Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

CI3725 - Traductores e Interpretadores

Abril - Junio 2015

Lanscii - Etapa II

Análisis Sintáctico (9%)

Especificación de la entrega

Esta segunda etapa del proyecto corresponde al módulo de análisis sintáctico de programas escritos en Lanscii. Esto incluye la definición de una gramática y la construcción del árbol sintáctico abstracto correspondiente.

Usted deberá escribir una gramática para Lanscii que será luego utilizada en la herramienta generadora de analizadores sintácticos de su lenguaje de preferencia.

Esto incluye la implementación de distintas clases (Java, Ruby y C++) o tipos de datos (Haskell) que representen el significado de las expresiones e instrucciones en Lanscii. Por ejemplo, las expresiones conformadas por dos subexpresiones y un operador binario, podría ser representando por la clase/ tipo de datos `BIN_EXPRESION`, que pudiera tener como información las dos *subexpresiones*, el operador utilizado y el tipo resultante de la expresión. También pudiera tenerse diferentes árboles de expresión binarios para los casos aritméticos, booleanos, relacionales y de canvas; ahorrando así, particularmente en el caso de Lanscii, la necesidad de calcular el tipo resultante de la expresión. Por otra parte, una instrucción de *repetición indeterminada* podría ser representada por la clase/tipo de datos `REPETICION_INDET`, que pudiera tener como información la condición de la iteración y la subinstrucción sobre la cual se esté iterando.

De encontrar algún error sintáctico, deberá reportar dicho error y abortar la ejecución del analizador. Usted **no** debe imprimir todos los errores sintácticos, solamente el primero encontrado.

En esta etapa, la gramática que utilice y los nombres que coloque a los símbolos *terminales* y *no terminales* presentes en la misma, son enteramente de su elección. Sin embargo, dicha gramática debe cumplir con algunas condiciones:

- Su gramática debe ser ambigua y recursiva por la izquierda. Los conflictos deberán ser tratados entonces por los mecanismos de resolución de conflictos que incluye cada herramienta.

- Su gramática **debe** reconocer completa y únicamente a las cadenas válidas en Lanscii.
- Los nombres escogidos para los símbolos *terminales* y *no terminales* **deben** ser apropiados y fácilmente reconocibles como parte del lenguaje. (Por ejemplo: `EXPR`, `EXPRESSION` y `EXPRES` son nombres apropiados para las expresiones).

Como ejemplo, si es suministrado el siguiente programa válido en Lanscii:

```
{ %foo bar |
    bar = 3;
    {- asignar a foo el doble del valor de bar, más treinta y cinco;
       si bar es mayor a dos -}
    ( bar > 2 ?
        foo = 35 + bar * 2
    )
}
```

El resultado de la ejecución de su analizador deberá tener un formatio similar al siguiente:

```
ASSIGN:
 |  VARIABLE:
 |   |  IDENTIFIER: bar
 |  EXPRESSION:
 |   |  NUMBER: 3
CONDITIONAL STATEMENT:
 |  CONDITION:
 |   |  OPERATION: >
 |   |   |  IDENTIFIER: bar
 |   |   |  NUMBER: 2
 |  THEN:
 |   |  ASSIGN:
 |   |   |  VARIABLE:
 |   |   |   |  IDENTIFIER: foo
 |   |   |  EXPRESSION:
 |   |   |   |  OPERATION: +
 |   |   |   |   |  NUMBER: 35
 |   |   |   |   |  OPERATION: *
 |   |   |   |   |   |  IDENTIFIER: bar
 |   |   |   |   |   |  NUMBER: 2
```

Note que la declaración y tipos de las variables no se ve reflejado en el árbol sintáctico abstracto. Esta funcionalidad corresponde a una estructura llamada tabla de símbolos y será construida en la siguiente entrega.

Implementación

Usted deberá desarrollar el analizador sintáctico de Lanscii como se indica a continuación:

1. Diseñe una gramática cuyo lenguaje generado sea Lanscii y escriba la misma en el lenguaje de especificación de la herramienta ofrecida para el lenguaje de su elección.
2. Escriba las reglas para construir e imprimir el árbol sintáctico abstracto correspondiente a una frase reconocida.
3. Escriba un programa escrito en Lanscii (posiblemente incorrecto) y genere el árbol sintáctico correspondiente en caso de ser correcto. Si el programa no es correcto por razones puramente léxicas, debe imprimir únicamente los errores de lexer econtrados. Si el programa tiene al menos un error sintáctico, debe imprimir únicamente el primero de tales errores.

Análisis Teórico-Práctico

Esta etapa no tendrá análisis teórico-práctico.

Entrega

Formato de Entrega

Deben enviar un correo electrónico a **todos los preparadores** con el asunto: `[CI3725]eXgY` donde `X` corresponde al número de la entrega e `Y` al número del equipo. El correo debe incluir lo siguiente:

Un archivo **.zip** con el nombre `eXgY` siguiendo las mismas instrucciones del asunto del correo referentes a los valores de `X` e `Y`, que contenga:

- Código fuente debidamente documentado.
- En caso de utilizar *Haskell*, deben incluir un archivo Makefile o un archivo de configuración para *Cabal*. En caso de utilizar *C++* se debe incluir un archivo Makefile. Si su proyecto no compila, el proyecto no será corregido.
- Un archivo de texto con el nombre `LEEME.txt` donde **brevemente** se expliquen:
  - Decisiones de implementación
  - Estado actual del proyecto
  - Problemas presentes
  - Cualquier comentario respecto al proyecto que consideren necesario
  - Este archivo **debe** estar identificado con los nombres, apellidos y carné de cada miembro del equipo de trabajo.

- Un archivo de texto con el nombre `analisistp.txt` con la respuesta a cada una de las preguntas correspondientes a la sección de **Análisis Teórico-Práctico**. En caso de ser necesario adjuntar imágenes para contestar alguna de las preguntas, éstas deben estar en formato **.jpg** o **.png** debidamente identificadas tanto en el archivo **.zip** como en el archivo `analisisTP.txt`.

Fecha de entrega

La fecha límite de entrega del proyecto es el día **jueves 28** de mayo de 2015 (semana 8) *hasta* las **11:00pm**, entregas hechas más tarde tendrán una **penalización del 20%** de la nota, esta penalización aplica por cada día de retraso.