Las gramáticas de contexto libre son utilizadas para establecer las reglas sintácticas de los lenguajes de programación.

Gramática de contexto libre $G = (V_N, V_T, P, S)$ que genera expresiones aritéticas donde

- $V_N = \{S, E\}$,
- $V_T = \{\boldsymbol{identificador}, =, +, *, (,), \boldsymbol{número}\}$,
- $S \in V_N$
- y el conjunto de producciones es:

$$
\begin{array}{rcl}
P & = & \{ \\
 & (1) & S \longrightarrow \boldsymbol{identificador} \ = \ E \\
 & (2) & E \longrightarrow E \ + \ E \\
 & (3) & E \longrightarrow E \ * \ E \\
 & (4) & E \longrightarrow (\ E\ ) \\
 & (5) & E \longrightarrow \boldsymbol{identificador} \\
 & (6) & E \longrightarrow \boldsymbol{número} \\
 & & \}
\end{array}
$$

Conjunto de reglas de producción de una gramática que permite generar algunas sentencias de control de un lenguaje de programación:

$$
\begin{array}{rcl}
P & = & \{ \\
 & & S \longrightarrow \boldsymbol{mientras}\ C\ \boldsymbol{hacer}\ S\ \boldsymbol{fin\_mientras} \\
 & & S \longrightarrow \boldsymbol{si}\ C\ \boldsymbol{entonces}\ S\ \boldsymbol{fin\_si} \\
 & & S \longrightarrow \boldsymbol{si}\ C\ \boldsymbol{entonces}\ S\ \boldsymbol{si\_no}\ S\ \boldsymbol{fin\_si} \\
 & & S \longrightarrow \boldsymbol{Asignación} \\
 & & C \longrightarrow \boldsymbol{Condición} \\
 & & \}
\end{array}
$$

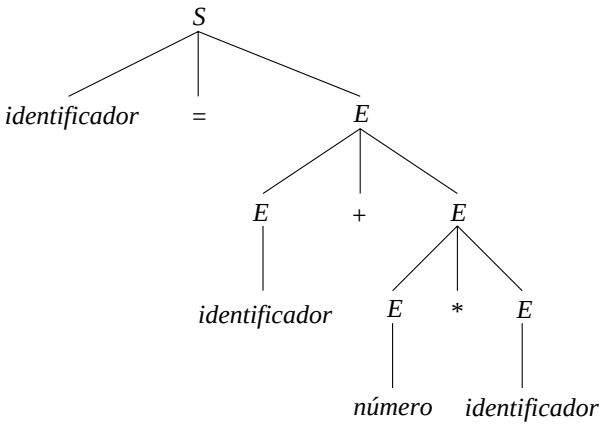
Derivación de una expresión aritmética

$$
\begin{aligned}
S &\underset{1}{\Rightarrow} \textbf{\textit{identificador}} = E \\
&\underset{2}{\Rightarrow} \textbf{\textit{identificador}} = E + E \\
&\underset{3}{\Rightarrow} \textbf{\textit{identificador}} = E + E * E \\
&\underset{6}{\Rightarrow} \textbf{\textit{identificador}} = E + \textbf{\textit{número}} * E \\
&\underset{5}{\Rightarrow} \textbf{\textit{identificador}} = E + \textbf{\textit{número}} * \textbf{\textit{identificador}} \\
&\underset{5}{\Rightarrow} \textbf{\textit{identificador}} = \textbf{\textit{identificador}} + \textbf{\textit{número}} * \textbf{\textit{identificador}}
\end{aligned}
$$

Ejemplo de derivación por la izquierda:

$$
\begin{aligned}
S &\underset{1}{\Rightarrow} \textbf{\textit{identificador}} = E \\
&\underset{2}{\Rightarrow} \textbf{\textit{identificador}} = E + E \\
&\underset{5}{\Rightarrow} \textbf{\textit{identificador}} = \textbf{\textit{identificador}} + E \\
&\underset{3}{\Rightarrow} \textbf{\textit{identificador}} = \textbf{\textit{identificador}} + E * E \\
&\underset{6}{\Rightarrow} \textbf{\textit{identificador}} = \textbf{\textit{identificador}} + \textbf{\textit{número}} * E \\
&\underset{5}{\Rightarrow} \textbf{\textit{identificador}} = \textbf{\textit{identificador}} + \textbf{\textit{número}} * \textbf{\textit{identificador}}
\end{aligned}
$$

Ejemplo de árbol de derivación de una gramática de contexto libre.



Sea G una gramática de contexto libre compuesta por el siguiente conjunto de reglas de producción:

$$\begin{aligned} P &= \{ \\ &(1) \quad S \longrightarrow aAa \\ &(2) \quad A \longrightarrow aAa \\ &(3) \quad A \longrightarrow bBb \\ &(4) \quad B \longrightarrow bBb \\ &(5) \quad B \longrightarrow c \\ &\quad \} \end{aligned}$$

El lenguaje que genera la gramatica G puede ser expresado como:

$$\begin{aligned} L(G) &= \{a^i b^j c b^j a^i | i, j \geq 1\} \\ &= \{abcba, abcba, abbcbba, \ldots, aacaa, aabbcbbaa, \ldots\} \end{aligned}$$

Este lenguaje se denomina "palíndromo impar" porque cada una de las palabras del lenguaje se puede leer igual de izquierda a derecha que de derecha a izquierda y tiene un elemento central que divide a la palabra.

La derivación que genera la palabra *aabbcbbaa* es la siguiente:

$$\begin{aligned} S &\underset{1}{\Rightarrow} aAa \\ &\underset{2}{\Rightarrow} aaAaa \\ &\underset{3}{\Rightarrow} aabBbaa \\ &\underset{4}{\Rightarrow} aabbBbbaa \\ &\underset{5}{\Rightarrow} aabbcbbaa \end{aligned}$$

La siguiente gramática es ambigua porque puede generar la cadena

identificador = identificador + número * identificador

mediante dos derivaciones por la izquierda diferentes.

$$
\begin{array}{rcl}
P & = & \{ \\
 & (1) & S \longrightarrow \textbf{\textit{identificador}} \ = \ E \\
 & (2) & E \longrightarrow E \ + \ E \\
 & (3) & E \longrightarrow E \ * \ E \\
 & (4) & E \longrightarrow (\ E\ ) \\
 & (5) & E \longrightarrow \textbf{\textit{identificador}} \\
 & (6) & E \longrightarrow \textbf{\textit{número}} \\
 & & \}
\end{array}
$$

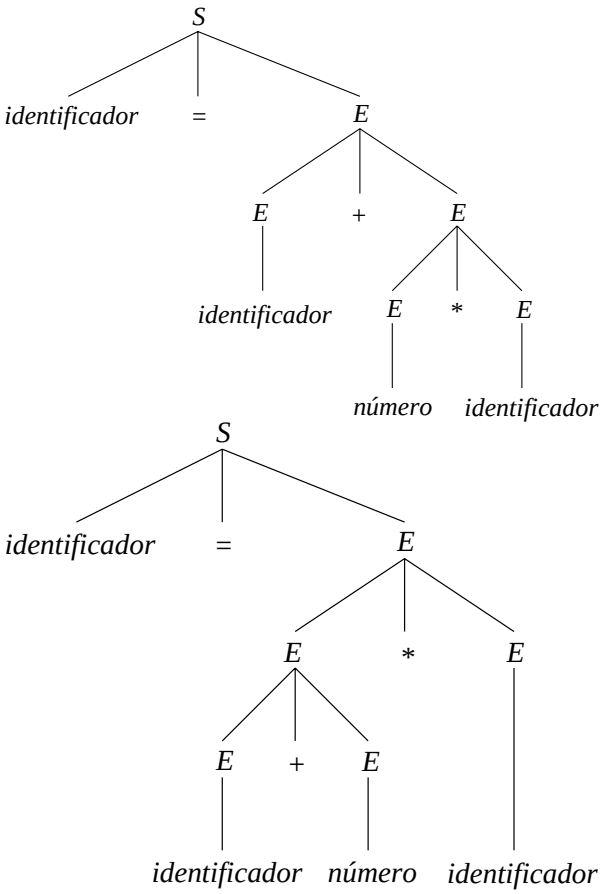
Primera derivación por la izquierda:

$$
\begin{array}{rcl}
S & \underset{1}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ E \\
 & \underset{2}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ E \ + \ E \\
 & \underset{5}{\Rightarrow} & \textbf{\textit{identificador}} \ = \textbf{\textit{identificador}} \ + \ E \\
 & \underset{3}{\Rightarrow} & \textbf{\textit{identificador}} \ = \textbf{\textit{identificador}} \ + \ E \ * \ E \\
 & \underset{6}{\Rightarrow} & \textbf{\textit{identificador}} \ = \textbf{\textit{identificador}} \ + \ \textbf{\textit{número}} \ * \ E \\
 & \underset{5}{\Rightarrow} & \textbf{\textit{identificador}} \ = \textbf{\textit{identificador}} \ + \ \textbf{\textit{número}} \ * \ \textbf{\textit{identificador}}
\end{array}
$$

Segunda derivación por la izquierda:

$$
\begin{array}{rcl}
S & \underset{1}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ E \\
 & \underset{3}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ E \ * \ E \\
 & \underset{2}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ E \ + \ E \ * \ E \\
 & \underset{5}{\Rightarrow} & \textbf{\textit{identificador}} \ = \textbf{\textit{identificador}} \ + \ E * \ E \\
 & \underset{5}{\Rightarrow} & \textbf{\textit{identificador}} \ = \textbf{\textit{identificador}} \ + \ \textbf{\textit{número}} \ * \ E \\
 & \underset{5}{\Rightarrow} & \textbf{\textit{identificador}} \ = \textbf{\textit{identificador}} \ + \ \textbf{\textit{número}} \ * \ \textbf{\textit{identificador}}
\end{array}
$$

Árboles de derivación diferentes para una misma cadena.



