

---

# Contenido

---

## 1 INTRODUCCIÓN 1

- 1.1 ¿Por qué compiladores? Una breve historia 2
- 1.2 Programas relacionados con los compiladores 4
- 1.3 Proceso de traducción 7
- 1.4 Estructuras de datos principales en un compilador 13
- 1.5 Otras cuestiones referentes a la estructura del compilador 14
- 1.6 Arranque automático y portabilidad 18
- 1.7 Lenguaje y compilador de muestra TINY 21
- 1.8 C-Minus: Un lenguaje para un proyecto de compilador 26
- Ejercicios 27
- Notas y referencias 29

## 2 RASTREO O ANÁLISIS LÉXICO 31

- 2.1 El proceso del análisis léxico 32
- 2.2 Expresiones regulares 34
- 2.3 Autómatas finitos 47
- 2.4 Desde las expresiones regulares hasta los DFA 64
- 2.5 Implementación de un analizador léxico TINY ("Diminuto") 75
- 2.6 Uso de Lex para generar automáticamente un analizador léxico 81
- Ejercicios 91
- Ejercicios de programación 93
- Notas y referencias 94

## 3 GRAMÁTICAS LIBRES DE CONTEXTO Y ANÁLISIS SINTÁCTICO 95

- 3.1 El proceso del análisis sintáctico 96
- 3.2 Gramáticas libres de contexto 97
- 3.3 Árboles de análisis gramatical y árboles sintácticos abstractos 106
- 3.4 Ambigüedad 114
- 3.5 Notaciones extendidas: EBNF y diagramas de sintaxis 123
- 3.6 Propiedades formales de los lenguajes libres de contexto 128
- 3.7 Sintaxis del lenguaje TINY 133
- Ejercicios 138
- Notas y referencias 142

## 4 ANÁLISIS SINTÁCTICO DESCENDENTE 143

- 4.1 Análisis sintáctico descendente mediante método descendente recursivo 144
- 4.2 Análisis sintáctico LL(1) 152
- 4.3 Conjuntos primero y siguiente 168
- 4.4 Un analizador sintáctico descendente recursivo para el lenguaje TINY 180
- 4.5 Recuperación de errores en analizadores sintácticos descendentes 183
  - Ejercicios 189
  - Ejercicios de programación 193
  - Notas y referencias 196

## 5 ANÁLISIS SINTÁCTICO ASCENDENTE 197

- 5.1 Perspectiva general del análisis sintáctico ascendente 198
- 5.2 Autómatas finitos de elementos LR(0) y análisis sintáctico LR(0) 201
- 5.3 Análisis sintáctico SLR(1) 210
- 5.4 Análisis sintáctico LALR(1) y LR(1) general 217
- 5.5 Yacc: un generador de analizadores sintácticos LALR(1) 226
- 5.6 Generación de un analizador sintáctico TINY utilizando Yacc 243
- 5.7 Recuperación de errores en analizadores sintácticos ascendentes 245
  - Ejercicios 250
  - Ejercicios de programación 254
  - Notas y referencias 256

## 6 ANÁLISIS SEMÁNTICO 257

- 6.1 Atributos y gramáticas con atributos 259
- 6.2 Algoritmos para cálculo de atributos 270
- 6.3 La tabla de símbolos 295
- 6.4 Tipos de datos y verificación de tipos 313
- 6.5 Un analizador semántico para el lenguaje TINY 334
  - Ejercicios 339
  - Ejercicios de programación 342
  - Notas y referencias 343

## 7 AMBIENTES DE EJECUCIÓN 345

- 7.1 Organización de memoria durante la ejecución del programa 346
- 7.2 Ambientes de ejecución completamente estáticos 349
- 7.3 Ambientes de ejecución basados en pila 352
- 7.4 Memoria dinámica 373
- 7.5 Mecanismos de paso de parámetros 381

- 7.6 Un ambiente de ejecución para el lenguaje TINY 386
  - Ejercicios 388
  - Ejercicios de programación 395
  - Notas y referencias 396

## 8 GENERACIÓN DE CÓDIGO 397

- 8.1 Código intermedio y estructuras de datos para generación de código 398
- 8.2 Técnicas básicas de generación de código 407
- 8.3 Generación de código de referencias de estructuras de datos 416
- 8.4 Generación de código de sentencias de control y expresiones lógicas 428
- 8.5 Generación de código de llamadas de procedimientos y funciones 436
- 8.6 Generación de código en compiladores comerciales: dos casos de estudio 443
- 8.7 TM: Una máquina objetivo simple 453
- 8.8 Un generador de código para el lenguaje TINY 459
- 8.9 Una visión general de las técnicas de optimización de código 468
- 8.10 Optimizaciones simples para el generador de código de TINY 481
  - Ejercicios 484
  - Ejercicios de programación 488
  - Notas y referencias 489

## Apéndice A: PROYECTO DE COMPILADOR 491

- A.1 Convenciones léxicas de C— 491
- A.2 Sintaxis y semántica de C— 492
- A.3 Programas de muestra en C— 496
- A.4 Un ambiente de ejecución de la Máquina Tiny para el lenguaje C— 497
- A.5 Proyectos de programación utilizando C— y TM 500

## Apéndice B: LISTADO DEL COMPILADOR TINY 502

## Apéndice C: LISTADO DEL SIMULADOR DE LA MÁQUINA TINY 545

## Bibliografía 558

## Índice 562

# Análisis sintáctico descendente

- |     |   |     |  |
|-----|---|-----|--|
| 4.1 | Análisis sintáctico descendente mediante método descendente recursivo | 4.4 | Un analizador sintáctico descendente recursivo para el lenguaje TINY |
| 4.2 | Análisis sintáctico LL(1)   | 4.5 | Recuperación de errores en analizadores sintácticos descendentes     |
| 4.3 | Conjuntos Primero y Siguiente   |     |  |

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**.<sup>1</sup> 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.