4.4. Gramáticas para lenguajes de programación

La sintaxis de los lenguajes de programación, o al menos una gran porción de ésta, se presenta usualmente por medio de gramáticas GIC. En tales casos se dice que el lenguaje está en la **forma Backus-Naur** o, simplemente, en la **forma BNF**. Los lenguajes que están en **BNF** ofrecen ventajas significativas para el diseño de analizadores sintácticos en compiladores.

Ejemplo A continuación se exhibe una gramática para generar los números reales sin signo, similar a la utilizada en muchos lenguajes de programación. Las variables aparecen encerradas entre paréntesis $\langle\ \rangle$ y $\langle real \rangle$ es la variable inicial de la gramática. El alfabeto de terminales es $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \texttt{.}, \texttt{+}, \texttt{-}, \texttt{E}\}$. En el contexto de los lenguajes de programación los terminales se denominan también **componentes léxicos**, **lexemas**, o **tokens**.

$$
\begin{array}{lcl}
\langle real \rangle & \rightarrow & \langle dígitos \rangle \langle decimal \rangle \langle exp \rangle \\
\langle dígitos \rangle & \rightarrow & \langle dígitos \rangle \langle dígitos \rangle \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
\langle decimal \rangle & \rightarrow & \texttt{.}\langle dígitos \rangle \mid \lambda \\
\langle exp \rangle & \rightarrow & \texttt{E}\langle dígitos \rangle \mid \texttt{E+}\langle dígitos \rangle \mid \texttt{E-}\langle dígitos \rangle \mid \lambda
\end{array}
$$

Esta gramática genera expresiones como 47.236, 321.25E+35, 0.8E9 y 0.8E+9. Las partes decimal y exponencial son "opcionales" debido a las producciones $\langle decimal \rangle \rightarrow \lambda$ y $\langle exp \rangle \rightarrow \lambda$, pero no se generan expresiones como .325, E125, 42.5E ni 0.1E+.

Ejemplo Gramática para generar los **identificadores** en lenguajes de programación, es decir, cadenas cuyo primer símbolo es una letra que va seguida de letras y/o dígitos. Las variables aparecen encerradas entre paréntesis $\langle\ \rangle$ e $\langle identificador \rangle$ es la variable inicial de la gramática. La variable $\langle lsds \rangle$ representa "letras o dígitos". Los terminales en esta gramática son las letras, minúsculas o mayúsculas, y los dígitos.

$$
\begin{array}{lcl}
\langle identificador \rangle & \rightarrow & \langle letra \rangle \langle lsds \rangle \\
\langle lsds \rangle & \rightarrow & \langle letra \rangle \langle lsds \rangle \mid \langle dígito \rangle \langle lsds \rangle \mid \lambda \\
\langle letra \rangle & \rightarrow & \text{a} \mid \text{b} \mid \text{c} \mid \cdots \mid \text{x} \mid \text{y} \mid \text{z} \mid \text{A} \mid \text{B} \mid \text{C} \mid \cdots \mid \text{Y} \mid \text{Z} \\
\langle dígito \rangle & \rightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

Ejercicios de la sección 4.4

1. Con la gramática del primer ejemplo de esta sección hacer derivaciones y árboles de derivación para las cadenas 235.101E+25 y 0.01E-12.

2. Encontrar una GIC, con una sola variable, para generar los identificadores, es decir, el lenguaje del segundo ejemplo de esta sección.

3. Una gramática similar a la siguiente se usa en muchos lenguajes de programación para generar expresiones aritméticas formadas por sumas y productos de números en binario:

$$
\begin{array}{lcl}
\langle \textit{expresión} \rangle & \to & \langle \textit{término} \rangle \mid \langle \textit{expresión} \rangle \texttt{+} \langle \textit{término} \rangle \\
\langle \textit{término} \rangle & \to & \langle \textit{factor} \rangle \mid \langle \textit{término} \rangle \texttt{*} \langle \textit{factor} \rangle \\
\langle \textit{factor} \rangle & \to & \langle \textit{número} \rangle \mid \texttt{(} \langle \textit{expresión} \rangle \texttt{)} \\
\langle \textit{número} \rangle & \to & \texttt{1} \langle \textit{número} \rangle \mid \texttt{0} \langle \textit{número} \rangle \mid \texttt{0} \mid \texttt{1}
\end{array}
$$

   El alfabeto de terminales de esta gramática es $\{\texttt{0}, \texttt{1}, \texttt{+}, \texttt{*}, \texttt{(}, \texttt{)}\}$.

   (i) Hacer derivaciones y árboles de derivación para las siguientes cadenas:

```
10+101*10
(101+10*10)
(10+111)*(100+10*101+1111)
```

   (ii) Demostrar que esta gramática no es ambigua.