Conjunto de reglas de producción de una gramática no ambigua que puede generar expresiones aritméticas

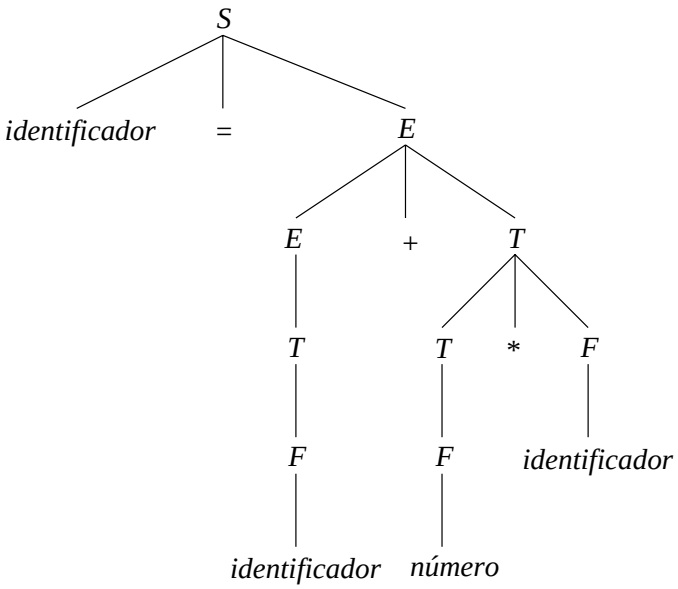
$$
\begin{array}{rcl}
P & = & \{ \\
 & (1) & S \longrightarrow \textbf{\textit{identificador}} \ = \ E \\
 & (2) & E \longrightarrow E \ + \ T \\
 & (3) & E \longrightarrow T \\
 & (4) & T \longrightarrow T \ * \ F \\
 & (5) & T \longrightarrow F \\
 & (6) & F \longrightarrow (\ E \ ) \\
 & (7) & F \longrightarrow \textbf{\textit{identificador}} \\
 & (8) & F \longrightarrow \textbf{\textit{número}} \\
 & & \}
\end{array}
$$

La derivación por la izquierda de la cadena

identificador = identificador + número * identificador

utilizando esta gramática es la siguiente:

$$
\begin{array}{rcl}
S & \underset{1}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ E \\
 & \underset{2}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ E \ + \ T \\
 & \underset{3}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ T \ + \ T \\
 & \underset{5}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ F \ + \ T \\
 & \underset{7}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ \textbf{\textit{identificador}} \ + \ T \\
 & \underset{4}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ \textbf{\textit{identificador}} \ + \ T \ * \ F \\
 & \underset{5}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ \textbf{\textit{identificador}} \ + \ F \ * \ F \\
 & \underset{8}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ \textbf{\textit{identificador}} \ + \ \textbf{\textit{número}} \ * \ F \\
 & \underset{7}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ \textbf{\textit{identificador}} \ + \ \textbf{\textit{número}} \ * \ \textbf{\textit{identificador}}
\end{array}
$$


S
identificador
=
E
E
+
T
T
T
*
F
F
F
identificador
identificador
número

El problema del "else danzante"

$$
\begin{array}{lcl}
 & \cdots & \\
(1)\ S & \longrightarrow & \textbf{\textit{if}}\ C\ S \\
(2)\ S & \longrightarrow & \textbf{\textit{if}}\ C\ S\ \textbf{\textit{else}}\ S \\
(3)\ S & \longrightarrow & I \\
 & \cdots &
\end{array}
$$

donde S genera sentencias de control, C genera expresiones condicionales e I genera otras sentencias, por ejemplo, de asignación.

Esta gramática es ambigua porque la sentencia

$$\textbf{\textit{if}}\ C\ \ \textbf{\textit{if}}\ C\ \ S\ \ \textbf{\textit{else}}\ S$$

puede ser generada mediante dos derivaciones que tienen dos árboles sintácticos diferentes:

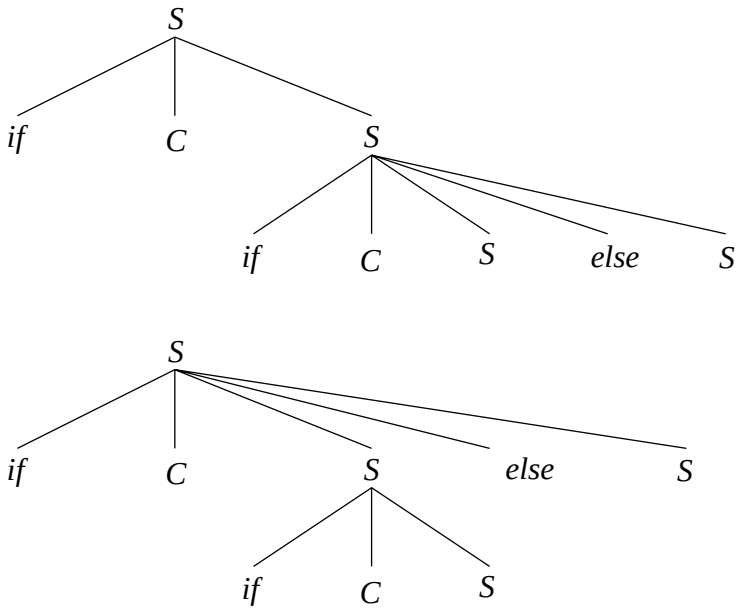
1. Primera derivación:

$$
\begin{array}{rcl}
S & \underset{1}{\Rightarrow} & \textbf{\textit{if}}\ C\ \underline{S} \\
 & \underset{2}{\Rightarrow} & \textbf{\textit{if}}\ C\ \underline{\textbf{\textit{if}}\ C\ S\ \textbf{\textit{else}}\ S}
\end{array}
$$

2. Segunda derivación:

$$
\begin{array}{rcl}
S & \underset{2}{\Rightarrow} & \textbf{\textit{if}}\ C\ \underline{S}\ \textbf{\textit{else}}\ S \\
 & \underset{1}{\Rightarrow} & \textbf{\textit{if}}\ C\ \underline{\textbf{\textit{if}}\ C\ S}\ \textbf{\textit{else}}\ S
\end{array}
$$

Problema del “else danzante”: árboles correspondientes a la primera y a la segunda derivación.



Puesto que el lenguaje C asosia el “else” al “if más cercano”, la derivación correcta es la primera.

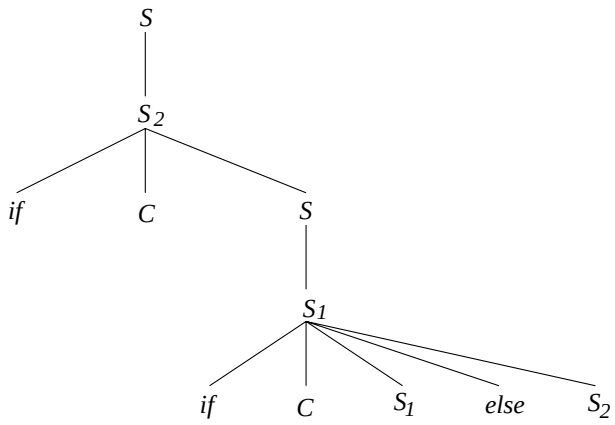
Afortunadamente, se puede reescribir la gramática para que tenga en cuenta este criterio semántico:

$$
\begin{array}{lllll}
 & & \ldots & & \\
(1) & S & \longrightarrow & S_1 & \\
(2) & S & \longrightarrow & S_2 & \\
(3) & S_1 & \longrightarrow & \textbf{\textit{if}}\ C\ S_1\ \textbf{\textit{else}}\ S_1 & \\
(4) & S_1 & \longrightarrow & I & \\
(5) & S_2 & \longrightarrow & \textbf{\textit{if}}\ C\ S & \\
(6) & S_2 & \longrightarrow & \textbf{\textit{if}}\ C\ S_1\ \textbf{\textit{else}}\ S_2 & \\
 & & \ldots & &
\end{array}
$$

donde S_1 genera la sentencia "if emparejada" mientras que S_2 genera la sentencia "if no emparejada".

La derivación de la cadena anterior es la siguiente:

$$
\begin{array}{lll}
S & \underset{2}{\Rightarrow} & S_2 \\
 & \underset{5}{\Rightarrow} & \textbf{\textit{if}}\ C\ \underline{S} \\
 & \underset{1}{\Rightarrow} & \textbf{\textit{if}}\ C\ \underline{S_1} \\
 & \underset{3}{\Rightarrow} & \textbf{\textit{if}}\ C\ \underline{\textbf{\textit{if}}\ C\ S_1\ \textbf{\textit{else}}\ S_2}
\end{array}
$$



Árbol sintáctico asociado a una derivación que asocia el "else al if más cercano".

Lenguajes intrínsecamente ambiguos

El siguiente lenguaje es un lenguaje de contexto libre intrínsecamente ambiguo porque todas las gramáticas de contexto libre que lo generan son ambiguas:

$$L = \{a^i b^i c^j | i, j \geq 1\} \cup \{a^i b^j c^j | i, j \geq 1\}$$

Este lenguaje es de contexto libre porque puede ser generado por una gramática de contexto libre como la siguiente:

$$\begin{array}{rcl} P & = & \{ \\ & (1) & S \longrightarrow AC \\ & (2) & S \longrightarrow BD \\ & (3) & A \longrightarrow aAb \\ & (4) & A \longrightarrow ab \\ & (5) & C \longrightarrow cC \\ & (6) & C \longrightarrow c \\ & (7) & B \longrightarrow aB \\ & (8) & B \longrightarrow a \\ & (9) & D \longrightarrow bDc \\ & (10) & D \longrightarrow bc \\ & & \} \end{array}$$

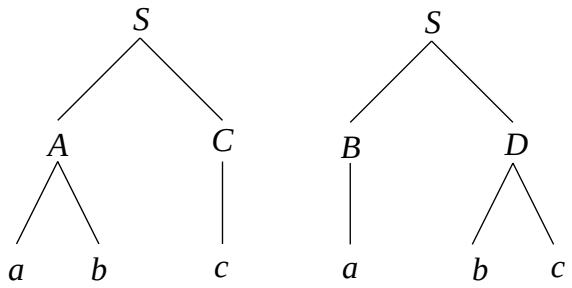
Esta gramática es ambigua porque puede derivar la cadena abc utilizando dos derivaciones por la izquierda diferentes:

1. Primera derivación:

$$\begin{aligned} S &\underset{1}{\Rightarrow} AC \\ &\underset{4}{\Rightarrow} abC \\ &\underset{6}{\Rightarrow} abc \end{aligned}$$

2. Segunda derivación:

$$\begin{aligned} S &\underset{2}{\Rightarrow} BD \\ &\underset{8}{\Rightarrow} aD \\ &\underset{10}{\Rightarrow} abc \end{aligned}$$



(a) (b)

Árboles de derivación diferentes de la cadena abc correspondientes a (a) la primera y a (b) la segunda derivación.

