Teoría de Lenguajes

Práctica 9: Parsing descendente

Versión del 18 de marzo de 2024

Ejercicio 1. Considerar la siguiente gramática, que permite generar expresiones aritméticas de forma no ambigua:

$$G = \langle \{E, T, F\}, \{\texttt{+}, \texttt{-}, \texttt{*}, \texttt{(}, \texttt{)}, \textbf{id}\}, P, E \rangle,$$

con P:

$$\begin{array}{rcl} E & \to & E \texttt{ + } T \mid E \texttt{ - } T \mid T \\ T & \to & T \texttt{ * } F \mid F \\ F & \to & \texttt{(} E \texttt{)} \mid \textbf{id} \end{array}$$

a. Dar una gramática G' equivalente a G que no posea recursión a la izquierda.

b. Dar los conjuntos de Primeros y Siguientes para cada símbolo no terminal de G'.

c. Determinar los símbolos directrices para cada producción de G' y dar su tabla de parsing LL(1).

d. ¿Es G' una gramática LL(1)? ¿Por qué?

e. Dar el árbol de derivación para la cadena **id * (id + id)** usando el método LL(1) con la gramática G'.

Ejercicio 2. Para cada G_i, decidir si es LL(1). En caso contrario, dar una gramática que sí sea LL(1) y genere G_i. En ambos casos, dar la tabla de parsing LL(1) de la gramática resultante.

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

$$\begin{array}{rcl} S & \to & aAS \mid b \\ A & \to & a \mid bSA \end{array}$$

b. $G_2 = \langle \{A\}, \{\texttt{+}, \texttt{-}, a\}, P_2, A \rangle$, con P_2:

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

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

$$\begin{array}{rcl} S & \to & A \mid B \\ A & \to & aA \mid \lambda \\ B & \to & bB \mid \lambda \end{array}$$

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

$$\begin{array}{rcl} S & \to & aAaa \mid bAba \\ A & \to & b \mid \lambda \end{array}$$

e. $G_5 = \langle \{A\}, \{\{, \}\}, P_5, A \rangle$, con P_5:

$$A \to A\{A\}A \mid \lambda$$

f. $G_6 = \langle \{S\}, \{a\}, P_6, S \rangle$, con P_6:

$$S \to aS \mid a$$

g. $G_7 = \langle \{A\}, \{0,1\}, P_7, A \rangle$, con P_7:

$$A \rightarrow 0A1 \mid 01$$

h. $G_8 = \langle \{S\}, \{a,b\}, P_8, S \rangle$, con P_8:

$$S \rightarrow aaSbb \mid a \mid \lambda$$

i. $G_9 = \langle \{S, A, L\}, \{\texttt{(}, \texttt{)}, \texttt{,}, f, x\}, P_9, S \rangle$, con P_9:

$$\begin{aligned} S &\rightarrow f\ A \\ A &\rightarrow \texttt{(}L\texttt{)} \\ L &\rightarrow x \mid x\texttt{,}L \mid S \end{aligned}$$

j. $G_{10} = \langle \{S, A\}, \{a,b\}, P_{10}, S \rangle$, con P_{10}:

$$\begin{aligned} S &\rightarrow SAa \mid Aa \\ A &\rightarrow Aa \mid b \end{aligned}$$

k. $G_{11} = \langle \{S, T\}, \{a,b\}, P_{11}, S \rangle$, con P_{11}:

$$\begin{aligned} S &\rightarrow b \mid Sb \mid Tb \\ T &\rightarrow aTb \mid ab \end{aligned}$$

l. $G_{12} = \langle \{S, A\}, \{a,d\}, P_{12}, S \rangle$, con P_{12}:

$$\begin{aligned} S &\rightarrow Aa \mid a \\ A &\rightarrow Sd \mid d \end{aligned}$$

m. $G_{13} = \langle \{S, A\}, \{a,c\}, P_{13}, S \rangle$, con P_{13}:

$$\begin{aligned} S &\rightarrow Sc \mid cA \mid \lambda \\ A &\rightarrow aA \mid a \end{aligned}$$

Ejercicio 3. 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*:

$$G = \langle \{S, E, C\}, \{\texttt{if}, \texttt{then}, \texttt{else}, \textbf{sent}, \textbf{cond}\}, P, S \rangle,$$

$$\text{con } P: \quad \begin{aligned} S &\rightarrow \texttt{if}\ C\ \texttt{then}\ S\ E \mid \textbf{sent} \\ E &\rightarrow \texttt{else}\ S \mid \lambda \\ C &\rightarrow \textbf{cond} \end{aligned}$$

a. Dar la tabla de parsing LL(1) para G y mostrar que G no es LL(1).

b. 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.

c. 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**

d. ¿Existe algún problema si se decide resolver el conflicto de la manera opuesta? Pista: ver qué sucede al intentar reconocer la cadena `if` **cond** `then` **sent** `else` **sent**.