Tema 3: Gramáticas Formales

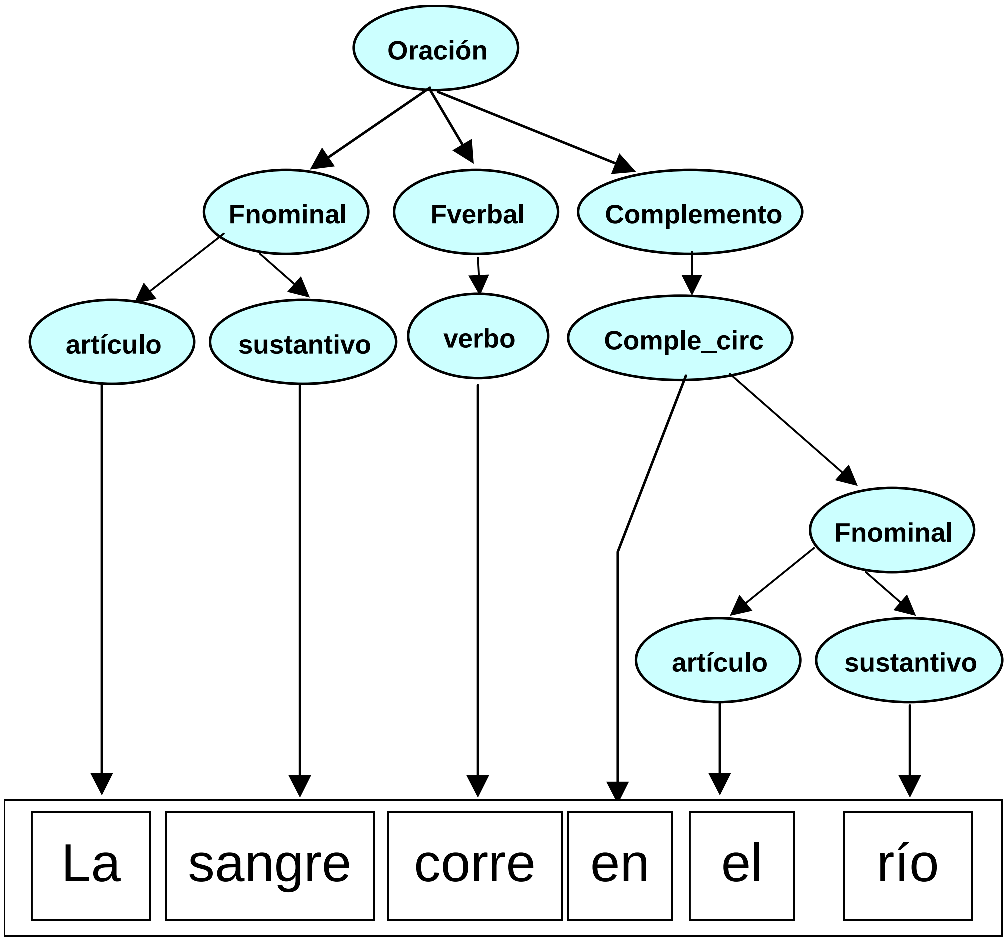
Informática Teórica I

Teoría de Gramáticas Formales. Bibliografía

- M. Alfonseca, J. Sancho y M. Martínez. “Teoría de Lenguajes, Gramáticas y Autómatas”, R.A.E.C., Madrid, (1998).
- P. Isasi, P. Martínez y D. Borrajo. “Lenguajes, Gramáticas y Autómatas, un Enfoque Práctico”. Addison-Wesley, (1997).
- John E. Hopcroft y Jefrey D. Ullman. “Introduction to Automata Theory, Languages and Computation”. Addison-Wesley, (1979).
- F.J. Sanchis y C. Galán. “Compiladores: teoría y construcción”. Ed Paraninfo, (1986).

Gramáticas Formales. Introducción

Una Gramática del castellano como diagrama sintáctico



Gramáticas Formales. Introducción

Una Gramática del castellano en notación de Backus

<Oración>::= <Fnominal> <Fverbal> | <Fnominal> <Fverbal><Complemento>

<Fnominal>::= <Sustantivo> | <NomPr> | <Artículo> <Sustantivo>

| <Artículo> <Sustantivo> <Adjetivo>

| <Artículo> <Sustantivo> <Adjetivo>

| < Fnominal >de < Fnominal >

<Fverbal>::= <Verb> | <Verb> <Adverbio>

<Complemento>::=<ComDir> |<ComIn> |<ComCir> | <ComDir> <ComIn> <ComCir>

<ComDir>::= <Fnominal> , <ComIn>::= a <Fnominal> | para <Fnominal> | ....

<ComCir> ::= en <Fnominal> | desde <Fnominal> | cuando <Fnominal> | ....

Donde: <Sustantivo>, <Adjetivo>, <Adverbio>, <Artículo>, <NomPr>, <Verbo>, etc toman como valores palabras propias de estas categorías gramaticales

Gramáticas Formales. Definición

"ente formal" para especificar de manera finita el conjunto de cadenas de símbolos que constituyen un lenguaje

- es una cuádrupla: **(Σ_T, Σ_N, S, P),** Σ_T y Σ_N son alfabetos:

  Σ_T: alfabeto de símbolos terminales. Todas las cadenas del lenguaje representado por la G (L(G)) están formadas con símbolos de este alfabeto.

  Σ_N: Conjunto de símbolos auxiliares introducidos como elementos auxiliares para la definición de G pero que no figuran en las cadenas de L(G).

  S: axioma o símbolo destacado. Es un símbolo NT a partir del que se comienzan a aplicar las reglas de P.

  P: conjunto de reglas de producción: **u::=v** donde $u \in \Sigma^+$ y $v \in \Sigma^*$ $u = xAy$ tal que $x, y \in \Sigma^*$ y $A \in \Sigma_N$.

Gramáticas Formales. Definición

- **Gramática formal:**

  Se cumple entre los alfabetos de G:

  $\Sigma_T \cup \Sigma_N$ = Alfabeto o vocabulario de G

  $\Sigma_T \cap \Sigma_N = \phi$ (son disjuntos)

- Notación de Backus: Si u::=v y u::= w son dos reglas de producción de P, entonces se puede escribir u ::= v | w

  Se denomina notación BNF: Forma Normal de Backus o Forma Normal de Backus-Naur

  Ejemplo: sea G = ({0,1,2,3,4,5,6,7,8,9}, {N,C}, N,P)

  P = {N::= CN | C

  C::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}

Gramáticas Formales. Definiciones

- **Forma sentencial:**

  Sea una *G*, sea $x \in \Sigma^*$

  x se denomina **forma sentencial de G** si se verifica:

  $$\mathbf{S} \overset{*}{\rightarrow} \mathbf{x}$$

  es decir, que existe una relación de Thue entre el axioma y *x*.

