ST0270 Lenguajes Formales y Compiladores
Análisis sintáctico

Andrés Sicard Ramírez

Universidad EAFIT

Semestre 2024-1

Preliminares

Referencias

Las referencias principales para estas diapositivas son las secciones § 4.4.1–4.4.5 de: inglés (Aho et al. 2006), español (Aho et al. 2008).

Convenciones

(i) Los números asignados a los definiciones, ejemplos, ejercicios, páginas y teoremas en estas diapositivas corresponden a los números asignados en (Aho et al. 2008).

(ii) El conjunto potencia de un conjunto A es denotado 2^A.

(iii) Una gramática (N, Σ, P, S) será denotada $(\mathcal{N}, \mathcal{T}, \mathcal{P}, S)$.

(iv) La relación $\alpha \xrightarrow[G]{*} \beta$ será denotada $\alpha \overset{*}{\Rightarrow} \beta$.

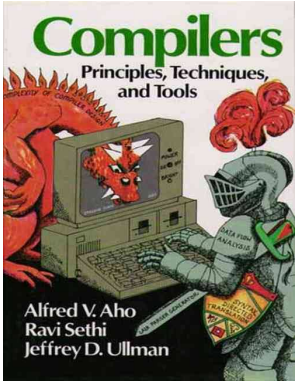
(v) Para escribir los elementos de las gramáticas seguiremos las convenciones en la sección § 4.2.2.

Preliminares

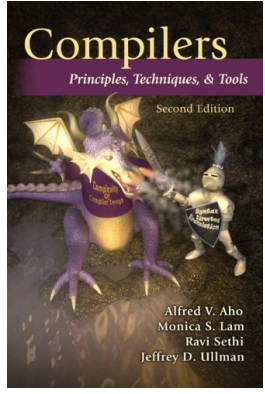
Agradecimientos

Agradezco a mi colega Oscar Eduardo García Quintero por señalarme algunas correcciones en una versión anterior de estas diapositivas.

El libro del drágon



(Primera edición, 1986)
(Primera edición en español, 1990)



(Segunda edición, 2006)
(Segunda edición en español, 2008)

Introducción

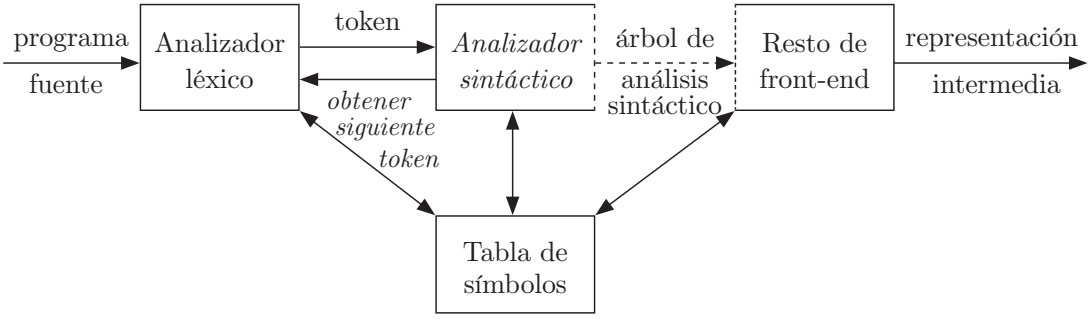
Descripción

En la introducción al curso mencionamos que un analizador sintáctico (*parser*) es una de las partes de un compilador:

> *El **análisis sintáctico (parsing)** es el problema de tomar una cadena de terminales y averiguar cómo derivarla a partir del símbolo inicial de la gramática, y si no puede derivarse a partir de este símbolo, entonces hay que reportar los errores dentro de la cadena. (pág. 45)*

Introducción

Posición del analizador sintáctico en el *front-end* (fig. 4.1)



Introducción

Descripción

*Un **árbol de análisis sintáctico (parser tree)** es una representación gráfica de una derivación que filtra el orden en el que se aplican las producciones para sustituir los no terminales. Cada nodo interior de un árbol de análisis sintáctico representa la aplicación de una producción. El nodo interior se etiqueta con el no terminal A en el encabezado de la producción; los hijos del nodo se etiquetan, de izquierda a derecha, mediante los símbolos en el cuerpo de la producción por la que se sustituyó esta A durante la derivación. (pág. 201)*