Vacuidad del lenguaje generado por una gramática de contexto libre

[1] ***inicio***
[2] $Viejo \leftarrow \emptyset$
[3] $Nuevo \leftarrow \{A | A \longrightarrow x \in P \wedge x \in V_T^*\}$
[4] ***mientras*** $(Nuevo \neq Viejo) y (S \notin Nuevo)$ ***hacer***
[5] $Viejo \leftarrow Nuevo$
[6] $Nuevo \leftarrow Viejo \cup \{A | A \longrightarrow \alpha \in P \wedge \alpha \in (V_T \cup Viejo)^*\}$
[7] ***fin mientras***
[8] ***si*** $S \in Nuevo$
[9] ***entonces escribir***(“La gramática genera un lenguaje no vacío”)
[10] ***si no escribir***(“La gramática genera un lenguaje vacío”)
[11] ***fin si***
[12] ***fin***

Sea G una gramática con el siguiente conjunto de reglas de producción:

$$
\begin{array}{rcl}
P & = & \{ \\
 & (1) & S \longrightarrow ABa \\
 & (2) & A \longrightarrow BDb \\
 & (3) & A \longrightarrow EB \\
 & (4) & B \longrightarrow CD \\
 & (5) & C \longrightarrow ab \\
 & (6) & C \longrightarrow aA \\
 & (7) & D \longrightarrow b \\
 & (8) & E \longrightarrow Sa \\
 & & \}
\end{array}
$$

La aplicación del algoritmo de comprobación de la vacuidad de una gramática de contexto libre consta de los siguientes pasos

Paso	***Viejo***	***Nuevo***
0	$\emptyset$	$\{C, D\}$
1	$\{C, D\}$	$\{B, C, D\}$
2	$\{B, C, D\}$	$\{A, B, C, D\}$
3	$\{A, B, C, D\}$	$\{S, A, B, C, D\}$

Como S pertenece a $Nuevo$, la gramática genera un lenguaje no vacío.

Supresión de símbolos no generadores

- Entrada: $G = (V_N, V_T, P, S)$ gramática de contexto libre.
- Salida: $G' = (V'_N, V_T, P', S)$ gramática de contexto libre sin símbolos no generadores.

[1] ***inicio***
[2] $Viejo \leftarrow \emptyset$
[3] $Nuevo \leftarrow \{A | A \in V_N \ \wedge \ \exists A \longrightarrow x \in P \ \wedge \ x \in V_T^*\}$
[4] ***mientras*** $(Nuevo \neq Viejo)$ ***hacer***
[5] $Viejo \leftarrow Nuevo$
[6] $Nuevo \leftarrow Viejo \cup \{A | A \in V_N \wedge \exists A \longrightarrow \alpha \in P \wedge \alpha \in (Viejo \cup V_T)^*\}$
[7] ***fin mientras***
[8] $V'_N \leftarrow Nuevo$
[9] $P' \leftarrow \{A \longrightarrow \alpha | A \longrightarrow \alpha \in P \wedge A \in V'_N \wedge \alpha \in (V'_N \cup V_T)^*\}$
[10] ***fin***

Las reglas de producción de la gramática G' se obtienen a partir de las reglas de producción de la gramática G que sólo tienen símbolos generadores.

Sea $G = (V_N, V_T, P, S)$ una gramática de contexto libre donde $V_N = \{S, A, B, C, D, E, F\}$, $V_T = \{a, b, c\}$ y

$$
\begin{aligned}
P \quad = \quad & \{ \\
& (1) \quad S \longrightarrow AB \\
& (2) \quad S \longrightarrow Ab \\
& (3) \quad A \longrightarrow aC \\
& (4) \quad B \longrightarrow bCa \\
& (5) \quad B \longrightarrow DbE \\
& (6) \quad C \longrightarrow b \\
& (7) \quad D \longrightarrow Fb \\
& (8) \quad E \longrightarrow ca \\
& (9) \quad F \longrightarrow aD \\
& \}
\end{aligned}
$$

Los pasos del algoritmo que elimina los símbolos no generadores de la gramática de contexto libre son:

Paso	***Viejo***	***Nuevo***
0	$\emptyset$	$\{C, E\}$
1	$\{C, E\}$	$\{A, B, C, E\}$
2	$\{A, B, C, E\}$	$\{S, A, B, C, E\}$
3	$\{S, A, B, C, E\}$	$\{S, A, B, C, E\}$

y el conjunto de producciones que se genera es:

$$
\begin{aligned}
P' \quad = \quad & \{ \\
& S \longrightarrow AB \mid Ab \\
& A \longrightarrow aC \\
& B \longrightarrow bCa \\
& C \longrightarrow b \\
& E \longrightarrow ca \\
& \}
\end{aligned}
$$

Se puede observar cómo se han suprimido los símbolos D y F porque no pueden generar ninguna cadena de símbolos terminales.

Supresión de símbolos no accesibles

- Entrada: $G' = (V'_N, V_T, P', S)$ gramática de contexto libre sin símbolos no generadores.
- Salida: $G'' = (V''_N, V'_T, P'', S)$ gramática de contexto libre sin símbolos no accesibles.

[1] ***inicio***
[2] $Viejo \leftarrow \{S\}$
[3] $Nuevo \leftarrow \{X | X \in (V'_N \cup V_T) \wedge \exists S \longrightarrow \alpha X \beta \in P' \wedge \alpha, \beta \in (V'_N \cup V_T)^*\}$
[4] ***mientras*** $(Nuevo \neq Viejo)$ ***hacer***
[5] $Viejo \leftarrow Nuevo$
[6] $Nuevo \leftarrow Viejo \cup \{X | \exists A \longrightarrow \alpha X \beta \in P' \ \wedge \ A \in Viejo \ \wedge$
[7] $X \in (V'_N \cup V_T) \ \wedge \ \alpha, \beta \in (V'_N \cup V_T)^*\}$
[8] ***fin mientras***
[9] $V''_N \leftarrow Nuevo \cap V'_N$
[10] $V'_T \leftarrow Nuevo \cap V_T$
[11] $P'' \leftarrow \{A \longrightarrow \alpha | A \longrightarrow \alpha \in P' \wedge A \in V''_N \wedge \alpha \in (V''_N \cup V'_T)^*\}$
[12] ***fin***

Las reglas de producción de la gramática G'' se obtienen a partir de las reglas de producción de la gramática G' que sólo tienen símbolos accesibles.

Sea $G' = (V'_N, V_T, P', S)$ la gramática sin símbolos no generadores obtenida en el ejemplo anterior.

La aplicación del algoritmo que suprime los símbolos no accesibles consta de los siguientes pasos

Paso	***Viejo***	***Nuevo***
0	$\{S\}$	$\{S, A, B, b\}$
1	$\{S, A, B, b\}$	$\{S, A, B, C, a, b\}$
2	$\{S, A, B, C, a, b\}$	$\{S, A, B, C, a, b\}$

y el conjunto de producciones que se genera es

$$
\begin{aligned}
P'' \;=\; & \{ \\
& S \longrightarrow AB \mid Ab \\
& A \longrightarrow aC \\
& B \longrightarrow bCa \\
& C \longrightarrow b \\
& \}
\end{aligned}
$$

Se han suprimido los símbolos E y c porque no son accesibles desde el símbolo inicial de la gramática.

Correcta eliminación de símbolos inútiles:

1. Eliminar los símbolos no generadores.
2. Eliminar los símbolos no accesibles.

Si estos algoritmos se aplican en orden inverso entonces no se garantiza que se eliminen todos los símbolos inútiles. Considérese de nuevo la gramática

$$
\begin{array}{rcl}
P & = & \{ \\
 & (1) & S \longrightarrow ABa \\
 & (2) & A \longrightarrow BDb \\
 & (3) & A \longrightarrow EB \\
 & (4) & B \longrightarrow CD \\
 & (5) & C \longrightarrow ab \\
 & (6) & C \longrightarrow aA \\
 & (7) & D \longrightarrow b \\
 & (8) & E \longrightarrow Sa \\
 & & \}
\end{array}
$$

Los pasos del algoritmo que elimina los símbolos no accesibles de la gramática de contexto libre son:

Paso	***Viejo***	***Nuevo***
0	$\{S\}$	$\{S, A, B, b\}$
1	$\{S, A, B, b\}$	$\{S, A, B, C, D, E, a, b\}$
2	$\{S, A, B, C, D, E, a, b\}$	$\{S, A, B, C, D, E, F, a, b, c\}$
3	$\{S, A, B, C, D, E, F, a, b\}$	$\{S, A, B, C, D, E, F, a, b, c\}$

Si se aplica primero el algoritmo que elimina los símbolos no accesibles, se puede observar que no se suprime ningún símbolo.

Paso	***Viejo***	***Nuevo***
0	$\emptyset$	$\{C, E\}$
1	$\{C, E\}$	$\{A, B, C, E\}$
2	$\{A, B, C, E\}$	$\{S, A, B, C, E\}$
3	$\{S, A, B, C, E\}$	$\{S, A, B, C, E\}$

Si, a continuación, se aplica el algoritmo que elimina los símbolos no generadores, el conjunto de producciones resultante contiene los símbolos E y c que no son accesibles.

$$
\begin{array}{rcl}
P & = & \{ \\
 & (1) & S \longrightarrow AB \\
 & (2) & S \longrightarrow Ab \\
 & (3) & A \longrightarrow aC \\
 & (4) & B \longrightarrow bCa \\
 & (6) & C \longrightarrow b \\
 & (8) & E \longrightarrow ca \\
 & & \}
\end{array}
$$

En resumen, para eliminar todos los símbolos inútiles de una gramática, se ha de aplicar primero el algoritmo que elimina los símbolos no generadores y, a continuación, el algoritmo que elimina los símbolos no accesibles.

Obtención de los símbolos anulables

- Entrada: $G = (V_N, V_T, P, S)$ gramática de contexto libre sin símbolos inútiles.
- Salida: Anulables, conjunto de símbolos anulables.

