Autómatas y Lenguajes Formales 2019-II
Facultad de Ciencias UNAM
Nota de Clase 10, Gramáticas libres de contexto

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

3 de febrero de 2020

1. Gramáticas

Un mecanismo relevante para generar un lenguaje es mediante el concepto de gramática formal. Las gramáticas formales fueron introducidas por Noam Chomsky en 1956. La intención era tener un modelo para la descripción de lenguajes naturales. Posteriormente, una clase especial de estos formalismos, llamados gramáticas libres de contexto, se utilizaron como herramienta para presentar la sintaxis de lenguajes de programación y para el diseño de analizadores léxicos de compiladores.

Definición 1. *Una gramática libre de contexto es una cuaterna* $G = \langle V, T, S, P\rangle$ *tal que:*

- V *es un alfabeto de* variables *o* símbolos no-terminales*, los cuales se denotan con mayúsculas* $A, B, C, \ldots$
- T *es un alfabeto de* símbolos terminales*, los cuales se denotan con minúsculas* $a, b, c, \ldots$. *Además se requiere que* $T \cap V = \varnothing$.
- $S \in V$ *es una variable distinguida llamada el* símbolo inicial *o el axioma.*
- P *es un conjunto finito de reglas de reescritura, llamadas* reglas de producción *o producciones. Cada regla es de la forma*

$$A \to \alpha$$

donde $A \in V,\ \alpha \in (V \cup T)^\star$.

Las reglas de producción sirven para generar cadenas, proceso que se formaliza mediante las derivaciones formales:

Definición 2 (Derivación formal)**.** *Dadas dos palabras* $w, v \in (V \cup T)^\star$ *decimos que* v *es* derivable *a partir de* w *en* ***un paso*** $(w \to v)$ *si y sólo si:*

Existe una regla $A \to \alpha$ *en* P *y cadenas* $\gamma_1, \gamma_2 \in (V \cup T)^\star$ *tales que:*

$$w = \gamma_1 A \gamma_2 \qquad\qquad v = \gamma_1 \alpha \gamma_2$$

Algunos autores utilizan $\Rightarrow$ en vez de $\to$ para denotar la relación de derivación. Nosotros preferimos sobrecargar el operador $\to$.

Definición 3. *Decimos que una cadena v es* derivable *a partir de w si existen palabras $\gamma_2, \ldots, \gamma_n$ tales que*

$$w = \gamma_1 \to \gamma_2 \ldots \gamma_{n+1} \to \gamma_n = v$$

En tal caso escribimos $w \to^\star v$.

Obsérvese que las relaciones $\to$ y $\to^\star$ entre cadenas se pueden también definir de manera recursiva con las siguientes reglas:

- Levantamiento de las producciones a cualquier contexto

$$\frac{A \to \alpha \ \in \mathcal{P}}{\gamma A \gamma' \to \gamma \alpha \gamma'}$$

  Esto es importante pues las derivaciones reescriben cadenas arbitrarias y las producciones libres de contexto sólo indican como reescribir una variable. Con esta regla la relación $\to$ se convierte en una relación entre cadenas.

- La relación $\to^\star$ incluye cero pasos de reescritura

$$\frac{}{\alpha \to^\star \alpha} \ (D1)$$

- $\to^\star$ consiste en dar muchos pasos (tal vez cero) y un paso de reescritura

$$\frac{\alpha \to^\star \beta \quad \beta \to \gamma}{\alpha \to^\star \gamma} \ (D2)$$

- Alternativamente, $\to^\star$ consisten en dar primero un paso de reescritura, seguido de muchos pasos (tal vez cero).

$$\frac{\alpha \to \beta \quad \beta \to^\star \gamma}{\alpha \to^\star \gamma} \ (D3)$$

La relación $\to^\star$ queda definida inductivamente por las reglas $(D1), (D2)$ o bien $(D1), (D3)$, de manera que según convenga, podra usarse $(D2)$ o $(D3)$ en una prueba particular.

Adicionalmente es necesario en muchos casos el hecho de que una derivación se preserva bajo cualquier contexto, lo cual queda formalizado con la siguiente regla que resulta admisible a partir de las anteriores:

$$\frac{\alpha \to^\star \beta}{\gamma \alpha \gamma' \to^\star \gamma \beta \gamma'} \ (D4)$$

Ahora ya podemos dar la definición formal del lenguaje generado por una gramática.

