Gramáticas Independientes del Contexto (GIC)

Definiciones y propiedades

¿Qué es una gramática?

- Modelo de estructuras recursivas.
- Definición de reglas para representar las expresiones de los lenguajes.
- Especificación rigurosa y explicita de estructura de un lenguaje.

Características:

- Ausencia de ambigüedad, por tanto bien definidas.
- Rigurosas (claridad, explicitud).
- Facilitan evaluación: comprobar, conclusiones, derivar.
- Hacer predicciones: generalización.
- Desarrollo de aplicaciones.

Existen varios tipos de Gramáticas, las que mas se usan en computación son las gramáticas generativas, definidas por Noam Chomsky.

Las Gramáticas Generativas constan de la siguiente tupla de elementos:

$$\mathbf{G} = (\mathbf{V}, \mathbf{T}, \mathbf{P}, \mathbf{S})$$

Donde:

V: conjunto finito de Variables (símbolos no terminales/categorías sintácticas)
T: conjunto finito de símbolos Terminales (alfabeto terminal o alfabeto de símbolos)
P: conjunto finito de Producciones o Reglas (definición recursiva del lenguaje)

Cada regla o producción consta de:

- Cabeza: variable.
- $\mapsto$: símbolo de producción.
- Cuerpo: cadena de 0 o mas símbolos terminales y/o variables.

Es decir una regla tiene la forma: Cabeza $\mapsto$ Cuerpo, por ejemplo: Por ejemplo: A$\mapsto$ aBA donde A,B en V, a en T

S: símbolo inicial

Nota: Se asume que de V $\cap$ T= ϕ

Las gramáticas generativas son modelos matemáticos finitos que nos permiten generar las cadenas o palabras de un lenguaje finito o infinito.

Según la Jerarquía de Chomsky: las gramáticas generativas se clasifican en 4 tipos. Esta clasificación es inclusiva, es decir tipo 3 $\subset$ tipo 2 $\subset$ tipo 1 $\subset$ tipo 0

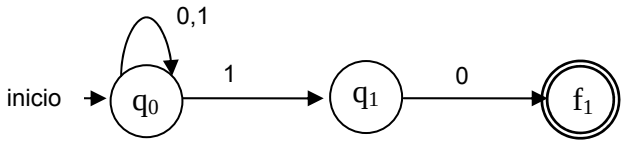
Tipo	Gramática	Restricciones a la forma de las Reglas	Lenguaje	Autómata
0	Irrestricta	Ninguna restricción $\mu_1\mu_2 \mapsto \beta_1\beta_2\beta_3$ donde $\mu_i\beta_i$ en (V U T)*	Enumerables recursivamente	Máquina de Turing
1	Dependientes del Contexto o Sensibles de Contexto	La parte derecha contiene como mínimo los símbolos de la parte izquierda $A\mu B \mapsto A\beta B$ Donde A,B en V y μ,β en (V U T)*	Dependiente del Contexto	Autómata lineales infinitos
2	Independiente del Contexto	La parte izquierda solo puede tener un símbolo $A \mapsto \beta$, Donde A en V, y β en (V U T)*	Independiente del Contexto	Autómata de Pilas
3	Regulares	La regla solo puede tener 2 formas: $A \mapsto aB$ y $A \mapsto a$, donde A,B en V y a en T	Regulares	Autómata finito

Gramáticas Regulares (GR)

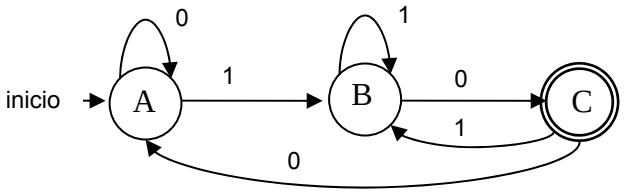
Son equivalen a los AF pues tratan sobre el mismo tipo de lenguajes: los lenguajes regulares. Las reglas de las GR tienen en la cabeza solo un variable y en su cuerpo o un símbolo Terminal o un símbolo Terminal seguido de una Variable, por tanto son de la forma:

$A \mapsto aB$ y $A \mapsto a$,
donde A,B en V y a en T

A partir de un AF que reconoce un lenguaje regular se puede obtener una GR que lo genere. Por ejemplo para el L = (0 + 1)*10, el siguiente AFND lo reconoce:



El AFD equivalente sería:



Para obtener la GR a partir del anterior AFD se debe hacer las siguientes equivalencias:

