

PRÁCTICA 2: GRAMÁTICAS Y GENERADORES

AUTOMÁTICOS

591000 - COMPILADORES

Prof. Katia Leal Algara, Universidad de Alcalá

13/10/2022

Objetivos de la práctica

- Aprender a manejar una herramienta de generación de intérprete de lenguajes.
- Ser capaz de generar analizadores léxicos que reconozcan un lenguaje en concreto.
- Ser capaz de generar analizadores sintácticos que reconozcan un lenguaje en concreto.
- Ser capaz de generar el árbol sintáctico abstracto asociado a una entrada.
- Iniciarse en el concepto de la tabla de símbolos y la gestión de errores.

Enunciado

Esta práctica consta de dos partes:

1. Primera parte: generación de árboles sintácticos para lenguajes específicos

Para esta primera parte se os proporcionarán características y un conjunto de ejemplos y, partiendo de los mismos se deberá:

Nivel básico (2/3 de la valoración):

- Ser capaz de detectar e identificar automáticamente cada uno de los elementos del lenguaje (es decir, generar el lexer y el parser correspondientes al lenguaje). El analizador debe ser generalista, es decir, debe ser capaz de validar otros programas escritos con la misma estructura que los ejemplos propuestos.
- Construir el AST correspondiente (aquí es suficiente con una captura de pantalla).

Nivel medio (1/3 de la valoración):

- Plantear alguna mejora que mejore o amplie la gramática. Es posible añadir algún elemento estructural o tipo, por ejemplo.

- Mostrar el fichero del árbol AST en un formato legible de texto plano - aquí es necesario el uso de listeners en ANTLR.

1.1. SQLmini (3 puntos)

Las consultas tipo SQL son muy habituales y pueden servirnos como caso sencillo para generar nuestro primer lexer y parser. Aquí tenéis ocho ejemplos distintos:

```
SELECT name FROM employees
SELECT name, age FROM employees
SELECT name, age FROM employees WHERE age > 30
SELECT name, age FROM employees ORDER BY age ASC
SELECT name, age FROM employees WHERE age >= 30 ORDER BY name DESC
SELECT * FROM products WHERE (price >= 100 AND stock > 0) OR (category = 'Electronics')
SELECT name, description FROM products WHERE (price >= 10.5 AND stock > 0) OR
        (category = 'Electronics') ORDER BY name ASC
SELECT name, age FROM employees WHERE
        (age > 30 AND (department = 'HR' OR department = 'Finance'))
        OR (salary > 50000)
```

Estas consultas de prueba cubren una amplia gama de escenarios, desde simples sentencias SELECT hasta consultas más complejas con cláusulas WHERE y cláusulas ORDER BY. En las últimas no hay un salto de línea en el medio, podéis suponer que son una sola línea. Podéis utilizar estos casos de prueba para validar vuestro lexer y parser para el SQLmini.

Nota: la AND siempre tendrá prioridad sobre la OR si están al mismo nivel.

Generad un analizador léxico y sintáctico para poder construir el AST correspondiente a una consulta SQL cualquiera que siga las estructuras de las consultas de ejemplo.

Requisitos de entrega:

- Para la primera parte, básica, es necesario:
 - El analizador completo (léxico y sintáctico) dividido en dos ficheros separados ANTLR, explicando las decisiones tomadas.
 - En la memoria hay que aportar las capturas de los árboles generados para:
 - Los ocho ejemplos propuestos.
 - Dos ejemplos adicionales de libre elaboración que tengan características similares (y, preferiblemente, que cubran símbolos o casos no cubiertos explícitamente en los ejemplos).
- Para la segunda parte, intermedia, es necesario adicionalmente la implementación requerida para un programa principal que:
 - Lea una entrada cualquiera desde la consola.
 - Escriba el árbol resultante correspondiente a la entrada leída y lo muestre por la consola (o fichero de texto) en formato de texto plano.

1.2. Linguine (6 puntos)

Linguine es un nuevo lenguaje que acabamos de inventar, aunque se basa en estructuras y dinámicas a las que estamos acostumbrados. Utilizando los programas colgados en la carpeta de entrega de la práctica 2 de la plataforma escritos en Linguine, generar un analizador léxico y sintáctico que sea capaz de generar los AST correspondientes.

