Universidad de Alcalá

Práctica 2: Gramáticas y generadores automáticos

780018 - Procesadores del Lenguaje

Prof. Marçal Mora Cantallops, Universidad de Alcalá Octubre 2024

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 tres partes separadas.

1. Primera parte: árbol sintáctico para un lenguaje limitado (la búsqueda del tesoro)

En esta primera parte se propone la construcción de un analizador léxico y sintáctico en ANTLR que sea capaz de extraer la información relevante de un texto que describe un mapa de la búsqueda del tesoro. Para ello, se os proporcionará un fichero de ejemplo y, partiendo del mismo se deberá realizar el análisis, suponiendo que, de inicio, la estructura es siempre la misma (pudiendo tener, eso sí, más o menos líneas). Notad que algunos elementos son constantes y estructurales, mientras otros son variables y, por lo tanto, hay que ser lo suficientemente flexible en el analizador como para tratarlos e incorporarlos en el árbol correspondiente. En este primer punto no es necesario implementar medidas de control sobre el contenido textual del mapa (por ejemplo, no es necesario controlar que los límites sean correctos, etc.). Las tareas se dividen, pues, en:

Nivel básico (2 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 (aquí es suficiente con una captura de pantalla).

Nivel medio (1 punto):

- Plantear al menos dos mejoras que amplien la gramática (es decir, que añada posibilidades o instrucciones adicionales a la definición del mapa o que aporte flexibilidad adicional).
- Mostrar el fichero del árbol AST en un formato legible de texto plano - aquí es necesario el uso de listeners en ANTLR.

2. Segunda parte: jugando a la búsqueda del tesoro

Esta parte es una extensión sobre la primera parte. **Nivel avanzado (2 puntos)**.
Implementar una estructura de juego en la que:

- Se pueda cargar un mapa desde un fichero externo. Si se desea, se puede asumir que el mapa tiene un tamaño limitado (e.g., 5x5).
- El jugador seleccione casillas en las que bucear.
- En caso de encontrar algún tesoro, habrá que indicarlo, junto a la puntuación acumulada. El tesoro, lógicamente, debe desaparecer.
- En caso de no encontrarlo, también se mostrará un mensaje informativo al jugador.
- El juego puede durar eternamente o se puede añadir una condición de fin de partida (e.g., se han encontrado todos los tesoros o se han terminado las casillas/intentos).

Añadir mecánicas adicionales (aquí se pueden plantear desde nuevas reglas, un segundo jugador o incluso restricciones semánticas adicionales, por ejemplo) o reconocer entradas de texto más ricas y elaboradas (ampliando el trabajo de la parte 1), con su correspondiente lógica y justificación, pueden suponer una bonificación adicional a la nota de esta parte, hasta un máximo de 0,5 puntos adicionales.

3. Tercera parte: árbol sintáctico para un lenguaje simple (MiniB)

MiniB es un lenguaje inventado aunque basado lejanamente en una especie de BASIC simplificado, por eso lo de mini. Se adjuntan una serie de ejemplos, y podéis basaros en cualquier recurso sobre BASIC para entender qué hacen la mayoría de las funciones incluidas. El título es informativo de lo que representa cada fragmento de código. En la misma línea, se incluyen varios ficheros con “error” en el título; serían ficheros que deberían dar error, aunque tened en cuenta que en este punto, si el error no es léxico o sintáctico no hay que detectarlo todavía.

Otra consideración importante es notar que los ficheros sirven solo de ejemplo pero que las estructuras se pueden combinar. Es decir, si existe un condicional y existe un bucle, lógicamente podría haber un condicional dentro de un bucle, aunque no se muestre en los ejemplos.

Nivel básico (4 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 - aquí es necesario el uso de listeners en ANTLR.

Requisitos de entrega

- Para la primera y tercera 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 ejemplos propuestos y, al menos, dos de propios.
- Para las partes intermedias 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.
- Para la parte avanzada, el juego, la estructura es libre mientras cumpla con los objetivos deseados.

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 3.

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. Entendido el funcionamiento de los Listeners es casi directo extraer la información para el juego de la segunda parte; solo quedará implementar el mecanismo de juego.

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 proporcionarán ficheros de entrada en formato texto que se deberán analizar léxica y semánticamente y cuyo AST deberá mostrarse por pantalla (y explicarse debidamente).

Bonificación en la parte 2

Se bonificará con hasta +0,5 puntos sobre la nota de la PL2 a aquellos grupos que implementen y justifiquen mejoras en la segunda parte, como por ejemplo (pero no limitado a):

- Tablero de tamaño variable según la entrada.
- Añadidos al juego: otra categoría de casillas, mapas tridimensionales, etc.
- Posibilidad de dos jugadores.
- Y otras aportaciones originales.

Nota: Es, por lo tanto, posible obtener más de 10 puntos en esta práctica.

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.

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 previsiblemente 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.
- 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.

- **Suspenso**: 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.