- V = Q = { q_0, q_1, f_1 } que por conveniencia se etiquetaron como { A, B, C }
- T = Σ = { 0, 1 }
- P, el conjunto de producciones se obtiene aplicando la siguiente transformación: Para cada transición del AFD de la forma δ(q,a) = p, tal que q,p en Q y a en Σ, se escribe la regla q → ap, y si p en F entonces se agrega la producción q → a, es decir como si p fuese ε.
- S = A = q_0, es decir el símbolo inicial de la gramática es el estado inicial del AFD

Se esta forma se obtiene que G = ({ A, B, C },{ 0, 1 }, P, A), donde P es el siguiente conjunto de producciones:

1. A → 1B
2. A → 0A
3. B → 1B
4. B → 0C
5. B → 0
6. C → 1B
7. C → 0A

Este conjunto de producciones son de la forma de una Gramática Regular. A partir de A, símbolo inicial, se sustituyen sucesivamente para generar una cadena formada solo por símbolos terminales que pertenecen al lenguaje (0+1)*10:

A ==>^{2} 0**A** ==>^{2} 00**A** ==>^{1} 001**B** ==>^{4} 0010

Desde A se obtiene la cadena 0010 que forma parte del lenguaje aplicando derivaciones, es decir sustituyendo una variable que aparece en la cabeza de alguna producción por su cuerpo.

Gramáticas Libres de Contexto

Las GIC (Gramáticas Independientes del Contexto) o GLC (Gramáticas Libres del Contexto) son llamadas también "Gramática en la Forma de Backus-Naur (BNF)" (usado para describir lenguajes de programación). Las GIC se usan para inferir si ciertas cadenas están en el lenguaje expresado por la gramática. Hay 2 tipos de inferencia:

- Inferencia recursiva (cuerpo a cabeza/de cadenas a variables)
- Derivación (cabeza a cuerpo, expansión de producciones)

Ejemplo: cadenas palíndromes en {0,1} S→ λ| 0 | 1 | 1S1 | 0S0

Derivación

Aplicación de las producciones de una Gramática para obtener una cadena de terminales. Consiste en sustituir la variable de la cabeza por el cuerpo de la producción.

Símbolo empleado es: $\Rightarrow$ (un paso de derivación)

$\beta A \mu \Rightarrow \beta \xi \mu$ si existe la producción $A \rightarrow \xi$
A en V y ξ, μ, β en $(V \cup T)^*$

$\Rightarrow^*$ Este símbolo indica múltiples pasos de derivación

Por ejemplo de la GLC anterior

$S \rightarrow \lambda \mid 0 \mid 1 \mid 1S1 \mid 0S0$

Una derivación sería la siguiente:

S ==>^{4} 1**S**1 ==>^{4} 11**S**11 ==>^{2} 110**S**011 ==>^{3} 1101011 esta cadena es binaria palindrome

Lenguaje

Si G es una GIC, G = (V, T, P, S) entonces
L (G) = { w esta en T* | $S \Rightarrow^* w$ }

El lenguaje generado por una GLC G es el conjunto de cadenas formadas por símbolos terminales que tienen derivaciones desde el símbolo inicial S de la gramática

Formas sentenciales

Las derivaciones a partir del símbolo inicial (S), $S \overset{*}{\Rightarrow} \alpha$

$S \overset{*}{\underset{mi}{\Rightarrow}} \alpha$ forma sentencial izquierda.

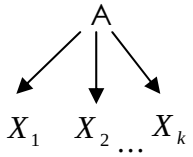
$S \overset{*}{\underset{md}{\Rightarrow}} \alpha$ forma sentencial derecha.

Árboles de Derivación

ES un Árbol formado a partir de la derivación de una gramática. Sirve para estudiar la ambigüedad (más de un árbol de derivación y por tanto de interpretaciones)

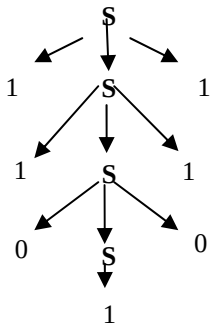
Características:

- Cada nodo interior es variable.
- Cada nodo hoja es Terminal o $\in$
- Si existe una producción $A \mapsto X_1 X_2 \ldots X_k$, ver árbol de derivación siguiente:



- Para que el X_i sea λ debe existir la producción $X_i \mapsto \lambda$
- La cadena resultado del árbol: las hojas del árbol concatenadas de izquierda a derecha, recorrido en pre orden.