[1] ***inicio***
[2] $Viejo \leftarrow \emptyset$
[3] $Nuevo \leftarrow \{A | A \in V_N \ \wedge \ \exists A \longrightarrow \epsilon \in P\}$
[4] ***mientras*** $(Nuevo \neq Viejo)$ ***hacer***
[5] $Viejo \leftarrow Nuevo$
[6] $Nuevo \leftarrow Viejo \cup \{A | A \in V_N \ \wedge \ \exists A \longrightarrow \alpha \in P \ \wedge \ \alpha \in Viejo^*\}$
[7] ***fin mientras***
[8] $Anulables \leftarrow Nuevo$
[9] ***fin***

Sea $G = (V_N, V_T, P, S)$ una gramática de contexto libre donde

$$V_N = \{S, A, B, C, D, E\}, V_T = \{a, b\}$$

y

$$\begin{array}{rcl} P & = & \{ \\ & (1) & S \longrightarrow AD \\ & (2) & S \longrightarrow B \\ & (3) & A \longrightarrow CDE \\ & (4) & B \longrightarrow CE \\ & (5) & C \longrightarrow S \\ & (6) & C \longrightarrow a \\ & (7) & C \longrightarrow \epsilon \\ & (8) & D \longrightarrow A \\ & (9) & D \longrightarrow b \\ & (10) & E \longrightarrow S \\ & (11) & E \longrightarrow a \\ & (12) & E \longrightarrow \epsilon \\ & & \} \end{array}$$

Los pasos para obtener el conjunto de símbolos anulables de esta gramática son:

Paso	***Viejo***	***Nuevo***
0	$\emptyset$	$\{C, E\}$
1	$\{C, E\}$	$\{B, C, E\}$
2	$\{B, C, E\}$	$\{S, B, C, E\}$
3	$\{S, B, C, E\}$	$\{S, B, C, E\}$

Gramática sin ϵ

- Entrada: $G = (V_N, V_T, P, S)$ gramática de contexto libre sin símbolos inútiles y el conjunto de símbolos anulables de G.
- Salida: $G' = (V'_N, V_T, P', S')$ gramática sin ϵ.

[1] ***inicio***
[2] $P' \leftarrow \{A \longrightarrow \alpha | A \longrightarrow \alpha \in P \ \wedge \ \alpha \neq \epsilon \ \wedge$
[3] α no contiene ningún símbolo anulable$\}$
[4] ***para*** $(A \longrightarrow \alpha \in P) \wedge (\alpha$ contiene símbolos anulables$)$ ***hacer***
[5] ***si*** $\alpha = \alpha_0 B_1 \alpha_1 \ldots B_k \alpha_k$
[6] $\wedge \forall i \ (B_i \in Anulables)$
[7] $\wedge \forall j \ (\alpha_j$ no contiene ningún símbolo anulable$)$
[8] ***entonces***
[9] $P' \leftarrow P' \cup \{A \longrightarrow \alpha_0 X_1 \alpha_1 \ldots X_k \alpha_k |$
[10] $\forall i \ (X_i = B_i \vee X_i = \epsilon) \ \wedge \ \alpha_0 X_1 \alpha_1 \ldots X_k \alpha_k \neq \epsilon\}$
[11] ***fin si***
[12] ***fin para***
[13] ***si*** $S \in Anulables$
[14] ***entonces***
[15] $V'_N \leftarrow V_N \cup \{S'\}$
[16] $P' \leftarrow P' \cup \{S' \longrightarrow \epsilon, S' \longrightarrow S\}$
[17] ***fin si***
[18] ***fin***

Sea $G = (V_N, V_T, P, S)$ la gramática de contexto libre del ejemplo anterior cuyo conjunto de símbolos anulables es $\{S, B, C, E\}$. La gramática sin ϵ equivalente es $G' = (V'_N, V_T, P', S')$ donde las reglas de producción de P' se obtienen a partir de las reglas de producción de P:

Paso 0: Se incluyen en P' las reglas que no contienen en su parte derecha ningún símbolo anulables y que no son producciones ϵ:

$$\begin{array}{rcl} P' & = & \{ \\ & (1) & S \longrightarrow AD \\ & (6) & C \longrightarrow a \\ & (8) & D \longrightarrow A \\ & (9) & D \longrightarrow b \\ & (11) & E \longrightarrow a \\ & & \} \end{array}$$

Paso 1: La regla $S \longrightarrow B$ contiene el símbolo anulable B lo que provoca la generación de las dos reglas siguientes:

$$\begin{array}{rcl} S & \longrightarrow & B \\ S & \longrightarrow & \epsilon \end{array}$$

Sólo se añade a P' la primera regla porque la otra es una producción ϵ.

Paso 2: La regla $A \longrightarrow CDE$ contiene los símbolos anulable C y E lo que provoca la generación de las cuatro reglas siguientes:

$$\begin{array}{rcl} A & \longrightarrow & CDE \\ A & \longrightarrow & DE \\ A & \longrightarrow & CD \\ A & \longrightarrow & D \end{array}$$

que se añaden a P' porque ninguna es una producción ϵ.

Paso 3: La regla $B \longrightarrow CE$ contiene los símbolos anulable C y E lo que provoca la generación de las cuatro reglas siguientes:

$$\begin{array}{rcl} B & \longrightarrow & CE \\ B & \longrightarrow & E \\ B & \longrightarrow & C \\ B & \longrightarrow & \epsilon \end{array}$$

Sólo se añaden a P' las tres primeras reglas porque la otra es una producción ϵ.

Paso 4: La regla $C \longrightarrow S$ contiene el símbolo anulable S lo que provoca la generación de las dos reglas siguientes:

$$\begin{aligned} C &\longrightarrow S \\ C &\longrightarrow \epsilon \end{aligned}$$

Sólo se añade a P' la primera regla porque la otra es una producción ϵ.

Paso 5: La regla $E \longrightarrow S$ contiene el símbolo anulable S lo que provoca la generación de las dos reglas siguientes:

$$\begin{aligned} E &\longrightarrow S \\ E &\longrightarrow \epsilon \end{aligned}$$

Sólo se añade a P' la primera regla porque la otra es una producción ϵ.

Paso final: Como el símbolo S es anulable, se añaden a la gramática las siguientes producciones:

$$\begin{aligned} S' &\longrightarrow \epsilon \\ S' &\longrightarrow S \end{aligned}$$

Se han suprimido las producciones ϵ

$$\begin{aligned} (7) \quad & C \longrightarrow \epsilon \\ (12) \quad & E \longrightarrow \epsilon \end{aligned}$$

El conjunto final de reglas de producción es:

$$\begin{aligned} P' \;=\; \{ & \\ & S' \longrightarrow \epsilon \mid S \\ & S \longrightarrow AD \mid B \\ & A \longrightarrow CDE \mid DE \mid CD \mid D \\ & B \longrightarrow CE \mid E \mid C \\ & C \longrightarrow S \mid a \\ & D \longrightarrow A \mid b \\ & E \longrightarrow S \mid a \\ & \} \end{aligned}$$

Eliminación de reglas unitarias

La gramática con el conjunto de producciones

$$
\begin{array}{rcl}
P & = & \{ \\
 & (1) & S \longrightarrow \textbf{\textit{identificador}} \ = \ E \\
 & (2) & E \longrightarrow E \ + \ T \\
 & (3) & E \longrightarrow T \\
 & (4) & T \longrightarrow T \ * \ F \\
 & (5) & T \longrightarrow F \\
 & (6) & F \longrightarrow (\ E\ ) \\
 & (7) & F \longrightarrow \textbf{\textit{identificador}} \\
 & (8) & F \longrightarrow \textbf{\textit{número}} \\
 & & \}
\end{array}
$$

permite generar expresiones aritméticas como la que se muestra a continuación:

$$
\begin{array}{rcl}
S & \underset{1}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ E \\
 & \underset{2}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ E \ + \ T \\
 & \underset{3}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ T \ + \ T \\
 & \underset{5}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ F \ + \ T \\
 & \underset{7}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ \textbf{\textit{identificador}} \ + \ T \\
 & \underset{4}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ \textbf{\textit{identificador}} \ + \ F \\
 & \underset{7}{\Rightarrow} & \textbf{\textit{identificador}} \ = \ \textbf{\textit{identificador}} \ + \ \textbf{\textit{identificador}}
\end{array}
$$

Las derivaciones inmediatas de los pasos 3, 4 y 5

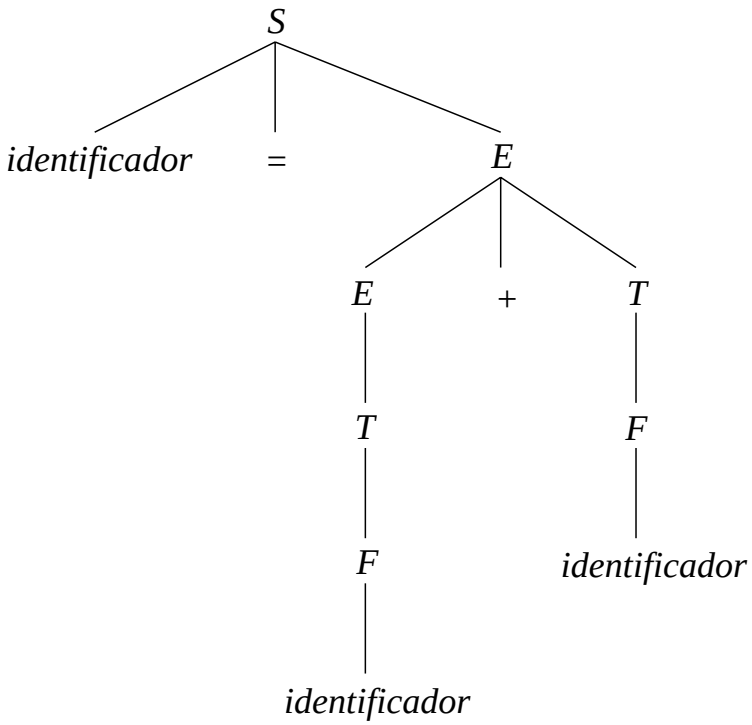
$$
\begin{array}{rcl}
E & \underset{3}{\Rightarrow} & T \\
 & \underset{5}{\Rightarrow} & F \\
 & \underset{7}{\Rightarrow} & \textbf{\textit{identificador}}
\end{array}
$$

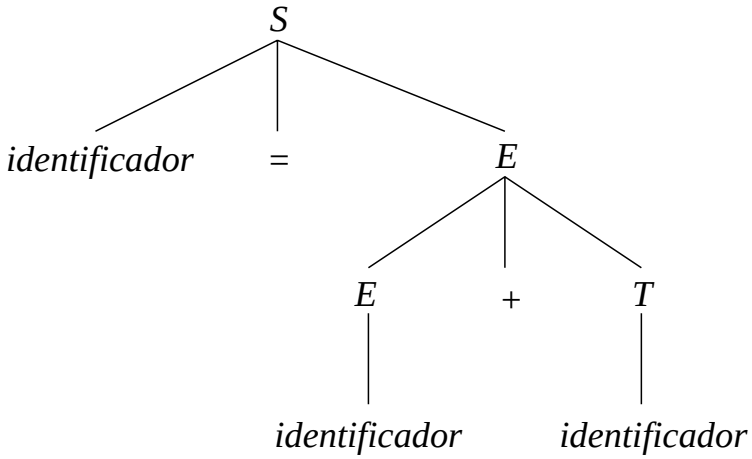
se podrían sustituir por la siguiente derivación inmediata

$$E \ \Rightarrow \ \textbf{\textit{identificador}}$$

donde $E \longrightarrow \textbf{\textit{identificador}}$ sería una nueva regla de producción de la gramática.