- Si $\mathbf{x} \in \Sigma_T$ se dice que **x** es una **Sentencia o instrucción del lenguaje descrito por G**

Gramáticas Formales. Definiciones

- **Concepto de Lenguaje:** asociado a una gramática

Sea una G

Se llama $\left\{ \begin{array}{l} \text{lenguaje asociado a esa G:} \\ \text{lenguaje generado por esa G} \\ \text{lenguaje descrito por esa G} \end{array} \right\}$ $\boxed{\textbf{a } \mathbf{L(G) = \{x \,/\, S \stackrel{*}{\rightarrow} x \text{ AND } x \in \Sigma_T^*\}}}$

Es decir, al conjunto de todas las sentencias de la Gramática

Gramáticas Formales. Definiciones

- Decir cuál es el lenguaje descrito por la G del ejemplo:

  sea G = ({0,1,2,3,4,5,6,7,8,9}, {N,C}, N,P)

  P = {N::= NC | C, C::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}

Gramáticas Formales. Definiciones: Gramáticas equivalentes

- **Gramáticas equivalentes:**

Dos gramáticas G1 y G2 son equivalentes si describen /generan el mismo lenguaje:

$$\boxed{G1 \approx G2}$$

Si

$$\boxed{L(G1) = L(G2)}$$

Gramáticas Formales. Definiciones: Frases y asideros

- Sea G
- Sea $v = xuy$ una forma sentencial de G ($S \overset{*}{\rightarrow} v$)
- Se dice que u es una **frase de la forma sentencial** v respecto del símbolo no terminal U ($U \in \Sigma_{NT}$) si se cumple:

  $$S \overset{*}{\rightarrow} x\,U\,y$$

  $$U \overset{+}{\rightarrow} u \qquad \textbf{derivación de longitud n}$$

- Si U es una forma sentencial de G, entonces todas las frases que derivan de U serán a su vez formas sentenciales de G
- u es **frase simple de** v si se cumple:

  $$S \overset{*}{\rightarrow} x\,U\,y$$

  $$U \rightarrow u \qquad \textbf{derivación directa}$$

- Se llama **asidero, handle o pivote** a la frase simple más a la izquierda de la forma sentencial

Gramáticas Formales. Definiciones: Frases y asideros. Ejemplo

- **Ejercicio alfonseca pg 50:**

En la gramática que describe los números enteros positivos, demostrar que N no es una frase de N1. Encontrar todas las frases de N1. ¿Cuáles son frases simples? ¿Cuál es el asidero?

G = ({0,1,2,3,4,5,6,7,8,9}, {N,C}, N,P)

P = {N::= NC │C, C::= 0│1│2│3│4│5│6│7│8│9}

¿y en la forma sentencial 2C0?

Gramáticas Formales. Definiciones: Frases y asideros. Ejemplo

- **Ejercicio alfonseca pg 50:**

v = xuy = N1

N1 N es la frase

N**1** 1 es la frase

N1 N1 es la frase

Gramáticas Formales. Definiciones: Recursividad

- Sea G
- Una G se llama **recursiva en *U***, $U \in \Sigma_{NT}$, si se cumple:

$$\boldsymbol{U \xrightarrow{+} x\, U\, y}$$

- Si $x = \lambda$ $(U \xrightarrow{+} U\, y)$ se dice que G es **recursiva a izquierdas**
- Si $y = \lambda$ $(U \xrightarrow{+} xU)$ se dice que G es **recursiva a derechas**
- Una regla de producción es recursiva si tiene la forma:

$$\boldsymbol{U ::= x\, U\, y}$$

- Si un lenguaje es infinito, la gramática que lo representa tiene que ser recursiva

Ejercicios: alfonseca página 50 ejercicios 1, 2 y 3

Jerarquía de Chomsky

$G= (\Sigma_T, \Sigma_N, S, P)$
$\Sigma_T \cup \Sigma_N = \Sigma$, alfabeto gramática, $\Sigma_T \cap \Sigma_N = \phi$

Tipos: según la forma de sus reglas de derivación
$G3 \subset G2 \subset G1 \subset G0$ Jerárquica restricciones en su conjunto P.

G0
No Restringidas
o con Estructura de Frase

u ::= v $u \in \Sigma^+$
$v \in \Sigma^*$

Única restricción: **λ::= v $\notin$P**

Forma sentencial:
$u = xAy$, $x,y \in \Sigma^*$, $A \in \Sigma_N$

Ejemplo: G= {(0,1), (S), S, P}, donde:
P={(S→000S111), (0S1→01)}

Con Estructura De Frases:
(xAy::=xvy) $\in$ P, donde:
$x,y \in \Sigma^*$, $A \in \Sigma_N$, $u \in \Sigma^+$
En xAy ::= xvy cuando v=λ,
xAy ::= xy reglas compresoras

- Los lenguajes que son representados por G0 se llaman lenguajes sin restricciones
- Chomsky 1959: todo L(G0) puede ser descrito por una G0 con estructura de frases. Ejemplo:

G= ({a,b},{A,B.C},A,P)	G'= {a,b},{A,B.C,X,Y},A,P')
P = {A::= aABC / abC	P' = {A::= aABC / abC
CB::= BC	CB::=XB, XB::=XY
	XY::=BY, BY::=BC
bB::=bb	bB::=bb
bC::=b}	bC::=b}

Jerarquía de Chomsky

$G= (\Sigma_T, \Sigma_N, S, P)$
$\Sigma_T \cup \Sigma_N = \Sigma$, alfabeto gramática
$\Sigma_T \cap \Sigma_N = \phi$

Tipos: según la forma de sus reglas de derivación
$G3 \subset G2 \subset G1 \subset G0$ Jerárquica restricciones en su conjunto P.

G1
Sensibles al Contexto

xAy ::= xvy

$x,y \in \Sigma^*$, $A \in \Sigma_N$
$v \in \Sigma^+$ No permite reglas compresoras
Excepción: **(S::= λ)$\in$P**

Ejemplo:
G= {(0,1,2,3,4,5,6,7,8,9,), (<número>, <dígito>),<número>, P},
P={(<número> → <dígito><número>,
<número> → <dígito>,
<dígito>→0|1|2|3|4|5|6|7|8|9)}

Sensibles al contexto:
se puede cambiar **A** por **v**, siempre en el contexto **x...y**

- Los lenguajes que son representados por G1 se llaman **lenguajes sensibles al contexto**
- $\lambda \in L(G1)$ Sii $(S::=\lambda) \in P$
- Ejemplo de G que no es G1:

G= ({a,b}, {S}, S, P)

P= {S::= aaaaSbbbb, aSb::= ab}

- Ejemplo de G que si es G1:

G= ({a,b}, {S}, S, P)

P= {S::= aaaaSbbbb, aSb::= abb}

Jerarquía de Chomsky

- **Propiedad de NO DECRECIMIENTO de las G1**:
  - Las cadenas que se obtienen en cualquier derivación de una G1 son de longitud no decreciente, es decir:

    $$\mathbf{u} \rightarrow \mathbf{v} \Rightarrow |\mathbf{v}| \geq |\mathbf{u}|$$

    la longitud de la parte decha de la producción es mayor o igual a la longitud de la parte izquierda

    - Demostración:

      $\alpha A\beta \rightarrow \alpha\gamma\beta$

      donde $\gamma\in(\Sigma_T\cup\Sigma_{NT})^+$ por definición de G1 (no compresoras)

      es decir, $\gamma \neq \lambda$ siempre, lo que implica $|\gamma| \geq 1$

      y como $|A| = 1$ como mínimo, queda demostrado

Jerarquía de Chomsky

$G= (\Sigma_T, \Sigma_N, S, P)$
$\Sigma_T \cup \Sigma_N = \Sigma$, alfabeto gramática
$\Sigma_T \cap \Sigma_N = \phi$

Tipos: según la forma de sus reglas de derivación
G3⊂G2⊂G1⊂G0 Jerárquica restricciones en su conjunto P.