Por ejemplo, en la GLC anterior (binario palindromo), donde
G = ({S},{0,1},P,S), y P = { S→ λ | 0 | 1 | 1S1 | 0S0 }
La derivación anterior de la cadena 1101011, se obtiene el siguiente árbol de derivación:



La derivación:
S ==>4 1**S**1 ==>4 11**S**11 ==>2 110**S**011 ==>3 1101011

Ejercicios

1. L = $\{0^n 1^n, n \geq 1\}$

   S → 01 | 0S1

2. L = $\{0^n 1^n, n \geq 0\}$

   S → λ | 01 | 0S1

3. L = $\{0^n 1^m, n \geq m\}$

   S → 0 | 0S | 0S1

Gramáticas ambiguas

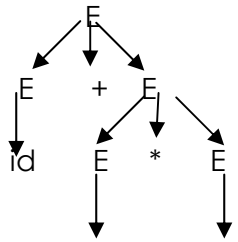
La ambigüedad se refiere a la estructura de la gramática

Si cada cadena del lenguaje posee una única estructura entonces la gramática es no ambigua. En cambio si existe al menos alguna cadena del lenguaje con más de una estructura (árbol de derivación) entonces la gramática es ambigua.

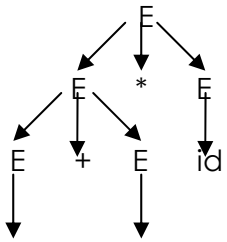
Ejemplo: E→ E+E | E* E | (E) | id

La forma sentencial o cadena E+E*E puede generarse mediante 2 derivaciones a partir de E:
1) E ⇒ E+E ⇒ id +E ⇒ id +E*E ⇒ id + id*E ⇒ id + id*id
2) E ⇒ E*E ⇒ E+E*E ⇒ id +E*E⇒ id + id*E ⇒ id + id*id

1-



2-



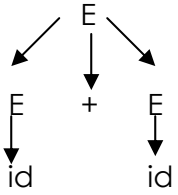
Dos derivadas izquierda distintas y se llega exactamente a la misma cadena.

id id id id

Una gramática es ambigua si tiene al menos una cadena w que tenga más de un árbol de derivación (o mas de una derivación izquierda o mas de una derivación derecha).

Las derivaciones no es la estructura, la estructura es el árbol de derivación. Por ejemplo si se tiene las siguientes derivaciones diferentes, se obtiene el mismo árbol.

2 derivaciones { $E \Rightarrow E+E \Rightarrow id + E \Rightarrow id + id$ (d^{mi})
$E \Rightarrow E+E \Rightarrow E + id \Rightarrow id + id$ (d^{md})



La multiplicidad de derivaciones no es la causa de la ambigüedad. La causa es la existencia de 2 o mas árboles de derivación para una cadena w en T* (por eso se sugiere escoger d^{mi} o d^{md} para verificar ambigüedad).

Teorema: para toda G = (V, T, P, S) y toda cadena w en T*, w tiene 2 árboles de derivación distintos si y solo si w tiene 2 derivaciones mas a la izquierda distintas desde S.

Para detectar la ambigüedad de las gramáticas no existe un algoritmo que nos diga si una gramática es o no ambigua. Las causas de la ambigüedad son:

-No se respeta la presencia de los operadores, necesitamos forzar que la estructura sea única.

-No se tiene una convención (izquierda o derecha) para agrupar operadores idénticos (E + E * E), la convención es por la izquierda.

Una técnica que puede eliminar la ambigüedad, consiste en introducir nuevas variables con el nivel de agrupamiento siguiente:

1- **F** Factor { -Identificador
-Expresión con paréntesis

No puede separase con * ò +

2- **T** Término { -productos de 2 ò más factores.

No puede separase con +

3- **E** Expresión { cualquier cosa.

Ejemplo: la gramática ambígua anterior $S \rightarrow S+S \mid S* S \mid (S) \mid id$, puede sustituirse por su equivalente no ambígua:

$S \rightarrow S + T \mid T$

$T \rightarrow T * F \mid F$
$F \rightarrow (S) \mid a$

Ambigüedad inherente:

Se dice que un Lenguaje Libre de Contexto es inherente ambiguo si todas las gramáticas que presentan ese lenguaje son ambiguas. Estos lenguajes existen aunque son difíciles de imaginar.

Un LLC es inherente ambiguo si todas las gramáticas son ambiguas.

Simplificación de Gramáticas Libres de Contexto

1- Eliminar producciones λ
2- Eliminar producciones unitarias
3- Eliminar los símbolos inútiles

1- Eliminar ε producciones ($A \mapsto \lambda$)

Primero se revisa si el lenguaje contiene a la cadena vacía λ:

- Si λ no está en L no necesitamos λ producciones, por tanto aplicamos el algoritmo.
- Si λ en L, entonces generamos una GIC para L – {λ} y después al final hacer que:
  $G' = (V \cup \{S'\}, T, P', S')$ y $V \cap S' = \phi$ y $P' = P \cup (S' \mapsto S \mid \lambda)$

Existe un algoritmo para detectar símbolos nulificables

A es nulificable si $A \overset{*}{\Rightarrow} \lambda$

- Si $A \Rightarrow \lambda$ en P entonces A es nulificable
- Si $A \Rightarrow \beta$ en P y todos los símbolos de β son nulificables $\Rightarrow$ A es nulificable.

