Resumen Normalización de Gramáticas Libres de Contexto Facultad de Ciencias UNAM [*]

Favio E. Miranda Perea A. Liliana Reyes Cabello Lourdes González Huesca

La normalización de gramáticas consiste en transformar todas las producciones de una gramática de manera que tengan cierta forma sintáctica en particular. Así la normalización de gramáticas libres de contexto es útil para homogeneizar la forma de las producciones además para optimizar los procesos de derivación de cadenas. Con las **formas normales** se facilita una solución al problema de la pertenencia, es decir decidir si una cadena pertenece o no a un lenguaje:

Dada una gramática G y una palabra w, ¿se cumple que $w \in L(G)$?
Es decir, pertenece w al lenguaje generado por G.

- Si la palabra w es generada por G, el proceso para la construcción de un árbol de derivación terminará eventualmente.
- En caso contrario **no** podemos saber cuándo detener la construcción de un árbol de derivación.

1. Normalización de Gramáticas

Para obtener gramáticas normales, son necesarias varias transformaciones que simplificarán y reorganizarán las producciones.

1.1. Eliminación de variables inútiles

Definición 1.1 *Decimos que una variable A es **accesible** o **alcanzable** si existen $u, v \in (V \cup T)^\star$ tales que $S \to^\star uAv$. Obsérvese que según esta definición S siempre es alcanzable.*

Definición 1.2 *Una variable A es **productiva** o **terminable** si existe $w \in T^\star$ tal que $A \to^\star w$. En particular si $A \to \varepsilon$ es una producción entonces A es productiva.*

Definición 1.3 *Una variable A es **inútil** si no es alcanzable o no es productiva.*

A continuación damos un algoritmo para hallar variables productivas, mediante la transformación de una gramática:

Iniciar el conjunto *Prod* con todas las producciones que contienen cadenas de terminales a la derecha:

$$Prod := \{A \in V \mid A \to w \ \in P,\ w \in T^\star\}$$

[*]Material para el curso de Autómatas y Lenguajes Formales, desarrollado bajo el proyecto UNAM-PAPIME PE102117 2017–2018.

Repetir la incorporación de variables cuyas producciones contienen variables productivas y símbolos terminales a la derecha de $\rightarrow$:

$$Prod := Prod \cup \{A \in V \mid A \rightarrow w,\ w \in (T \cup Prod)^\star\}$$

Hasta que no se añaden nuevas variables a $Prod$.

Ejemplo: Calculemos las variables productivas de la gramática:

$$\begin{array}{ll} S \rightarrow & ACD \mid bBd \mid ab \\ A \rightarrow & aB \mid aA \mid C \\ B \rightarrow & aDS \mid aB \\ C \rightarrow & aCS \mid CB \mid CC \\ D \rightarrow & bD \mid ba \\ E \rightarrow & AB \mid aDb \end{array}$$

Iniciamos con $Prod = \{S, D\}$. Las iteraciones nos llevan a que C es la única variable improductiva, se elimina esta variable junto con todas las reglas donde figure:

$$\begin{array}{ll} S \rightarrow & bBd \mid ab \\ A \rightarrow & aB \mid aA \\ B \rightarrow & aDS \mid aB \\ D \rightarrow & bD \mid ba \\ E \rightarrow & AB \mid aDb \end{array}$$

El siguiente algoritmo permite hallar variables accesibles:

Iniciar $Acc := \{S\}$

Repetir la incorporación de variables que aparecen a la derecha de $\rightarrow$ en producciones de variables accesibles:

$$Acc := Acc \cup \{A \in V \mid \exists\ B \rightarrow uAv \in P,\ B \in Acc,\ u, v \in (V \cup T)^\star\}$$

Hasta que no se añaden nuevas variables a Acc.