G2
de Contexto Libre

- Los lenguajes que son representados por G2 se llaman **lenguajes no sensibles al contexto o de contexto libre**
- $\forall$ L(G2) $\exists$ l(G2') sin reglas A::= λ (A $\neq$ S) Si $\lambda \in$ L(G2). Ver algoritmo eliminación reglas NO generativas.

A ::= v

$v \in \Sigma^*$
$A \in \Sigma_N$ puede aparecer A::= λ

r $\in$ P se caracterizan por tener un sólo símbolo NT en su parte izquierda

(S::= λ) $\in$P y **(A::= λ) $\notin$P**
(algoritmo limpieza reglas NO generativas)

Ejemplo:
G= {(a,b), (S,A),S, P}
P={(S $\to$ aS, S $\to$ aA, A$\to$ bA, A $\to$ b)}

Contexto Libre: se puede cambiar **A** por **v**, en cualquier contexto

Jerarquía de Chomsky

$G = (\Sigma_T, \Sigma_N, S, P)$
$\Sigma_T \cup \Sigma_N = \Sigma$, alfabeto gramática
$\Sigma_T \cap \Sigma_N = \phi$

Tipos: según la forma de sus reglas de derivación
$G3 \subset G2 \subset G1 \subset G0$ Jerárquica restricciones en su conjunto P.

G3
Gramáticas Regulares

G3 Lineales por la Izda.	**G3 Lineales por la Dcha.**
A::= a	**A::= a**
A::= Va	**A::= aV**
S::= λ	**S::= λ**

$a \in \Sigma_T$
$A,V \in \Sigma_N$ S es axioma

r $\in$ P un sólo símbolo NT en su parte izda y su parte dcha comienza por un T seguido o no de NT (al revés en lineal derecha)

Ejemplo:
G= {(a,b), (S,A),S, P},
P={(S $\rightarrow$ aS, S $\rightarrow$ aA, A$\rightarrow$ bA, A $\rightarrow$ b)}

- Los lenguajes que son representados por G3 se llaman **lenguajes regulares**
- $\forall$ L(G3) $\exists$ l(G3') sin reglas A::= λ (A $\neq$ S) Si λ $\in$L(G3). Ver algoritmo eliminación reglas NO generativas.

Jerarquía de Chomsky. Ejemplos I

- **Ejemplos de Gramáticas:** Isasi, Martínez y Borrajo, páginas 16 a 19

  G_1= ({0,1}, {A,B,S},S, P), P={S::= A0, A0::= 1B1, 1A::= 0B0, B::=λ / 1 / 0}

  G_2= ({0,1}, {A,B},A, P), P={A::= 1B1 / 11, 1B1::= 101 / 111}

  G_3= ({0,1}, {A,B},A, P), P={A::= 1B1 / 11, B::= 0 / 1}

  G_4= ({0,1}, {A,B, C},A, P), P={A::= 1B, B::=1/ 0C / 1C, C::= 1)}

Jerarquía de Chomsky. Ejemplos I

- **Ejemplos de Gramáticas:** Isasi, Martínez y Borrajo, páginas 16 a 19

G_1= ({0,1}, {A,B,S},S, P), P={S::= A0, A0::= 1B1, 1A::= 0B0, B::=λ / 1 / 0}

B::=λ es una regla compresora
A0::= 1B1 y 1A::= 0B0 no guardan el contexto
Se trata de G0 sin restricciones
L = {11, 101, 111}
¿Necesita un lenguaje tan sencillo de una gramática del mayor nivel en la jerarquía de Chomsky?

Jerarquía de Chomsky. Ejemplos I

- **Ejemplos de Gramáticas:** Isasi, Martínez y Borrajo, páginas 16 a 19

G_2= ({0,1}, {A,B},A, P), P={A::= 1B1 / 11, 1B1::= 101 / 111}

No hay reglas compresoras

1B1::= 101 / 111 guardan el contexto

Se trata de G1 con estructura de frases

L = {11, 101, 111}

¿Necesita un lenguaje tan sencillo de una gramática del mayor nivel en la jerarquía de Chomsky?

Jerarquía de Chomsky. Ejemplos I

- **Ejemplos de Gramáticas:** Isasi, Martínez y Borrajo, páginas 16 a 19

  G_3= ({0,1}, {A,B},A, P), P={A::= 1B1 / 11, B::= 0 / 1}

No hay reglas compresoras

Un solo símbolo a la izda

Libertad a la dcha ⇒ G independiente del contexto G2

L = {11, 101, 111}

¿Necesita un lenguaje tan sencillo de una gramática alto nivel en la jerarquía de Chomsky?

Jerarquía de Chomsky. Ejemplos I

- **Ejemplos de Gramáticas:** Isasi, Martínez y Borrajo, páginas 16 a 19

G_4= ({0,1}, {A,B, C},A, P), P={A::= 1B, B::= 1/ 0C / 1C, C::= 1)}

No hay reglas compresoras

Un solo símbolo a la izda

Un solo símbolo a la izda

Sin libertad a la dcha → G regular lineal derecha G3 LD

L = {11, 101, 111}

¿Necesita un lenguaje tan sencillo de una gramática alto nivel en la jerarquía de Chomsky? NO, es una G3 la que habría que construir

Jerarquía de Chomsky

- Un Lenguaje se denomina de tipo i (i= 0, 1, 2, 3) si existe una G tipo i tal que L= L(Gi)
- Jerarquía: relación de inclusión:

$$\boxed{G3 \subset G2 \subset G1 \subset G0}$$

Gramáticas Formales. Gramáticas Regulares

GRAMÁTICAS EQUIVALENTES

1. **∀ G3 lineal derecha con reglas del tipo A::= aS**

∃ una G3' lineal derecha equivalente sin reglas del tipo A::= aS

2. **∀ G3 lineal derecha**

∃ una G3' lineal izquierda equivalente

Gramáticas Formales. Gramáticas Regulares
Gramáticas Equivalentes

1. $\forall$ G3 lineal derecha con reglas del tipo A::= aS

$\exists$ una G3' lineal derecha equivalente sin reglas del tipo A::= aS

Procedimiento de transformación

1 se añade un nuevo símbolo en el alfabeto Σ_N

2 $\forall$ S::= x, donde $x \in \Sigma^+$, se añade una regla B::= x

3 Se transforman las reglas A::= aS (que desaparecen) en reglas del tipo A::= aB.

4 Las reglas tipo S::= λ no se ven afectadas por este algoritmo.

Gramáticas Formales. Gramáticas Regulares
Gramáticas Equivalentes

2. ∀ **G3 lineal derecha** ∃ **una G3' lineal izquierda equivalente** Procedimiento de transformación