Introducción

Ejemplo (§ 2.4.1)

$$
\begin{array}{rcl}
\mathit{instr} & \rightarrow & \textbf{expr}\ ; \\
& | & \textbf{if}\ (\ \textbf{expr}\ )\ \mathit{instr} \\
& | & \textbf{for}\ (\ \mathit{expropc}\ ;\ \mathit{expropc}\ ;\ \mathit{expropc}\ )\ \mathit{instr} \\
& | & \textbf{otras} \\
 & & \\
\mathit{expropc} & \rightarrow & \epsilon \\
& | & \textbf{expr}
\end{array}
$$

Introducción

Ejemplo (§ 2.4.1)

$$
\begin{array}{rcl}
instr & \rightarrow & \textbf{expr}\ ; \\
 & | & \textbf{if}\ (\ \textbf{expr}\ )\ instr \\
 & | & \textbf{for}\ (\ expropc\ ;\ expropc\ ;\ expropc\ )\ instr \\
 & | & \textbf{otras} \\
 & & \\
expropc & \rightarrow & \epsilon \\
 & | & \textbf{expr}
\end{array}
$$

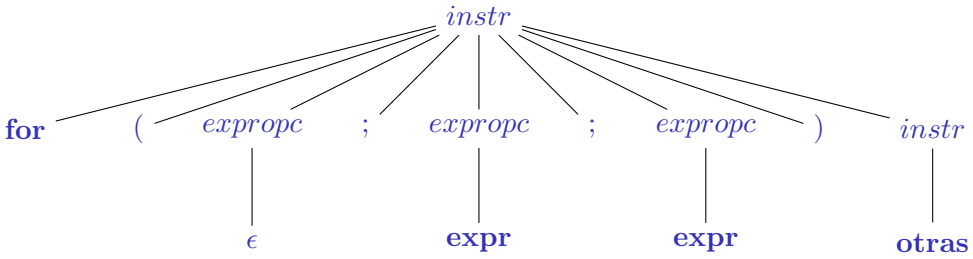
De acuerdo a las convenciones para las gramáticas de la sección § 4.2.2, los componentes de la gramática $(\mathcal{N}, \mathcal{T}, \mathcal{P}, S)$ son:

$$
\begin{aligned}
\mathcal{N} &= \{instr, expropc\}, \\
\mathcal{T} &= \{\textbf{expr}, \textbf{if}, \textbf{for}, \textbf{otras}, ;,), (, \}, \\
S &= instr.
\end{aligned}
$$

Introducción

Ejemplo (§ 2.4.1, continuación)

Árbol de análisis sintáctico (*parser tree*) para la palabra **for** (; **expr** ; **expr**) **otras**.



Introducción

Métodos de análisis sintácticos

Los siguientes métodos hacen referencia a la forma en la que se construye el árbol de análisis sintáctico (*parser tree*).

(i) Métodos descendentes (*top-down*)
La construcción empieza en la raíz y procede hacia las hojas.

(ii) Métodos ascendentes (*bottom-up*)
La construcción empieza en las hojas y procede hacia la raíz.

Análisis sintáctico descendente

Descripción

El **análisis sintáctico descendente (*top-down parsing*)** puede entenderse como:

(i) la construcción de un árbol de análisis sintáctico (*parser tree*) para una palabra de entrada empleando un recorrido en preorden (una clase de recorrido primero en profundidad (*depth-first*) en la cual una acción se realiza cuando se visita el nodo por primera vez), o equivalentemente,

(ii) buscar una derivación más a la izquierda para una palabra de entrada.

Análisis sintáctico descendente

Algunos métodos (algoritmos) de análisis sintáctico descendente

(i) Análisis sintáctico descendente recursivo (*recursive-descent parsing*)
- Analizadores sintácticos predictivos (*predictive parsers*)

(ii) Analizadores sintáctico no recursivos (*nonrecursive parsing*)

Análisis sintáctico descendente

Descripción

El **análisis sintáctico descendente recursivo (*recursive-descent parsing*)** está formado por un conjunto de procedimientos, uno por cada símbolo no terminal (Fig. 4.13).

```
     void A() {
1)       Elegir una producción A, A → X1X2 ··· Xk;
2)       for ( i = 1 a k ) {
3)           if ( Xi es un no terminal )
4)               llamar al procedimiento Xi();
5)           else if ( Xi es igual al símbolo de entrada actual a )
6)               avanzar la entrada hasta el siguiente símbolo;
7)           else /* ha ocurrido un error */;
         }
     }
```

Análisis sintáctico descendente

Ejemplo (§ 2.4.1)

Construcción de un árbol de análisis sintáctico (*parser tree*) empleando el algoritmo de análisis sintáctico descendente recursivo (*recursive-descent parsing*) para la palabra **for** (; **expr** ; **expr**) **otras**.

