

Práctica 4

Análisis sintáctico descendente

Ejercicio 1. Para la siguiente gramática $G = (\{S\}, \{0, 1\}, P, S)$:

$$S \rightarrow 0S1 \mid 01$$

1. Dibujar el árbol de los estados (entrada y pila) que atraviesa un analizador sintáctico predictivo con *backtracking* para analizar la cadena $000111 \in L(G)$.
2. Dibujar el árbol de los estados que atraviesa un analizador sintáctico predictivo con *backtracking* para analizar la cadena $00111 \notin L(G)$.

Ejercicio 2. Para la gramática $G = (\{S, A, B, C, D\}, \{b, c, d, \bullet\}, P, S)$:

$$\begin{aligned} S &\rightarrow A \bullet S \mid \epsilon \\ A &\rightarrow BC \mid D \\ B &\rightarrow bB \mid \epsilon \\ C &\rightarrow cC \mid \epsilon \\ D &\rightarrow dD \mid \epsilon \end{aligned}$$

1. Calcular los conjuntos FIRST y FOLLOW para cada símbolo no terminal.
2. Calcular la tabla de análisis sintáctico LL(1) para G .
3. Verificar que la gramática es LL(1), observando que todas las entradas de la tabla tienen a lo sumo una producción.
4. Analizar sintácticamente la cadena $b \bullet c \in L(G)$. Exhibir la tabla que muestra la evolución del estado de la pila y la cadena de entrada a lo largo del proceso de análisis sintáctico. Reconstruir la derivación más a la izquierda obtenida.
5. Analizar sintácticamente la cadena $\bullet bd \notin L(G)$ y verificar que la cadena se rechaza (obteniendo un error de sintaxis).

Ejercicio 3. Para la gramática:

$$G = (\{P, S, E, E', T\}, \{;, \text{id}, :=, \text{while}, \text{do}, \{, \}, \text{op}, \text{const}\}, P, P)$$

$$\begin{aligned} P &\rightarrow S; P \mid \epsilon \\ S &\rightarrow \text{id} := E \mid \text{while } E \text{ do } S \mid \{P\} \\ E &\rightarrow TE' \\ E' &\rightarrow \epsilon \mid \text{op } E' \\ T &\rightarrow \text{id} \mid \text{const} \end{aligned}$$

1. Calcular la tabla de análisis sintáctico LL(1) para G y verificar que la gramática es LL(1).
2. Analizar sintácticamente la cadena **while id do id := const;** $\in L(G)$ y reconstruir la derivación más a la izquierda obtenida.
3. Analizar sintácticamente la cadena **{const := id;}** $\notin L(G)$.

Ejercicio 4. Decidir si las siguientes gramáticas son LL(1). Para las que no lo sean, dar una gramática que genere el mismo lenguaje y sea LL(1) usando técnicas de normalización de gramáticas.

1. $G_1 = (\{E\}, \{+, *, \mathbf{n}\}, P, E)$

$$E \rightarrow EE+ \mid EE* \mid \mathbf{n}$$

2. $G_2 = (\{E\}, \{+, *, \mathbf{n}\}, P, E)$

$$E \rightarrow +EE \mid *EE \mid \mathbf{n}$$

3. $G_3 = (\{S\}, \{a, b\}, P, S)$

$$S \rightarrow aaa \mid aba \mid bab \mid bbb$$

4. $G_4 = (\{E\}, \{\&\&, ||, !, p, q\}, P, S)$

$$E \rightarrow E\&\&E \mid E||E \mid !E \mid p \mid q$$

5. $G_5 = (\{S, A\}, \{a, b\}, P, S)$

$$\begin{aligned} S &\rightarrow aAb \\ A &\rightarrow b \mid \epsilon \end{aligned}$$

Ejercicio 5. Supongamos que la tabla de análisis sintáctico para una gramática G no es LL(1) porque algunas entradas de la tabla tienen dos o más producciones. ¿La tabla de análisis sintáctico construida es completamente inútil? ¿Cómo se podría usar para mejorar un analizador sintáctico predictivo con *backtracking*?

Ejercicio 6. Dada la gramática $G = (\{S\}, \{a, b\}, P, S)$:

$$S \rightarrow aSbS$$

analizar sintácticamente las siguientes cadenas utilizando el algoritmo de Earley:

1. $abab \in L(G)$

2. $aabb \in L(G)$

3. $aab \notin L(G)$

Ejercicio 7. Dada la gramática $G = (N, \Sigma, P, \text{grup_verb})$: con:

$$N = \{\text{grup_verb}, \text{sn}, \text{espec_ms}, \text{n_ms}, \text{verb}\}$$

$$\Sigma = \{\text{el}, \text{un}, \text{gato}, \text{perro}, \text{cielo}, \text{ladra}, \text{mira}\}$$

$$\begin{array}{ll} \text{grup_verb} & \rightarrow \text{sn verb} \mid \text{sn verb sn} \\ \text{sn} & \rightarrow \text{espec_ms n_ms} \\ \text{espec_ms} & \rightarrow \text{el} \mid \text{un} \\ \text{n_ms} & \rightarrow \text{gato} \mid \text{perro} \mid \text{cielo} \\ \text{verb} & \rightarrow \text{ladra} \mid \text{mira} \end{array}$$

analizar sintácticamente las siguientes cadenas utilizando el algoritmo de Earley:

1. **un gato mira el cielo** $\in L(G)$

2. **el ladra perro** $\notin L(G)$