Los ejemplos:

```
(1) ----- Asignación sencilla -----
let x = 5;

(2) ----- Asignación con operación -----
let y = 2 + 3;

(3) ----- Condicional -----
if (x > 2) then let y = 1 else let y = 0;

(4) ----- Función simple -----
fun add(a, b) -> a + b;
let result = add(3, 4);
show(result);

(5) ----- Función recursiva -----
fun factorial(n) -> if (n <= 1) then 1 else n * factorial(n - 1);
let result = factorial(5);
show(result+2);

(6) ----- Estructura match -----
match x with
| 1 -> "One"
| 2 -> "Two"
| ? -> "Other";

(7) ----- Funciones anidadas -----
fun add(a, b) -> a + b;
fun square(x) -> x * x;
let result = square(add(2, 3));

(8) ----- Asignación con condicional -----
let a = 5;
let b = 3;
let result = if (a > b) then (a + b) * (a - b) else a / b;

(9) ----- Función recursiva dentro de un match -----
fun fib(n) -> match n with
| 0 -> 0
| 1 -> 1
| ? -> fib(n - 1) + fib(n - 2);
let result = fib(6);

(10) ----- Función pasada como parámetro -----
```

```
fun apply(func, x) -> func(x);  
fun square(x) -> x * x;  
let result = apply(square, 5);  
-----
```

Algunas notas relevantes:

- Notad que las instrucciones dentro de una función o de un condicional tienen una sola línea.
- La estructura match es como un switch/case pero que devuelve el valor de la derecha en función de la entrada.
- 'show' imprime por pantalla.
- Asumid siempre que si existe una operación pueden existir también las comunes similares. Por ejemplo, si existe '>' (mayor que) también existen el resto de comparaciones, como '<' (menor que), etc. Si se puede poner un if dentro de una asignación, también un match. Y así sucesivamente.
- Cualquier decisión que se tome a nivel de diseño se debe explicar.

Requisitos de entrega:

- Para la primera parte, básica, es necesario:
 - El analizador completo (léxico y sintáctico) dividido en dos ficheros separados ANTLR, explicando las decisiones tomadas.
 - En la memoria hay que aportar las capturas de los árboles generados para los diez ejemplos propuestos.
- Para la segunda parte, intermedia, es necesario adicionalmente la implementación requerida para un programa principal que:
 - Lea una entrada cualquiera **desde un fichero de texto** como los proporcionados en los ejemplos.
 - Escriba el árbol resultante correspondiente a la entrada leída y lo muestre por la consola (o fichero de texto) en formato de texto plano.

2. Segunda parte: iniciación a la tabla de símbolos (1 punto)

Este apartado sirve únicamente para introducirnos a la tabla de símbolos, que será usada más extensivamente en la última práctica. Consiste en **añadir una funcionalidad a vuestro programa del apartado 1.2** que devuelva, tras analizar un fragmento de código, una lista con las variables que se han declarado en él y los valores que tienen asignadas.

Secuencia recomendada

Siempre elaborando la memoria en paralelo a lo avanzado:

1. Realizar y entender el nivel básico de 1.1 y 1.2.
2. Hechas ambas cosas, entender el funcionamiento básico de los Listeners de ANTLR para construir el árbol en formato de texto plano y sacarlo por pantalla o por fichero. Así se pueden realizar directamente ambos niveles medios.
3. Pensar en posibles mejoras o pequeñas extensiones de las propuestas 1.1 y 1.2 e implementarlas.
4. Entendido el funcionamiento de los Listeners es casi directo extraer la información para la parte “avanzada” de la segunda parte; solo quedará implementar el mecanismo.

Defensa

En la defensa de la práctica se deberá exponer y explicar los distintos componentes que formen el sistema programado por el alumno, respondiendo a las preguntas del profesor, si corresponde. Adicionalmente, se podrán proporcionar ficheros de entrada en formato texto que se deberán analizar léxica y semánticamente y cuyo AST deberá mostrarse por pantalla (y explicarse debidamente).

Ejemplos

Se encuentran incluidos en el enunciado.

Material Adicional

Tenéis también seis vídeos explicativos de ANTLR (desde su instalación hasta su funcionamiento) en la plataforma. La recomendación es visualizarlos completos y en orden, e intentar realizar a la vez los casos que se plantean en ellos, que son sencillos pero muy ilustrativos.



