ST0270 Lenguajes Formales y Compiladores
Análisis sintáctico

Andrés Sicard Ramírez

Universidad EAFIT

Semestre 2024-1

(Última actualización: 25 de julio de 2025)

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 $G = (N, \Sigma, P, S)$ será denotada $\mathfrak{G} = (\mathcal{N}, \mathcal{T}, \mathcal{P}, S)$.

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

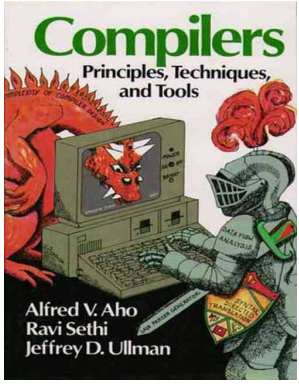
(v) Para escribir los elementos de las gramáticas seguiremos las convenciones en la sección § 4.2.2.

Preliminares

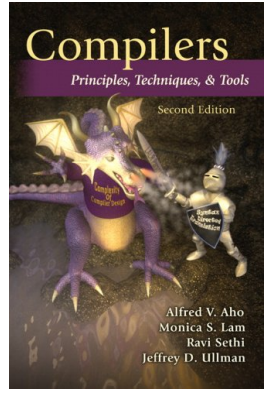
Agradecimientos

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

El libro del drágon



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



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

Introducción

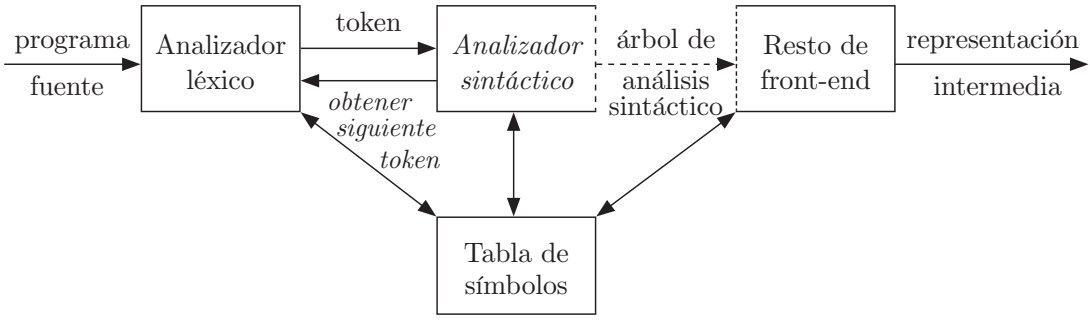
Descripción

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)

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

Introducción

Ejemplo (§ 2.4.1)

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

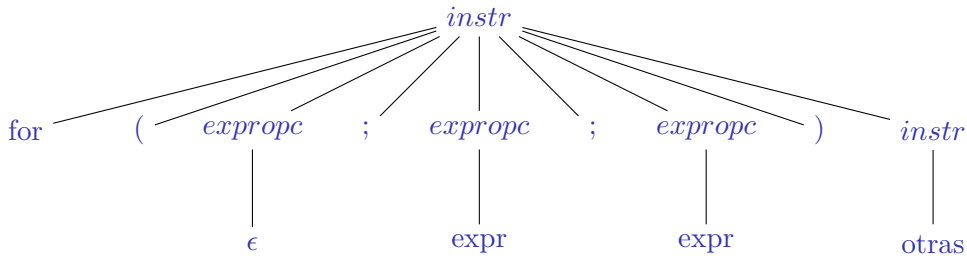
De acuerdo a las convenciones para las gramáticas de la sección § 4.2.2, los componentes de la gramática $(\mathcal{N}, \mathcal{T}, \mathcal{P}, S)$ son:

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

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*)
- Análisis sintáctico predictivo (*predictive parser*)

(ii) Análisis sintáctico predictivo no recursivo (*nonrecursive predictive 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

$$\begin{aligned} S &\rightarrow c\ A\ d \\ A &\rightarrow a\ b \mid a \end{aligned}$$

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 $\mathsf{Pr} : (\mathcal{N} \cup \mathcal{T})^* \to 2^{\mathcal{T} \cup \{\epsilon\}}$.