Definición 4 (Lenguaje de una gramática)**.** *Dada una gramática $G = \langle V, T, S, P \rangle$ definimos al lenguaje generado por G, denotado $L(G)$, como el conjunto de palabras de símbolos* ***terminales*** *derivables a partir del símbolo inicial S. Es decir,*

$$L(G) = \{w \in T^\star \mid S \to^\star w\}$$

Veamos algunos ejemplos:

Ejemplo: Las siguientes son gramáticas libres de contexto para algunos lenguajes regulares:

- $L = a^\star$

$$S \to aS \mid \varepsilon$$

- $L = a^\star b^\star$

$$\begin{array}{lcl} S & \to & aS \mid bA \mid \varepsilon \\ A & \to & bA \mid b \mid \varepsilon \end{array}$$

- $L = 0^+1^+$

$$\begin{array}{lcl} S & \to & CU \\ C & \to & 0C \mid 0 \\ U & \to & 1U \mid 1 \end{array}$$

Ejemplo:

- $L = \{a^n b a^m \mid n, m \geq 1\} = a^+ b a^+$

$$\begin{array}{lcl} S & \to & aS \mid aB \\ B & \to & bC \\ C & \to & aC \mid a \end{array}$$

- $L = \{a^n b^n \mid n \in \mathbb{N}\}$ (no es regular)

$$S \to aSb \mid \varepsilon$$

- $L = \{w \in \{a, b\}^\star \mid w = w^R\}$ (no es regular)

$$S \to aSa \mid bSb \mid a \mid b \mid \varepsilon$$

Ejemplo: Considere el siguiente lenguaje $L = (a+b)^\star$, es el lenguaje en que cualquier cadena con a y b debe generarse. Veamos la gramática correspondiente:

- La cadena vacía debe generarse: $S \to \varepsilon$
- Si $w \in L$ entonces $wa \in L$: $S \to Sa$
- Si $w \in L$ entonces $wb \in L$: $S \to Sb$

La cadena $w = ababb$ se deriva como sigue:

$$S \to Sb \to Sbb \to Sabb \to Sbabb \to Sababb \to ababb$$

Ejemplo: Proporcionar una gramática que genere las cadenas en $L = \{a^i b^j \mid i, j \in \mathbb{N}, i \leq j\}$.

- La cadena vacía debe generarse ($i = j = 0$): $S \to \varepsilon$
- Debe haber al menos tantas b como a, primero las a y luego las b:

$$S \to aSb$$

- Puede haber más b al final:

$$S \to Sb$$

Ejemplo de derivación para la cadena $w = aabbb$

$$S \to aSb \to aaSbb \to aaSbbb \to aabbb$$

Ejemplo: El lenguaje $L = \{a^i b^j a^j b^i \mid i, j \in \mathbb{N}\}$ es generado por la siguiente gramática:

- Primero generamos el centro de la palabra, $b^j a^j$:

$$S \to B \qquad B \to bBa \qquad B \to \varepsilon$$

- Después los extremos $a^i,\ b^i$: $\qquad S \to aSb$

Un ejemplo de derivación es la generación de la cadena $w = aababb$

$$S \to aSb \to aaSbb \to aaBbb \to aabBabb \to aababb$$

Ejemplo: Dado $L = \{a^i b^i a^j b^j \mid i, j \in \mathbb{N}\}$, la siguiente gramática genera el lenguaje:

- Primero generaremos el lenguaje $\{a^i b^i \mid i \in \mathbb{N}\}$ mediante:

$$P \to \varepsilon \qquad P \to aPb$$

- Para generar a L simplemente agregamos:

$$S \to PP$$

Ejemplo de derivación, la cadena $w = aabbab$ se genera como sigue:

$$S \to PP \to aPbP \to aPbaPb \to aaPbbaPb \to aaPbbab \to aabbab$$

Ejemplo: El lenguaje $L = \{a^i b^i \mid i \in \mathbb{N}\} \cup \{b^i a^i \mid i \in \mathbb{N}\}$ se genera mediante:

- Primero el lenguaje $\{a^i b^i \mid i \in \mathbb{N}\}$ se genera con la siguiente gramática:

$$P \to \varepsilon \qquad P \to aPb$$

- Después generamos el lenguaje $\{b^i a^i \mid i \in \mathbb{N}\}$ mediante:

$$Q \to \varepsilon \qquad Q \to bQa$$

- Finalmente, para generar a L simplemente agregamos:

$$S \to P \qquad S \to Q$$

Ejemplo de derivación para $w = bbbaaa$

$$S \to P \to aPb \to aaPbb \to aaaPbbb \to aaabbb$$

1.1. Derivaciones y árboles

Una derivación formal usando gramáticas libres de contexto no suele ser única, pues cada paso depende de la variable elegida para su reescritura. Esto genera un no determinismo, que puede eliminarse utilizando una convención, la cual suele ser de dos tipos, eligiendo siembre la variable más a la izquierda o bien más a la derecha para el proceso de reescritura.

