Teoría de Lenguajes

Práctica 8
(Parsers Descendentes)

1. Para cada G_i, decidir si es LL(1). En caso contrario, dar una gramática que sí sea LL(1) y genere $L(G_i)$. En ambos casos, indicar los símbolos directrices de cada producción de la gramática resultante.

   (a) $G_1 = \langle \{S, A\}, \{\mathtt{a}, \mathtt{b}\}, P_1, S \rangle$, con P_1:

   $$\begin{array}{l} S \longrightarrow \mathtt{a}AS \mid \mathtt{b} \\ A \longrightarrow \mathtt{a} \mid \mathtt{b}SA \end{array}$$

   (b) $G_2 = \langle \{S\}, \{\mathtt{a}, \mathtt{b}\}, P_2, S \rangle$, con P_2:

   $$S \longrightarrow \mathtt{aa}S\mathtt{bb} \mid \mathtt{a} \mid \lambda$$

   (c) $G_3 = \langle \{S, A, B\}, \{\mathtt{a}, \mathtt{b}\}, P_3, S \rangle$, con P_3:

   $$\begin{array}{l} S \longrightarrow A \mid B \\ A \longrightarrow \mathtt{a}A \mid \lambda \\ B \longrightarrow \mathtt{b}B \mid \lambda \end{array}$$

   (d) $G_4 = \langle \{S, A\}, \{\mathtt{a}, \mathtt{b}\}, P_4, S \rangle$, con P_4:

   $$\begin{array}{l} S \longrightarrow \mathtt{a}A\mathtt{aa} \mid \mathtt{b}A\mathtt{ba} \\ A \longrightarrow \mathtt{b} \mid \lambda \end{array}$$

   (e) $G_5 = \langle \{S\}, \{\mathtt{a}\}, P_5, S \rangle$, con P_5:

   $$S \longrightarrow \mathtt{a}S \mid \mathtt{a}$$

   (f) $G_6 = \langle \{S, A\}, \{\mathtt{a}, \mathtt{d}\}, P_6, S \rangle$, con P_6:

   $$\begin{array}{l} S \longrightarrow A\mathtt{a} \mid \mathtt{a} \\ A \longrightarrow S\mathtt{d} \mid \mathtt{d} \end{array}$$

   (g) $G_7 = \langle \{S, A\}, \{\mathtt{a}, \mathtt{c}\}, P_7, S \rangle$, con P_7:

   $$\begin{array}{lcl} S & \longrightarrow & S\mathtt{c} \mid \mathtt{c}A \mid \lambda \\ A & \longrightarrow & \mathtt{a}A \mid \mathtt{a} \end{array}$$

(h) $G_8 = \langle \{S, A, L\}, \{\texttt{(}, \texttt{)}, \texttt{,}, \texttt{f}, \texttt{x}\}, P_8, S \rangle$, con P_8:

$$\begin{array}{lcl} S & \longrightarrow & \texttt{f}A \\ A & \longrightarrow & \texttt{(}L\texttt{)} \\ L & \longrightarrow & \texttt{x} \mid \texttt{x,}L \mid S \end{array}$$

(i) $G_9 = \langle \{S, A\}, \{\texttt{a}, \texttt{b}\}, P_9, S \rangle$, con P_9:

$$\begin{array}{lcl} S & \longrightarrow & SA\texttt{a} \mid A\texttt{a} \\ A & \longrightarrow & A\texttt{a} \mid \texttt{b} \end{array}$$

(j) $G_{10} = \langle \{S, T\}, \{\texttt{a}, \texttt{b}\}, P_{10}, S \rangle$, con P_{10}:

$$\begin{array}{lcl} S & \longrightarrow & \texttt{b} \mid S\texttt{b} \mid T\texttt{b} \\ T & \longrightarrow & \texttt{a}T\texttt{b} \mid \texttt{ab} \end{array}$$

(k) $G_{11} = \langle \{A\}, \{\texttt{\{}, \texttt{\}}\}, P_{11}, A \rangle$, con P_{11}:

$$A \longrightarrow A\texttt{\{}A\texttt{\}}A \mid \lambda$$

(l) $G_{12} = \langle \{A\}, \{\texttt{+}, \texttt{-}, \texttt{a}\}, P_{12}, A \rangle$, con P_{12}:

$$A \longrightarrow \texttt{+}AA \mid \texttt{-}AA \mid \texttt{a}$$

(m) $G_{13} = \langle \{A\}, \{\texttt{0}, \texttt{1}\}, P_{13}, A \rangle$, con P_{13}:

$$A \longrightarrow \texttt{0}A\texttt{1} \mid \texttt{01}$$

2. A veces es posible utilizar un parser predictivo para reconocer el lenguaje de una gramática que tiene conflictos LL(1). Esto puede realizarse resolviendo los conflictos en favor de alguna de las producciones involucradas. Por ejemplo, la siguiente gramática representa el problema del $if - then - else$:

   $G_1 = \langle \{S, E, C\}; \{\texttt{if}, \texttt{then}, \texttt{sent}, \texttt{else}, \texttt{cond}\}; P_1; S \rangle$, con P_1:

$$\begin{array}{lcl} S & \longrightarrow & \texttt{if}\ C\ \texttt{then}\ S\ E \mid \texttt{sent} \\ E & \longrightarrow & \texttt{else}\ S \mid \lambda \\ C & \longrightarrow & \texttt{cond} \end{array}$$

   (a) Mostrar que G_1 no es $LL(1)$.