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

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

Conjuntos Primero y Siguiente

Descripción

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

(i) El conjunto $\text{Sig}(A)$ 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 $\text{Sig} : \mathcal{N} \to 2^{\mathcal{T} \cup \{\$\}}$.

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

(ii) Es decir, $\text{Sig}(A)$ 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 $\text{Sig} : \mathcal{N} \to 2^{\mathcal{T} \cup \{\$\}}$.

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

(ii) Es decir, $\text{Sig}(A)$ 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 $\$ \in \text{Sig}(A)$.

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 Sig(A)

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

Conjuntos Primero y Siguiente

Ejemplo 4.30

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

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

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:

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

En el tablero.

(continua en la próxima diapositiva)

Conjuntos Primero y Siguiente

Ejemplo 4.30 (continuación)

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

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)

Acrónimo

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:

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

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

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

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 M[A, $] ← M[A, $] ∪ {A → α};
11 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:

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

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:

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

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	$F \to \mathrm{id}$			$F \to (\ E\ )$		

Análisis sintáctico predictivo no recursivo

Algunos métodos (algoritmos) de análisis sintáctico descendente

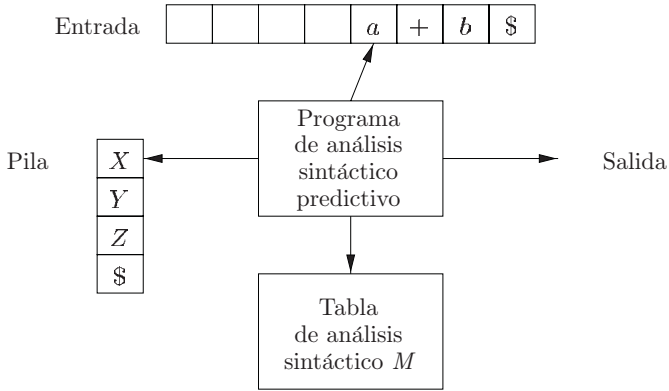
(i) Análisis sintáctico descendente recursivo (*recursive-descent parsing*)
- Análisis sintáctico predictivo (*predictive parser*)

(ii) Análisis sintáctico predictivo no recursivo (*nonrecursive predictive parsing*)

Análisis sintáctico predictivo no recursivo

Introducción

Para una gramática $\mathrm{LL}(1)$ podemos construir un analizador sintáctico predictivo (buscando una derivación más a la izquierda) que en vez de realizar llamadas recursivas utilice una pila (Fig. 4.19).



Análisis sintáctico predictivo no recursivo

Algoritmo 4.34: Análisis sintáctico predictivo no recursivo

Entrada: Una palabra w y una tabla de análisis sintáctico $M[A, a]$ para una gramática $\mathfrak{G}$.

Entrada: En el tope de la pila está el símbolo inicial de la gramática y debajo está el símbolo $\$$.

Salida: Si $w \in \mathrm{L}(\mathfrak{G})$, una derivación por la izquierda de w; en caso contrario, un error.

(continua en la próxima diapositiva)

Análisis sintáctico predictivo no recursivo

Algoritmo 4.34: Análisis sintáctico predictivo no recursivo (continuación)

```
1  let a el primer símbolo de w;
2  let X el símbolo del tope de la pila;
3  while X ≠ $ do
4  |   if X = a then desempilar y let a el siguiente símbolo
   |     de w;
5  |   else if X es un terminal then error();
6  |   else if M[X, a] es un error then error();
7  |   else if M[X, a] = X → Y1Y2 ... Yk then
8  |   |   imprimir la producción X → Y1Y2 ... Yk;
9  |   |   desempilar;
10 |   |   empilar YkYk−1 ... Y1, con Y1 en el tope de la pila;
11 |   end
12 |   let X el símbolo del tope de la pila;
13 end
```

Análisis sintáctico predictivo no recursivo

Ejemplo 4.35

Apliquemos el algoritmo 4.34 a la gramática del ejemplo 4.32.