Definición 5. *Una derivación $S \to^\star w$ es una derivación por la izquierda o más a la izquierda si en cada paso se reescribe la variable más a la izquierda en la palabra.*

Definición 6. *Una derivación $S \to^\star w$ es una derivación por la derecha o más a la derecha si en cada paso se reescribe la variable más a la derecha en la palabra.*

Ejemplo: En la gramática

$$S \to aAs \mid a \qquad\qquad A \to SbA \mid SS \mid ba$$

1. La siguiente es una derivación por la izquierda de la cadena $aabbaa$

$$\mathbf{S} \to a\mathbf{A}S \to a\mathbf{S}bAS \to aab\mathbf{A}S \to aabba\mathbf{S} \to aabbaa$$

2. La siguiente es una derivación por la derecha de $aabbaa$

$$\mathbf{S} \to aA\mathbf{S} \to a\mathbf{A}a \to aSb\mathbf{A}a \to a\mathbf{S}bbaa \to aabbaa$$

intuitivamente es claro que siempre se puede construir una derivación más a la izquierda o derecha, como lo asegura la siguiente

Proposición 1. *Si $w \in L(G)$ entonces w tiene una derivación más a la izquierda y una derivación más a la derecha.*

Demostración. Sea $w \in L(G)$.El resultado se sigue haciendo inducción sobre $S \to^\star w$. □

1.2. Construcción de árboles de derivación

El uso y razonamiento con cadenas es adecuado para el humano pero no resulta una forma adecuada de implementar lenguajes. En su lugar deben utilizarse representaciones mejor estructuradas estas estructuras adecuadas para la implementación en compiladores, en específico para el análisis sintáctico[1] de programas fuente.

Definición 7. *Dada una gramática libre de contexto $G = \langle V, T, S, P\rangle$, un árbol de derivación en G se construye como sigue:*

1. *La raíz contiene al símbolo inicial S.*
2. *Cada nodo interior contiene una variable.*
3. *Cada hoja contiene un símbolo de $V \cup T \cup \{\varepsilon\}$.*
4. *Si un nodo interior contiene una variable A entonces sus hijos contienen símbolos (de izquierda a derecha) $a_1, \ldots, a_n$ si y sólo si $A \to a_1 a_2 \ldots a_n$ está en P.*
5. *La palabra generada se puede reconstruir al leer las hojas de izquierda a derecha.*

[1] Parsing

2. Propiedades de Cerradura

La clase de lenguajes libres de contexto es cerrada bajo las siguientes operaciones:

- Unión: si L_1, L_2 son lenguajes libres del contexto entonces $L_1 \cup L_2$ es un lenguaje libre del contexto.
- Concatenación: si L_1, L_2 son lenguajes libres del contexto entonces $L_1 L_2$ es un lenguaje libre del contexto.
- Estrella de Kleene: si L_1 es un lenguaje libres del contexto entonces $L_1^\star$ es un lenguaje libre del contexto.

Veamos más a detalle estas operaciones:

- Cerradura bajo la unión
  Si $G_1 = \langle V_1, T, S_1, P_1 \rangle$, $G_2 = \langle V_2, T, S_2, P_2 \rangle$ son dos gramáticas libres de contexto donde $L_1 = L(G_1)$ y $L_2 = L(G_2)$ entonces $L_1 \cup L_2 = L(G)$ y G es la gramática

  $$G = \langle V_1 \cup V_2 \cup \{S\}, T, S, P \rangle$$

  y las reglas de producción están dadas por $P_1 \cup P_2$ más las producciones:

  $$S \to S_1 \qquad S \to S_2$$

- Cerradura bajo la concatenación
  Si $G_1 = \langle V_1, T, S_1, P_1 \rangle$, $G_2 = \langle V_2, T, S_2, P_2 \rangle$ son dos gramáticas libres de contexto donde $L_1 = L(G_1)$ y $L_2 = L(G_2)$ entonces $L_1 L_2 = L(G)$ donde G es la gramática

  $$G = \langle V_1 \cup V_2 \cup \{S\}, T, S, P \rangle$$

  y P está dado por $P_1 \cup P_2$ más la producción:

  $$S \to S_1 S_2$$

- Cerradura bajo la estrella de Kleene
  Si $G_1 = \langle V_1, T, S_1, P_1 \rangle$ es una gramática libre de contexto con $L_1 = L(G_1)$ entonces $L_1^\star = L(G)$ donde G es la gramática

  $$G = \langle V_1 \cup \{S\}, T, S, P \rangle$$

  y P está dado por P_1 más la producciones:

  $$S \to S_1 S_1 \qquad S \to \varepsilon$$

En general las siguientes propiedades **no son válidas** para lenguajes libres del contexto.

- Cerradura bajo la intersección.
- Cerradura bajo el complemento.

- Cerradura bajo la diferencia.