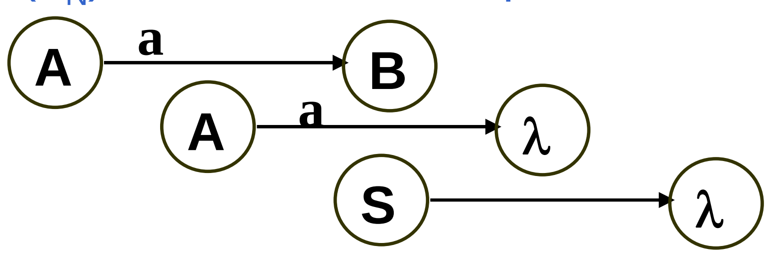
   1. **creación de un grafo dirigido:**

      a) número de nodos = $C(\Sigma_N)$ +1, cada nodo etiquetado con símbolos de Σ_N y otro con λ

      b) cada A::=aB ∈P

      c) cada A::=a ∈P

      d) si S::= λ ∈P



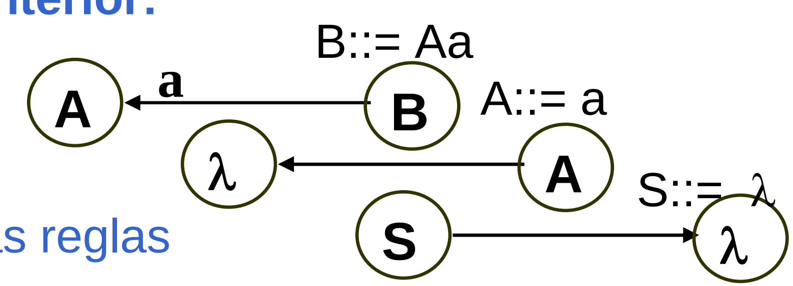
   2. **transformación del grafo dirigido anterior:**

   a) intercambiar etiquetas de S y λ

   b) invertir sentido de todos arcos

   c) deshacer el camino y generar las nuevas reglas

   ⇒ interpretar el grafo para obtener la G3LI equivalente



Gramáticas Formales. Gramáticas Regulares Gramáticas Equivalentes

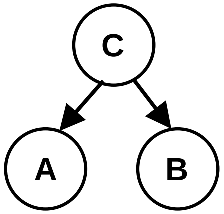
- Ejercicio: alfonseca pg 58: Sea la G3 LD, G1 = ({0,1}, {S,B}, S, P), donde

  P = {S::= 1B /1, B::= 0S}, hallar la G3 LI equivalente.

  Demostrar que el lenguaje descrito por ambas es el mismo.

Gramáticas Formales. Árboles de derivación

- A las derivaciones de las G tipo 1, 2 y 3 les corresponde un árbol equivalente, También llamado **árbol sintáctico o parse tree**
- Representa las producciones aplicadas durante la generación de una palabra, es decir, su estructura de acuerdo con la G
- Es un árbol ordenado y etiquetado que se construye:
  - La raíz se denota por el axioma de G
  - Una derivación directa se representa por un conjunto de ramas que salen de un nodo dado (parte izquierda de la P)
  - Aplicar una regla un símbolo de la parte izda queda sustituido por una palabra *u* de la parte dcha. Por cada uno de los símbolos de u se dibuja una rama que sale del NT a ese símbolo: pej sea u=AB y sea la P= C→u



El símbolo más a la izda en la P queda más a la izda en el árbol

Gramáticas Formales. Árboles de derivación

- En una G1, además, se debe conservar el contexto
- Para cada rama:
  - el nodo de partida se llama **padre** del nodo final
  - el nodo final es **hijo** del nodo padre
  - dos nodos hijos del mismo padre se llaman **hermanos**
  - un nodo es **ascendiente** de otro si es su padre o ascendiente de su padre
  - un nodo es **descendiente** de otro si es su hijo o descendiente de sus hijos
- A lo largo del proceso de construcción del árbol, los nodos finales de cada paso sucesivo, leídos de izda a dcha dan la **forma sentencial** obtenida por la derivación representada por el árbol.
- El conjunto de las hojas del árbol (nodos denotados por símbolos terminales o λ) leídos de izda a dcha nos dan la **sentencia** generada por la derivación

Gramáticas Formales. Árboles de derivación

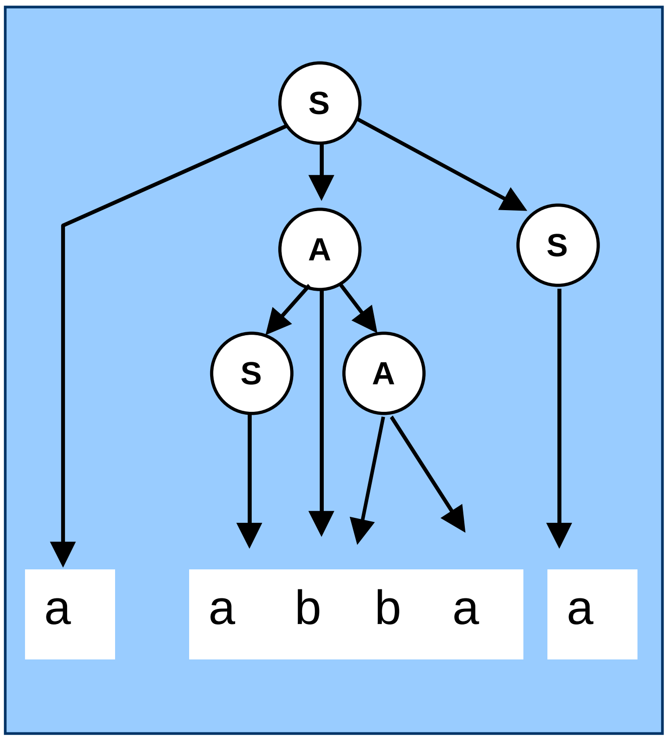
- Dada la G = ({a,b}, {A,S}, S, {S::= aAS / a, A::= SbA / SS / ba}).

