
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

3.5 NOTACIONES EXTENDIDAS: EBNF Y DIAGRAMAS DE SINTAXIS

3.5.1 Notación EBNF

Las construcciones repetitivas y opcionales son muy comunes en los lenguajes de programación, lo mismo que las reglas de gramática de la BNF. Por lo tanto, no debería sorprendernos que la notación BNF se extienda en ocasiones con el fin de incluir notaciones especiales para estas dos situaciones. Estas extensiones comprenden una notación que se denomina **BNF extendida**, o **EBNF** (por las siglas del término en inglés).

Considere en primer lugar el caso de la repetición, como el de las secuencias de sentencias. Vimos que la repetición se expresa por la recursión en las reglas gramaticales y que se puede utilizar tanto la recursión por la izquierda como la recursión por la derecha, indicadas por las reglas genéricas

$$A \rightarrow A \alpha \mid \beta \quad (\text{recursiva por la izquierda})$$

y

$$A \rightarrow \alpha A \mid \beta \quad (\text{recursiva por la derecha})$$

donde α y β son cadenas arbitrarias de terminales y no terminales y donde en la primera regla β no comienza con A y en la segunda β no finaliza con A .

Se podría emplear la misma notación para la repetición que la que se utiliza para las expresiones regulares, a saber, el asterisco $*$ (también conocido como cerradura de Kleene en expresiones regulares). Entonces estas dos reglas se escribirían como las reglas no recursivas

$$A \rightarrow \beta \alpha^*$$

y

$$A \rightarrow \alpha^* \beta$$

En vez de eso EBNF prefiere emplear llaves $\{ \dots \}$ para expresar la repetición (así hace clara la extensión de la cadena que se va a repetir), y escribimos

$$A \rightarrow \beta \{ \alpha \}$$

y

$$A \rightarrow \{ \alpha \} \beta$$

para las reglas.

El problema con cualquier notación de repetición es que oculta cómo se construye el árbol de análisis gramatical, pero, como ya se expuso, a menudo no importa. Tomemos por ejemplo el caso de las secuencias de sentencias (ejemplo 3.9). Escribimos la gramática como sigue, en forma recursiva por la derecha:

$$\begin{aligned} \text{secuencia-sent} &\rightarrow \text{sent} ; \text{secuencia-sent} \mid \text{sent} \\ \text{sent} &\rightarrow s \end{aligned}$$

Esta regla tiene la forma $A \rightarrow \alpha A \mid \beta$, con $A = \text{secuencia-sent}$, $\alpha = \text{sent} ;$ y $\beta = \text{sent}$. En EBNF esto aparecería como

$$\text{secuencia-sent} \rightarrow \{ \text{sent} ; \} \text{sent} \quad (\text{forma recursiva por la derecha})$$

También podríamos haber usado una regla recursiva por la izquierda y obtenido la EBNF

$$\text{secuencia-sent} \rightarrow \text{sent} \{ ; \text{sent} \} \quad (\text{forma recursiva por la izquierda})$$

De hecho, la segunda forma es la que generalmente se utiliza (por razones que analizaremos en el siguiente capítulo).

Un problema más importante se presenta cuando se involucra la asociatividad, como ocurre con las operaciones binarias como la resta y la división. Por ejemplo, consideremos la primera regla gramatical en la gramática de expresión simple de la anterior subsección:

$$\text{exp} \rightarrow \text{exp opsuma term} \mid \text{term}$$

Esto tiene la forma $A \rightarrow A \alpha \mid \beta$, con $A = \text{exp}$, $\alpha = \text{opsuma term}$ y $\beta = \text{term}$. De este modo, escribimos esta regla en EBNF como

$$\text{exp} \rightarrow \text{term} \{ \text{opsuma term} \}$$

Ahora también debemos suponer que esto implica asociatividad por la izquierda, aunque la regla misma no lo establezca explícitamente. Debemos suponer que una regla asociativa por la derecha estaría implicada al escribir

$$\text{exp} \rightarrow \{ \text{term opsuma} \} \text{term}$$

pero éste no es el caso. En cambio, una regla recursiva por la derecha como

$$\text{secuencia-sent} \rightarrow \text{sent} ; \text{secuencia-sent} \mid \text{sent}$$

se visualiza como si fuera una *sentencia* seguida opcionalmente por un signo de punto y coma y una *secuencia-sent*.

Las construcciones opcionales en EBNF se indican encerrándolas entre corchetes [...]. Esto es similar en esencia a la convención de la expresión regular de colocar un signo de interrogación después de una parte opcional, pero tiene la ventaja de encerrar la parte opcional sin requerir paréntesis. Por ejemplo, las reglas gramaticales para las sentencias *if* con partes opcionales *else* (ejemplos 3.4 y 3.6) se escribirían en EBNF como se describe a continuación:

$$\begin{aligned} \text{sentencia} &\rightarrow \text{sent-if} \mid \text{otro} \\ \text{sent-if} &\rightarrow \text{if } (\text{exp}) \text{sentencia} \mid \text{else } \text{sentencia} \mid \\ \text{exp} &\rightarrow 0 \mid 1 \end{aligned}$$

Una regla recursiva por la derecha tal como

$$\text{secuencia-sent} \rightarrow \text{sent} ; \text{secuencia-sent} \mid \text{sent}$$

también se escribe como

$$\text{secuencia-sent} \rightarrow \text{sent} [; \text{secuencia-sent}]$$

(compare esto con el uso de las llaves anterior para escribir esta regla en forma recursiva por la izquierda).

