Universidad de Alcalá

Práctica 2: Gramáticas y generadores automáticos

591000 - Compiladores

Prof. Marçal Mora Cantallops, Universidad de Alcalá 14/10/2022

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.

Consideraciones generales

- La práctica se realizará en grupos de hasta 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 tanto el uso de Java como el de Python para la implementación. En caso de realizarse en Python, la nota máxima posible se verá incrementada en un 10 %.
- El entregable será tanto el código generado como una memoria explicativa del proceso seguido (para cada uno de los casos propuestos); dicha memoria será el principal factor en la valoración final (previa comprobación de que el código funciona adecuadamente).
- En caso de no realizar alguna de las partes, hay que remarcarlo claramente poniendo el título del apartado e indicando los motivos por los que no se ha podido o se ha decidido no implementarlo.

Enunciado

Esta práctica consta de cuatro ejercicios en dos partes separadas.

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

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

Nivel básico (1 puntos):

- 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).
- Construir el AST correspondiente.

Nivel medio (1 punto):

- Plantear alguna mejora que mejore o amplie la gramática.
- Mostrar el fichero del árbol AST en un formato legible de texto plano.

1.1. Árbol sintáctico de un CSV

Un fichero separado por comas es un formato muy habitual para conjuntos de datos en columnas. Algunos ejemplos podrían ser los siguientes:

```
Nombre; Frase; Tipo; Lanzamiento
Aatrox; the Darkin Blade; Juggernaut; 2013-06-13
Ahri; the Nine-Tailed Fox; Burst; 2011-12-14
Akali; the Rogue Assassin; Assassin; 2010-05-11
```

```
Type,Fast Moves,DPS,Power-PvP,Energy-PvP,Cast Time,Turns,DPT,EPT
Ste,Steel Wing,"13,8",11-7,6-5,"0,8s",2,"3,5","2,5"
Dra,Dragon Tail,"13,6",15-9,9-10,"1,1s",3,"3,0","3,3"
```

Generad un analizador léxico y sintáctico para poder construir el AST correspondiente a un CSV cualquiera, suponiendo que los separadores pueden ser la coma (,), el punto y coma (;) o la barra (|).

1.2. Un lenguaje para las formas

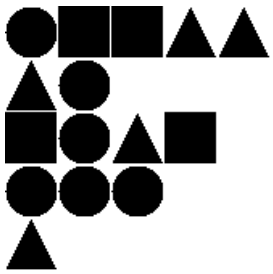
Vamos a crear un lenguaje simple para definir imágenes que estén compuestas de formas. El lenguaje *formitas* va a tener las siguientes características:

- El código fuente estará en una sola línea.

- El primero de los elementos será el tamaño del marco (o de la imagen de salida). Lo llamaremos imgdim y será, por ejemplo, imgdim:256, que significará que la imagen va a tener un tamaño de 256x256 píxeles.
- El tamaño de las formas individuales se especifica a nivel global y será con shpdim (e.g. shpdim:64 para imágenes de 64x64.
- La descripción de las imágenes empieza con un delimitador (>>>).
- Las filas se separan con una pipe (|). No puede haber un símbolo | ni justo después del inicio (>>>) ni justo antes del final (<<<),
- Las formas individuales se separan por comas (,). Una fila no puede empezar o terminar con una coma.
- Cada fila está formada por una lista de formas (por ejemplo: (triangulo, cuadrado, circulo)).
- El cierre se hace con tres símbolos de menor (<<<).

```
imgdim:180,shpdim:32
>>>circulo,cuadrado,cuadrado,triangulo,triangulo|triangulo,circulo|
cuadrado,circulo,triangulo,cuadrado|circulo,circulo,circulo|triangulo<<<
```

No es el objetivo en esta PL, pero la salida final de una línea como esta sería algo como esto, para que os hagáis una idea de lo que representa:



1.3. Lenguaje de programación mínimo

Imaginemos que tenemos un lenguaje muy simple de programación que es, además, más natural en su lenguaje. Lo llamaremos E++. Un ejemplo de programa podría ser el siguiente:

```
1 # Inicializacion de variable
2 asignar a = 20 ;P
3
4 # if-else sencillo
5 a > 2 ???
6 si ->
7     a = 5 - 2 ;P
8 terminar
```

```
9
10 # imprimir por pantalla
11 mostrar a ;P
```

Generad la gramática capaz de generar el AST para este lenguaje, que tiene:

- Asignación inicial para inicializar variables (asignar).
- Estructuras tipo if con (???) para la condición, (si ->, no->) para el condicional y (terminar) para cerrar el bloque.
- Las comparaciones pueden ser las habituales en los lenguajes de programación.
- Usa (;P) como fin de línea.
- Imprime por pantalla con (mostrar).
- Admite enteros, floats y cadenas de caracteres - strings.
- Los identificadores pueden tener la forma habitual en los lenguajes de programación.
- Solo puede sumar y restar como operadores aritméticos por ahora.
- Los comentarios van con almohadilla y siempre en una línea para ellos solos (es decir, no hay comentarios en la misma línea que hay una instrucción).

2. Segunda parte: las recetas de la abuela

Nivel básico (3 puntos):

- Definir un lenguaje en el que poder expresar recetas de cocina. Debe, por lo tanto, cubrir las instrucciones habituales, los verbos de cocina, los utensilios y las indicaciones de cantidades o productos. Debéis también limitar las elecciones posibles a un conjunto razonable, obviamente. En general, debe ser lo más informativo posible de la forma más sistemática que se os ocurra. Se valorará tanto la sencillez como el potencial (es decir, que de la forma más sistemática y condensada posible cubra gran cantidad de posibilidades).
- 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 que habéis diseñado). Es, por lo tanto, necesario que generéis un juego de pruebas exhaustivo - de recetas de ejemplo en este caso.
- Construir el AST correspondiente. Fijaros que una vez diseñado lo anterior, el ejercicio se reduce a hacer lo mismo que en los apartados de la primera parte (la diferencia es que habéis diseñado vosotros el lenguaje y no viene dado).

Nivel avanzado (1 puntos):

- Implementar un analizador para la lista de la compra que:

- Pueda leer una receta.
- Escriba la lista de ingredientes con sus cantidades por pantalla.
- Escriba la lista de utensilios a utilizar.
- (Y otras cosas que se os ocurran en función de vuestro diseño).
- Nota/pista: aquí se trata de poner en práctica un simulacro de tabla de símbolos para acumular la información vista al visitar el árbol.

Secuencia recomendada

Siempre elaborando la memoria en paralelo a lo avanzado:

1. Realizar y entender el nivel básico de la parte 1.
2. Aplicando lo anterior, realizar y entender el nivel básico de la parte 2.
3. 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.
4. Pensar en posibles mejoras o pequeñas extensiones de las propuestas 1.1, 1.2 y 1.3 e implementarlas.
5. 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).

Para los casos de la segunda parte, los estudiantes deberán proporcionar su propio juego (exhaustivo) de pruebas.

Ejemplos

Se encuentran como materiales anexos.

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.