Ejemplo: Calculemos las variables accesibles de la gramática:

$$\begin{array}{ll} S \rightarrow & aS \mid AaB \mid ACS \\ A \rightarrow & aS \mid AaB \mid AC \\ B \rightarrow & bB \mid DB \mid BB \\ C \rightarrow & aDa \mid ABD \mid ab \end{array} \qquad \begin{array}{ll} D \rightarrow & aD \mid DD \mid ab \\ E \rightarrow & FF \mid aa \\ F \rightarrow & aE \mid EF \end{array}$$

Iniciamos con $Acc = \{S\}$. El resultado es: $Acc = \{S, A, B, C, D\}$ ya que E y F son variables inaccesibles, se eliminan junto con todas las reglas donde figuren:

$$\begin{array}{ll} S \rightarrow & aS \mid AaB \mid ACS \\ A \rightarrow & aS \mid AaB \mid AC \\ B \rightarrow & bB \mid DB \mid BB \\ C \rightarrow & aDa \mid ABD \mid ab \\ D \rightarrow & aD \mid DD \mid ab \end{array}$$

Para eliminar variables inútiles se aplican los dos algoritmos anteriores **en el siguiente orden**:

1. Eliminar variables no productivas.
2. Eliminar variables no accesibles.

La importancia del orden de los algoritmos radica en que si se aplican los algoritmos en orden inverso el resultado puede ser una gramática que aún contenga variable inútiles. Veamos un ejemplo:

Ejemplo: Considere la siguiente gramática

$$S \to a \mid AB \qquad\qquad A \to aA \mid \boldsymbol{\varepsilon}$$

Al eliminar primero las variables no accesibles se obtiene la misma gramática, al ser todas las variables accesibles.
Posteriormente al eliminar variables improductivas resulta

$$S \to a \qquad\qquad A \to aA \mid \boldsymbol{\varepsilon}$$

y claramente A es inútil por ser inaccesible.

1.2. Eliminación de ε-producciones

Las gramáticas libres de contexto permiten el uso de producciones de la forma $A \to \boldsymbol{\varepsilon}$. La eliminación de estas producciones llamadas ε-producciones genera una transformación.

Definición 1.4 *Una variable A se llama **anulable** si $A \to^\star \boldsymbol{\varepsilon}$, es decir si una derivación que empieza en A genera la cadena vacía.*

Veamos un algoritmo para hallar variables anulables:

Iniciar el conjunto *Anul* con las variables que tienen $\boldsymbol{\varepsilon}$ como producción

$$Anul := \{A \in V \mid A \to \boldsymbol{\varepsilon} \ \in P\}$$

Repetir la incorporación de variables que tienen producciones cadenas de variables anulables

$$Anul := Anul \cup \{A \in V \mid \exists\, A \to w \ \in P,\ \ w \in Anul^\star\}$$

Hasta que no se añaden nuevas variables a *Anul*

Una vez que se han identificado las variables anulables, la siguiente transformación de una gramática libre de contexto elimina exactamente las ε-producciones:

> Para cada producción en la gramática que tenga la forma $A \to w_1 \dots w_n$ se deben agregar las producciones $A \to v_1 \dots v_n$ que son resultantes de los cambios de símbolos donde:
>
> - $v_i = w_i$ si $w_i \notin Anul$, se respetan las variables no anulables
> - $v_i = w_i$ ó $v_i = \boldsymbol{\varepsilon}$ si $w_i \in Anul$, las variables anulables pueden dejarse o eliminarse
>
> Verificando que no se anulen todos los v_i al mismo tiempo.

Es decir, se van a respetar las producciones existentes y si alguna contiene las variables anulables se agregarán las producciones que resulten de eliminar las variables anulables en esa producción. Las $\boldsymbol{\varepsilon}$-producciones desaparecerán.

Ejemplo: Eliminación de $\boldsymbol{\varepsilon}$-producciones de la gramática

$$\begin{array}{ll} S \to & AB \mid ACA \mid ab \\ A \to & aAa \mid B \mid CD \\ B \to & bB \mid bA \\ C \to & cC \mid \varepsilon \\ D \to & aDc \mid CC \mid ABb \end{array}$$