Entregables

La entrega de la práctica constará de:

- Breve memoria donde se indique cómo ejecutar la práctica para cada uno de los casos y qué estructura se ha seguido para cada uno de los casos propuestos. Se detallarán además las dificultades encontradas, y se resaltarán (remarcándolo claramente) tanto las partes no resueltas o problemáticas (y la razón por la que lo son) como las partes añadidas y los aspectos que vayan más allá de los mínimos exigidos.

- El código fuente de la solución a la memoria listo para ejecutar los ejemplos. Solo son necesarios los ficheros ANTLR y los ficheros Java/Python esenciales para su ejecución.

Consideraciones generales

- La práctica se realizará en grupos de tres personas. La recomendación es que se formen grupos que también puedan, posteriormente, realizar la PL3 conjuntamente, porque estarán relacionadas.
- Para la realización de la práctica se deberá usar ANTLR. Está permitido el uso de plugins para IntelliJ, NetBeans, Eclipse, Visual Studio Code, Visual Studio IDE, y jEdit.
- Se permite  el uso de Java  para la implementación.
- La entrega sólo se considera lista para calificar si está en condiciones de funcionamiento, si tiene una documentación que cumple los requisitos establecidos, y si el grupo se presenta al completo en la defensa, en caso de ser convocado. Si no se satisfacen todas estas condiciones, se considerará no entregada / no presentada.
- La entrega sólo se podrá realizar entre las fechas de publicación de la práctica y la fecha límite de entrega, por lo que el alumno cuenta con tiempo de sobra para cumplir con el proceso.
- La entrega sólo se considerará hecha si se realiza a través de la actividad correspondiente de la plataforma BlackBoard: cualquier otro método de entrega será no válido, y se contará como no presentado.
- Todos los ficheros deben ser subidos a la plataforma: no se permiten entregas que contengan enlaces a servicios externos como Dropbox, OneDrive, Google Drive, etc. En caso de enviar la práctica con estos enlaces (o por correo electrónico), se considerará la práctica como no completa y no entregada / no presentada.

Criterios de calificación

- **No presentado:** si no se cumplen las condiciones de entrega en uno o varios puntos de los ya expuestos, la práctica será calificada como no presentada, perdiendo el alumno la posibilidad de conseguir puntos por ella (calificará como 0 puntos) de cara a la calificación final de la asignatura, y teniéndose en cuenta para los criterios de No Presentado en el global de la asignatura según lo indicado en la guía docente.
- **Suspense:** La práctica no tiene un funcionamiento correcto en alguno o varios de sus puntos, y/o la documentación adjunta es deficiente y no muestra adecuadamente el trabajo realizado.

- **Aprobado:** La práctica tiene un funcionamiento correcto en cuanto a procesamiento y resultados conseguidos en todos sus puntos. La documentación y defensa es correcta. El AST generado recoge las estructuras del lenguaje.
- **Notable:** Además de las condiciones de aprobado, la práctica muestra un trabajo cuidado del alumno, con una buena limpieza y organización del código, así como una buena estructuración del problema, y una buena documentación y defensa. El AST generado recoge las estructuras del lenguaje, y lo hace de manera que su acceso y organización sea adecuada, incluyendo la aplicación y dominio de los Listeners.
- **Sobresaliente:** El alumno ha desarrollado la práctica de manera correcta y ha ido más allá de lo solicitado, mostrando un dominio de los problemas que la práctica abarca, tanto a nivel técnico como a nivel de conocimiento del problema y sus soluciones posibles. El AST generado recoge las estructuras del lenguaje, y presenta una generalización tal que permite la extensión del lenguaje y sus operaciones de manera sencilla sin necesitar cambios profundos en la gramática. Es capaz de organizar el AST mediante una estructura totalmente jerárquica para todos sus elementos en formato texto plano.

Plagio

Si se detecta la copia total o parcial entre prácticas, o se encuentran estructuras de código similares entre distintas prácticas, se considerará automáticamente que los alumnos implicados han copiado o no han realizado su práctica de manera original, tal como establecen las condiciones de la misma. Todos los alumnos implicados serán calificados automáticamente como suspenso, con calificación 0, tal y como establece la normativa de la universidad.