Contenido

Capítulo 4

Análisis sintáctico descendente

Un algoritmo de análisis sintáctico **descendente** analiza una cadena de tokens de entrada mediante la búsqueda de los pasos en una derivación por la izquierda. Un algoritmo así se denomina descendente debido a que el recorrido implicado del árbol de análisis gramatical es un recorrido de preorden y, de este modo, se presenta desde la raíz hacia las hojas (véase la sección 3.3 del capítulo 3). Los analizadores sintácticos descendentes existen en dos formas: **analizadores sintácticos inversos** o **en reversa** y **analizadores sintácticos predictivos**. Un analizador sintáctico predictivo intenta predecir la siguiente construcción en la cadena de entrada utilizando uno o más tokens de búsqueda por adelantado, mientras que un analizador sintáctico inverso intentará las diferentes posibilidades para un análisis sintáctico de la entrada, respaldando una cantidad arbitraria en la entrada si una posibilidad falla. Aunque los analizadores sintácticos inversos son más poderosos que los predictivos, también son mucho más lentos, ya que requieren una cantidad de tiempo exponencial en general y, por consiguiente, son inadecuados para los compiladores prácticos. Aquí no estudiaremos los analizadores sintácticos inversos (pero revise la sección de notas y referencias y los ejercicios para unos cuantos apuntes y sugerencias sobre este tema).

Las dos clases de algoritmos de análisis sintáctico descendente que estudiaremos aquí se denominan **análisis sintáctico descendente recursivo** y **análisis sintáctico LL(1)**. El análisis sintáctico descendente recursivo además de ser muy versátil, es el método más adecuado para un analizador sintáctico manuscrito, por lo tanto, lo estudiaremos primero. El análisis sintáctico LL(1) se abordará después, aunque ya no se usa a menudo en la práctica, es útil estudiarlo como un esquema simple con una pila explícita, y puede servirnos como un preludio para los algoritmos ascendentes más poderosos (pero también más complejos) del capítulo siguiente. Nos ayudará también al formalizar algunos de los problemas que aparecen en el modo descendente recursivo. El método de análisis sintáctico LL(1) debe su nombre a lo siguiente. La primera "L" se refiere al hecho de que se procesa la entrada de izquierda a derecha, del inglés "Left-right" (algunos analizadores sintácticos antiguos empleaban para procesar la entrada de derecha a izquierda, pero eso es poco habitual en la actualidad). La segunda "L" hace referencia al hecho de que rastrea una derivación por la izquierda para la cadena de entrada. El número 1 entre paréntesis significa que se utiliza sólo un símbolo de entrada para predecir la dirección del análisis sintáctico. (También es posible tener análisis sintáctico LL(k), utilizando k símbolos de búsqueda hacia delante. Estudiaremos esto a grandes rasgos más adelante en el capítulo, pero un símbolo de búsqueda hacia delante es el caso más común.)

Tanto el análisis sintáctico descendente recursivo como el análisis sintáctico LL(1) requieren en general del cálculo de conjuntos de búsqueda hacia delante que se conocen como conjuntos **Primero** y **Siguiente**.[1] Puesto que se pueden construir analizadores sin construir estos conjuntos de manera explícita, pospondremos un análisis de los mismos hasta que hayamos introducido los algoritmos básicos. Entonces continuaremos con un análisis de un analizador sintáctico de TINY construido de manera descendente recursiva y cerraremos el capítulo con una descripción de los métodos de recuperación de errores en el análisis sintáctico descendente.

1. Estos conjuntos también son necesarios en algunos de los algoritmos para el análisis sintáctico ascendente que se estudian en el capítulo siguiente.