En el tablero.

Análisis sintáctico predictivo no recursivo

Ejemplo 4.35

Apliquemos el algoritmo 4.34 a la gramática del ejemplo 4.32.

En el tablero.

(continua en la próxima diapositiva)

Análisis sintáctico predictivo no recursivo

Ejemplo 4.35 (continuación)

pila	entrada	acción
$E\ \$$	$\mathrm{id} + \mathrm{id} * \mathrm{id}\ \$$	
$T\ E'\ \$$	$\mathrm{id} + \mathrm{id} * \mathrm{id}\ \$$	salida $E \to T\ E'$
$F\ T'\ E'\ \$$	$\mathrm{id} + \mathrm{id} * \mathrm{id}\ \$$	salida $T \to F\ T'$
$\mathrm{id}\ T'\ E'\ \$$	$\mathrm{id} + \mathrm{id} * \mathrm{id}\ \$$	salida $F \to \mathrm{id}$
$T'\ E'\ \$$	$+\ \mathrm{id} * \mathrm{id}\ \$$	match id
$E'\ \$$	$+\ \mathrm{id} * \mathrm{id}\ \$$	salida $T' \to \epsilon$
$+\ T\ E'\ \$$	$+\ \mathrm{id} * \mathrm{id}\ \$$	salida $E' \to +\ T\ E'$
$T\ E'\ \$$	$\mathrm{id} * \mathrm{id}\ \$$	match $+$
$F\ T'\ E'\ \$$	$\mathrm{id} * \mathrm{id}\ \$$	salida $T \to F\ T'$
$\mathrm{id}\ T'\ E'\ \$$	$\mathrm{id} * \mathrm{id}\ \$$	salida $F \to \mathrm{id}$
$T'\ E'\ \$$	$*\ \mathrm{id}\ \$$	match id
$*\ F\ T'\ E'\ \$$	$*\ \mathrm{id}\ \$$	salida $T' \to *\ F\ T'$
$F\ T'\ E'\ \$$	$\mathrm{id}\ \$$	match $*$
$\mathrm{id}\ T'\ E'\ \$$	$\mathrm{id}\ \$$	salida $F \to \mathrm{id}$
$T'\ E'\ \$$	$\$$	match id
$E'\ \$$	$\$$	salida $T' \to \epsilon$
$\$$	$\$$	salida $E' \to \epsilon$

Análisis sintáctico ascendente

Introducción

En un analizador sintáctico ascendente (*bottom-up parse*) la construcción del árbol de análisis sintáctico (*parser tree*) empieza en las hojas y procede hacia la raíz.

Análisis sintáctico ascendente

Ejemplo

Gramática:

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

Palabra: $\mathrm{id} * \mathrm{id}$

En el tablero.

Análisis sintáctico ascendente

Acrónimo

$\mathrm{LR}(k)$: «*the* L *is for left-to-right scanning of the input, the* R *for constructing a rightmost derivation in reverse, and the* k *for the number of input symbols of lookahead that are used in making parsing decisions.*» (pág. 241)

Análisis sintáctico ascendente

Descripción

El ***shift-reduce parsing* (análisis sintáctico de desplazamiento-reducción)** es un tipo de análisis sintáctico ascendente (*bottom-up parsing*) que se emplea para analizar sintácticamente las gramáticas LR.

Análisis sintáctico ascendente

Descripción

El ***shift-reduce parsing* (análisis sintáctico de desplazamiento-reducción)** es un tipo de análisis sintáctico ascendente (*bottom-up parsing*) que se emplea para analizar sintácticamente las gramáticas LR.

«Podemos considerar el análisis sintáctico ascendente como el proceso de "reducir" una cadena w al símbolo inicial de la gramática. En cada paso de **reducción**, se sustituye una subcadena específica que coincide con el cuerpo de una producción por el no terminal que se encuentra en el encabezado de esa producción.» (pág. 234)

Análisis sintáctico ascendente

Ejemplo

En el tablero.