Eliminación de reglas unitarias: árbol de derivación de una gramática con reglas unitarias y árbol de derivación de una gramática equivalente sin reglas unitarias.





Símbolos no terminales accesibles mediante reglas unitarias

[1] ***inicio***
[2] $Viejo \leftarrow \emptyset$
[3] $Nuevo \leftarrow \{A\}$
[4] ***mientras*** $(Nuevo \neq Viejo)$ ***hacer***
[5] $Viejo \leftarrow Nuevo$
[6] $Nuevo \leftarrow Viejo \cup \{B | A \longrightarrow B \in P \wedge A \in Viejo\}$
[7] ***fin mientras***
[8] $N_A \leftarrow Nuevo$
[9] ***fin***

Considérese la gramática que genera expresiones aritméticas.

$$
\begin{array}{lll}
P & = & \{ \\
 & (1) & S \longrightarrow \textbf{\textit{identificador}} \ = \ E \\
 & (2) & E \longrightarrow E \ + \ T \\
 & (3) & E \longrightarrow T \\
 & (4) & T \longrightarrow T \ * \ F \\
 & (5) & T \longrightarrow F \\
 & (6) & F \longrightarrow (\ E\ ) \\
 & (7) & F \longrightarrow \textbf{\textit{identificador}} \\
 & (8) & F \longrightarrow \textbf{\textit{número}} \\
 & & \}
\end{array}
$$

El cálculo del conjunto N_E tiene los siguientes pasos:

Paso 0: Inicialmente el conjunto $Viejo$ está vacío. El símbolo no terminales E se incluye en $Nuevo$

Paso 1: Como los conjuntos $Viejo$ y $Nuevo$ son diferentes, se asigna a $Viejo$ el valor de $Nuevo$ y se añaden a $Nuevo$ el símbolo T debido a la existencia de la producción unitaria $E \longrightarrow T$.

Paso 2: Como la condición de salida del bucle no se cumple, se asigna $Viejo$ el valor de $Nuevo$ y se incluye en $Nuevo$ el símbolo no terminal F debido a la existencia de la producción unitaria $T \longrightarrow F$.

Paso 3: Puesto que la condición de salida todavía no se cumple, se asigna el valor de $Nuevo$ a $Viejo$.

Paso final: El bucle termina porque los conjuntos $Viejo$ y $Nuevo$ son iguales

Paso	***Viejo***	***Nuevo***
0	$\emptyset$	$\{E\}$
1	$\{E\}$	$\{E,T\}$
2	$\{E,T\}$	$\{E,T,F\}$
3	$\{E,T,F\}$	$\{E,T,F\}$

Por tanto, $N_E = \{E,T,F\}$. Análogamente se pueden calcular los conjuntos $N_S = \{S\}, N_T = \{T,F\}$ y $N_F = \{F\}$.

Eliminación de reglas unitarias

- Entrada: $G = (V_N, V_T, P, S)$ gramática de contexto libre y los conjuntos $N_A \ \forall A \in V_N$.
- Salida: $G' = (V'_N, V_T, P', S)$ gramática sin reglas unitarias.

[1] ***inicio***
[2] $P' \leftarrow \emptyset$
[3] ***para*** cada $A \in V_N$ ***hacer***
[4] ***para*** cada $B \in N_A$ ***hacer***
[5] ***si*** $B \longrightarrow \alpha \in P$ no es una regla unitaria
[6] ***entonces*** $P' \leftarrow P' \cup \{A \longrightarrow \alpha\}$
[7] ***fin si***
[8] ***fin para***
[9] ***fin para***
[10] $V'_N \leftarrow \{A | A \in V_N \wedge \exists A \longrightarrow \alpha \in P'\}$
[11] ***fin***

Considérese la gramática con reglas unitarias

$$
\begin{array}{lll}
P & = & \{ \\
 & (1) & S \longrightarrow \textbf{\textit{identificador}} \ = \ E \\
 & (2) & E \longrightarrow E \ + \ T \\
 & (3) & E \longrightarrow T \\
 & (4) & T \longrightarrow T \ * \ F \\
 & (5) & T \longrightarrow F \\
 & (6) & F \longrightarrow (\ E\ ) \\
 & (7) & F \longrightarrow \textbf{\textit{identificador}} \\
 & (8) & F \longrightarrow \textbf{\textit{número}} \\
 & & \}
\end{array}
$$

Para eliminar las reglas unitarias de esta gramática, se necesitan los conjuntos N_S, N_E, N_T y N_F calculados en el ejemplo anterior. Los pasos para generar la nueva gramática sin reglas unitarias son:

Paso 1: Como el conjunto $N_S = \{S\}$, se añaden a P' todas las producciones de S, ya que ninguna es una regla unitaria.

$$S \longrightarrow \textbf{\textit{identificador}} \ = \ E$$

Paso 2: Las alternativas de las producciones no unitarias de los símbolos no terminales del conjunto $N_E = \{E, T, F\}$ se convierten en alternativas de las producciones del símbolo E.

$$
\begin{array}{lll}
E & \longrightarrow & E \ + \ T \\
E & \longrightarrow & T \ * \ F \\
E & \longrightarrow & (\ E\ ) \\
E & \longrightarrow & \textbf{\textit{identificador}} \\
E & \longrightarrow & \textbf{\textit{número}}
\end{array}
$$

Paso 3: Las alternativas de las producciones no unitarias de los símbolos no terminales del conjunto $N_T = \{T, F\}$ se convierten en alternativas de las producciones del símbolo T.

$$
\begin{array}{lll}
T & \longrightarrow & T \ * \ F \\
T & \longrightarrow & (\ E\ ) \\
T & \longrightarrow & \textbf{\textit{identificador}} \\
T & \longrightarrow & \textbf{\textit{número}}
\end{array}
$$

Paso 4: Como el conjunto $N_F = \{F\}$, se añaden a P' todas las producciones de F, ya que ninguna es una regla unitaria.

$$
\begin{aligned}
F &\longrightarrow (\ E \ ) \\
F &\longrightarrow \textbf{\textit{identificador}} \\
F &\longrightarrow \textbf{\textit{número}}
\end{aligned}
$$

El nuevo conjunto de reglas de producción que se ha generado es:

$$
\begin{aligned}
P' \ = \ \{ & \\
& S \longrightarrow \textbf{\textit{identificador}} \ = \ E \\
& E \longrightarrow E \ + \ T \mid T \ * \ F \mid (\ E \ ) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \\
& T \longrightarrow T \ * \ F \mid (\ E \ ) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \\
& F \longrightarrow (\ E \ ) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \\
& \}
\end{aligned}
$$

Puede darse el caso en el que al eliminar las reglas unitarias surjan símbolos inútiles, como se puede observar en el siguiente ejemplo.

Considérese el conjunto de producciones P' de una gramática sin ϵ:

$$\begin{aligned} P \;=\; \{ & \\ & S' \longrightarrow \epsilon \mid S \\ & S \longrightarrow AD \mid B \\ & A \longrightarrow CDE \mid DE \mid CD \mid D \\ & B \longrightarrow CE \mid E \mid C \\ & C \longrightarrow S \mid a \\ & D \longrightarrow A \mid b \\ & E \longrightarrow S \mid c \\ & \} \end{aligned}$$

Los conjuntos de símbolos no terminales que son accesibles mediante producciones unitarias son:

$$\begin{aligned} N_{S'} &= \{S', S, B, C, E\} \\ N_S &= \{S, B, C, E\} \\ N_A &= \{A, D\} \\ N_B &= \{S, B, C, E\} \\ N_C &= \{S, B, C, E\} \\ N_D &= \{A, D\} \\ N_E &= \{S, B, C, E\} \end{aligned}$$

La aplicación del algoritmo que elimina las reglas unitarias permite generar el siguiente conjunto de producciones:

$$\begin{aligned} P' \;=\; \{ & \\ & S' \longrightarrow \epsilon \mid AD \mid CE \mid a \mid c \\ & S \longrightarrow \; AD \mid CE \mid a \mid c \\ & A \longrightarrow CDE \mid DE \mid CD \mid b \\ & B \longrightarrow CE \mid AD \mid a \mid c \\ & C \longrightarrow AD \mid CE \mid a \mid c \\ & D \longrightarrow CDE \mid DE \mid CD \mid b \\ & E \longrightarrow AD \mid CE \mid a \mid c \\ & \} \end{aligned}$$

Como se puede observar, los símbolos S y B no son accesibles, porque no aparecen en la parte derecha de ninguna regla de producción. Por tanto, estos dos símbolos son inútiles y se pueden omitir.

Ejemplos de gramáticas recursivas por la izquierda:

- Parte izquierda de una sentencia de asignación múltiple en el lenguaje C: dado el siguiente conjunto de producciones

$$
\begin{array}{rcl}
P & = & \{ \\
 & (1) & S \longrightarrow LE \\
 & (2) & L \longrightarrow L\ \textbf{\textit{identificador}}\ = \\
 & (3) & L \longrightarrow \textbf{\textit{identificador}}\ = \\
 & (4) & E \longrightarrow E\ +\ T \\
 & & \cdots \\
 & & \}
\end{array}
$$

  se puede generar la siguiente derivación recursiva

$$
\begin{array}{rcl}
S & \underset{1}{\Rightarrow} & LE \\
 & \underset{2}{\Rightarrow} & L\ \textbf{\textit{identificador}}\ =\ E \\
 & \underset{2}{\Rightarrow} & L\ \textbf{\textit{identificador}}\ =\ \textbf{\textit{identificador}}\ =\ E \\
 & \underset{3}{\Rightarrow} & \textbf{\textit{identificador}}\ =\ \textbf{\textit{identificador}}\ =\ \textbf{\textit{identificador}}\ =\ E
\end{array}
$$

- Lista de parámetros de un procedimiento o función:

$$
\begin{array}{rcl}
P & = & \{ \\
 & & S \longrightarrow \textbf{\textit{identificador}}\ (L) \\
 & & L \longrightarrow L, \textbf{\textit{identificador}} \\
 & & L \longrightarrow \textbf{\textit{identificador}} \\
 & & \cdots \\
 & & \}
\end{array}
$$

- Componente de un *array* de varias dimensiones:

$$
\begin{array}{rcl}
P & = & \{ \\
 & & S \longrightarrow \textbf{\textit{identificador}}\ D \\
 & & D \longrightarrow D[\textbf{\textit{número}}] \\
 & & D \longrightarrow [\textbf{\textit{número}}] \\
 & & \cdots \\
 & & \}
\end{array}
$$

Eliminación de la recursividad inmediata por la izquierda

- Entrada: $G = (V_N, V_T, P, S)$ gramática de contexto libre con reglas recursivas por la izquierda.
- Salida: $G' = (V'_N, V_T, P', S)$ gramática sin reglas de producción recursivas por la izquierda.