ALGORITMO para eliminar producciones λ:

INICIO

i= 0

$N^0 = \{A \mid A \rightarrow \lambda$ una P y A en V$\}$

REPETIR

i= i +1

$N^i = N^{i-1} \cup \{A \mid A \rightarrow \beta$ en P y β en $(N^{i-1})+\}$

HASTA $(N^i = N_{i-1})$

P'= Si $A \mapsto X_1 X_2 \ldots X_n$ en P, entonces agregamos a P' las producciones $A \mapsto \beta_1\beta_2\ldots\beta_n$:

- Si X^i no esta en N^i entonces $\beta_i = X_i$
- Si X^i esta en N^i entonces $\beta_i = X_i$ o $\beta_i = \lambda$
- No todos los β_i son λ

FIN

Ejemplo: Dada la GLC G = ({S,A,B,C,D},{a,b},P,S), eliminar sus producciones λ

$S \rightarrow ABCBD$

$A \rightarrow CD$
$B \rightarrow Cb$
$C \rightarrow a \mid \lambda$
$D \rightarrow bD \mid \lambda$

Calcular N^i

$N^0 = \{C, D\}$

$N^1 = N^0 \cup \{A\} = \{C, D\} \cup \{A\} = \{C, D, A\}$

$N^2 = N^1 \cup \phi = \{C, D, A\} \cup \phi = \{C, D, A\}$

P'=
$S \rightarrow ABCBD \mid BCBD \mid ABCB \mid ABBD \mid ABB \mid BBD \mid BCB \mid BB$
$A \rightarrow CD \mid C \mid D$
$B \rightarrow Cb \mid b$
$C \rightarrow a$
$D \rightarrow bD \mid b$

2- Eliminar producciones unitarias ($A \mapsto B$)

Son producciones de la forma $A \mapsto B$, donde A y B en V

Definición de "derivable de A":

- Si $A \rightarrow B$ es una producción, entonces B es derivable de A.
- Si C es derivable de A, $C \rightarrow B$ es una producción y $B \neq A$, entonces B es derivable de A.

ALGORITMO para eliminar producciones unitarias:

INICIO

PRECONDICION = { La GLC G = (V, T, P, S) no contiene λ-producciones }

1. $P_1 = P$
2. Para todos A en V = { B | B es derivable de A }
3. Para cada par (A, B) tal que B es derivable de A y cada producción no unitaria $B \rightarrow \beta$, añadir la producción $A \rightarrow \beta$ a P_1 sino esta presente ya en P_1.
4. Eliminar todas las producciones unitarias de P_1.

FIN

Ejemplo: Tomamos el resultado del ejercicio anterior

1- $P_1 = P =$
$S \rightarrow ABCBD \mid BCBD \mid ABCB \mid ABBD \mid ABB \mid BBD \mid BCB \mid BB$
$A \rightarrow CD \mid C \mid D$
$B \rightarrow Cb \mid b$
$C \rightarrow a$
$D \rightarrow bD \mid b$

2- $S = \Phi$

A= {C, D}

B= Φ
C= Φ
D= Φ

3- Pares (A, C), Como C $\rightarrow$ a y A $\rightarrow$ C entonces A $\rightarrow$ a
(A, D), Como D $\rightarrow$ bD y D $\rightarrow$ b y A $\rightarrow$ D entonces A $\rightarrow$ bD y A $\rightarrow$ b

4- $P' = P_1 \cup \left\{ \begin{matrix} A \rightarrow a \\ A \rightarrow bD \text{ y } A \rightarrow b \end{matrix} \right\} - \left\{ \begin{matrix} A \rightarrow C \\ A \rightarrow D \end{matrix} \right\}$

Resultando la siguiente GLC:

S $\rightarrow$ ABCBD | BCBD | ABCB | ABBD | ABB | BBD | BCB | BB
A $\rightarrow$ CD | a | bD | b
B $\rightarrow$ Cb | b
C $\rightarrow$ a
D $\rightarrow$ bD | b

3- Eliminar símbolos inútiles

Los símbolos útiles (símbolos generadores de terminales y alcanzables desde S).