(continua en la próxima diapositiva)

Análisis sintáctico descendente

Ejemplo (§ 2.4.1, continuación)

```
void instr() {
    switch ( preanálisis ) {
    case expr:
        coincidir(expr); coincidir(';'); break;
    case if:
        coincidir(if); coincidir('('); coincidir(expr); coincidir(')'); instr();
        break;
    case for;
        coincidir(for); coincidir('(');
        expropc(); coincidir(';'); expropc(); coincidir(';'); expropc();
        coincidir(')'); instr(); break;
    case otras;
        coincidir(otras); break;
    default:
        reportar("error de sintaxis");
    }
}
```

Análisis sintáctico descendente

Ejemplo (§ 2.4.1, continuación)

```
void expropc() {
        if ( preanálisis == expr ) coincidir(expr);
}

void coincidir(terminal t) {
        if ( preanálisis == t ) preanálisis = siguienteTerminal;
        else reportar("error de sintaxis");
}
```

Análisis sintáctico descendente

Ejemplo 4.29

Construcción de un árbol de análisis sintáctico (*parser tree*) empleando el algoritmo de análisis sintáctico descendente recursivo (*recursive-descent parsing*) para la gramática

$$\begin{array}{lcl} S & \to & c\ A\ d \\ A & \to & a\ b \mid a \end{array}$$

y la palabra cad.

En el tablero.

Análisis sintáctico descendente

Ejemplo 4.29

Construcción de un árbol de análisis sintáctico (*parser tree*) empleando el algoritmo de análisis sintáctico descendente recursivo (*recursive-descent parsing*) para la gramática

$$\begin{array}{lcl} S & \to & c\,A\,d \\ A & \to & a\,b \mid a \end{array}$$

y la palabra cad.

En el tablero.

Como mostró el ejemplo anterior, en algunas ocasiones el análisis sintáctico descendente recursivo puede requerir un rastreo hacia atrás (*backtracking*).

Conjuntos Primero y Siguiente

Introducción

Sea $(\mathcal{N}, \mathcal{T}, \mathcal{P}, S)$ una gramática y sea $\$ \notin \mathcal{N} \cup \mathcal{T}$ un símbolo que delimita a la derecha una palabra. Vamos a definir dos funciones

$$\begin{aligned} \mathrm{Pr} &: (\mathcal{N} \cup \mathcal{T})^* \to 2^{\mathcal{T} \cup \{\epsilon\}} && \text{(Primero)}, \\ \mathrm{Sig} &: \mathcal{N} \to 2^{\mathcal{T} \cup \{\$\}} && \text{(Siguiente)}. \end{aligned}$$

Sean $\alpha \in (\mathcal{N} \cup \mathcal{T})^*$ y $A \in \mathcal{N}$. Los conjuntos $\mathrm{Pr}(\alpha)$ y $\mathrm{Sig}(A)$ serán usados por los analizadores sintácticos para seleccionar que producción de la gramática deben procesar.

Conjuntos Primero y Siguiente

Descripción

Sea $\alpha \in (\mathcal{N} \cup \mathcal{T})^*$ y sea la función $\mathrm{Pr} : (\mathcal{N} \cup \mathcal{T})^* \to 2^{\mathcal{T} \cup \{\epsilon\}}$.

(i) El conjunto $\mathrm{Pr}(\alpha)$ es el conjunto de símbolos terminales que comienzan cadenas derivadas desde α.

Conjuntos Primero y Siguiente

Descripción

Sea $\alpha \in (\mathcal{N} \cup \mathcal{T})^*$ y sea la función $\mathrm{Pr} : (\mathcal{N} \cup \mathcal{T})^* \to 2^{\mathcal{T} \cup \{\epsilon\}}$.

(i) El conjunto $\mathrm{Pr}(\alpha)$ es el conjunto de símbolos terminales que comienzan cadenas derivadas desde α.

(ii) Además, si $\alpha \stackrel{*}{\Rightarrow} \epsilon$ entonces $\epsilon \in \mathrm{Pr}(\alpha)$.