Primero obtenemos las variables anulables, iniciando con $Anul = \{C\}$:

$$Anul = \{C, D, A, S\}$$

El proceso de anulación de variables hace que se elimine la producción $C \to \boldsymbol{\varepsilon}$, se dejan las producciones existentes y se agregan las producciones que eliminan los elementos de $Anul$. Se obtiene la siguiente gramática:

$$\begin{array}{ll} S \to & AB \mid ACA \mid ab \mid \mathbf{B} \mid \mathbf{CA} \mid \mathbf{AA} \mid \mathbf{AC} \mid \mathbf{A} \mid \mathbf{C} \\ A \to & aAa \mid B \mid CD \mid \mathbf{aa} \mid \mathbf{C} \mid \mathbf{D} \\ B \to & bB \mid bA \mid \mathbf{b} \\ C \to & cC \mid \mathbf{c} \\ D \to & aDc \mid CC \mid ABb \mid \mathbf{ac} \mid \mathbf{C} \mid \mathbf{Bb} \end{array}$$

Acerca de la palabra vacía Si originalmente se tenía $\boldsymbol{\varepsilon} \in L(G)$ la eliminación de $\boldsymbol{\varepsilon}$-producciones genera una gramática que **no** genera a $\boldsymbol{\varepsilon}$. Es posible saber si se pierde la palabra vacía al eliminar $\boldsymbol{\varepsilon}$-producciones verificando si $S \in Anul$.
Si se quiere recuperar a $\boldsymbol{\varepsilon}$ debe agregarse un nuevo símbolo inicial S' así como las producciones $S' \to S$ y $S' \to \varepsilon$. De esta forma, $S' \to \varepsilon$ es la única $\boldsymbol{\varepsilon}$-producción permitida.

1.3. Eliminación de producciones unitarias

Definición 1.5 *Una producción de la forma $A \to B$ donde A y B son ambas variables se llama **producción unitaria**. El **conjunto unitario** de A se define como sigue:*

$$Unit(A) = \{B \in V \mid A \to^{\star} B \text{ usando sólo producciones unitarias } \}$$

Obsérvese que por definición se tiene $A \in Unit(A)$ si existe $A \to B$.
Ahora veamos un algoritmo para hallar el conjunto $Unit(A)$ para cualquier variable o símbolo no-terminal:

Iniciar $Unit(A) := \{A\}$

Repetir la incorporación de variables que tengan producciones unitarias

$$Unit(A) := Unit(A) \cup \{B \in V \mid \exists\, C \to B,\ \ C \in Unit(A)\ \}$$

Hasta que no se añaden nuevas variables a $Unit(A)$.

Para la eliminación de producciones unitarias en una gramática se debe realizar la siguiente transformación, para cada variable se debe calcular su conjunto unitario y realizar las siguientes acciones:

- Para cada $B \in Unit(A)$ y cada producción $A \to w$ agregar la producción

$$B \to w$$

- Eliminar todas las producciones unitarias

Ejemplo: Eliminación de producciones unitarias de la gramática:

$$\begin{array}{ll} S \to & AS \mid AA \mid BA \mid \varepsilon \\ A \to & aA \mid a \\ B \to & bB \mid bC \mid C \\ C \to & aA \mid bA \mid B \mid ab \end{array}$$

Los conjuntos unitarios para cada variable son:

$$\begin{array}{ll} Unit(S) = \{S\} & Unit(A) = \{A\} \\ Unit(B) = \{B, C\} & Unit(C) = \{C, B\} \end{array}$$

Así la gramática obtenida al eliminar producciones unitarias es:

$$\begin{array}{ll} S \to & AS \mid AA \mid BA \mid \varepsilon \\ A \to & aA \mid a \\ B \to & bB \mid bC \mid aA \mid bA \mid ab \\ C \to & aA \mid bA \mid ab \mid bB \mid bC \end{array}$$

1.4. Remplazo simple de producciones

Esta transformación eliminará reglas de la forma $A \to uBv$ donde $B \to w_1 \mid w_2 \mid \cdots \mid w_n$ y las cadenas w_i son de símbolos terminales. Esto se logra al redefinir el conjunto de reglas P en la gramática G con

$$P' = (P - \{A \to uBv\}) \cup \{A \to uw_1v \mid uw_2v \mid \ldots uw_nv\}$$

Ejemplo: Reemplazar la regla $D \to ABF$ de la gramática siguiente:

$$\begin{array}{ll} S \to & DA \mid aF \\ D \to & ABF \mid a \\ A \to & a \\ B \to & b \mid ba \\ F \to & AF \mid A \end{array}$$

