

Práctico Nro. 4: Gramáticas Libres del Contexto - Autómatas Push-Down

Análisis Sintáctico

Los ejercicios marcados se deben comprobar su resolución con la notebook
Note_AyL_Pract4.ipynb

Parte A: Gramáticas Libres del Contexto

Ejercicio 1.

Dada la siguiente gramática $G = \langle \{S, A\}, \{a, b\}, P, S \rangle$ donde

$$P = \{ \begin{array}{l} (1) S \rightarrow aaSA \\ (2) S \rightarrow ASaa \\ (3) S \rightarrow \lambda \\ (4) S \rightarrow SS \\ (5) A \rightarrow b \\ (6) A \rightarrow bbb \end{array} \}$$

- Obtener, utilizando la relación deriva, al menos dos cadenas diferentes derivadas desde G tal que, en una de ellas se use una secuencia de derivaciones de más a la derecha, y en la otra se use una secuencia de derivaciones más a la izquierda.
- Utilizando notación de conjunto, describir por comprensión el lenguaje que genera la gramática G .

Ejercicio 2.

- Diseñar una GLC para cada uno de los siguientes lenguajes:

$$L_1 = \{a^i b^j \mid i, j > 0, j = i - 1\}$$

$$L_2 = \{t^j u^{2*i} v^i s^{j+1} \mid i > 0, j \geq 0\}$$

$$L_3 = \{a^i b^{i+2} c^j d^k \mid i, j, k > 0, k > j\}$$

$$L_4 = \{(ab)^k c^i (ba)^k \mid k, j > 0, i = 2 * j + 1\}$$

- Determinar cuáles de estas afirmaciones son verdaderas, en caso afirmativo dé una secuencia de derivaciones de más a la izquierda o de más a la derecha:

$$1) w = abbbcddd \in L_3$$

$$2) w = abcccbaba \in L_4$$

- En los ítems del punto anterior en los cuales sea posible, mostrar un árbol de derivación cuya frontera sea la cadena.

Ejercicio 3.

Construir una GLC que genere código en HTML como se describe a continuación: El código HTML solamente tiene una sección de cuerpo delimitada por los tags `< body >` y `< /body >`. En el cuerpo puede tener como contenido párrafos delimitados por los tags: `< p >` y `< /p >` y es posible no tener ningún párrafo. Un párrafo debe tener al menos un texto y se puede mostrar en negrita delimitado por los tags: `< strong >` y `< /strong >`. El siguiente es un ejemplo de un código válido para este lenguaje. Asuma que está definido el no terminal que genera textos.

```
<body>
  <p> Materia <strong> Autómatas </strong> Licenciatura </p>
  <p> Gramáticas </p>
</body>
```

Ejercicio 4.

Construir una BNF que genere el lenguaje descrito a continuación: Se necesita generar una estructura de bloques anidados como la provista por el lenguaje C. El bloque tiene la siguiente estructura: El delimitador de apertura es `{` y el de cierre `}`, sus componentes tienen el siguiente formato: un terminal `s` al principio y el siguiente puede ser otro bloque pero esto es optativo. Un bloque sin componentes no es válido. El siguiente es un ejemplo de una cadena válida para este lenguaje.

```
{
  s
  {
    s
  }
}
```

Ejercicio 5.

Convertir las siguientes BNF's a BNFE's:

BNF1: `< declaracion clase > ::= < atributo > class i < cuerpo clase >`
 `< atributo > ::= λ | public`
 `< cuerpo clase > ::= { < bloque > }`
 `< bloque > ::= s; | s; < bloque >`

BNF2:

`< opción constante > ::= < lista opcional >`
 `< lista opcional > ::= λ | = { < lista inicializadores > }`
 `< lista inicializadores > ::= c < resto lista inic >`
 `< resto lista inic > ::= λ | , c < resto lista inic >`

Ejercicio 6.

Decir si las siguientes GLC's son ambiguas o no, justificando su respuesta.

$$G_1 = \langle \{S, A\}, \{a, b\}, P, S \rangle$$

$$P = \{S \rightarrow abS \mid aAb \mid \lambda \\ A \rightarrow bSa \mid ba\}$$

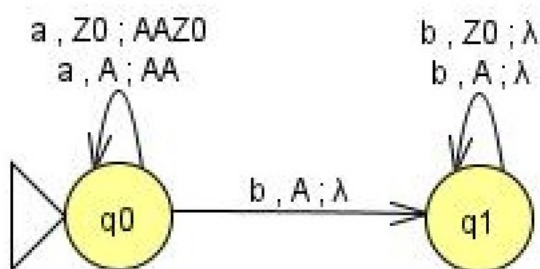
$$G_2 = \langle \{S, A\}, \{a, b\}, P, S \rangle$$

$$P = \{S \rightarrow AA \\ A \rightarrow AAA \mid bA \mid a \mid Ab\}$$

Parte B: Autómatas Push-Down

Ejercicio 1.

Dado el siguiente Autómata Push-Down (APD) M :



- ¿Cómo es la aceptación de cadenas en M , es por estado final o pila vacía? ¿es un APD determinístico?
- ☒ Determinar cuál de estas dos cadenas es reconocida por el APD y mostrar la secuencia de configuraciones para esa cadena:
 - 1) $aabbbb$
 - 2) $aabbb$
- Describir por comprensión, utilizando la notación de conjunto, el lenguaje reconocido por M .