Conjuntos Primero y Siguiente

Descripción

Sea $A \in \mathcal{N}$ y sea la función $\mathrm{Sig} : \mathcal{N} \to 2^{\mathcal{T} \cup \{\$\}}$.

(i) El conjunto $\mathrm{Sig}(A)$ es el conjunto de símbolos terminales que pueden aparecer inmediatamente a la derecha de A en una forma sentencial.

Conjuntos Primero y Siguiente

Descripción

Sea $A \in \mathcal{N}$ y sea la función $\mathrm{Sig} : \mathcal{N} \to 2^{\mathcal{T} \cup \{\$\}}$.

(i) El conjunto $\mathrm{Sig}(A)$ es el conjunto de símbolos terminales que pueden aparecer inmediatamente a la derecha de A en una forma sentencial.

(ii) Es decir, $\mathrm{Sig}(A)$ es el conjunto de símbolos terminales a tales que exista una derivación

$$S \overset{*}{\Rightarrow} \alpha A a \beta,$$

para algún $\alpha, \beta \in (\mathcal{N} \cup \mathcal{T})^*$.

Conjuntos Primero y Siguiente

Descripción

Sea $A \in \mathcal{N}$ y sea la función $\mathrm{Sig} : \mathcal{N} \to 2^{\mathcal{T} \cup \{\$\}}$.

(i) El conjunto $\mathrm{Sig}(A)$ es el conjunto de símbolos terminales que pueden aparecer inmediatamente a la derecha de A en una forma sentencial.

(ii) Es decir, $\mathrm{Sig}(A)$ es el conjunto de símbolos terminales a tales que exista una derivación

$$S \overset{*}{\Rightarrow} \alpha A a \beta,$$

para algún $\alpha, \beta \in (\mathcal{N} \cup \mathcal{T})^*$.

(iii) Además, si A puede ser el símbolo más a la derecha de una forma sentencial entonces $\$ \in \mathrm{Sig}(A)$.

Conjuntos Primero y Siguiente

Computación de $\Pr(X)$

Sean $X, Y_1, Y_2, \ldots Y_k \in \mathcal{N} \cup \mathcal{T}$. Para computar $\Pr(X)$ aplicamos los siguientes pasos hasta que no más símbolos terminales o ϵ puedan ser adicionados a cualquier conjunto Primero:

1) Si $X \in \mathcal{T}$ entonces $\Pr(X) = \{X\}$.
2) Si $X \in \mathcal{N}$ y $X \to Y_1 Y_2 \cdots Y_k$ entonces
   a) Si $a \in \Pr(Y_i)$ y $\epsilon \in \bigcap_{j=1}^{i-1} \Pr(Y_j)$, entonces adicionar el símbolo terminal a al conjunto $\Pr(X)$.
   b) Si $\epsilon \in \bigcap_{j=1}^{k} \Pr(Y_j)$, entonces adicionar ϵ al conjunto $\Pr(X)$.
   c) Si $X \to \epsilon$ entonces adicionar ϵ al conjunto $\Pr(X)$.

Conjuntos Primero y Siguiente

Computación de $\mathrm{Pr}(\alpha)$

Sean $\alpha = X_1 X_2, \ldots X_n \in (\mathcal{N} \cup \mathcal{T})^*$. Para computar $\mathrm{Pr}(\alpha)$:

1) Adicionar al conjunto $\mathrm{Pr}(\alpha)$ todos los símbolos diferentes a ϵ de $\mathrm{Pr}(X_1)$.
2) Si $\epsilon \in \mathrm{Pr}(X_1)$ entonces adicionar al conjunto $\mathrm{Pr}(\alpha)$ todos los símbolos diferentes a ϵ de $\mathrm{Pr}(X_2)$.
3) Si $\epsilon \in \mathrm{Pr}(X_1) \cap \mathrm{Pr}(X_2)$ entonces adicionar al conjunto $\mathrm{Pr}(\alpha)$ todos los símbolos diferentes a ϵ de $\mathrm{Pr}(X_3)$. Y así, sucesivamente.
4) Finalmente, si $\epsilon \in \bigcap_{i=1}^{n} \mathrm{Pr}(X_i)$, entonces adicionar ϵ al conjunto $\mathrm{Pr}(\alpha)$.

