraréis en la carpeta `asint` con el código para generar un analizador sintáctico con CUP, tenéis que ejecutar:

```
$ java java_cup.Main -parser AnalizadorSintacticoTiny -symbols ClaseLexica -nopositions Tiny.cup
```

Esto generará los archivos `AnalizadorSintacticoTiny.java` y `ClaseLexica.java` cuyo contenido debería coincidir con los archivos con el mismo nombre que ya existían. La carpeta `asint` contiene el analizador léxico que se trata como vimos en la anterior sesión. Compilad el código java generado y ya podéis usar el analizador. En la versión del `Main` que se adjunta, se lee de la entrada del fichero `input.txt` (podéis modificarlo para que use un fichero que se pasa por parámetro). Probad el analizador generado sobre el ejemplo siguiente

```
evalua
166.386 * euros + 1.66386 * (centimos1 + centimos2)
donde
euros = 567,
centimos1 = 456,
centimos2 = 10
```

Ejercicio 2: *Lenguaje de Listas*. Tenemos que hacer un analizador sintáctico 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 = L1#L2              // L3 es la concatenación de L1 y L2, es decir L2
L4 = [[[1,2],3],4]      // L4 es una lista heterogénea
L5 = lreduce + L4       // L5 tiene un único elemento: la suma de los números de L4
L6 = lmap - 1 L4        // L6 se obtiene restando 1 a los números de L4
print L5                // Se imprime [10]
print L6                // Se imprime [[[0,1],2],3]
L7 = lfilter != 1 L4    // L7 es una copia de L4 eliminando los números iguales a 1
print L7                // Se imprime [[[2],3],4]
L8 = lfilter > 2 L7     // L8 será [[[],3],4]
```

En cualquier sitio que se espere una lista podemos encontrar una expresión que tenga como resultado una lista, pero en algunos casos será necesario el uso de paréntesis para romper la ambigüedad. Por ejemplo, podemos tener

```
L9 = lfilter != 1 (L4 # L2)
```

Considerad distintas posibilidades según la prioridad que refleje vuestra gramática. Por otro lado, si no es necesario no hay que obligar a poner paréntesis. Por ejemplo hay que aceptar

```
fil2 = lfilter != 1 [1,2,3]
```

La operación `lfilter` tiene como parámetros una operación relacional $(>, <, ==, !=)$, un número y una lista. La operación `lmap` tiene como parámetros una operación aritmética $(+, -, *, /)$, un número y una lista. La operación `lreduce` tiene como parámetros una operación aritmética $(+, -, *, /)$ y una lista.