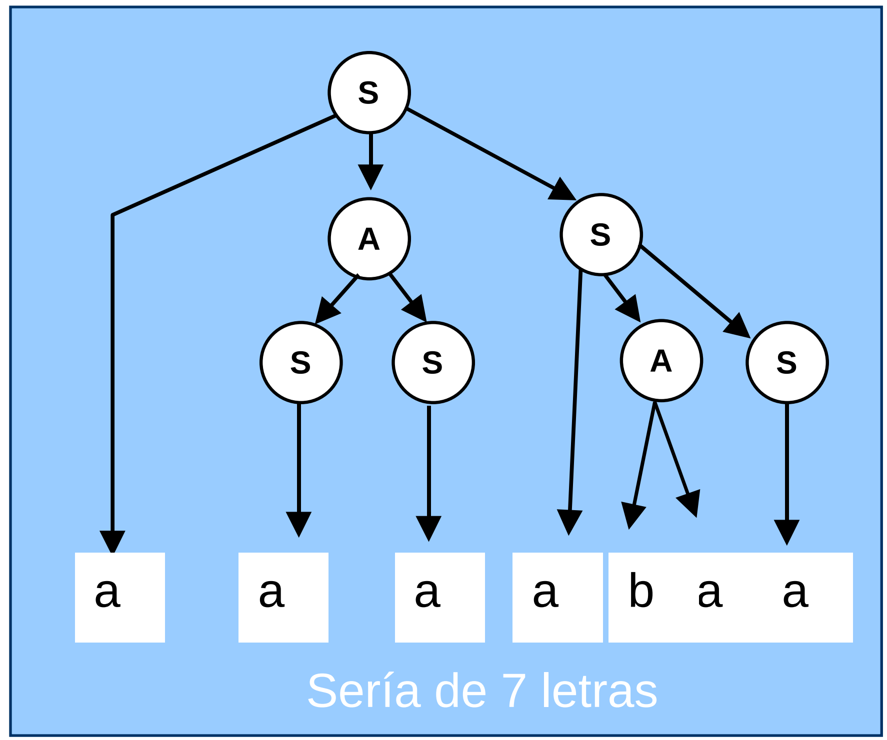
  Hallar un árbol de derivación para una sentencia de 6 letras.

Gramáticas Formales. Árboles de derivación

- Dada la G = ({a,b}, {A,S}, S, {S::= aAS / a, A::= SbA / SS / ba}).

  Hallar un árbol de derivación para una sentencia de 6 letras.

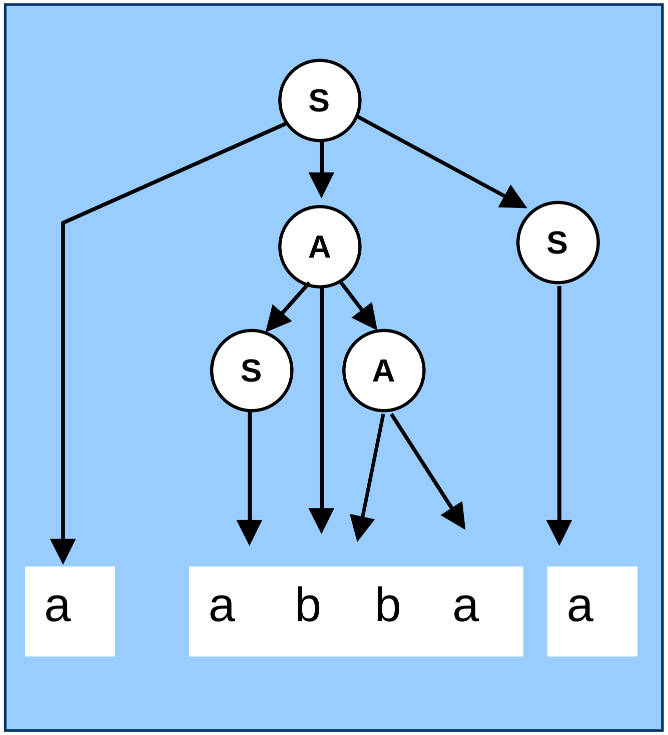


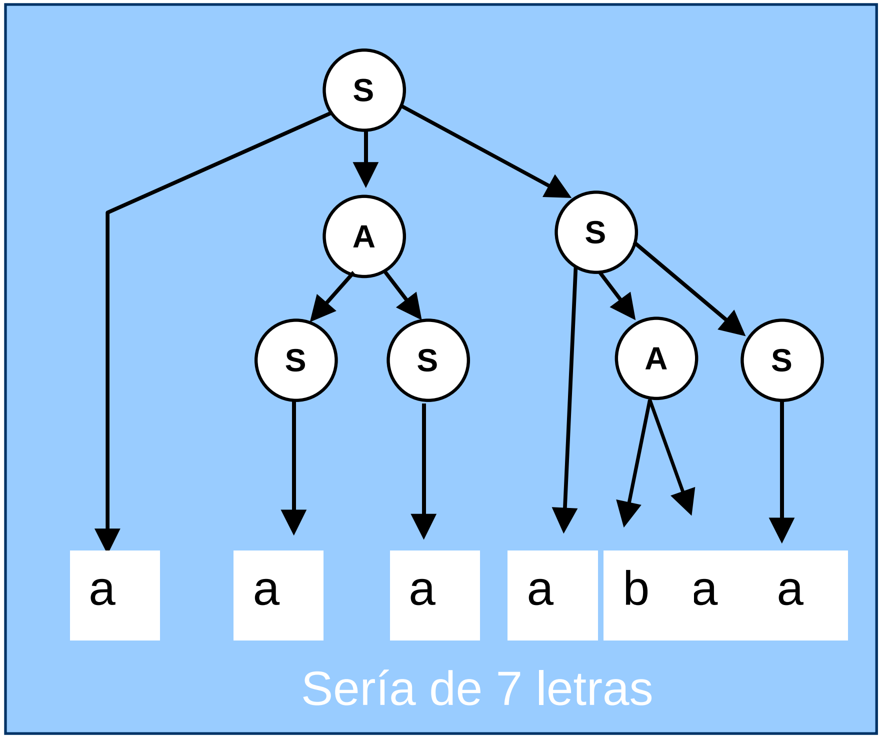


Gramáticas Formales. Árboles de derivación

- Dada la G = ({a,b}, {A,S}, S, {S::= aAS / a, A::= SbA / SS / ba}).

  Hallar un árbol de derivación para una sentencia de 6 letras.





Gramáticas Formales. Árboles de derivación

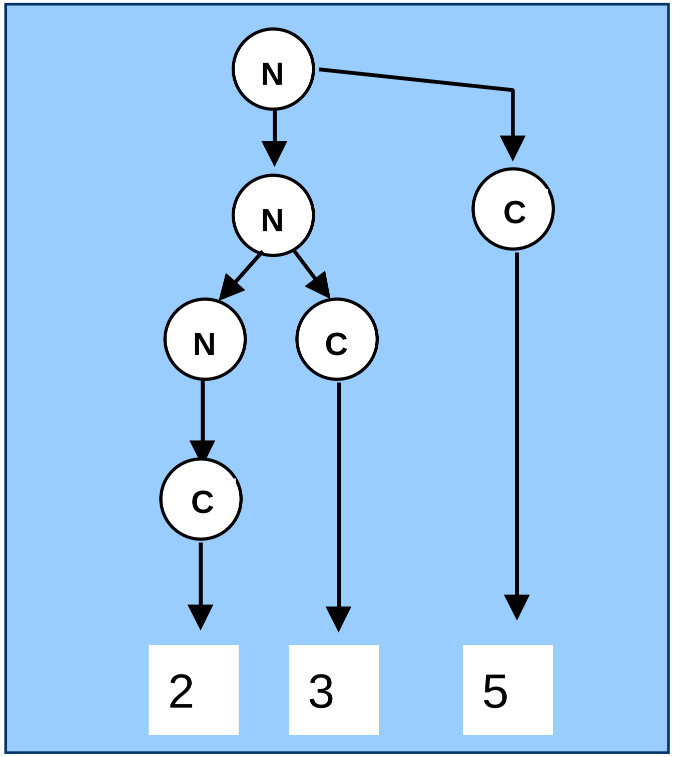
- **Dada la G = ({0,1,2,3,4,5,6,7,8,9}, {N,C}, N, {N ::= NC / C, C ::= 0 /.../ 9})**