Conjuntos Primero y Siguiente

Computación de $\mathrm{Sig}(A)$

Para computar $\mathrm{Sig}(A)$, para todos los símbolos terminales A, aplicamos los siguientes pasos hasta que no pueda adicionar nada más a cualquier conjunto Siguiente:

1) Adicionar al conjunto $\mathrm{Sig}(S)$ el símbolo $\$$.
2) Si hay una producción $A \to \alpha B \beta$, entonces adicionar al conjunto $\mathrm{Sig}(B)$ el conjunto $\mathrm{Pr}(\beta)$ excepto ϵ.
3) Si hay una producción $A \to \alpha B$ entonces adicionar al conjunto $\mathrm{Sig}(B)$ el conjunto $\mathrm{Sig}(A)$.
4) Si hay una producción $A \to \alpha B \beta$, con $\epsilon \in \mathrm{Pr}(\beta)$, entonces adicionar al conjunto $\mathrm{Sig}(B)$ el conjunto $\mathrm{Sig}(A)$.

Conjuntos Primero y Siguiente

Ejemplo 4.30

Calculemos los conjuntos Primero y Siguiente para los símbolos no terminales de la siguiente gramática:

$$
\begin{array}{lcll}
E & \rightarrow & T\ E' & (1) \\
E' & \rightarrow & +\ T\ E' & (2) \\
E' & \rightarrow & \epsilon & (3) \\
T & \rightarrow & F\ T' & (4) \\
T' & \rightarrow & *\ F\ T' & (5) \\
T' & \rightarrow & \epsilon & (6) \\
F & \rightarrow & (\ E\ ) & (7) \\
F & \rightarrow & \mathbf{id} & (8)
\end{array}
$$

En el tablero.

Conjuntos Primero y Siguiente

Ejemplo 4.30

Calculemos los conjuntos Primero y Siguiente para los símbolos no terminales de la siguiente gramática:

$$
\begin{array}{lcll}
E & \to & T\ E' & (1) \\
E' & \to & +\ T\ E' & (2) \\
E' & \to & \epsilon & (3) \\
T & \to & F\ T' & (4) \\
T' & \to & *\ F\ T' & (5) \\
T' & \to & \epsilon & (6) \\
F & \to & (\ E\ ) & (7) \\
F & \to & \mathbf{id} & (8)
\end{array}
$$

En el tablero.

(continua en la próxima diapositiva)

Conjuntos Primero y Siguiente

Ejemplo 4.30 (continuación)

$$\begin{aligned}
\mathrm{Pr}(E) &= \{(, \mathbf{id}\}, & \mathrm{Sig}(E) &= \{), \$\},\\
\mathrm{Pr}(E') &= \{+, \epsilon\}, & \mathrm{Sig}(E') &= \{), \$\},\\
\mathrm{Pr}(T) &= \{(, \mathbf{id}\}, & \mathrm{Sig}(T) &= \{+,), \$\},\\
\mathrm{Pr}(T') &= \{*, \epsilon\}, & \mathrm{Sig}(T') &= \{+,), \$\},\\
\mathrm{Pr}(F) &= \{(, \mathbf{id}\}, & \mathrm{Sig}(F) &= \{+, *,), \$\}.
\end{aligned}$$

Gramáticas LL(1)

Introducción

Un analizador sintáctico predictivo (*predictive parser*) es un analizador sintáctico descendente recursivo (*recursive-descent parsing*) que no requiere un rastreo hacia atrás (*backtracking*).

Un analizador sintáctico predictivo (*predictive parser*) se puede construir para las gramáticas LL(1).

Gramáticas LL(1)

LL(1): «*The first* L *in* LL(1) *stands for scanning the input from left to right, the second* L *for producing a leftmost derivation, and the* 1 *for using one input symbol of lookahead at each step to make parsing action decisions.*» (pág. 222).

Gramáticas $\mathrm{LL}(1)$

Definición

Una gramática es $\mathrm{LL}(1)$ sii para cada par de producciones $A \to \alpha \mid \beta$, con $\alpha \neq \beta$, se cumplen las siguientes condiciones:

(i) Para ningún símbolo terminal a, tanto α como β derivan cadenas que empiecen con a.

(ii) Sólo α o β, pero no ambas, puede derivar la cadena vacía.

(iii)
- a) Si $\beta \overset{*}{\Rightarrow} \epsilon$ entonces α no deriva a ninguna cadena que empiece con un símbolo terminal en $\mathrm{Sig}(A)$.
- b) Si $\alpha \overset{*}{\Rightarrow} \epsilon$ entonces β no deriva a ninguna cadena que empiece con un símbolo terminal en $\mathrm{Sig}(A)$.