Sea G = (V, T, P, S), X es un símbolo útil si $S \overset{*}{\Rightarrow} \beta X \mu \overset{*}{\Rightarrow} w$ para β y μ en (V U T)* y en w T*

ALGORITMO para eliminar variables no generadoras

VV: Variables Viejas
VN: Variables Nuevas
INICIO
VV = ϕ
VN = { A | A $\rightarrow$ w en P y w en T* }
MIENTRAS VN $\neq$ VV HACER
VV = VN
VN = VV U { A | A $\rightarrow$ β en P y β en (VV U T)* }
FINMIENTRAS
V' = VN
P' = { A $\rightarrow$ β | A en V' y β en (V' U T)* }
FIN

Se eliminan todas las producciones en donde intervienen 1 o más de los símbolos no generadores de terminales. Las V' = VN son solo variables generadoras.

Ejemplo: Elimine las variables no generadoras de la siguiente GLC G = ({S,A},{a},P,S), con las producciones P:

S $\rightarrow$ AB | a
A $\rightarrow$ a

(0)
VV = ϕ, T = {a}
VN = {A, S}

(1)
VV = {A, S}
VN = {A, S} U {A, S} = {A, S}

Parada (VV = VN) ➔ {A, S} = {A, S}
V'= {A, S}

P'= $\begin{cases} S \rightarrow a \\ A \rightarrow a \end{cases}$

Decimos que X es "generador" si $X \Rightarrow^* w$ (con sucesivas derivaciones puede derivar una cadena de terminales) para alguna cadena terminal w. Todo símbolo terminal es generador, ya que w puede ser ese mismo símbolo terminal, que se deriva en cero pasos.

ALGORITMO para determinar si son alcanzables

INICIO
VV = TV = TN = ϕ
VN = {S}
MIENTRAS (VV ≠ VN) o (TV ≠ TN) HACER
VV = VN;
TV = TN;
VN = VV U { X | A $\rightarrow \beta_1 | \beta_2 | \ldots | \beta_n$ en P, X en V, X en algún β_i y A en VV}
TN = TV U {X | A $\rightarrow \beta_1 | \beta_2 | \ldots | \beta_n$ en P, X en T, X en algún β_i y A en VV}
FINMIENTRAS
V'= VN
T'= TN
P'= { A $\rightarrow \beta$ | A en V', β en (V' U T`)*, A $\rightarrow \beta$ en P }
P' es el conjunto de producciones P contenido de símbolos de (V' U T')
FIN

Ejemplo: Elimine los símbolos no alcanzables de la siguiente GLC G = ({S,A,B},{a},P,S), con las producciones:

S $\rightarrow$ Aa | a
A $\rightarrow$ a
B $\rightarrow$ SA

0- VN = {S}

1- VV ={S}
TV = Φ
VN = {S, A}
TN = {a}

2- VV = {S, A}
TV = {a}
VN = {S, A}
TN = {a}

V' = {S, A}
T' = {a}
P' = $\begin{cases} S \rightarrow Aa \mid a \\ A \rightarrow a \end{cases}$

Teorema: Si eliminamos
1ª los símbolos que NO son generadores $X \Rightarrow^* w$, para algún w en T*
2ª los símbolos que NO son alcanzables $S \Rightarrow \beta X \mu$ para algún β y μ en (V U T)*
Solo quedan los símbolos útiles de la GLC.

Formas Normales de las GLC

Forma Normal de Chomsky (FNC) (Chomsky, 1959)

Una gramática GLC, esta en la FNC si cada una de sus producciones es de los tipos siguientes:

$A \rightarrow BC$
$A \rightarrow a$

Donde A, B y C son variables (en V) y a es un símbolo Terminal (en T)

Teorema: Cualquier GLC sin λ-producciones puede ser transformada a una gramática equivalente en donde las producciones son de de la forma $A \rightarrow BC$ o $A \rightarrow a$

Antes de aplicar el algoritmo hay que eliminar:
1- producciones λ
2- producciones unitarias
3- símbolos inútiles.

ALGORITMO para llevar a la Forma Nornal de Chomsky
INICIO
Sea $A \rightarrow X_1 X_2 \ldots X_m$ en P
1- Crear producciones del tipo $A \rightarrow a$ adecuadamente
Si X_i es Terminal y $X_i = a$ entonces agregamos a P' la producción $C_a \rightarrow a$ y remplazamos X_i por C_a. Entonces todas las producciones son de forma:
$A \rightarrow B_1 B_2 \ldots B_m$ y $A \rightarrow a$

2- Cuando creamos producciones $A \rightarrow BC$ adecuadamente
Para los casos en que $m \geq 3$, hacemos:
$A \rightarrow B_1 D_1$; $D_1 \rightarrow B_2 D_2$;…, $D_{m-2} \rightarrow B_{m-1} B_m$

Agregamos las respectivas variables y producciones
$V' = V \cup \{ D_1, D_2 \ldots D_{m-2} \}$
FIN

