Universidad Nacional Autónoma de México

Facultad de Ciencias

Compiladores. 2025-1 | 7008

Práctica 3: Implementación de un Analizador Sintáctico de descenso recursivo

Kevin Steve Quezada Ordoñez

stevequezada@ciencias.unam.mx

01/10/2024

- **Dada la gramática** $G = (N, \Sigma, P, S)$

  Donde P es:

  programa → declaraciones sentencias
  declaraciones → declaraciones declaracion | declaracion
  declaracion → tipo lista_var ;
  tipo → `int` | `float`
  lista_var → lista_var , `identificador` | `identificador`
  sentencias → sentencias sentencia | sentencia
  sentencia → `identificador` = expresion ; | `if` (expresion) sentencias `else` sentencias
  | `while` (expresion) sentencias
  expresion → expresion + expresion | expresion − expresion | expresion ∗ expresion
  | expresion / expresion | `identificador` | `numero`
  expresion → (expresion)

1. **Se determinan los conjuntos** N, Σ **y el símbolo inicial** S**.** `(0.5 pts.)`

   - N *(No terminales)*:

     $$N = \{\text{programa}, \text{declaraciones}, \text{declaracion}, \text{tipo}, \text{lista\_var}, \text{sentencias}, \text{sentencia}, \text{expresion}\}$$

   - Σ *(Terminales)*:

     $$\Sigma = \{\texttt{int}, \texttt{float}, \texttt{,}, \texttt{identificador}, =, \texttt{;}, \texttt{if}, \texttt{else}, \texttt{while}, \texttt{(}, \texttt{)}, +, \texttt{-}, \texttt{*}, \texttt{/}, \texttt{numero}\}$$

   - S *(Símbolo inicial)*:

     $$S = \{\text{programa}\}$$

2. **Se realiza el proceso de eliminación de ambigüedad.** `(1 pts.)`

   i. **Se identifica la ambigüedad**

$$\begin{aligned} \text{expresion} \rightarrow\ & \text{expresion} + \text{expresion} \\ \mid\ & \text{expresion} - \text{expresion} \\ \mid\ & \text{expresion} * \text{expresion} \\ \mid\ & \text{expresion} \,/\, \text{expresion} \\ \mid\ & \texttt{identificador} \\ \mid\ & \texttt{numero} \\ \mid\ & (\text{expresion}) \end{aligned}$$

   Esta gramática es ambigua porque no especifica la precedencia ni la asociatividad de los operadores, permitiendo derivaciones múltiples para una misma expresión. La multiplicación y división tienen mayor precedencia que la suma y resta. Se necesita separar la gramática para reflejar esta diferencia.

   ii. **Se define la recursividad según la asociatividad**

   Los operadores $+$, $-$, $*$ y $/$ son asociativos a la izquierda. Por lo tanto, las producciones deben ser recursivas a la izquierda para reflejar esta propiedad.

   iii. **Se reescribe la gramática respetando precedencia y asociatividad**

   - **Nivel más bajo:**

$$\text{factor} \rightarrow (\text{expresion}) \mid \texttt{identificador} \mid \texttt{numero}$$

   - **Nivel intermedio:**

$$\text{termino} \rightarrow \text{termino} * \text{factor} \mid \text{termino} \,/\, \text{factor} \mid \text{factor}$$

   - **Nivel superior:**

$$\text{expresion} \rightarrow \text{expresion} + \text{termino} \mid \text{expresion} - \text{termino} \mid \text{termino}$$

   Gramática Actualizada:

$$\begin{aligned} \text{programa} &\rightarrow \text{declaraciones sentencias} \\ \text{declaraciones} &\rightarrow \text{declaraciones declaracion} \mid \text{declaracion} \\ \text{declaracion} &\rightarrow \text{tipo lista\_var} \ ; \\ \text{tipo} &\rightarrow \texttt{int} \mid \texttt{float} \\ \text{lista\_var} &\rightarrow \text{lista\_var} \ , \ \texttt{identificador} \mid \texttt{identificador} \\ \text{sentencias} &\rightarrow \text{sentencias sentencia} \mid \text{sentencia} \\ \text{sentencia} &\rightarrow \texttt{identificador} = \text{expresion} \ ; \\ &\quad \mid \texttt{if} \ (\ \text{expresion}\ ) \ \text{sentencias} \ \texttt{else} \ \text{sentencias} \\ &\quad \mid \texttt{while} \ (\ \text{expresion}\ ) \ \text{sentencias} \\ \text{expresion} &\rightarrow \text{expresion} + \text{termino} \mid \text{expresion} - \text{termino} \mid \text{termino} \\ \text{termino} &\rightarrow \text{termino} * \text{factor} \mid \text{termino} \,/\, \text{factor} \mid \text{factor} \\ \text{factor} &\rightarrow (\text{expresion}) \mid \texttt{identificador} \mid \texttt{numero} \end{aligned}$$

3. **Se muestra el proceso de eliminación de la recursividad izquierda.** `(1 pts.)`

   i. **Se identifica la recursividad izquierda**

   - declaraciones → declaraciones declaracion | declaracion
   - lista_var → lista_var , `identificador` | `identificador`
   - sentencias → sentencias sentencia | sentencia
   - expresion → expresion + termino | expresion − termino | termino
   - termino → termino ∗ factor | termino / factor | factor

   ii. **Se elimina la recursividad izquierda.**

   - **declaraciones:**

     declaraciones → declaracion declaraciones'
     declaraciones' → declaracion declaraciones' | ϵ