Se obtiene:

$$\begin{array}{ll} S \to & DA \mid aF \\ D \to & AbF \mid AbaF \mid a \\ A \to & a \\ B \to & b \mid ba \\ F \to & AF \mid A \end{array}$$

1.5. Remoción de producciones recursivas por la izquierda

Una producción recursiva por la izquierda es de la forma $A \to Aw$ con $w \in (V \cup T)^*$.
Ellas serán reemplazadas por producciones cuya recursión es por la derecha de la siguiente forma, para cada variable o símbolo no-terminal A crear dos categorías de reglas:

1. $A_{izq} = \{A \to Au_i \in P \mid u_i \in (V \cup T)^*\}$
2. $A_{der} = \{A \to v_j \in P \mid v_j \in (V \cup T)^*\}$

A continuación se obtendrán las reglas para la variable A como sigue:

$$A \to v_1 \mid \cdots \mid v_m \mid v_1 Z \mid \cdots \mid v_n Z$$

Además se incorporarán las siguientes reglas a P:

$$Z \to u_1 Z \mid \cdots \mid u_n Z \mid u_1 \mid \cdots \mid u_n$$

Ejemplo: Considere las siguientes producciones:

$$A \to Aa \mid Aab \mid bb \mid b$$

Se eliminarán las producciones recursivas por la izquierda:

$$A_{izq} = \{A \to Aa,\ A \to Aab\} \qquad A_{der} = \{A \to bb,\ A \to b\}$$

generando la siguiente gramática:

$$\begin{array}{ll} A \to & bb \mid b \mid bbZ \mid bZ \\ Z \to & aZ \mid abZ \mid a \mid ab \end{array}$$

1.6. Problema de la pertenencia

Dada una gramática G *sin ε-producciones ni producciones unitarias* y una palabra w de longitud n, ¿se cumple que $w \in L(G)$? Analicemos esta situación:

- Si la palabra w es generada por G la construcción de un árbol de derivación terminará eventualmente.
- En caso contrario basta con construir un árbol hasta el nivel $\mathbf{2n-1}$ para concluir que $w \notin L(G)$, es decir un árbol que pueda tener como hojas a la cadena w.
- En cada paso de la construcción de un árbol se obtiene un nuevo terminal (a lo más n pasos) o se aumenta la longitud de la palabra en 1 (a lo más $n-1$ pasos)

2. Formas Normales

Después de haber introducido varias transformaciones de gramáticas libres de contexto, veamos ahora dos formas normales de gran utilidad.

2.1. Forma Normal de Chomsky

Definición 2.1 *Una gramática libre de contexto G está en* ***forma normal de Chomsky (FNC)*** *si:*

- *G no contiene variables inútiles.*
- *G no contiene producciones unitarias ni ε-producciones (salvo $S \to \boldsymbol{\varepsilon}$)*
- *Todas las producciones son binarias con variables o terminales, es decir de la forma:*

$$A \to BC \quad \acute{o} \quad A \to a \qquad \text{donde } B, C \in V,\ a \in T$$

Cualquier gramática libre de contexto es equivalente a una gramática en **FNC**, lo cual se logra parcialmente como sigue:

1. Eliminar las variables inútiles.
2. Eliminar las ε-producciones, salvo cuando la cadena $\boldsymbol{\varepsilon}$ pertenece al lenguaje original $L(G)$ y en ese caso se agrega un nuevo símbolo inicial S' y $S' \to \boldsymbol{\varepsilon} \mid S$.
3. Eliminar producciones unitarias.
4. Las producciones restantes son todas de la forma $A \to a$ con $a \in T$ ó $A \to w$ con $|w| \geq 2$

El proceso restante es la eliminación de producciones $A \to w$ donde $|w| \geq 2$. A este proceso le llamaremos **simulación de producciones**, para ello basta hacer lo siguiente para cada producción P de la forma $A \to \alpha_1\alpha_2 \ldots \alpha_n$ con $\alpha_i \in V \cup T$ y $n \geq 2$ (obsérvese que si $n = 2$, al menos uno de α_1, α_2 debe ser terminal, pues si no la producción ya es válida para FNC)

