UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ciencias

Integrantes:
Adrián Aguilera Moreno
Sebastián Alejandro Gutierrez Medina

Compiladores

Tarea 02

1. Considera la siguiente gramática:

$$S \to aSbS|bSaS|\epsilon$$

Construye dos derivaciones por la derecha para la cadena abaabb.
¿Cómo es el árbol se sintaxis concreta para esta cadena, es único?

Solución. A continuación se dan las dos derivaciones por la derecha requeridas, estas son

Derivación 1.

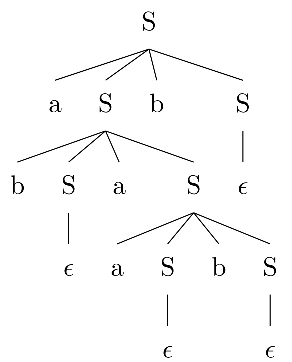
$$\begin{aligned} S &\to aSbS \\ &\to abSaSb\epsilon \\ &\to ab\epsilon aaSbSb \\ &\to abaa\epsilon b\epsilon b \\ &\to abaabb \end{aligned}$$

Derivación 2.

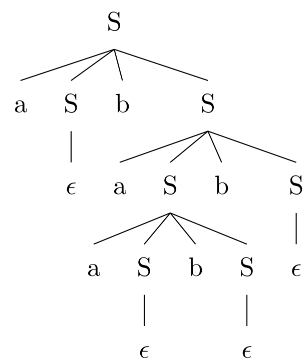
$$\begin{aligned} S &\to aSbS \\ &\to a\epsilon baSbS \\ &\to abaaSbSb\epsilon \\ &\to abaa\epsilon b\epsilon b \\ &\to abaabb \end{aligned}$$

Los árboles de sintáxis abstracta se muestran a continuación

Derivación 1.



Derivación 2.



Como se puede ver, el árbol de sintáxis abstracta no es único.

2. ¿Qué lenguaje genera la gramática siguiente?

$$S \to aAb \mid aAA \mid aB \mid bbA \quad A \to aAb \mid ab \quad B \to bBa \mid ba$$

Solución. Primero observemos que

$$A \to aAb \mid ab$$

puede ser traducido como expresión regular $(a^k b^k + ab)$ con $k \in \mathbb{N}/\{0\}$ y

$$B \to bBa \mid ba$$

puede ser traducido como la expresión regular $(ba + b^n a^n)$ con $n \in \mathbb{N}/\{0\}$. Así, tenemos que el lenguaje generado por la gramática es

$$\big(a(a^k b^k + ab)b + a(a^n b^n + ab)^2 + a(ba + b^m a^m) + bb(a^p b^p + ab)\big)$$

con $m, p \in \mathbb{N}/\{0\}$.

3. Considere la siguiente gramática:

$$\begin{array}{rcl} S & \to & aSb \mid A \\ A & \to & aAd \mid cBd \\ B & \to & aBb \mid ab \end{array}$$

Aplica el algoritmo CYK para verificar si las cadenas cabd y aabb se pueden generar con la gramática. Muestra y explica el proceso de ejecución del algoritmo y si realizas alguna transformación de la gramática indica el motivo y la técnica usada.

Solución. Primero, llevemos nuestra gramática a la FNC[1]

1. Eliminamos producciones unitarias:

$$\begin{array}{rcl} S & \to & aSb \mid aAd \mid cBd \\ A & \to & aAd \mid cBd \\ B & \to & aBb \mid ab. \end{array}$$

2. Agregando producciones en FNC:

$$\begin{array}{rcl} S & \to & aSb \mid aAd \mid cBd \\ A & \to & aAd \mid cBd \\ B & \to & aBb \mid ab \\ A' & \to & a \\ B' & \to & b \\ C' & \to & c \\ D' & \to & d \\ T_1 & \to & A'S \\ T_2 & \to & A'A \\ T_3 & \to & C'B \\ T_4 & \to & A'B. \end{array}$$

3. Sustituyendo en nuestra gramática original tenemos que

$$\begin{array}{rcl} S & \to & T_1B \mid T_2D' \mid T_3D' \\ A & \to & T_2D' \mid T_3D' \\ B & \to & T_4D' \mid A'B' \\ A' & \to & a \\ B' & \to & b \\ C' & \to & c \\ D' & \to & d \\ T_1 & \to & A'S \\ T_2 & \to & A'A \\ T_3 & \to & C'B \\ T_4 & \to & A'B. \end{array}$$

Ahora que nuestra gramática esta en FNC, podemos aplicar el algoritmo CYK[2]:

[1]Forma Normal de Chomsky.

[2]Me basaré en el libro: Teoría de autómatas, lenguajes y computación, de JOHN E. HOPCROFT, RAJEEV MOTWANI, y JEFFREY D. ULLMAN.

Caso cabd:

S			
T_3	-		
-	B	-	
C'	A'	B'	D'
c	a	b	d

Caso aabb:

B			
T_4	-		
-	B	-	
A'	A'	B'	B'
a	a	b	b

Como podemos ver, el caso "cabd" logra procesar la cadena y por tanto la gramática la genera. En el caso "aabb" no se logra procesar la cadena hasta S y por tanto nuestra gramática no le genera.

4. La siguiente gramática describe un lenguaje de consultas simple en donde `STRING` es un símbolo terminal:

```
 Session → Fact Session
 Session → Question
 Session → ( Session ) Session
    Fact → ! STRING
Question → ? STRING
```

Calcula las funciones FIRST y FOLLOW para los símbolos no-terminales de la gramática.

- FIRST
  - FIRST(Session) = {FIRST(Fact), FIRST(Question), (}
  - FIRST(Fact) = {!}
  - FIRST(Question) = {?}
- FOLLOW
  - FOLLOW(Session) = {FIRST(Session), STRING}
  - FOLLOW(Fact) = {STRING}
  - FOLLOW(Question) = {STRING}

5. Considera la siguiente gramática:

$$\begin{array}{rcl} E & \to & -E \mid (E) \mid VE' \\ E' & \to & -E \mid \varepsilon \\ V & \to & \texttt{id}V' \\ V' & \to & (E) \mid \varepsilon \end{array}$$

1. Muestra el cálculo de los conjuntos First y Follow.
    - FIRST
        - FIRST(E) = {-, (, FIRST(V)}
        - FIRST(E') = {-}
        - FIRST(V) = {i}
        - FIRST(V') = {(}
    - FOLLOW
        - FOLLOW(E) = {FIRST(E), FIRST(E')}
        - FOLLOW(E') = {FIRST(E)}
        - FOLLOW(V) = {d}
        - FOLLOW(V') = {FIRST(E)}
2. Muestra dos árboles de sintaxis, uno abstracta y otro concreta para la cadena $-\texttt{id}(-\texttt{id})-\texttt{id}$.

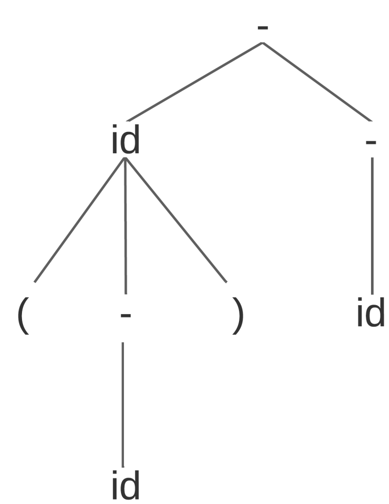


Figure 1: Arbol de Sintaxis Abstracta