Si deseáramos escribir una operación aritmética tal como la suma en forma asociativa derecha, escribiríamos

$$\text{exp} \rightarrow \text{term} [\text{opsuma } \text{exp}]$$

en lugar de utilizar llaves.

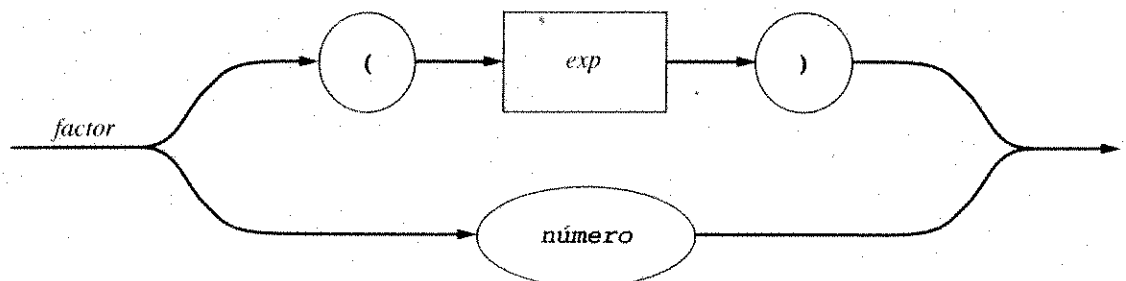
3.5.2 Diagramas de sintaxis

Las representaciones gráficas para simbolizar de manera visual las reglas de la EBNF se denominan **diagramas de sintaxis**. Se componen de cajas que representan terminales y no terminales, líneas con flechas que simbolizan secuencias y selecciones, y etiquetas de no terminales para cada diagrama que representan la regla gramatical que define ese no terminal. Para indicar terminales en un diagrama se emplea una caja de forma oval o redonda, mientras que para indicar no terminales se utiliza una caja rectangular o cuadrada.

Considere como ejemplo la regla gramatical

$$\text{factor} \rightarrow (\text{exp}) \mid \text{número}$$

Ésta se escribe como un diagrama sintáctico de la siguiente manera:

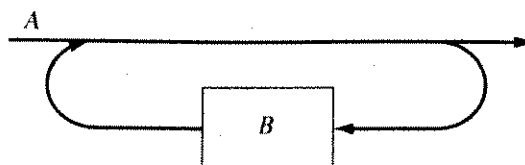


Advierta que *factor* no está situado en una caja, pero se utiliza como una etiqueta para el diagrama de sintaxis, lo cual indica que el diagrama representa la definición de la estructura de ese nombre. Advierta también el uso de las líneas con flechas para indicar la selección y la secuenciación.

Los diagramas de sintaxis se escriben desde la EBNF más que desde la BNF, de modo que necesitamos diagramas que representen construcciones de repetición y opcionales. Dada una repetición tal como

$$A \rightarrow \{ B \}$$

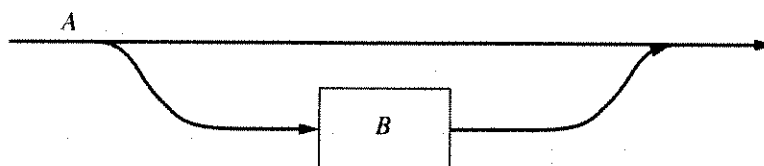
el diagrama de sintaxis correspondiente se dibuja por lo regular como sigue:



Advierta que el diagrama debe tener en cuenta que no aparezca ninguna B .
Una construcción opcional tal como

$$A \rightarrow [B]$$

se dibuja como



Concluiremos nuestro análisis de los diagramas de sintaxis con algunos ejemplos en los que se utilizan ejemplos anteriores de EBNF.

Ejemplo 3.10

Considere nuestro ejemplo de ejecución de expresiones aritméticas simples. Éste tiene la BNF (incluyendo asociatividad y precedencia).