- Si $\alpha_i \in T$, digamos $\alpha_i = a$, entonces
  1. Agregar la producción $T_a \to a$, donde T_a es una nueva variable.
  2. Cambiar α_i por T_a en la producción P
- Para cada producción P de la forma $A \to B_1B_2 \ldots B_m$ con $B_i \in V,\ m \geq 3$
  1. Agregar $(m-2)$ nuevas variables $D_1, \ldots, D_{m-2}$ y reemplazar a P con las siguientes producciones:

$$A \to B_1D_1 \quad D_1 \to B_2D_2 \quad \ldots \quad D_{m-2} \to B_{m-1}B_m$$

Ejemplo: Simulación de producciones $A \to w_1w_2 \ldots w_n,\ n \geq 2$.
La producción $A \to abBaC$ se simula con producciones simples y binarias como sigue:

- Agregamos las nuevas variables T_a, T_b y las producciones

$$A \to T_aT_bBT_aC \qquad T_a \to a \qquad T_b \to b$$

- Para simular la producción $A \to T_aT_bBT_aC$ agregamos nuevas variables D_1, D_2, D_3 y las producciones binarias necesarias obteniendo finalmente la gramática:

$$A \to T_aD_1 \qquad D_1 \to T_bD_2 \qquad D_2 \to BD_3 \qquad D_3 \to T_aC$$
$$T_a \to a \qquad T_b \to b$$

Ejemplo: Transformar la siguiente gramática a **FNC**:

$$\begin{array}{lll} S \to & AB \mid aBC \mid SBS & \qquad A \to \quad aA \mid C \\ B \to & bbB \mid b & \qquad C \to \quad cC \mid \varepsilon \end{array}$$

La gramática resultante equivalente es:

$$\begin{array}{lll} S \to & AB \mid T_aD_1 \mid SD_2 \mid T_aB \mid T_bD_3 \mid b \\ A \to & T_aA \mid T_cC \mid a \mid c \\ B \to & T_bD_3 \mid b \\ C \to & T_cC \mid c \end{array} \qquad \begin{array}{lll} D_1 \to & BC \\ D_2 \to & BS \\ D_3 \to & T_bB \end{array} \qquad \begin{array}{lll} T_a \to & a \\ T_b \to & b \\ T_c \to & c \end{array}$$

2.2. Forma Normal de Greibach

Otra forma normal es la llamada de Greibach, esta forma normal es útil en algoritmos de parsing ya que se da prioridad a los prefijos terminales de las cadenas generadas por las reglas de producción de la gramática. Esto asegura que no existe recursión por la izquierda de ninguna forma.

Definición 2.2 *Una gramática libre de contexto G está en* ***forma normal de Greibach (FNG)*** *si:*

- *La variable inicial S no es recursiva, es decir, no figura en el lado derecho de las producciones.*
- *G no tiene variables inútiles ni ε-producciones salvo $S \to \varepsilon$.*
- *Todas las producciones son de la forma $A \to a\alpha$ con $a \in T$ y $\alpha \in V^\star$.*

Veamos cómo transformar una gramática en una equivalente en forma normal de Greibach, obsérvese que cualquier gramática libre de contexto es equivalente a una gramática en **FNG**.

El proceso requiere primero que la gramática esté en Forma Normal de Chomsky, de esta forma se eliminan las producciones inútiles y las ε-producciones.
Después, se aplica el siguiente método para transformar a **FNG**:

1. Verificar que el símbolo inicial no sea recursivo, si lo es entonces agregar un nuevo símbolo inicial con las producciones del anterior.
2. Eliminar las producciones recursivas por izquierda.
3. Enumerar u ordenar las variables o símbolos no-terminales de la gramática: iniciar con el símbolo inicial y el resto de las variables siguen cualquier orden.
4. Reemplazar las producciones que contengan variables de orden mayor en el lado derecho de $\to$, es decir todas las reglas deben tener la forma:
   - $A \to Bw$ donde la variable A debe estar por debajo de la variable B en el orden propuesto.
   - $A \to aw$ con $a \in T$

Finalmente la gramática resultante está en forma normal de Greibach, esta transformación permite simplificar el problema de la pertenencia: dada una gramática G **en forma normal de Greibach** y una palabra w de longitud n, ¿Se cumple $w \in L(G)$? Es decir, pertenece w al lenguaje generado por G.