Ejercicio 2.

- Diseñar un APD para cada uno de los siguientes lenguajes:

$$L_1 = \{a^i c^k \mid i > 0, k = i + 1\}$$

$$L_2 = \{d^j c^i e^k \mid j, i > 0, k > i\}$$

$$L_3 = \{a^i b^j c^k \mid i, j, k > 0, k = i + 2 * j + 1\}$$

$$L_4 = \{a^i b^j c^k d^s \mid i, j, k, s > 0, s = 2 * i + 1, k = j\}$$

- b. Determinar cuáles son APD determinísticos. Para los no determinísticos, decidir si es posible diseñar uno determinístico, y en caso de ser así, construirlo.
- c. Decidir cuáles de los siguientes ítems son verdaderos, para los afirmativos, utilizando el correspondiente autómata construido en el punto a, mostrar la secuencia de descripciones instantáneas para aceptar la cadena:
 - 1) $w = dceee \in L_2$
 - 2) $w = abbccdd \in L_4$

Ejercicio 3.

A partir del APD del ejercicio 1 de la Parte B, obtener el APD M' que acepte por estado final. Aplicar la transformación vista en teoría.

Ejercicio 4.

Construir un APD M que acepte por estado final el lenguaje L , y luego, desde M , obtener el APD M' que acepte por pila vacía, aplicar la transformación vista en teoría:

$$L = \{a^i b^j c^k \mid i, j, k > 0, j = i + 2 * k\}$$

Ejercicio 5.

Determinar el valor de verdad de cada uno de los siguientes ítems y justificar adecuadamente:

- a. Para reconocer una cadena un APD que reconoce por pila vacía, requiere solamente que se vacíe la pila.
- b. Para todo APD no determinístico se puede construir un APD determinístico que reconozca el mismo lenguaje.
- c. Un APD puede determinar si un programa escrito en C será aceptado por el compilador del lenguaje C.

Ejercicio 6.

- a. Construir los analizadores sintácticos top-down y bottom-up basados en APD para cada una de las siguientes gramáticas:

G_1 = Gramática L_2 del ejercicio 2 de la parte A. Se debe rotular las producciones.

$$G_2 = \langle \{S, A, B\}, \{<, >, ,, 1\}, P, S \rangle$$

$$P = \{ \begin{array}{l} (1) S \rightarrow < A > \\ (2) S \rightarrow < > \\ (3) A \rightarrow B, A \\ (4) A \rightarrow B \\ (5) B \rightarrow S \\ (6) B \rightarrow 1 \end{array} \}$$

- b. Determinar si es posible construir un árbol de derivación para cada una de las cadenas. De ser posible, mostrar la secuencia de descripciones instantáneas y el parser izquierdo utilizando el analizador top-down construido en el punto a:
- $iw = tuvss \in L(G_1)$?
 - $\boxed{\odot} iw = \langle\langle >, 1 > \in L(G_2)$?
- c. Idem al punto b pero en este caso utilizar el analizador bottom-up construido en el punto a y mostrar el parser derecho:
- $iw = tuuvss \in L(G_1)$?
 - $iw = < 1 \in L(G_2)$?

Ejercicios Complementarios

Ejercicio 1.

Diseñar una GLC para cada uno de los siguientes lenguajes:

$$L_1 = \{a^i b^{i+1} c^j d^{2*j} / i, j > 0\}$$

$$L_2 = \{(ab)^k b^s c^i / k, j, s > 0, i = 2 * j, s = k + 3\}$$

Ejercicio 2.

Convertir las siguientes BNF's a BNFE's:

BNF1:

$$\begin{aligned} \langle \text{EntradaSalida} \rangle &::= \text{cin}' \gg' \langle \text{Identificador} \rangle \langle \text{Resto} \rangle \\ \langle \text{Resto} \rangle &::= \lambda \mid ' \gg' \langle \text{Identificador} \rangle \langle \text{Resto} \rangle \end{aligned}$$

BNF2:

$$\begin{aligned} \langle \text{Term} \rangle &::= \neg \langle \text{Expr} \rangle \mid \langle \text{Expr} \rangle \\ \langle \text{Expr} \rangle &::= \langle \text{Valor} \rangle \mid \langle \text{Expr} \rangle = \langle \text{Valor} \rangle \\ \langle \text{Valor} \rangle &::= 0 \mid 1 \end{aligned}$$

Ejercicio 3.

Diseñar un APD para cada uno de los siguientes lenguajes:

$$L_1 = \{ww^R / w \in \{a, b\}^*\}$$

$$L_2 = \{(ba)^i c^k a^l b^j / i, k, l, j > 0, j = i + 1 \text{ y } k > l\}$$

Ejercicio 4.

Construir los analizadores sintácticos top-down y bottom-up basados en APD para la siguiente gramática:

$$G_2 = \langle \{S, A, B\}, \{-, \cup, \neg, cj\}, P, S \rangle$$

$$P = \{ \begin{aligned} (1) & S \rightarrow B \\ (2) & S \rightarrow S - B \\ (3) & B \rightarrow A \cup B \\ (4) & B \rightarrow A \\ (5) & A \rightarrow cj \\ (6) & A \rightarrow \neg cj \end{aligned} \}$$