[1] ***inicio***
[2] $P' \leftarrow \emptyset$
[3] ***para*** cada $A \in V_N$ ***hacer***
[4] ***si*** A no tiene producciones recursivas
[5] ***entonces*** se añaden a P' las producciones de A
[6] ***si no***
[7] ***si*** $A \longrightarrow A\alpha_1|A\alpha_2|\cdots|A\alpha_p|\beta_1|\beta_2|\cdots|\beta_q \in P$
[8] donde $\alpha_i \neq \epsilon \ \forall i \in \{1, 2, \ldots, p\}$
[9] y β_j no empieza por $A \ \forall j \in \{1, 2, \ldots, q\}$
[10] ***entonces***
[11] se añaden a P' las producciones
[12] $A \longrightarrow \beta|\beta_j A' \quad \forall i \in \{1, 2, \ldots, p\}$
[13] $A' \longrightarrow \alpha_i|\alpha_i A' \quad \forall j \in \{1, 2, \ldots, q\}$
[14] donde A' es un nuevo símbolo no terminal
[15] ***fin si***
[16] ***fin si***
[17] ***fin para***
[18] ***fin***

Considérese el conjunto de producciones de la siguiente gramática:

$$
\begin{aligned}
P \;=\; & \{ \\
& S \longrightarrow \textbf{\textit{identificador}} \;=\; E \\
& E \longrightarrow E \;+\; T \mid T \;*\; F \mid (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \\
& T \longrightarrow T \;*\; F \mid (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \\
& F \longrightarrow (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \\
& \}
\end{aligned}
$$

Paso 1: La producción de S no es recursiva por la izquierda y, por tanto, se añade a P'.

Paso 2: El símbolo E tiene una regla recursiva por la izquierda y cuatro más que no lo son. Las nuevas reglas que se añaden a P' son:

$$
\begin{aligned}
E \;\longrightarrow\; & T \;*\; F \mid (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \mid \\
& T \;*\; F \;E' \mid (\; E \;)\; E' \mid \textbf{\textit{identificador}}\; E' \mid \textbf{\textit{número}}\; E' \\
E' \;\longrightarrow\; & +\; T \mid +\; T \;E'
\end{aligned}
$$

Paso 3: El símbolo T tiene una regla recursiva por la izquierda y tres más que no lo son. Las nuevas reglas que se añaden a P' son:

$$
\begin{aligned}
T \;\longrightarrow\; & (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \mid \\
& (\; E \;)\; T' \mid \textbf{\textit{identificador}}\; T' \mid \textbf{\textit{número}}\; T' \\
T' \;\longrightarrow\; & *\; F \mid *\; F \;T'
\end{aligned}
$$

Paso 4: El símbolo F no posee reglas recursivass. Por tanto, todas ellas se añaden a P'

El conjunto de producciones resultante es:

$$
\begin{aligned}
P' \;=\; & \{ \\
& S \longrightarrow \textbf{\textit{identificador}} \;=\; E \\
& E \longrightarrow T \;*\; F \mid (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \mid \\
& \quad T \;*\; F \;E' \mid (\; E \;)\; E' \mid \textbf{\textit{identificador}}\; E' \mid \textbf{\textit{número}}\; E' \\
& E' \longrightarrow +\; T \mid +\; T \;E' \\
& T \longrightarrow (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \mid \\
& \quad (\; E \;)\; T' \mid \textbf{\textit{identificador}}\; T' \mid \textbf{\textit{número}}\; T' \\
& T' \longrightarrow *\; F \mid *\; F \;T' \\
& F \longrightarrow (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \\
& \}
\end{aligned}
$$

Para comprobar que las dos gramáticas son equivalentes, considérense las siguientes derivaciones. La primera derivación se ha generado utilizando la gramática original con producciones recursivas por la izquierda:

$$
\begin{aligned}
S &\Rightarrow \textbf{\textit{identificador}} = E \\
&\Rightarrow \textbf{\textit{identificador}} = E + T \\
&\Rightarrow \textbf{\textit{identificador}} = \textbf{\textit{identificador}} + T \\
&\Rightarrow \textbf{\textit{identificador}} = \textbf{\textit{identificador}} + \textbf{\textit{identificador}}
\end{aligned}
$$

La segunda derivación genera la misma cadena que la anterior pero utiliza las producciones de la nuevas gramática sin recursividad por la izquierda:

$$
\begin{aligned}
S &\Rightarrow \textbf{\textit{identificador}} = E \\
&\Rightarrow \textbf{\textit{identificador}} = \textbf{\textit{identificador}}\ E' \\
&\Rightarrow \textbf{\textit{identificador}} = \textbf{\textit{identificador}} + T \\
&\Rightarrow \textbf{\textit{identificador}} = \textbf{\textit{identificador}} + \textbf{\textit{identificador}}
\end{aligned}
$$

Eliminación de la recursividad general por la izquierda

- Entrada: $G = (V_N, V_T, P, S)$ gramática de contexto libre propia, es decir, sin ciclos, sin producciones ϵ ni símbolos inútiles.
- Salida: $G' = (V'_N, V_T, P', S)$ gramática sin recursividad por la izquierda.

[1] ***inicio***
[2] $P' \leftarrow \emptyset$
[3] Ordénense los símbolos no terminales de la gramática: $\{A_1, A_2, \ldots, A_n\}$
[4] ***para*** i ***de*** 1 ***a*** n ***hacer***
[5] ***para*** j ***de*** 1 ***a*** $i-1$ ***hacer***
[6] ***si*** $A_i \longrightarrow A_j\ \gamma\ \in P$
[7] ***entonces***
[8] Añadir a P' las producciones
[9] $A_i \longrightarrow \delta_1\ \gamma \mid \cdots \mid \delta_k\ \gamma$
[10] donde
[11] $A_j \longrightarrow \delta_1 \mid \cdots \mid \delta_k$
[12] son las producciones actuales de A_j
[13] ***fin si***
[14] ***fin para***
[15] Eliminar la recursividad inmediata por la izquierda
[16] de las producciones de A_i.
[17] ***fin para***
[18] ***fin***

Considérese la siguiente gramática recursiva por la izquierda:

$$
\begin{array}{rcl}
P & = & \{ \\
 & (1) & S \longrightarrow A\ B \\
 & (2) & S \longrightarrow c \\
 & (3) & A \longrightarrow B\ b \\
 & (4) & A \longrightarrow S\ d \\
 & (5) & A \longrightarrow a \\
 & (6) & B \longrightarrow S\ b \\
 & (7) & B \longrightarrow A\ a \\
 & & \}
\end{array}
$$

Ordenamiento de los símbolos no terminales: $\{S, A, B\}$

Paso exterior 1: Producciones de S

> ***Paso interior 1:*** S no tiene ninguna producción que comience por un símbolo con un número de orden inferior al suyo.
>
> ***Eliminación de la recusividad inmediata:*** S no tiene recursividad inmediata por la izquierda.

Paso exterior 2: Producciones de A

> ***Paso interior 1:*** Sustitución de las producciones de A que comienzan por S: la regla número (4) $A \longrightarrow S\ d$ se sustituye por las reglas
>
> $$A \longrightarrow A\ B\ d \mid c\ d$$
>
> quedando las reglas de A de la siguiente forma:
>
> $$A \longrightarrow A\ B\ d \mid B\ b \mid c\ d \mid a$$
>
> ***Eliminación de la recusividad inmediata:*** se sustituyen las producciones de A por las siguientes producciones:
>
> $$
> \begin{array}{rcl}
> A & \longrightarrow & B\ b \mid c\ d \mid a \mid \\
> & & B\ b\ A' \mid c\ d\ A' \mid a\ A' \\
> A' & \longrightarrow & B\ d \mid B\ d\ A'
> \end{array}
> $$

Paso exterior 3: Producciones de B

> ***Paso interior 1:*** Sustitución de las producciones de B que comienzan por S: la regla número (6) $B \longrightarrow S\ b$ se sustituye por las reglas
>
> $$B \longrightarrow A\ B\ b \mid c\ b$$
>
> quedando las reglas de B de la siguiente forma:
>
> $$B \longrightarrow A\ B\ b \mid c\ b \mid A\ a$$

Paso interior 2: Sustitución de las producciones de B que comienzan por A: las regla $B \longrightarrow A\ B\ b$ se sustituye por las reglas

$$\begin{array}{rcl} B & \longrightarrow & B\ b\ B\ b \mid c\ d\ B\ b \mid a\ B\ b \mid \\ & & B\ b\ A'\ B\ b \mid c\ d\ A'\ B\ b \mid a\ A'\ B\ b \end{array}$$

y la regla $B \longrightarrow A\ a$ se sustituye por las reglas

$$\begin{array}{rcl} B & \longrightarrow & B\ b\ a \mid c\ d\ a \mid a\ a \mid \\ & & B\ b\ A'\ a \mid c\ d\ A'\ a \mid a\ A'\ a \end{array}$$

quedando las reglas de B de la siguiente forma:

$$\begin{array}{rcl} B & \longrightarrow & B\ b\ B\ b \mid B\ b\ A'\ B\ b \mid B\ b\ a \mid B\ b\ A'\ a \mid \\ & & c\ d\ B\ b \mid a\ B\ b \mid c\ d\ A'\ B\ b \mid a\ A'\ B\ b \mid \\ & & b\ a \mid c\ d\ a \mid a\ a \mid c\ d\ A'\ a \mid a\ A'\ a \mid \\ & & c\ b \end{array}$$

Eliminación de la recusividad inmediata: se sustituyen las producciones de B por las siguientes producciones:

$$\begin{array}{rcl} B & \longrightarrow & c\ d\ B\ b \mid a\ B\ b \mid c\ d\ A'\ B\ b \mid a\ A'\ B\ b \mid \\ & & b\ a \mid c\ d\ a \mid a\ a \mid c\ d\ A'\ a \mid a\ A'\ a \mid \\ & & c\ b \mid \\ & & c\ d\ B\ b\ B' \mid a\ B\ b\ B' \mid c\ d\ A'\ B\ b\ B' \mid a\ A'\ B\ b\ B' \mid \\ & & b\ a\ B' \mid c\ d\ a\ B' \mid a\ a\ B' \mid c\ d\ A'\ a\ B' \mid a\ A'\ a\ B' \mid \\ & & c\ b\ B' \\ B' & \longrightarrow & b\ B\ b \mid b\ A'\ B\ b \mid b\ a \mid b\ A'\ a \mid \\ & & b\ B\ b\ B' \mid b\ A'\ B\ b\ B' \mid b\ a\ B' \mid b\ A'\ a\ B' \end{array}$$