- Si w es generada por G la construcción de un árbol de derivación entonces debe terminará eventualmente.
- En caso contrario basta con construir el árbol hasta el nivel **n** para concluir que $w \notin L(G)$.
- En cada paso se obtiene un nuevo terminal (a lo más n pasos) y por lo general hay una menor ramificación.

Ejemplo: Transformar la siguiente gramática G a **FNG**.

$$S \to SaB \mid aB \qquad B \to bB \mid \varepsilon$$

1. Eliminación de variables inútiles: buscar las que no sean productivas o inaccesibles. Para ello se calculan los conjuntos de variables productivas y accesibles:

$$Prod = \{B\} \qquad Acc = \{S, B\}$$

2. Eliminación de ε-producciones: se identifican las variables que son anulables, es decir que generan la cadena vacía de forma directa o combinando variables que generen la vacía. Para ello se calcula el conjunto de variables anulables:

$$Anul = \{B\} \qquad \begin{array}{rl} S \to & SaB \mid aB \mid Sa \mid a \\ B \to & bB \mid b \end{array}$$

3. Eliminación de producciones unitarias: se calculan las producciones unitarias para cada variable de la gramática. Si el conjunto tiene elementos diferentes a la variable en cuestión se sustituyen por las producciones.

$$Unit(S) = \{S\} \qquad Unit(B) = \{B\}$$

4. Simulación de producciones
   Símbolos terminales: se agregan variables T_a por cada símbolo terminal y se sustituyen en el resto de las producciones:

$$\begin{array}{rl} S \to & S\mathbf{T_a}B \mid \mathbf{T_a}B \mid S\mathbf{T_a} \mid a \\ B \to & \mathbf{T_b}B \mid b \end{array} \qquad \begin{array}{rl} \mathbf{T_a} \to & a \\ \mathbf{T_b} \to & b \end{array}$$

   Producciones binarias: se simplifican las producciones para hacerlas de tamaño uno (con un terminal) o dos (con dos no-terminales) agregando nuevas variables **D**:

$$\begin{array}{rl} S \to & S\mathbf{D} \mid T_aB \mid ST_a \mid a \\ B \to & T_bB \mid b \\ \mathbf{D} \to & T_aB \end{array} \qquad \begin{array}{rl} T_a \to & a \\ T_b \to & b \end{array}$$

5. Símbolo inicial no-recursivo: se cambia el símbolo inicial por uno nuevo S' con las producciones de S:

$$\begin{array}{rl} \mathbf{S'} \to & SD \mid T_aB \mid ST_a \mid a \\ S \to & SD \mid T_aB \mid ST_a \mid a \\ B \to & T_bB \mid b \\ D \to & T_aB \end{array} \qquad \begin{array}{rl} T_a \to & a \\ T_b \to & b \end{array}$$

6. Eliminación de producciones recursivas por la izquierda: se identifican los conjuntos de producciones izquierdas para cada variable de la gramática:

$$\begin{array}{ll} S'_{izq} = \varnothing & S_{izq} = \{S \to SD_1,\ S \to ST_a\} \\ B_{izq} = \varnothing & D_{1izq} = \varnothing \\ T_{bizq} = \varnothing & T_{bizq} = \varnothing \end{array}$$

A continuación se conservan las producciones derechas y se reemplazan las izquierdas con las subcadenas v_j y una nueva variable Z. Esta nueva variable tiene como producciones las subcadenas u_i siguiendo la forma descrita en la sección 1.5:

$$\begin{array}{rlrl} S' \to & SD \mid T_aB \mid ST_a \mid a & T_a \to & a \\ S \to & T_aB \mid a \mid T_aB\mathbf{Z} \mid a\mathbf{Z} & T_b \to & b \\ B \to & T_bB \mid b & & \\ D \to & T_aB & & \\ \mathbf{Z} \to & D\mathbf{Z} \mid T_a\mathbf{Z} \mid D \mid T_a & & \end{array}$$