   (b) Indicar los símbolos directrices de cada producción.

   (c) Para los terminales que sean símbolo directriz de más de una producción del mismo no terminal, asignárselos a una sola de las producciones de manera que en el árbol de derivación resultante cada `else` quede asociado con el `if` más cercano.

   (d) Realizar el seguimiento de análisis sintáctico con el método LL(1) y dar el árbol de derivación resultante para:
   `if cond then if cond then sent else sent`

3. Escribir gramáticas extendidas ELL(1) para:

   (a) lista de sentencias terminadas en `;`

   (b) lista de valores separados por `,`

   (c) expresiones aritméticas con `+`, `-`, `*`, `(`, `)`, `id`

   (d) `if ... then ... elseif ... then ... else ... endif`, con las partes de `elseif` y `else` opcionales.

4. Dar gramáticas ELL(1) para cada uno de los lenguajes de la práctica 7. Escribir los parsers recursivos-iterativos correspondientes.

5. Dada la gramática $G = \langle \{L, R, A, D\}, \{\texttt{f}, \texttt{d}, \texttt{h}, \texttt{+}, \texttt{t}\}, P, L \rangle$, con P:

$$\begin{array}{rcl} L & \longrightarrow & R \mid R\texttt{+}L \\ R & \longrightarrow & A \mid RA \\ A & \longrightarrow & \texttt{d}L\texttt{h}D \mid \texttt{t} \mid \texttt{d}L\texttt{h} \\ D & \longrightarrow & \lambda \mid \texttt{f} \end{array}$$

   (a) Es G ELL(1)? Si no lo es, dar una gramática extendida G' que sea ELL(1) y tal que $L(G') = L(G)$ usando al menos una vez cada uno de los siguientes operadores: `*`, `+` y `?`.

   (b) Dar el parser recursivo-iterativo para la gramática ELL(1).

6. La siguiente gramática extendida representa una versión (simplificada!) de declaraciones de variables y funciones en el lenguaje C.

   $G_2 = \langle \{Declaracion, Declarador, LParDecl, ParDecl, LId, Tipo\},$
   $\{\texttt{(}, \texttt{)}, \texttt{[}, \texttt{]}, \texttt{,}, \texttt{*}, \texttt{int}, \texttt{char}, \texttt{ID}\}, P, Declaracion \rangle$, siendo P:

$$\begin{array}{rcl} Declaracion & \longrightarrow & Tipo\ Declarador\ (\ \texttt{,}\ Declarador\ )^* \\ Declarador & \longrightarrow & (\ \texttt{*}\ )^*\ (\ \texttt{ID} \mid \texttt{(}\ Declarador\ \texttt{)}\ ) \\ & & (\ \texttt{(}\ (\ LParDecl \mid LId\ )?\ \texttt{)} \mid \texttt{[}\ \texttt{NUM}\ ?\ \texttt{]}\ )^* \\ LParDecl & \longrightarrow & ParDecl\ (\ \texttt{,}\ ParDecl\ )^* \\ ParDecl & \longrightarrow & Tipo\ (\ Declarador\ )? \\ LId & \longrightarrow & \texttt{ID}\ (\ \texttt{,}\ \texttt{ID}\ )^* \\ Tipo & \longrightarrow & \texttt{int} \mid \texttt{char} \end{array}$$

   Escribir un parser recursivo-iterativo para este lenguaje, verificando que la gramática sea ELL(1).

7. Dada la gramática extendida:

   $G = \langle \{S, C, D\}, \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{,}\}, S, P \rangle$, con P:

$$\begin{array}{rcl} S & \longrightarrow & (\texttt{ab}C \mid \texttt{ac}D)^*\texttt{a} \\ C & \longrightarrow & \texttt{c}(\texttt{,c})^* \\ D & \longrightarrow & \texttt{d}(\texttt{d?})^* \end{array}$$

   - ¿Es G ELL(1)? Si no, dar una gramática G' que sí lo sea, tal que $L(G') = L(G)$.
   - Dar el parser recursivo-iterativo para la gramática ELL(1).

8. Dada la siguiente gramática que genera las expresiones regulares sobre los terminales $\{\texttt{a}, \texttt{b}, \texttt{c}\}$, dar una gramática extendida que sea ELL(1) y genere el mismo lenguaje.

   $G = \langle \{E\}, \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{(}, \texttt{)}, \texttt{|}, \texttt{*}\}, E, P \rangle$, con P:

$$E \longrightarrow E\texttt{|}E \;\mid\; EE \;\mid\; E^{\texttt{*}} \;\mid\; \texttt{(}E\texttt{)} \;\mid\; \texttt{a} \;\mid\; \texttt{b} \;\mid\; \texttt{c}$$