   - **lista_var:**

     lista_var → `identificador` lista_var'
     lista_var' → , `identificador lista_var'` | ϵ

   - **sentencias:**

     sentencias → sentencia sentencias'
     sentencias' → sentencia sentencias' | ϵ

   - **expresion:**

     expresion → termino expresion'
     expresion' → + termino expresion'
     | − termino expresion' | ϵ

   - **termino:**

     termino → factor termino'
     termino' → ∗ factor termino'
     | / factor termino' | ϵ

4. Se muestra por qué no es necesario aplicar factorización izquierda. (1 pts.)

La gramática no requiere factorización izquierda debido a que no se presentan producciones con prefijos comunes. Esto significa que las producciones de la gramática comienzan de manera diferente, lo que permite que un parser determine cuál regla aplicar sin ambigüedades.

5. Se muestran los nuevos conjuntos N y P. (0.5 pts.)

- *N (No terminales)*:

$$N = \{\text{programa}, \text{declaraciones}, \text{declaraciones'}, \text{declaracion}, \text{tipo}, \text{lista\_var}, \text{lista\_var'}, \text{sentencias}, \text{sentencias'}, \text{sentencia}, \text{expresion}, \text{expresion'}, \text{termino}, \text{termino'}, \text{factor}\}$$

- *P (Producciones)*:

$$\begin{aligned}
\text{programa} &\rightarrow \text{declaraciones sentencias} \\
\text{declaraciones} &\rightarrow \text{declaracion declaraciones'} \\
\text{declaraciones'} &\rightarrow \text{declaracion declaraciones'} \mid \epsilon \\
\text{declaracion} &\rightarrow \text{tipo lista_var ;} \\
\text{tipo} &\rightarrow \texttt{int} \mid \texttt{float} \\
\text{lista_var} &\rightarrow \texttt{identificador} \text{ lista_var'} \\
\text{lista_var'} &\rightarrow \text{ , } \texttt{identificador lista_var'} \mid \epsilon \\
\text{sentencias} &\rightarrow \text{sentencia sentencias'} \\
\text{sentencias'} &\rightarrow \text{sentencia sentencias'} \mid \epsilon \\
\text{sentencia} &\rightarrow \texttt{identificador} = \text{expresion ;} \\
&\mid \texttt{if} \text{ (expresion) sentencias } \texttt{else} \text{ sentencias} \\
&\mid \texttt{while} \text{ (expresion) sentencias} \\
\text{expresion} &\rightarrow \text{termino expresion'} \\
\text{expresion'} &\rightarrow + \text{ termino expresion'} \\
&\mid - \text{ termino expresion'} \mid \epsilon \\
\text{termino} &\rightarrow \text{factor termino'} \\
\text{termino'} &\rightarrow * \text{ factor termino'} \\
&\mid / \text{ factor termino'} \mid \epsilon \\
\text{factor} &\rightarrow \text{(expresion)} \\
&\mid \texttt{identificador} \\
&\mid \texttt{numero}
\end{aligned}$$

○ Gramática Final.

Para implementar las producciones en `Java`, fue necesario modificar la gramatica:

$$
\begin{aligned}
\text{programa} &\rightarrow \text{declaraciones sentencias} \\
\\
\text{declaraciones} &\rightarrow \text{declaracion declaraciones'} \\
\text{declaraciones'} &\rightarrow \text{declaracion declaraciones'} \mid \epsilon \\
\\
\text{declaracion} &\rightarrow \text{tipo lista\_var ;} \\
\text{tipo} &\rightarrow \texttt{int} \mid \texttt{float} \\
\text{lista\_var} &\rightarrow \texttt{identificador}\ \text{lista\_var'} \\
\text{lista\_var'} &\rightarrow \text{ , } \texttt{identificador}\ \text{lista\_var'} \mid \epsilon \\
\\
\text{sentencias} &\rightarrow \{\text{sentencias'}\} \mid \text{sentencia sentencias'} \\
\text{sentencias'} &\rightarrow \text{sentencia sentencias'} \mid \epsilon \\
\\
\text{sentencia} &\rightarrow \texttt{identificador} = \text{expresion ;} \\
&\mid \texttt{if}\ (\text{expresion})\ \text{sentencias}\ \texttt{else}\ \text{sentencias} \\
&\mid \texttt{while}\ (\text{expresion})\ \text{sentencias} \\
\\
\text{expresion} &\rightarrow \text{termino expresion'} \\
\text{expresion'} &\rightarrow + \text{ termino expresion'} \\
&\mid - \text{ termino expresion'} \\
&\mid < \text{ termino expresion'} \\
&\mid > \text{ termino expresion'} \\
&\mid <= \text{ termino expresion'} \\
&\mid >= \text{ termino expresion'} \\
&\mid == \text{ termino expresion'} \\
&\mid != \text{ termino expresion'} \\
&\mid \epsilon \\
\\
\text{termino} &\rightarrow \text{factor termino'} \\
\text{termino'} &\rightarrow * \text{ factor termino'} \\
&\mid / \text{ factor termino'} \\
&\mid \epsilon \\
\\
\text{factor} &\rightarrow (\text{expresion}) \\
&\mid \texttt{identificador} \\
&\mid \texttt{numero}
\end{aligned}
$$