  Hallar un árbol de derivación para N $\rightarrow$ 235

Gramáticas Formales. Árboles de derivación

- **Dada la G = ({0,1,2,3,4,5,6,7,8,9}, {N,C}, N, {N ::= NC / C, C ::= 0 /.../ 9})**

  Hallar un árbol de derivación para N $\rightarrow$ 235



Gramáticas Formales. Árboles de derivación

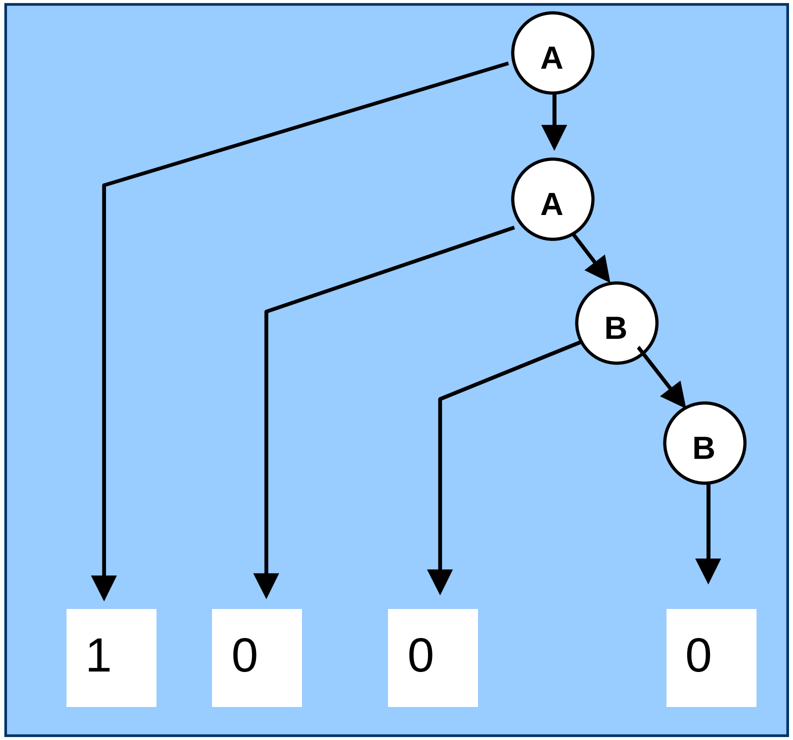
- **Dada la G = ({0,1}, {A,B}, A, {A::= 1A / OB, B::= 0B / 0}) Una de cuyas derivaciones válidas es: A $\rightarrow$ 1A $\rightarrow$ 1OB $\rightarrow$ 100B $\rightarrow$ 1000**

  Hallar un árbol de derivación para A $\rightarrow$ 1000

Gramáticas Formales. Árboles de derivación

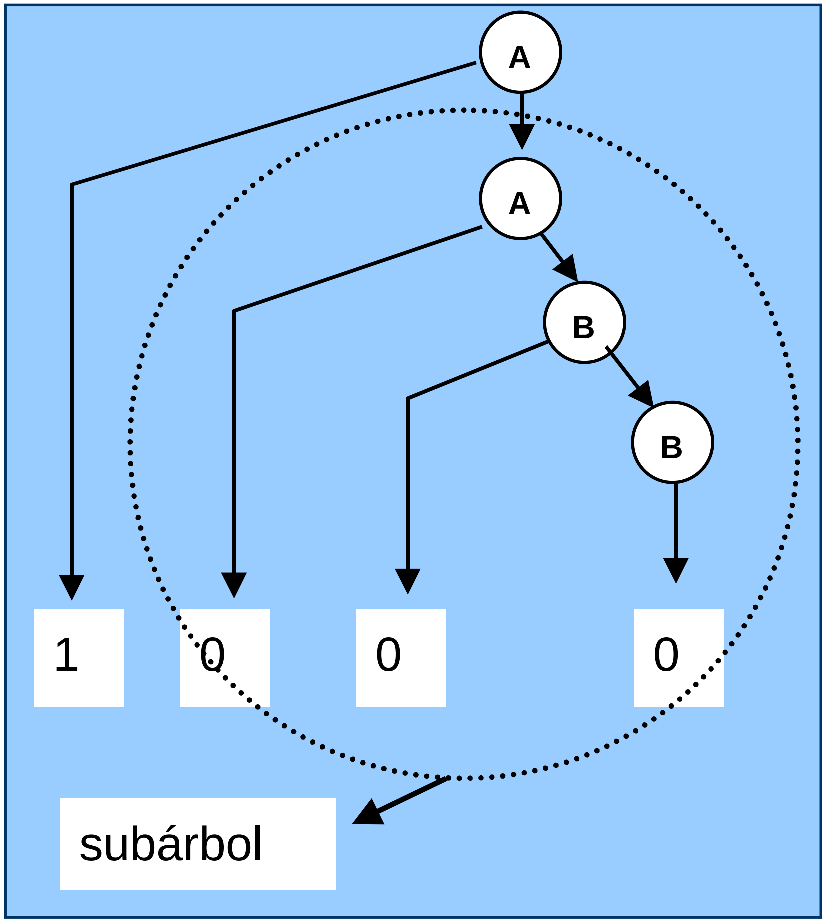
- **Dada la G = ({0,1}, {A,B}, A, {A::= 1A / OB, B::= 0B / 0}) Una de cuyas derivaciones válidas es: A → 1A → 1OB → 100B → 1000**