Ejemplo: Dada la GLC G = ({S,A,B},{a,b},P,S), obtener su FNC

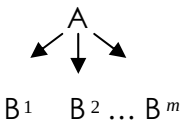
$$P \quad \begin{cases} S \rightarrow bA \mid aB \\ A \rightarrow bAA \mid aS \mid a \\ B \rightarrow aBB \mid bS \mid B \end{cases}$$

(1) $A \rightarrow a$
$S \rightarrow C_bA \mid C_aB$
$A \rightarrow C_bAA \mid C_aS \mid a$
$B \rightarrow C_aBB \mid C_bS \mid b$
$C_a \rightarrow a$
$C_b \rightarrow b$

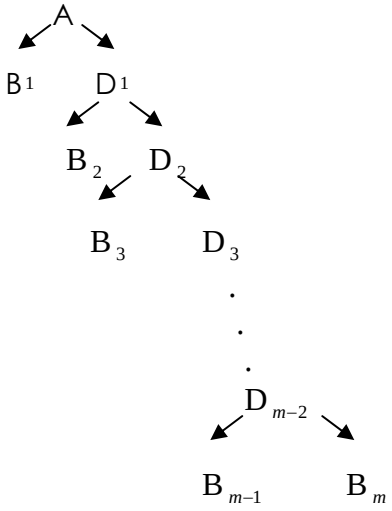
(2) $S \rightarrow C_bA \mid C_aB$
$A \rightarrow C_bD_1 \mid C_aS \mid a$
$B \rightarrow C_aD_2 \mid C_bS \mid b$
$D_1 \rightarrow AA$
$D_2 \rightarrow BB$
$C_a \rightarrow a$
$C_b \rightarrow b$

FNC no parece tener aplicaciones importantes en lingüística natural, aunque si tiene aplicación como una forma eficiente de comprobar si una cadena pertenece a un LLC. Los árboles de derivación son binarios.

$A \rightarrow B^1 B^2 \ldots B^m \Rightarrow$



$A \rightarrow B^1 D^1; D^1 \rightarrow B^2 D^2;$
$D^2 \rightarrow B^3 D^3; \ldots D^{m-2} \rightarrow B^{m-1} B^{m-2} \Rightarrow$



Forma Normal de Greybach (FNG) (Sheila Greibach, 1965)

Teorema: todo LLC sin λ-producciones pueden ser generado por una GCL en donde las producciones son de forma:

$A \rightarrow a\beta$, donde a en T y β esta en V*

El cuerpo de las producciones comienzan con un Terminal y puede seguirla cualquier cantidad de variables (inclusive 0 variables)

Su importancia radica en que, la FNG genera símbolos terminales por cada forma sentencial (==>), entonces una cadena de longitud n tiene exactamente n pasos de derivación en esta GLC, su derivación depende directamente del tamaño de la entrada (obtener una GLC en la FNG es costoso computacionalmente). Además una forma normal de Greybach permite generar Autómatas de Pila sin reglas λ (es decir, transiciones espontáneas).

PRECONDICION: Se comienza con una gramática en FNC.

Notas: Para aplicar el algoritmo cada variable se etiqueta en un conjunto ordenado de variables (con subíndices en secuencia). Por ejemplo, Si V = {S, A, B}, entonces V = {A_1, A_2, A_3}. La cardinalidad de V es llamada m, es decir m es el número de variables de la GLC.
Truco: Se busca enumerar con el valor más alto (K = 3) a una variable con alguna producción de la forma A → a

ALGORITMO para llevar a la Forma Normal de Greybach (FNG).

INICIO
- Paso 1: Convertir todas las producciones de una variable en la FNG

PARA K= m a 1 HACER
PARA J=1 a K-1 HACER
PARA Cada producción de la forma $A_k \rightarrow A_j \alpha$ HACER
PARA Todas las producciones $A_j \rightarrow \beta$ HACER
Agregar producciones $A_k \rightarrow \beta \alpha$
FINPARA
FINPARA
Remover producción $A_k \rightarrow A_j \alpha$
FINPARA
SI existe $A_k \rightarrow A_k \alpha$ ENTONCES
PARA Cada producción de la forma $A_k \rightarrow A_k \alpha$ HACER
Agregar producción $B_k \rightarrow \alpha$ y $B_k \rightarrow \alpha B_k$
FINPARA
Remover producción $A_k \rightarrow A_k \alpha$
PARA Cada producción $A_k \rightarrow \beta$ donde β no comienza con A_k HACER
Agregar la producción $A_k \rightarrow \beta B_k$
FINPARA
FINSI
FINPARA

- Paso 2: Convertir todas las producciones: Las reglas de la forma $A_j \rightarrow a\alpha$ se sustituye en las reglas de P hasta que todas sean FNG.