Considérense las siguientes producciones:

$$\begin{aligned} S \longrightarrow\ & \textbf{\textit{si}}\ E\ \textbf{\textit{entonces}}\ S\ \textbf{\textit{si no}}\ S\ \textbf{\textit{fin si}}\ \mid \\ & \textbf{\textit{si}}\ E\ \textbf{\textit{entonces}}\ S\ \textbf{\textit{fin si}} \end{aligned}$$

Al leer el componente léxico ***si*** no se sabe aún qué producción elegir para expandir la sentencia S. Sin embargo, se puede posponer esta decición si se utilizan las siguientes reglas de producción.

$$\begin{aligned} S &\longrightarrow \textbf{\textit{si}}\ E\ \textbf{\textit{entonces}}\ S\ S' \\ S' &\longrightarrow \textbf{\textit{si no}}\ S\ \textbf{\textit{fin si}}\ \mid\ \textbf{\textit{fin si}} \end{aligned}$$

En general, si $A \longrightarrow \alpha\ \beta_1 \mid \alpha\ \beta_2$ son dos producciones de A y la entrada a analizar comienza por una cadena no vacía derivada a partir de α, no se sabe si expandir A a $\alpha\ \beta_1$ o a $\alpha\ \beta_2$. Para ello, las producciones se transforman en las siguientes producciones:

$$\begin{aligned} A &\longrightarrow \alpha\ A' \\ A' &\longrightarrow \beta_1 \mid \beta_2 \end{aligned}$$

Factorización por la izquierda

- Entrada: $G = (V_N, V_T, P, S)$ gramática de contexto libre propia.
- Salida: $G' = (V'_N, V_T, P', S)$ gramática factorizada por la izquierda.

[1] ***inicio***
[2] ***para*** cada símbolo no terminal A ***hacer***
[3] ***mientras*** A tenga dos producciones actuales con el mismo prefijo
[4] ***hacer***
[5] ***si*** α es el prefijo más largo de dos o más
[6] alternativas de A y $\alpha \neq \epsilon$
[7] ***entonces***
[8] Sustituir todas las producciones
[9] $A \longrightarrow \alpha\ \beta_1 \mid \cdots \mid \alpha\ \beta_p \mid \gamma_1 \mid \cdots \mid \gamma_q$
[10] donde γ_i no empieza por $\alpha\ \forall i \in \{1, 2, \ldots, q\}$
[11] por las producciones
[12] $A \longrightarrow \alpha\ A' \mid \gamma_1 \mid \cdots \mid \gamma_q$
[13] $A' \longrightarrow \beta_1 \mid \cdots \mid \beta_p$
[14] ***fin si***
[15] ***fin mientras***
[16] ***fin para***
[17] ***fin***

Considérese el siguiente conjunto de producciones de una gramática de contexto libre:

$$
\begin{aligned}
P \;=\; & \{ \\
& S \longrightarrow A\ B\ c \mid A\ B\ d\ e \mid A\ B\ d\ f \mid A\ B\ S \\
& A \longrightarrow a \\
& B \longrightarrow b \\
& \}
\end{aligned}
$$

Paso 1: $\alpha_1 = ABd$ es el prefijo más largo de dos producciones de S. Por tanto, las reglas de S se sustituyen por las siguientes:

$$
\begin{aligned}
S &\longrightarrow A\ B\ d\ S' \mid A\ B\ c \mid A\ B\ S \\
S' &\longrightarrow e \mid f
\end{aligned}
$$

La cadena $\alpha_2 = AB$ es ahora el prefijo más largo de tres producciones actuales de S. Por tanto, las reglas de S se sustituyen por las siguientes:

$$
\begin{aligned}
S &\longrightarrow A\ B\ S'' \\
S'' &\longrightarrow d\ S' \mid c \mid S \\
S' &\longrightarrow e \mid f
\end{aligned}
$$

Pasos 2 y 3: Las producciones de A y B no requieren factorización.

Eliminación de la recursividad inmediata por la izquierda y factorización por la izquierda

- Entrada: $G = (V_N, V_T, P, S)$ gramática de contexto libre con reglas recursivas por la izquierda.
- Salida: $G' = (V'_N, V_T, P', S)$ gramática sin reglas de producción recursivas por la izquierda y factorizada por la izquierda.

[1] ***inicio***
[2] $P' \leftarrow \emptyset$
[3] ***para*** cada $A \in V_N$ ***hacer***
[4] ***si*** A no tiene producciones recursivas
[5] ***entonces*** se añaden a P' las producciones de A
[6] ***si no***
[7] ***si*** $A \longrightarrow A\alpha_1|A\alpha_2|\cdots|A\alpha_p|\beta_1|\beta_2|\cdots|\beta_q \in P$
[8] donde $\alpha_i \neq \epsilon \ \forall i \in \{1, 2, \ldots, p\}$
[9] y β_j no empieza por $A \ \forall j \in \{1, 2, \ldots, q\}$
[10] ***entonces***
[11] se añaden a P' las producciones
[12] $A \longrightarrow \beta_j A' \quad \forall i \in \{1, 2, \ldots, p\}$
[13] $A' \longrightarrow \alpha_i A'|\epsilon \quad \forall j \in \{1, 2, \ldots, q\}$
[14] donde A' es un nuevo símbolo no terminal
[15] ***fin si***
[16] ***fin si***
[17] ***fin para***
[18] ***fin***

Sea la gramática sin reglas unitarias que genera las expresiones aritméticas

$$
\begin{aligned}
P \;=\; & \{ \\
& S \longrightarrow \textbf{\textit{identificador}} \;=\; E \\
& E \longrightarrow E \;+\; T \mid T \;*\; F \mid (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \\
& T \longrightarrow T \;*\; F \mid (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \\
& F \longrightarrow (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \\
& \}
\end{aligned}
$$

La aplicación del algoritmo que factoriza y elimina la recurisividad inmediata por la izquierda genera la la siguiente gramática:

$$
\begin{aligned}
P' \;=\; & \{ \\
& S \longrightarrow \textbf{\textit{identificador}} \;=\; E \\
& E \longrightarrow T \;*\; F \;E' \mid (\; E \;)\; E' \mid \textbf{\textit{identificador}}\; E' \mid \textbf{\textit{número}}\; E' \\
& E' \longrightarrow \;+\; T \;E' \mid \epsilon \\
& T \longrightarrow \;(\; E \;)\; T' \mid \textbf{\textit{identificador}}\; T' \mid \textbf{\textit{número}}\; T' \\
& T' \longrightarrow \;*\; F \;T' \mid \epsilon \\
& F \longrightarrow (\; E \;) \mid \textbf{\textit{identificador}} \mid \textbf{\textit{número}} \\
& \}
\end{aligned}
$$

Forma normal de Chomsky

Se dice que una gramática de contexto libre está en la ***forma normal de Chomsky (F.N.C.)*** si sus reglas son de una de estas dos formas:

$$\begin{aligned} A &\longrightarrow B\ C \\ A &\longrightarrow a \end{aligned}$$

donde $A, B, C \in V_N$ y $a \in V_T$

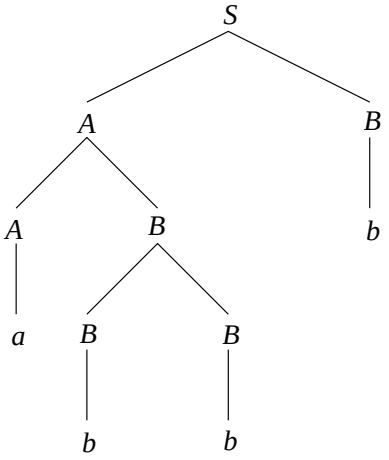
Sea una gramática en la forma normal de Chomsky con el siguiente conjunto de producciones:

$$\begin{aligned} P = \{ & \\ (1)\ & S \longrightarrow A\ B \\ (2)\ & A \longrightarrow A\ B \\ (3)\ & A \longrightarrow a \\ (4)\ & B \longrightarrow B\ B \\ (5)\ & B \longrightarrow b \\ & \} \end{aligned}$$

Derivación generada por una gramática en la forma normal de Chomsky:

$$\begin{aligned} S &\underset{1}{\Rightarrow} A\ B \\ &\underset{2}{\Rightarrow} A\ B\ B \\ &\underset{3}{\Rightarrow} a\ B\ B \\ &\underset{4}{\Rightarrow} a\ B\ B\ B \\ &\underset{5}{\Rightarrow} a\ b\ B\ B \\ &\underset{5}{\Rightarrow} a\ b\ b\ B \\ &\underset{6}{\Rightarrow} a\ b\ b\ b \end{aligned}$$

Árbol binario correspondiente a la derivación anterior:



Si $G = (V_N, V_T, P, S)$ es una gramática de contexto libre propia entonces se va a generar una gramática en la forma normal de Chomsky mediante la aplicación de los siguientes pasos:

1. Generación de una gramática $G_1 = (V_{N_1}, V_T, P_1, S)$ donde las reglas de P_1 son de la forma

$$\begin{aligned} A &\longrightarrow B_1\ B_2\ \cdots\ B_k \quad \text{donde } k \geq 2 \\ A &\longrightarrow a \end{aligned}$$

   verificándose que $L(G) = L(G_1)$.

2. A partir de la gramática G_1 obtenida en el paso anterior, se genera otra gramática G_2 que estará en la forma normal de Chomsky y que será equivalente a G, es decir, $L(G) = L(G_2)$.

Paso 1: Sea $A \longrightarrow X_1\ X_2\ \cdots\ X_k \in P$:

1. Si $k = 1$ entonces la regla es simplemente $A \longrightarrow X_1$, donde $X_1 \in V_T$, porque la gramática no tiene reglas unitarias al ser una gramática propia. En este caso, la regla $A \longrightarrow X_1$ se añade a P_1.
2. Si $k \geq 2$ entonces se añade a P_1 la regla $A \longrightarrow B_1\ B_2\ \cdots\ B_k$ donde
   - $B_i = X_i$ si $X_i \in V_N$
   - o B_i es un nuevo símbolo no terminal si $X_i = a_i \in V_T$, en cuyo caso, también se añade a P_1 la regla $B_i \longrightarrow X_i$.