Veamos los contraejemplos pertinentes, suponiendo que el lenguaje

$$L = \{a^i b^i c^i \mid i \geq 1\}$$

no es libre de contexto. Esto se verifica mediante el lema del bombeo para esta clase de lenguajes, el cual es una generalización del lema del bombeo para lenguajes regulares

- Intersección
  La intersección de dos lenguajes libres de contexto puede ser un lenguaje que **no** es libre del contexto.
  Por ejemplo considérense $L_1 = \{a^i b^i c^j \mid i, j \geq 1\}$ libre del contexto:

  $$S \to AB,\ A \to aAb \mid ab,\ B \to cC \mid c$$

  y $L_2 = \{a^i b^j c^j \mid i, j \geq 1\}$ también libre del contexto:

  $$S \to AB,\ A \to aA \mid a,\ B \to bBc \mid bc$$

  Pero $L_1 \cap L_2 = L$ que no es libre de contexto.

- Complemento
  Si el complemento de un lenguaje libre de contexto L, $\bar{L}$ fuera también libre del contexto entonces la intersección también lo será pues:

  $$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

- Diferencia
  Si la diferencia fuera un lenguaje libre de contexto, entonces también lo será el complemento pues:

  $$\overline{L} = \Sigma^\star - L$$

3. Normalización de Gramáticas

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 parar en la construcción de un árbol de derivación.
- Pero si la gramática está en cierta forma normal, sí es posible saber cuando parar en la búsqueda de árboles de derivación para contestar el problema de la pertenencia.

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

3.1. Eliminación de variables inútiles

Definición 8. *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 9. *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 10. *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 de $\to$:
>
> $$Prod := \{A \in V \mid A \to w \ \in P,\ w \in T^\star\}$$
>
> **Repetir** la incorporación de variables cuyas producciones contienen variables productivas y símbolos terminales a la derecha de $\to$:
>
> $$Prod := Prod \cup \{A \in V \mid A \to w,\ w \in (T \cup Prod)^\star\}$$
>
> **Hasta** que no se añaden nuevas variables a $Prod$

Ejemplo: Calculemos las variables productivas de la gramática:

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

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:

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

El siguiente algoritmo permite hallar variables accesibles:

> **Iniciar**
>
> $$Acc := \{S\}$$
>
> **Repetir** la incorporación de variables que aparecen a la derecha de $\to$ en producciones de variables accesibles:
>
> $$Acc := Acc \cup \{A \in V \mid \exists\ B \to uAv \in P,\ B \in Acc,\ u, v \in (V \cup T)^\star\}$$
>
> **Hasta** que no se añaden nuevas variables a Acc

Ejemplo: Calculemos las variables accesibles de la gramática:

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

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:

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

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

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

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

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

y claramente A es inútil por ser inaccesible.

3.2. 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 11. *Una variable A se llama **anulable** o* **nilpotente** *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 generan a $\boldsymbol{\varepsilon}$ con una producción directa

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

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.

3.3. Eliminación de producciones unitarias

Definición 12. *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 $A \to w$ se agrega la producción

$$B \to w$$

- Se eliminan 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:

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

Así la gramática obtenida al eliminar producciones unitarias es:

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

La importancia de la normalización de gramáticas radica en el hecho de que dada una gramática G sin ε-producciones ni producciones unitarias y una cadena u de longitud n, entonces la construcción de un árbol de derivación terminará eventualmente, mostrando que $u \in L(G)$. En caso contrario basta con construir el árbol hasta el nivel $2n-1$ para concluir que $u \notin L(G)$. Esto sucede pues en cada paso se obtiene un nuevo terminal (a lo mas n pasos) o se aumenta la longitud de la palabra en 1 (a lo mas $n-1$ pasos)

4. Formas normales

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

4.1. Forma Normal de Chomsky

Definición 13. *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:*

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

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 la producción $S' \to \boldsymbol{\varepsilon} \mid S$.
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 $A \to w_1 w_2 \dots w_n,\ n \geq 2$.
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

$$A \to T_a T_b B T_a C \qquad T_a \to a \qquad T_b \to b$$

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

$$A \to T_a D_1 \qquad D_1 \to T_b D_2 \qquad D_2 \to B D_3 \qquad D_3 \to T_a C$$

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

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

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

La gramática equivalente es:

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

4.2. 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 14. *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 $\alpha \in V^\star$.*

El algoritmo para transformar una gramática en FN de Greibach se omite, el proceso es directo aunque tedioso y requiere como precondición que la gramática esté en forma normal de Chomsky. La importancia de la FNG radica en el hecho de que el problema de la pertenencia se resuelve no sólo linealmente sino que en exactamente n pasos, siendo n la longitud de la cadena en cuestión, esto es claro pues cada producción en la FNG genera un nuevo terminal.