FIN

Ejemplo: Dada la GLC G siguiente se quiere obtener una GLC equivalente en la FNG:

$G = (\{A_1, A_2, A_3\}, \{a, b\}, P, A_1)$

$P = A_1 \rightarrow A_2A_3$
$A_2 \rightarrow A_3A_1 \mid b$
$A_3 \rightarrow A_1A_2 \mid a$

Paso 1: K = 3 , la variable A_3

J = 1

$A_3 \rightarrow A_2A_3A_2 \mid a$
$A_2 \rightarrow A_3A_1 \mid b$
$A_1 \rightarrow A_2\ A_3$

J = 2

$A_1 \rightarrow A_2A_3$
$A_2 \rightarrow A_3A_1 \mid b$
$A_3 \rightarrow \underbrace{A_3A_1A_3A_2}_{\alpha} \mid bA_3A_2 \mid a$

$A_1 \rightarrow A_2A_3$
$A_2 \rightarrow A_3A_1 \mid b$
$A_3 \rightarrow bA_3A_2 \mid a$
$B_3 \rightarrow A_1A_3A_2 \mid A_1A_3A_2B_3$

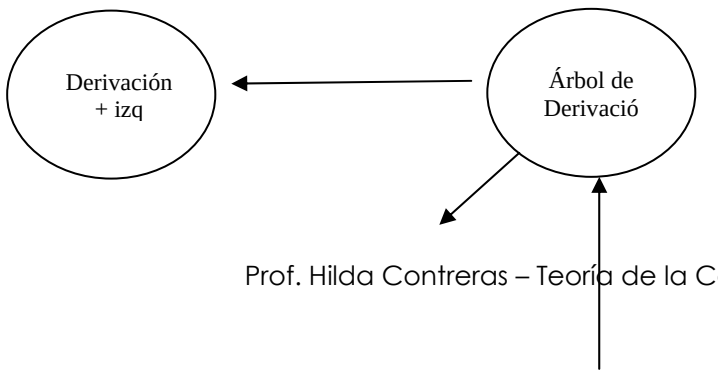
$A_1 \rightarrow A_2A_3$
$A_2 \rightarrow A_3A_1 \mid b$
$A_3 \rightarrow bA_3A_2 \mid a \mid bA_3A_2B_3 \mid aB_3$
$B_3 \rightarrow A_1A_3A_2 \mid A_1A_3A_2B_3$

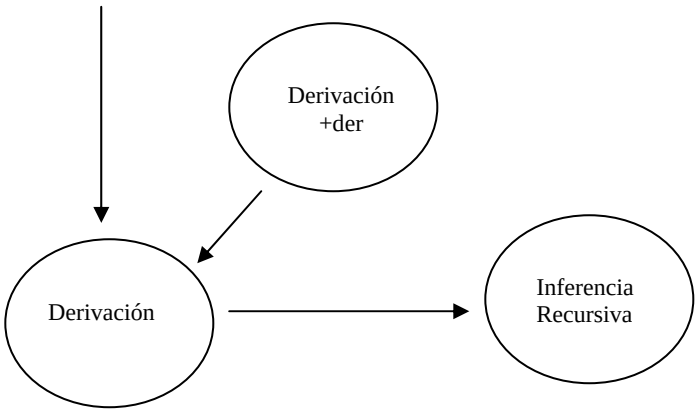
Paso 2:

$A_3 \rightarrow bA_3A_2 \mid a \mid bA_3A_2B_3 \mid aB_3$
o sustituir A_3 en A_2
$A_2 \rightarrow bA_3A_2A_1 \mid a\ A_1 \mid bA_3A_2B_3A_1 \mid aB_3A_1 \mid b$
o sustituir A_2 en A_1
o sustituir A_1 en B_3

La GLC queda con todas sus producciones en la FNG

Relación entre GLC

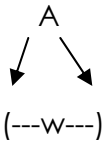




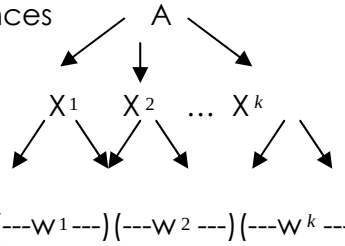
A = GLC = (V, T, P, S)

Si w esta en L(A) entonces existe un árbol de derivación con raíz A y resultado w.

