Universidad de Alcalá

Práctica 2: Gramáticas y generadores automáticos

Prof. Marçal Mora Cantallops, Universidad de Alcalá Octubre 2025

Objetivos de la práctica

- Aprender a manejar una herramienta de generación de intérpretes 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 intención 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 (y cualquier otro equivalente).
- Se permite tanto el uso de Java como el de Python para la implementación.
- 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 programa 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.
- En caso de usar algún tipo de IA Generativa es necesario especificar, dentro de la memoria y en detalle, en qué partes se ha usado y cómo.

Enunciado

Esta práctica consta de tres ejercicios en dos partes separadas.

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

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 (|).

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):

- Mostrar el fichero del árbol AST en un formato **legible** de texto plano que solo contenga los elementos indispensables (es decir, no es válido replicar la salida de ANTLR directamente).

1.2. 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.
- Es posible anidar condicionales.
- 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).

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 (2 punto):

- Realizar ampliaciones en la gramática que estén justificadas y que amplien la funcionalidad del lenguaje para acercarlo más a un lenguaje de programación habitual. (0,25 por cada una hasta un máximo de 4).
- Mostrar el fichero del árbol AST en un formato **legible** de texto plano que solo contenga los elementos indispensables (es decir, no es válido replicar la salida de ANTLR directamente).

2. Segunda parte: Un problema de librerías (de IKEA)

Nivel básico (4 puntos):

- Definir un lenguaje en el que poder expresar el montaje de una librería/estantería de IKEA (se aportan las instrucciones de algunas de las más comunes). Debe, por lo tanto, cubrir las instrucciones necesarias de la forma más generalista posible, los verbos de montaje, los herrajes y herramientas y las indicaciones de cantidades o elementos concretos. Debéis también reducir las posibilidades lingüísticas 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).
- Proporcionar las traducciones a instrucciones para cada uno de los manuales de montaje proporcionados. En caso que un manual tenga más de un camino posible (por ejemplo, colgar en la pared o no hacerlo), las deberéis considerar como instancias distintas de montaje (por ejemplo, la BILLY es un solo manual, pero la KALLAX es como si tuviese dos, cambiando los pasos del 15 al 19).
- Ser capaz de detectar e identificar automáticamente cada uno de los elementos del lenguaje generado (es decir, generar el lexer y el parser correspondientes al lenguaje que habéis diseñado). Para ello os deberéis basar en los manuales proporcionados de ejemplo y se considerará suficiente que reconozca las traducciones generadas para los ejemplos proporcionados.
- Construir el AST correspondiente para todos los ejemplos proporcionados. 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 secuencia de montaje:
  - Pueda leer los pasos.
  - Escriba la lista de herrajes con sus cantidades por pantalla.
  - Escriba la lista de herramientas a utilizar.
  - (Si, en función de vuestro diseño, se os ocurren otras cosas que se pueden añadir, se valorarán positivamente).
  - 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 para ambos ejemplos.
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 extensiones de la propuesta 1.2 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 conjunto de instrucciones de montaje generadas en su lenguaje para su análisis.

Ejemplos

Se encuentran como materiales anexos:

- Ejemplos CSV
- Ejemplo E++
- Ejemplo posible lenguaje para montaje de la caja KNAGGLIG, que no tiene que ver con las librerías ni hay que cubrirlo en vuestro lenguaje, pero puede ayudar a comprender la tarea a realizar.
- Manuales del resto de muebles del apartado 2.

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.