(continua en la próxima diapositiva)

Gramáticas $\mathrm{LL}(1)$

Definición (continuación)

Equivalentemente,

una gramática es $\mathrm{LL}(1)$ sii para cada par de producciones $A \to \alpha \mid \beta$, con $\alpha \neq \beta$, se cumplen las siguientes condiciones:

$$\mathrm{Pr}(\alpha) \cap \mathrm{Pr}(\beta) = \emptyset, \qquad \text{(i,ii)}$$

Si $\epsilon \in \mathrm{Pr}(\beta)$, entonces $\mathrm{Pr}(\alpha) \cap \mathrm{Sig}(A) = \emptyset$, (iii.a)

Si $\epsilon \in \mathrm{Pr}(\alpha)$, entonces $\mathrm{Pr}(\beta) \cap \mathrm{Sig}(A) = \emptyset$, (iii.b)

Gramáticas LL(1)

Algoritmo 4.31: Construcción de una tabla de análisis sintáctico predictivo
Entrada: Gramática $(\mathcal{N}, \mathcal{T}, \mathcal{P}, S)$
Salida: Tabla de análisis sintáctico $M(A, a)$

```
1  for cada producción A → α ∈ P do
2  |   for cada a ∈ Pr(α) do
3  |   |   M[A, a] ← M[A, a] ∪ {A → α}
4  |   end
5  |   if ϵ ∈ Pr(α) then
6  |   |   for cada terminal b ∈ Sig(A) do
7  |   |   |   M[A, b] ← M[A, b] ∪ {A → α}
8  |   |   end
9  |   end
10 |   if ϵ ∈ Pr(α) ∧ $ ∈ Sig(A) then
11 |   |   M[A, $] ← M[A, $] ∪ {A → α}
12 |   end
13 end
```

Gramáticas LL(1)

Ejemplo 4.32

Construcción de la tabla de análisis sintáctico predictivo $M(A, a)$ empleando el algoritmo 4.31 para la siguiente gramática:

$$
\begin{array}{lcll}
E & \rightarrow & T\ E' & (1) \\
E' & \rightarrow & +\ T\ E' & (2) \\
E' & \rightarrow & \epsilon & (3) \\
T & \rightarrow & F\ T' & (4) \\
T' & \rightarrow & *\ F\ T' & (5) \\
T' & \rightarrow & \epsilon & (6) \\
F & \rightarrow & (\ E\ ) & (7) \\
F & \rightarrow & \mathbf{id} & (8)
\end{array}
$$

En el tablero.

Gramáticas LL(1)

Ejemplo 4.32

Construcción de la tabla de análisis sintáctico predictivo $M(A, a)$ empleando el algoritmo 4.31 para la siguiente gramática:

$$
\begin{array}{lcll}
E & \rightarrow & T\ E' & (1) \\
E' & \rightarrow & +\ T\ E' & (2) \\
E' & \rightarrow & \epsilon & (3) \\
T & \rightarrow & F\ T' & (4) \\
T' & \rightarrow & *\ F\ T' & (5) \\
T' & \rightarrow & \epsilon & (6) \\
F & \rightarrow & (\ E\ ) & (7) \\
F & \rightarrow & \mathbf{id} & (8)
\end{array}
$$

En el tablero.

(continua en la próxima diapositiva)

Gramáticas LL(1)

Ejemplo 4.32 (continuación)

No terminal	Símbolo de entrada					
	id	$+$	$*$	$($	$)$	\$
E	$E \to T\ E'$			$E \to T\ E'$		
E'		$E' \to +\ T\ E'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to F\ T'$			$T \to F\ T'$		
T'		$T' \to \epsilon$	$T' \to *\ F\ T'$		$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to \mathbf{id}$			$F \to (\ E\ )$		

Referencias


Aho, Alfred V., Monica S. Lam, Ravi Sethi y Jeffrey D. Ullman [1986] (2006). *Compilers: Principles, Techniques, & Tools*. 2.ª ed. Addison-Wesley (vid. pág. 2).


— [1986] (2008). *Compiladores: Principos, Técnicas y Herramientas*. Trad. por Alfonso Vidal Romero Elizondo. 2.ª ed. Pearson Educación (vid. pág. 2).