Si $A \rightarrow w$ entonces



Si $A \Rightarrow w$ entonces



$A \rightarrow X_1 X_2 \ldots X_k$

$X_i \overset{*}{\Rightarrow} w_i \quad i=1 \ldots k$

Ejercicios:

1) Encuentre la GLC para

a.- $\{a^i b^j , i \geq j\}$
b.- $\{a^i b^j c^k , i = j+k\}$
c.- $\{ww^k , k>0, w \text{ en } (0+1)^*\}$
d.- $\{ w \mid w \text{ en } (011 + 1)^*(01)^* \}$

2) Si G es una GLC en FNC y w en L (G) con |w| = K ¿Cuál es el numero de pasos de una derivación de w en G?

3) Genere una gramática no ambigua para la GLC siguiente: $S \rightarrow aS \mid aSbS \mid c$

4) Realice una GLC para las instrucciones if, then, else y su anidamiento en el lenguaje C.

5) Encuentre la FNC para las siguientes GLC:

a) S → AaA | CA | BaB
A → aaBa | CDA | aa | DC
B → bB | bBA | bb | aS
C → Ca | bC | D
D → bD | λ

b) S → ASB | λ
A → aAS | a

c) S →AB | ABC
A → BA | BC | λ | a
B →AC | CB | λ | b
C → BC | AB | c

6) Realice una GLC para las instrucciones aritméticas con paréntesis equilibrados en el lenguaje C:

a = (a+b)*c;

7) Defina una GLC (Variables, Terminales, Producciones y Variable inicial) para generar todas las posibles páginas Web sencillas, según la siguiente tabla:

Etiquetas	
<HTML>	
<HEAD> </HEAD>	<TITLE></TITLE>
<BODY> </BODY>	 <HR> <P></P> <B></B> <FONT></FONT>
</HTML>	

Por ejemplo la siguiente cadena pertenece al lenguaje de la páginas Web sencillas:

```
<HTML>
  <HEAD>
    <TITLE>Tercer Parcial</TITLE>
  </HEAD>
  <BODY>
    <P> Teoría de la <B>Computación </B> – Ingeniería
    <HR>
    </P>
<HTML>
```

Utilice solo las etiquetas en mayúscula que se muestran en la tabla izquierda. Represente el contenido de la página con el conjunto alfabeto ALF = {A …Z} U {a …z).

8) Considere el alfabeto Σ = { a, b, (,), |, * , Φ } y construya una GLC que genere todas las expresiones regulares válidas sobre {a,b}.

9) La siguiente GLC genera un lenguaje regular pese a no serlo ella misma. (a) Encuentre una Gramática Regular que genere el lenguaje generado por:

G = ({S}, {a,b}, {S → SSS | a | ab},S)

(b) Dé la expresión regular del lenguaje L(G)

(c) Genere una cadena del lenguaje L(G) de tamaño mayor que 3 y muestre su árbol de derivación a partir de la GLC.

10) Escriba una GLC G que genere "todas" las posibles Gramáticas libres de contexto sobre el alfabeto {a,b} y que usan las variables S,A y B. Cada posible GLC es una secuencia de producciones separadas por comas, entre paréntesis y usando el símbolo igual "=" para indicar el símbolos de producción "→". Por ejemplo, una cadena generada por G es (S=ASB,A=a,B=b,S=λ). Indique claramente cuales son los símbolos terminales de G. Nota: Asuma que G puede generar reglas repetidas, es decir no necesita validar que la producción ya fue derivada anteriormente, por tanto la cadena (S=Aa,A=b,S=Aa) es igual a la cadena (S=Aa,A=b).

11) Dadas las etiquetas <?xml version="1.0"?>, <a1>, <a2>, <a3>, </a1>, </a2> y </a3>, escriba una GLC G que genere "todas" los posibles archivos XML "bien formados" con dichas etiquetas. Un archivo XML está bien formado si: (a) Tiene un solo elemento raíz, (b) La ultima etiqueta que abre es la primera que cierra, (c) El texto de los elementos no contiene ninguna de las cinco entidades &, <, >, ', ", excepto si esta encerrado entre un CDATA (abre con <![CDATA[y cierra con]>).(d) Los comentarios abren con <!—y cierran con -->. Nota: no considere la ocurrencia de atributos en los elementos o etiquetas. Indique claramente los terminales y variables de su GLC

12) Encuentre una GLC equivalente en la Forma Normal de Greibach (FNG) de la siguiente GLC G = ({S,A,B,C},{0,1},P,S), Justifique sus pasos:
S → 0A0 | 1B1 | BB
A → C
B → S | A
C → S | λ

13) Dada la siguiente GLC G encuentre una GLC G` en la FNG (Forma Normal de Greibach):
G = (V, T, P, S)

V = { S, A, B }
T = { a , b }

P = S → ABa | A
A → aA | λ
B → bB | λ