  Hallar un árbol de derivación para A → 1000



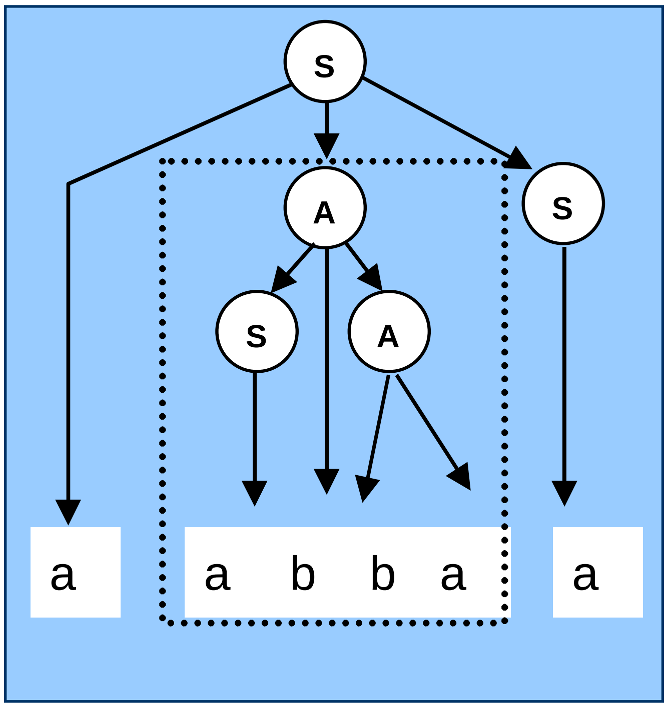
Gramáticas Formales. Subárboles de derivación

- Dado un árbol A correspondiente a una derivación, se llama **subárbol de A** al árbol cuya raíz es un nodo cualquiera de A, cuyos nodos son todos los descendientes de la raíz del subárbol en A y cuyas ramas son todas las que unen dichos nodos entre si en A.



Gramáticas Formales. Subárboles de derivación

- **Teorema**: las hojas de un subárbol, leídas de izda a dcha, forman una frase respecto al símbolo NT raíz del subárbol



abba es una frase de la forma sentencial aabbaa respecto del símbolo A

Gramáticas Formales. Ambigüedad

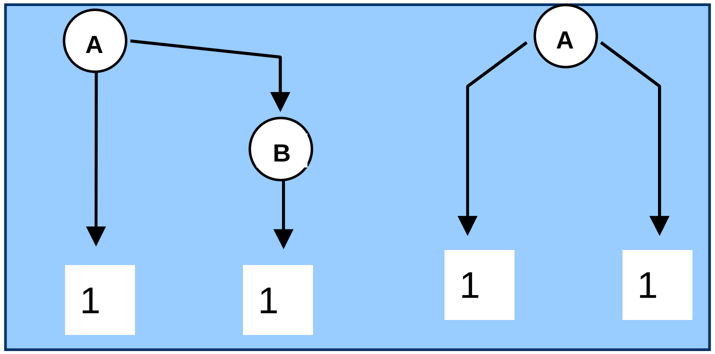
Concepto relacionado con el de árbol de derivación:

- si una sentencia puede obtenerse en una G por medio de dos o más árboles de derivación diferentes, **la sentencia es ambigua**
- una G es ambigua si contiene al menos una sentencia ambigua
- aunque una G sea ambigua, es posible que el lenguaje que describe no sea ambiguo [Floyd 1962] $\Rightarrow$ es posible encontrar una G equivalente que no lo sea
- existen lenguajes para los que NO es posible encontrar G no ambiguas $\Rightarrow$ Lenguajes Inherentemente Ambiguos [Gross 1964]
- la propiedad de ambigüedad es indecidible. Tan solo es posible encontrar condiciones suficientes que aseguren que una G es no ambigua
- indecidible: no existe un algoritmo que acepte una G y determine con certeza y en un tiempo finito si una G es ambigua o no.

Gramáticas Formales. Ambigüedad

- De las definiciones se deduce que existen 3 niveles de ambigüedad:
  - **Sentencia:** una sentencia es ambigua si puede obtenerse por medio de dos o más árboles de derivación diferentes

ej: G = ({1}, {A,B}, A, {A::= 1B / 11, B::= 1})



  - Gramática: es ambigua si contiene al menos una sentencia ambigua, ej: la G anterior
  - **Lenguaje:** un L es ambiguo si existe una G ambigua que lo genera, ej: L = {11} es ambiguo, pero como todos los lenguajes son ambiguos, es un término que no da información.

Gramáticas Formales. Ambigüedad

- **Lenguajes Inherentemente Ambiguos:** para los que NO es posible encontrar G no ambiguas $\Rightarrow$ ejemplo L = $\{\{a^n b^m c^m d^n\} \cup \{a^n b^n c^m d^m\} \ / \ m,n \geq 1\}$

ejemplo: L = {11} no es inherentemente ambiguo

G' = ({1}, {A}, A, {A..=11})

Gramáticas Formales. Ambigüedad

- Dada la G = ({a,b}, {A,B,S}, S, P)