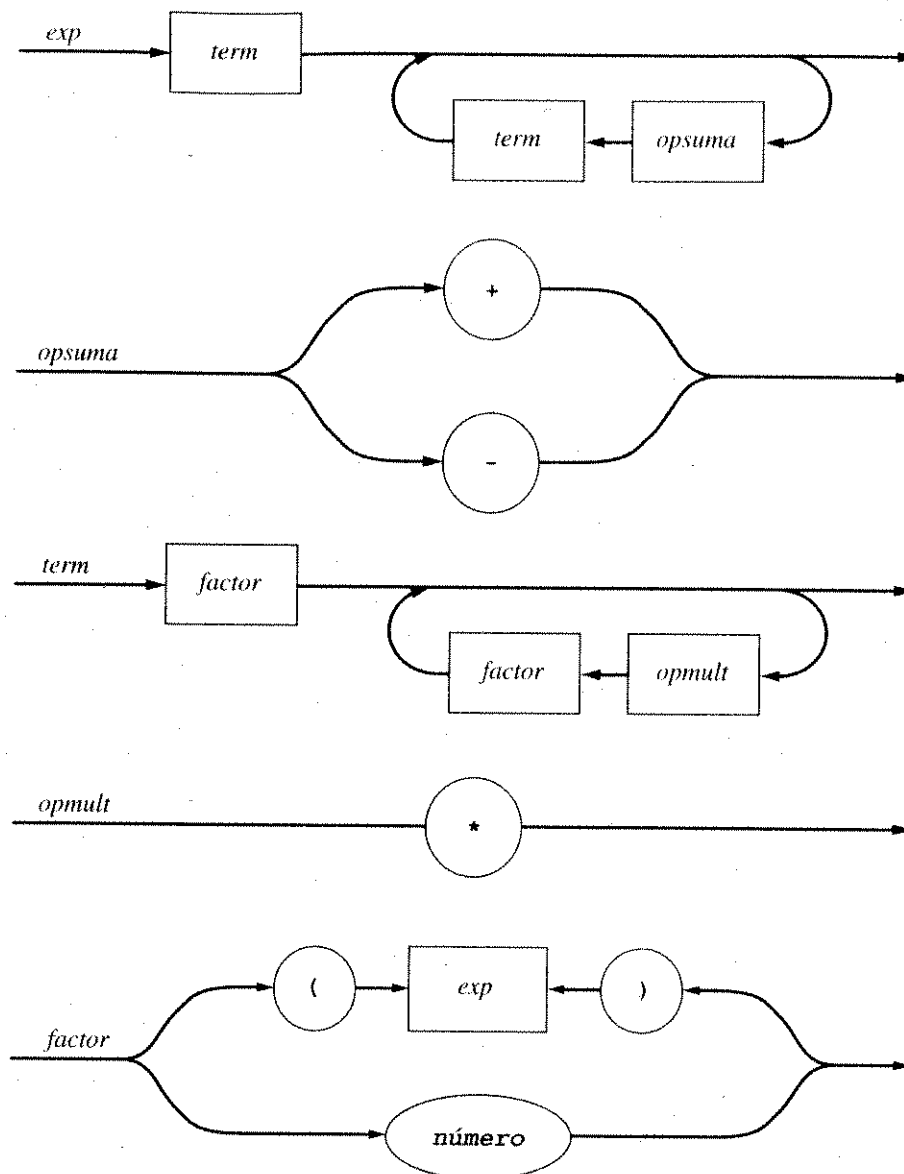
$$\begin{aligned} \text{exp} &\rightarrow \text{exp opsuma term} \mid \text{term} \\ \text{opsuma} &\rightarrow + \mid - \\ \text{term} &\rightarrow \text{term opmult factor} \mid \text{factor} \\ \text{opmult} &\rightarrow * \\ \text{factor} &\rightarrow (\text{exp}) \mid \text{número} \end{aligned}$$

La EBNF correspondiente es

$$\begin{aligned} \text{exp} &\rightarrow \text{term} \{ \text{opsuma term} \} \\ \text{opsuma} &\rightarrow + \mid - \\ \text{term} &\rightarrow \text{factor} \{ \text{opmult factor} \} \\ \text{opmult} &\rightarrow * \\ \text{factor} &\rightarrow (\text{exp}) \mid \text{número} \end{aligned}$$

Los diagramas de sintaxis correspondientes se proporcionan en la figura 3.4 (el diagrama de sintaxis para *factor* se dio con anterioridad).

Figura 3.4
Diagramas de sintaxis para
la gramática del ejemplo 3.10



Ejemplo 3.11

Considere la gramática de las sentencias if simplificadas del ejemplo 3.4 (página 103). Ésta tiene la BNF

$$\begin{aligned}
 \text{sentencia} &\rightarrow \text{sent-if} \mid \text{otro} \\
 \text{sent-if} &\rightarrow \text{if (exp) sentencia} \\
 &\quad \mid \text{if (exp) sentencia else sentencia} \\
 \text{exp} &\rightarrow 0 \mid 1
 \end{aligned}$$

y la EBNF

$$\begin{aligned}
 \text{sentencia} &\rightarrow \text{sent-if} \mid \text{otro} \\
 \text{sent-if} &\rightarrow \text{if (exp) sentencia} [\text{else sentencia}] \\
 \text{exp} &\rightarrow 0 \mid 1
 \end{aligned}$$

Los diagramas de sintaxis correspondientes se ofrecen en la figura 3.5.

Figura 3.5

Diagramas de sintaxis para
la gramática del ejemplo 3.11