Se verifica que $L(G_1) = L(G) - \{\epsilon\}$.

Paso 2: Para generar las reglas de P_2 se han analizar las reglas de P_1:

1. Si la regla de P_1 es de la forma $A \longrightarrow a$ entonces se añade a P_2.
2. Si $A \longrightarrow B_1\ B_2\ \cdots\ B_k \in P_1$ entonces se pueden presentar dos casos:
   (a) Si $k = 2$ entonces la regla está en la forma normal de Chomsky y se añade a P_2.
   (b) Si $k \geq 3$ entonces se añaden a P_2 el siguiente conjunto de reglas:

$$\begin{aligned} A &\longrightarrow B_1\ C_1 \\ C_1 &\longrightarrow B_2\ C_2 \\ &\cdots \\ C_{k-1} &\longrightarrow B_{k-2}\ C_{k-2} \\ C_{k-2} &\longrightarrow B_{k-1}\ B_k \end{aligned}$$

   Todas estas reglas están en la forma normal de Chomsky.

Se verifica que $L(G_2) = L(G_1) = L(G) - \{\epsilon\}$. G_2 es la gramática G' del enunciado del teorema.

Sea el siguiente conjunto de producciones de una gramática de contexto libre propia

$$
\begin{aligned}
P \;=\; & \{ \\
& S \longrightarrow a\ A\ B \\
& A \longrightarrow a\ B\ b \\
& A \longrightarrow a \\
& B \longrightarrow b\ b \\
& \}
\end{aligned}
$$

Paso 1: Se genera el siguiente conjunto de producciones:

$$
\begin{aligned}
P_1 \;=\; & \{ \\
& S \longrightarrow B_1\ A\ B \\
& A \longrightarrow B_1\ B\ B_2 \\
& A \longrightarrow a \\
& B \longrightarrow B_2\ B_2 \\
& B_1 \longrightarrow a \\
& B_2 \longrightarrow b \\
& \}
\end{aligned}
$$

Paso 2:

$$
\begin{aligned}
P_2 \;=\; & \{ \\
& S \longrightarrow B_1\ C_1 \\
& C_1 \longrightarrow A\ B \\
& A \longrightarrow B_1\ C_2 \\
& C_2 \longrightarrow B\ B_2 \\
& A \longrightarrow a \\
& B \longrightarrow B_2\ B_2 \\
& B_1 \longrightarrow a \\
& B_2 \longrightarrow b \\
& \}
\end{aligned}
$$

Forma normal de Greibach

Se dice que una gramática de contexto libre está en la ***forma normal de Greibach (F.N.G.)*** si sus reglas son de la forma:

$$A \longrightarrow a\ \alpha$$

donde $A \in V_N$, $a \in V_T$ y $\alpha \in V_N^*$

Si $G = (V_N, V_T, P, S)$ es una gramática de contexto libre que está en la forma normal de Chomsky, se va a construir otra gramática que estará en la forma normal de Greibach mediante la aplicación de los siguientes pasos:

1. Aplicación del algoritmo que elimina la recursividad general por la izquierda.
2. Transformación de las reglas de los símbolos no terminales de la gramática original.
3. Transformación de las reglas de los símbolos no terminales obtenidos al eliminar la recursividad inmediata.

Paso 1: Aplicación del algoritmo que elimina la recursividad general por la izquierda.

Las reglas de producción resultantes serán de la forma:

$$\begin{aligned} A_i &\longrightarrow A_j\gamma \quad \forall j > i \wedge i, j \in \{i, 2, \ldots, n\} \\ A_i &\longrightarrow a\gamma \\ A'_i &\longrightarrow \gamma \end{aligned}$$

donde $A_i, A_j \in V_N$, $a \in V_T$, los símbolos A'_i ($i \in \{1, 2, \ldots, m\}$) han sido generados al eliminar la recursividad inmediata por la izquierda y $\gamma \in (V_N \cup \{A'_1, A'_2, \ldots, A'_n\})*$. En particular, la reglas del símbolo A_n serán de la forma $A_n \longrightarrow a\ \gamma$ y ya estarán en la forma normal de Greibach.

Paso 2: Transformación de las reglas de los símbolos no terminales de la gramática original mediante la aplicación del siguiente algoritmo:

Transformación de las reglas de los símbolos no terminales de la gramática original

[1] ***inicio***
[2] ***para*** i ***de*** $n-1$ ***a*** 1 ***hacer***
[3] ***para*** j ***de*** $i+1$ ***a*** n ***hacer***
[4] ***para*** cada producción actual de A_i de la forma $A_i \longrightarrow A_j\ \gamma$
[5] ***hacer***
[6] ***si*** $A_j \longrightarrow a_1\ \alpha_1 \mid a_2\ \alpha_2 \mid \cdots \mid a_p\ \alpha_p$
[7] son las producciones actuales de A_j
[8] ***entonces*** $A_i \longrightarrow a_1\ \alpha_1\ \gamma \mid a_2\ \alpha_2\ \gamma \mid \cdots \mid a_p\ \alpha_p\ \gamma$
[9] pasan a ser producciones actuales de A_i
[10] ***fin si***
[11] ***fin para***
[12] ***fin para***
[13] ***fin para***
[14] ***fin***

Las reglas de producción resultantes serán de la forma:

$$\begin{aligned} A_i &\longrightarrow a\ \gamma \\ A_i' &\longrightarrow \gamma \end{aligned}$$

En particular, todas la reglas de los símbolos no terminales originales estarán en la forma normal de Greibach.

Paso 3: Aplicación del siguiente algoritmo para transformar las reglas de los símbolos no terminales obtenidos al eliminar la recursividad inmediata:

Transformación de las reglas de los símbolos obtenidos al eliminar la recursividad inmediata

[1] ***inicio***
[2] ***para*** i ***de*** 1 ***a*** m ***hacer***
[3] ***para*** j ***de*** 1 ***a*** n ***hacer***
[4] ***para*** cada producción actual de A'_i de la forma $A'_i \longrightarrow A_j \ \gamma$
[5] ***hacer***
[6] ***si*** $A_j \longrightarrow a_1 \ \alpha_1 \mid a_2 \ \alpha_2 \mid \cdots \mid a_p \ \alpha_p$
[7] son las producciones actuales de A_j
[8] ***entonces*** $A'_i \longrightarrow a_1 \ \alpha_1 \ \gamma \mid a_2 \ \alpha_2 \ \gamma \mid \cdots \mid a_p \ \alpha_p \ \gamma$
[9] pasan a ser producciones actuales de A'_i
[10] ***fin si***
[11] ***fin para***
[12] ***fin para***
[13] ***fin para***
[14] ***fin***

Las reglas de producción resultantes serán de la forma:

$$\begin{aligned} A_i &\longrightarrow a \ \gamma \\ A'_i &\longrightarrow a \ \gamma \end{aligned}$$

Por tanto, todas la reglas estarán en la forma normal de Greibach.

Se va a obtener la gramática en la forma normal de Greibach equivalente a la siguiente gramática, que ya está en la forma normal de Chomsky:

$$\begin{array}{rcl} P & = & \{ \\ & (1) & S \longrightarrow A\ B \\ & (2) & A \longrightarrow S\ B \\ & (3) & A \longrightarrow a \\ & (4) & B \longrightarrow B\ A \\ & (5) & B \longrightarrow d \\ & & \} \end{array}$$

Paso 1: Aplicación del algoritmo de la recursividad general por la izquierda:

1. La regla de S no necesita transformarse.
2. Se sustituye S por su alternativa en la regla $A \longrightarrow S\ B$ generando la regla:
$$A \longrightarrow A\ B\ B$$
3. Eliminación de la recursividad por la izquierda de las reglas de A. Las reglas actuales de A son
$$A \longrightarrow A\ B\ B \mid a$$
y las reglas que se generan al eliminar la recursividad inmendiata son:
$$\begin{array}{rcl} A & \longrightarrow & a \mid a\ A' \\ A' & \longrightarrow & B\ B \mid B\ B\ A' \end{array}$$
4. No se tiene que sustituir ningún símbolo en las reglas de B.
5. Eliminación de la recursividad por la izquierda de las reglas de B. Las reglas actuales de B son
$$B \longrightarrow B\ A \mid d$$
y las reglas que se generan al eliminar la recursividad inmendiata son:
$$\begin{array}{rcl} B & \longrightarrow & d \mid d\ B' \\ B' & \longrightarrow & A \mid A\ B' \end{array}$$

$$\begin{array}{rcl} P_1 & = & \{ \\ & & S \longrightarrow A\ B \\ & & A \longrightarrow a \mid a\ A' \\ & & B \longrightarrow d \mid d\ B' \\ & & A' \longrightarrow B\ B \mid B\ B\ B' \\ & & B' \longrightarrow A \mid A\ B' \\ & & \} \end{array}$$

Paso 2: Transformación de las reglas de los símbolos no terminales originales.

1. Las reglas de B ya están en la forma normal de Greibach.
2. Las reglas de A ya están en la forma normal de Greibach.
3. Se sustituye A por sus alternativas en la regla $S \longrightarrow A\ B$ generando las siguientes reglas de S:

$$S \longrightarrow a\ B \mid a\ A'\ B$$

Paso 3: Transformación de las reglas de los símbolos no terminales generados al eliminar la recursividad inmediata por la izquierda.

1. Trasformación de las reglas de A': $A' \longrightarrow B\ B \mid B\ B\ A'$. Se sustituye B por sus alternativas, generándose las siguientes reglas de A':

$$A' \longrightarrow d\ B \mid d\ B\ A' \mid d\ B'\ B \mid d\ B'\ B\ A'$$

2. Transformación de las reglas de B': $B' \longrightarrow A \mid A\ B'$. Se sustituye A por sus alternativas, generándose las siguientes reglas de B':

$$B' \longrightarrow a \mid a\ A' \mid a\ B' \mid a\ A'\ B'$$

3. Las reglas de B' no requieren ninguna sustitución.

El conjunto de producciones de la gramática en la forma normal de Greibach es:

$$\begin{aligned} P_1 = \{ & \\ & S \longrightarrow a\ B \mid a\ A'\ B \\ & A \longrightarrow a \mid a\ A' \\ & B \longrightarrow d \mid d\ B' \\ & A' \longrightarrow d\ B \mid d\ B\ A' \mid d\ B'\ B \mid d\ B'\ B\ A' \\ & B' \longrightarrow a \mid a\ A' \mid a\ B' \mid a\ A'\ B' \\ & \} \end{aligned}$$