Contenido

Capítulo 3

Gramáticas libres de contexto y análisis sintáctico

El análisis gramatical es la tarea de determinar la sintaxis, o estructura, de un programa. Por esta razón también se le conoce como **análisis sintáctico**. La sintaxis de un lenguaje de programación por lo regular se determina mediante las **reglas gramaticales** de una **gramática libre de contexto**, de manera similar como se determina mediante expresiones regulares la estructura léxica de los tokens reconocida por el analizador léxico. En realidad, una gramática libre de contexto utiliza convenciones para nombrar y operaciones muy similares a las correspondientes en las expresiones regulares. Con la única diferencia de que las reglas de una gramática libre de contexto son **recursivas**. Por ejemplo, la estructura de una sentencia if debe permitir en general que otras sentencias if estén anidadas en ella, lo que no se permite en las expresiones regulares. Este cambio aparentemente elemental para el poder de la representación tiene enormes consecuencias. La clase de estructuras reconocible por las gramáticas libres de contexto se incrementa de manera importante en relación con las de las expresiones regulares. Los algoritmos empleados para reconocer estas estructuras también difieren mucho de los algoritmos de análisis léxico, ya que deben utilizar llamadas recursivas o una pila de análisis sintáctico explícitamente administrada. Las estructuras de datos utilizadas para representar la estructura sintáctica de un lenguaje ahora también deben ser recursivas en lugar de lineales (como lo son para lexemas y tokens). La estructura básica empleada es por lo regular alguna clase de árbol, que se conoce como **árbol de análisis gramatical** o **árbol sintáctico**.

Como en el capítulo anterior, necesitamos estudiar la teoría de las gramáticas libres de contexto antes de abordar los algoritmos de análisis sintáctico y los detalles de los analizadores sintácticos reales que utilizan estos algoritmos. Sin embargo, al contrario de lo que sucede con los analizadores léxicos, donde sólo existe, en esencia un método algorítmico (representado por los autómatas finitos), el análisis sintáctico involucra el tener que elegir entre varios métodos diferentes, cada uno de los cuales tiene distintas propiedades y capacidades. De hecho existen dos categorías generales de algoritmos: de **análisis sintáctico descendente** y de **análisis sintáctico ascendente** (por la manera

en que construyen el árbol de análisis gramatical o árbol sintáctico). Pospondremos un análisis detallado de estos métodos de análisis sintáctico para capítulos subsiguientes. En este capítulo describiremos de manera general el proceso de análisis sintáctico y posteriormente estudiaremos la teoría básica de las gramáticas libres de contexto. En la sección final proporcionaremos la sintaxis del lenguaje TINY en términos de una gramática libre de contexto. El lector familiarizado con la teoría de las gramáticas libres de contexto y los árboles sintácticos puede omitir las secciones medias de este capítulo (o utilizarlas como repaso).

3.1 EL PROCESO DEL ANÁLISIS SINTÁCTICO

La tarea del analizador sintáctico es determinar la estructura sintáctica de un programa a partir de los tokens producidos por el analizador léxico y, ya sea de manera explícita o implícita, construir un árbol de análisis gramatical o árbol sintáctico que represente esta estructura. De este modo, se puede ver el analizador sintáctico como una función que toma como su entrada la secuencia de tokens producidos por el analizador léxico y que produce como su salida el árbol sintáctico

$$\textit{secuencia de tokens} \xrightarrow{\text{analizador sintáctico}} \textit{árbol sintáctico}$$

La secuencia de tokens por lo regular no es un parámetro de entrada explícito, pero el analizador sintáctico llama a un procedimiento del analizador léxico, como `getToken`, para obtener el siguiente token desde la entrada a medida que lo necesite durante el proceso de análisis sintáctico. De este modo, la etapa de análisis sintáctico del compilador se reduce a una llamada al analizador léxico de la manera siguiente:

```
syntaxTree = parse();
```

En un compilador de una sola pasada el analizador sintáctico incorporará todas las otras fases de un compilador, incluyendo la generación del código, y no es necesario construir ningún árbol sintáctico explícito (las mismas etapas del analizador sintáctico representarán de manera implícita al árbol sintáctico), y, por consiguiente, una llamada

```
parse();
```

lo hará. Por lo común, un compilador será de múltiples pasadas, en cuyo caso las pasadas adicionales utilizarán el árbol sintáctico como su entrada.

La estructura del árbol sintáctico depende en gran medida de la estructura sintáctica particular del lenguaje. Este árbol por lo regular se define como una estructura de datos dinámica, en la cual cada nodo se compone de un registro cuyos campos incluyen los atributos necesarios para el resto del proceso de compilación (es decir, no sólo por aquellos que calcula el analizador sintáctico). A menudo la estructura de nodos será un registro variante para ahorrar espacio. Los campos de atributo también pueden ser estructuras que se asignen de manera dinámica a medida que se necesite, como una herramienta adicional para ahorrar espacio.

Un problema más difícil de resolver para el analizador sintáctico que para el analizador léxico es el tratamiento de los errores. En el analizador léxico, si hay un carácter que no puede ser parte de un token legal, entonces es suficientemente simple generar un token de error y consumir el carácter problemático. (En cierto sentido, al generar un token de error, el analizador léxico transfiere la dificultad hacia el analizador sintáctico.) Por otra parte, el analizador sintáctico no sólo debe mostrar un mensaje de error, sino que debe **recuperarse** del error y continuar el análisis sintáctico (para encontrar tantos errores como sea posible). En

ocasiones, un analizador sintáctico puede efectuar **reparación de errores**, en la cual infiere una posible versión de código corregida a partir de la versión incorrecta que se le haya presentado. (Esto por lo regular se hace sólo en casos simples.) Un aspecto particularmente importante de la recuperación de errores es la exhibición de mensajes de errores significativos y la reanudación del análisis sintáctico tan próximo al error real como sea posible. Esto no es fácil, puesto que el analizador sintáctico puede no descubrir un error sino hasta mucho después de que el error real haya ocurrido. Como las técnicas de recuperación de errores dependen del algoritmo de análisis sintáctico que se haya utilizado en particular, se aplazará su estudio hasta capítulos subsiguientes.

1. Pero vea más adelante en el capítulo el análisis sobre reglas gramaticales como ecuaciones.