Autómata $\mathrm{LR}(0)$

Descripción

El **autómata $\mathbf{LR(0)}$** para una gramática le indica a un *shift-reduce parser* si debe realizar una acción de *shift* o de *reduce*.

Autómata $\mathrm{LR}(0)$

Definición

Un **elemento** de una gramática es una producción de la gramática con un punto $(\cdot)$ en alguna posición del cuerpo.

Autómata $\mathrm{LR}(0)$

Definición

Un **elemento** de una gramática es una producción de la gramática con un punto $(\cdot)$ en alguna posición del cuerpo.

Ejemplo

Para la producción $A \to X\ Y\ Z$ hay cuatro elementos:

$$A \to \cdot\ X\ Y\ Z$$
$$A \to X \cdot\ Y\ Z$$
$$A \to X\ Y \cdot\ Z$$
$$A \to X\ Y\ Z\ \cdot$$

Autómata $\mathrm{LR}(0)$

Definición

Los **estados** de un autómata $\mathrm{LR}(0)$ son conjuntos de elementos y éstos forman **la colección canónica** de elementos $\mathrm{LR}(0)$.

Autómata $\mathrm{LR}(0)$

Descripción

Para construir un autómata $\mathrm{LR}(0)$ para una gramática necesitamos:

(i) Una gramática extendida.

(ii) Una función cerradura(I), donde I es un conjunto de elementos.

(iii) Una función goto(I, X), donde I es un conjunto de elementos y X es un símbolo de la gramática, es decir, $X \in \mathcal{N} \cup \mathcal{T}$.

Autómata $\mathrm{LR}(0)$

Definición

Sea $\mathfrak{G}$ una gramática con símbolo inicial S. La **gramática aumentada** para $\mathfrak{G}$, denotada $\mathfrak{G}'$ es la gramática $\mathfrak{G}$, con un nuevo símbolo inicial S' y una nueva producción $S' \to S$.

Autómata $\mathrm{LR}(0)$

Definición

Sea $\mathcal{I}$ el conjunto de todos los elementos de una gramática, la función

$$\text{cerradura} : 2^{\mathcal{I}} \to 2^{\mathcal{I}}$$

para $I \in \mathcal{I}$ está definida por las siguientes reglas:

(i) Adicionar I a cerradura(I).

(ii) Si $A \to \alpha \cdot B\ \beta \in$ cerradura(I) y $B \to \gamma$ es una producción, entonces adicionar $B \to \gamma$ a cerradura(I). Repetir esta regla hasta que no se puedan adicionar nuevos elementos a cerradura(I).

Autómata $\mathrm{LR}(0)$

Ejemplo

En el tablero.

Autómata $\mathrm{LR}(0)$

Definición

Sea $\mathcal{I}$ el conjunto de todos los elementos de una gramática, la función

$$\text{goto} : 2^{\mathcal{I}} \times (\mathcal{N} \cup \mathcal{T}) \to 2^{\mathcal{I}}$$

para $I \in \mathcal{I}$ y $X \in \mathcal{N} \cup \mathcal{T}$ está definida por: $\text{goto}(I, X)$ es la cerradura de los elementos $A \to \alpha\, X \cdot \beta$, tales que $A \to \alpha \cdot X\, \beta \in I$.

Autómata $\mathrm{LR}(0)$

Ejemplo

En el tablero.

Autómata $\mathrm{LR}(0)$

Algoritmo: Computación de la colección canónica de elementos $\mathrm{LR}(0)$

Entrada: Gramática aumentada $\mathfrak{G}$
Salida: Colección canónica de elementos $\mathrm{LR}(0)$

```
1  C ← cerradura(S' → S);
2  repeat
3      for cada conjunto de elementos I en C do
4          for cada símbolo gramatical X do
5              if goto(I, X) no es vacío y no está en C then
6                  adicionar goto(I, X) a C
7              end
8          end
9      end
10 until no se agreguen nuevos conjuntos de elementos a C en una
       iteración;
```

Referencias


Alfred V. Aho, 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).