7. Reemplazo simple de producciones: esta transformación determinará un orden de las variables para que la reescritura de producciones sólo sea entre variables de mayor orden. El orden de las variables no es único, pero se debe respetar al símbolo inicial como el de menor orden. Todas las producciones deben empezar con un símbolo terminal seguido de cero o varias variables:

orden	*producciones*	*reemplazar*	
0	$S' \to SD \mid T_aB \mid ST_a \mid a$	S	ya que se debe empezar con un terminal y la variable S no lo es
		T_a	ya que se debe empezar con un terminal
1	$S \to T_aB \mid a \mid T_aBZ \mid aZ$	T_a	ya que se debe empezar con un terminal
2	$B \to T_bB \mid b$	T_b	ya que se debe empezar con un terminal
3	$D \to T_aB$	T_a	ya que se debe empezar con un terminal
4	$Z \to DZ \mid T_aZ \mid D \mid T_a$	D	ya que el orden de $D(3)$ es menor que el de $Z(4)$
		T_a	ya que se debe empezar con un terminal
5	$T_a \to a$	$--$	
6	$T_b \to b$	$--$	

La gramática resultante se obtiene con reemplazos iterativos, se hizo comenzando por D, luego Z, seguido de S y al final S':

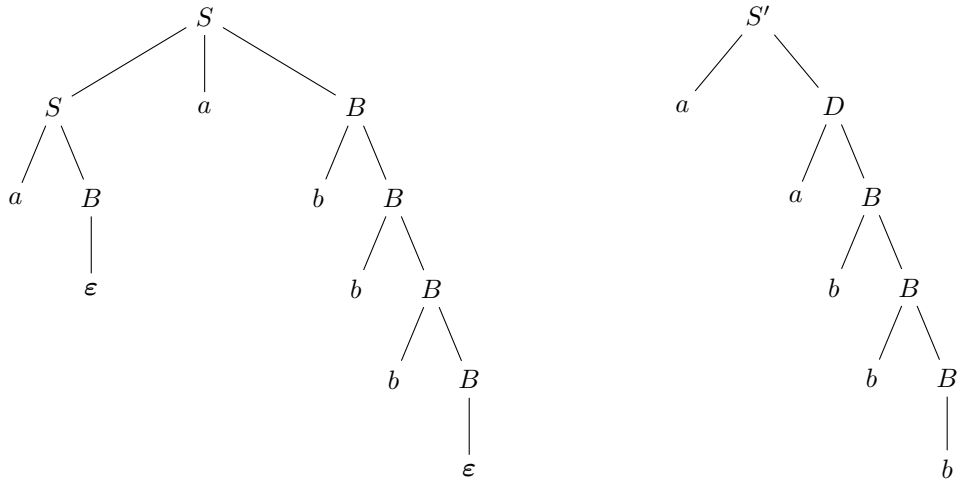
$$\begin{array}{rl} S' \to & aBD \mid aD \mid aBZD \mid aZD \mid aB \mid aBT_a \mid aT_a \mid aBZT_a \mid aZT_a \mid a \\ S \to & aB \mid a \mid aBZ \mid aZ \\ B \to & bB \mid b \\ D_1 \to & aB \\ Z \to & aBZ \mid aZ \mid aB \mid a \\ T_a \to & a \\ T_b \to & b \end{array}$$

Observemos lo siguiente:

- El lenguaje aceptado es:

$$L(G) = \{w \ \mid \ w = a^i b^j \text{ con } 1 < i < j \text{ o } 1 < j < i \text{ o } w = (ab)^+\}$$

- En particular la gramática del ejemplo no es ambigua.
- La obtención de la forma normal permite crear árboles con menor ramificación para el análisis de cadenas. Consideremos la cadena $w = aabbb$, el árbol izquierdo se obtuvo con la gramática sin transformar y el de la derecha, que tiene menos ramas, usando la gramática en forma normal de Greibach:



Referencias

[1] J.C. Martin. *Introduction to Languages and the Theory of Computation.* McGraw-Hill higher education. McGraw-Hill, 2003.