Ejercicios Laboratorio 3

Lenguajes de programación y procesadores de lenguaje

14 de Noviembre de 2024

En el Campus Virtual de la asignatura en el apartado de "Laboratorio 3: Procesamiento dirigido por Sintaxis" encontraréis un enlace para obtener CUP como archivo `cup.jar` y otro para obtener el código Java para generar los analizadores sintácticos en los que vamos a trabajar en la sesión de hoy `evaluadorExpresiones.zip` y `constructorAST.zip`. Descargad todos estos archivos en una carpeta y descomprimid los ficheros `.zip`.

Para más información o versiones más recientes de CUP visitad `http://www2.cs.tum.edu/projects/cup/`.

Podéis instalar CUP 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: Generador AST

En este apartado, vamos a seguir trabajando con el lenguaje de las prácticas anteriores pero ahora generando el AST del programa en vez de únicamente analizar si era correcto sintácticamente. Para ello hemos generado una estructura de constructores y nodos para poder almacenar el AST. Esta estructura es general y podrá ser reutilizada para otros compiladores y vuestros proyectos (está en la carpeta ast). En ella vemos por ejemplo que tenemos una interfaz para un nodo general (ASTNode), que a su vez puede ser una expresión E o una instrucción Instrucción. Para las expresiones, tenemos una clase abstracta E que será la que represente una expresión. Ahora mismo dentro de las expresiones tenemos las expresiones binarias (EBin), que pueden ser sumas o mayor; o por otro lado números (Num), booleanos (BoolValue) e identificadores (Identificador).

Las reglas de análisis sintáctico llaman a estos constructores para generar el AST. Observa que las reglas van generando el árbol que reconoce el programa. Para generar el analizador sintáctico (`ConstructorAST`) utilizamos::

```
java -cp cup.jar java_cup.Main -parser ConstructorAST
      -symbols ClaseLexica -nopositions Tiny.cup
```

Después para compilar el proyecto hacemos desde el directorio constructor, llamamos a:

```
javac -cp ../cup.jar */*.java
```

Finalmente, al ejecutar el programa generaremos el AST y lo imprimimos por pantalla. En este caso, estamos generando el AST representando el programa almacenado en input.txt:

```
java -cp ../cup.jar: constructorast.Main  ../input.txt
```

Modificaciones a realizar

Para practicar, podemos añadir los cambios que hicimos para extender el análisis léxico y sintáctico a este constructor del ast. De cara a añadir los cambios debemos:

1. Modificar los ficheros `mini_lexico.l` y `AlexOperations.java` añadiendo las nuevas unidades léxicas a reconocer y sus constructores.
2. Modificar el fichero `Tiny.cup` añadiendo las nuevas unidades léxicas como terminales. ES IMPORTANTE UTILIZAR LOS MISMOS NOMBRES QUE EN `AlexOperations.java`.
3. Modificar el fichero `Tiny.cup` añadiendo los nuevos no terminales necesarios y las nuevas expresiones a añadir en la gramática.
4. Generar los nodos necesarios en el ast para las nuevas operaciones o expresiones.
5. Hacer que las reglas del analizador sintáctico llamen a las constructoras de los nodos del ast.

Algunos cambios que podéis realizar son los siguientes (ordenados de más sencillos a más complejos) son los siguientes (son los mismos en los que trabajamos en la práctica anterior):

- Añadir el operador para la multiplicación
- Añadir otros operadores relacionales y las operaciones entre booleanos

- Añadir instrucciones para permitir bucles
- Añadir números reales y el tipo de número real
- ...

Recuerda que de cara al proyecto final será necesario extender el lenguaje para que incluya las instrucciones básicas presentes en la mayoría de los lenguajes de programación, así como las incluidas en los requisitos vía carta.