  P = {

  S::= bA / aB

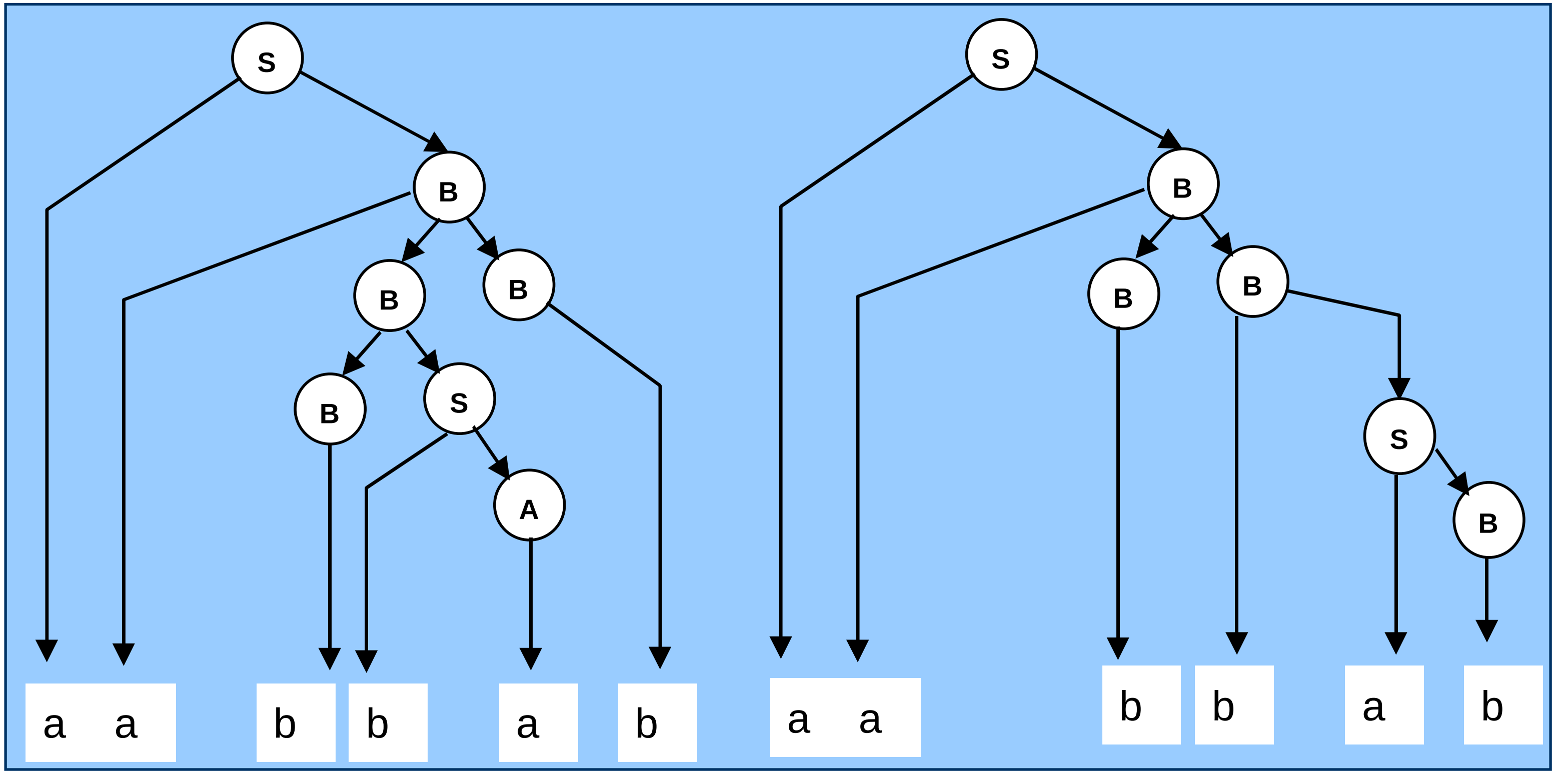
  A::= bAA / a / aS

  B ::= b / bS / aBB}

  demostrar que es una G ambigua al serlo la sentencia “aabbab”

Gramáticas Formales. Ambigüedad

- Dada la G = ({a,b}, {A,B,S}, S, P)
  P = {
  S::= bA / aB, A::= bAA / a / aS, B ::= b / bS / aBB}
  demostrar que es una G ambigua al serlo la sentencia “aabbab”



Gramáticas Formales.
Gramáticas Independientes del Contexto

- Son las gramáticas de tipo 2 en la jerarquía de Chomsky.
- Alfonseca: tema 9
- Importancia:
  - son las empleadas en la definición de lenguajes de programación y en la compilación de los mismos
  - Compilador
    - analizador $\rightarrow$ aceptador de la G
      - explorador (AF)
      - reconocedor (AP)
    - generador de código $\rightarrow$ Traductor

Gramáticas Formales.
Gramáticas Independientes del Contexto

- A los lenguajes generados por gramáticas del tipo 2 de la jerarquía de Chomsky se les denomina:
  - Lenguajes independientes del contexto o lenguajes de contexto libre
  - Se representan como L(G2)
  - Existen algoritmos que permiten reconocer si un L(G2) es vacío, finito o infinito

Gramáticas Formales.
Gramáticas Independientes del Contexto

- Lenguaje, vacío o no?:

  Sea G2, m = $C(\Sigma_{NT})$, **L(G2)** $\neq \phi$ si $\exists$ x $\in$ L(G2) tal que x puede generarse con un árbol de derivación en el que todos los caminos tienen longitud $\leq$ m

  Se generan todos los árboles de derivación con caminos $\leq$ m = $C(\Sigma_{NT})$ mediante el algoritmo:

  a) conjunto de árboles con longitud 0 (un árbol con S como raíz y sin ramas)

  b) a partir del conjunto de árboles de longitud n, generamos el conjunto de longitud n+1 < m +1 aplicando al conjunto de partida una producción que no haga duplicarse algún NT en el camino considerado

  c) se aplica el paso b) recursivamente hasta que no puedan generarse más árboles con caminos de longitud $\leq$ m. Al ser m y el número de reglas de P finito $\Rightarrow$ el algoritmo termina

L(G2) = ϕ si ninguno de los árboles genera una sentencia

Gramáticas Formales.
Gramáticas Independientes del Contexto

- Ejemplo de L(G2) = ϕ

  Sea G = ({a,b}, {A,B,C,S}, S, P)

  P = {

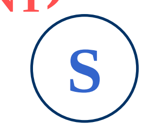
  S ::= aB / aA

  A ::= B / abB

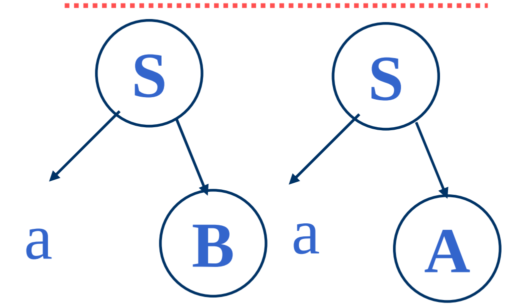
  B ::= bC}

$m \leq C(\Sigma_{NT}) = 4$

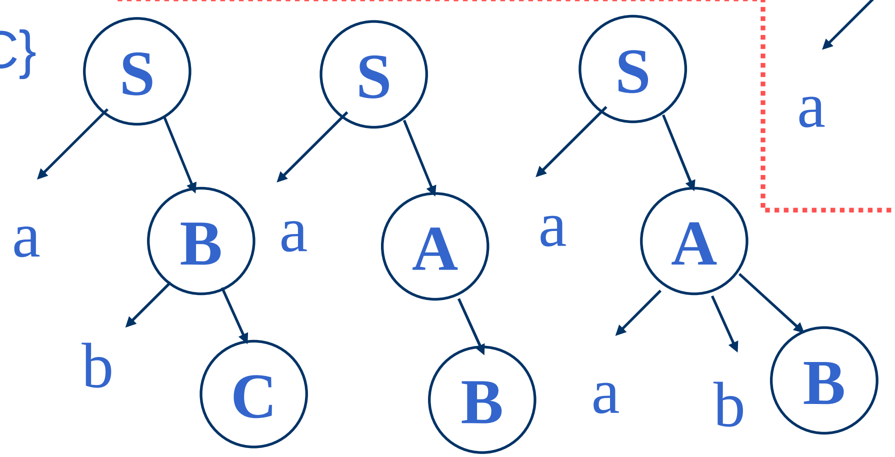
1. m=0



2. m=1



3. m=2



4. m=3

No se generan sentencias y salen NT ya obtenidos
Lenguaje vacío

Gramáticas Formales.
Gramáticas Independientes del Contexto

- Si L(G2) es no vacío, comprobar si **L(G2) = $\propto$**

Se construye un grafo cuyos nodos están etiquetados con los símbolos de (Σ_{NT}) mediante el algoritmo:

a) si $\exists$ una producción A::= α B β, se crea un arco de A a B donde A,B $\in \Sigma_{NT}$ y $\alpha,\beta \in \Sigma^*$

b) si no existen ciclos en el grafo el L(G2) = finito

c) L(G2) = $\propto$ si existen ciclos accesibles desde el axioma que corresponden a derivaciones de la forma A $\rightarrow^+$ α A β, donde /α/+ /β/ >0 (que no sean λ las dos a la vez).

L(G2) $\neq \propto$ si no hay ciclos en el grafo

Gramáticas Formales.
Gramáticas Independientes del Contexto

- Ejemplo de **L(G2) = ∝**

  Sea G = ({a,b,c}, {A,B,C,S}, S, P)

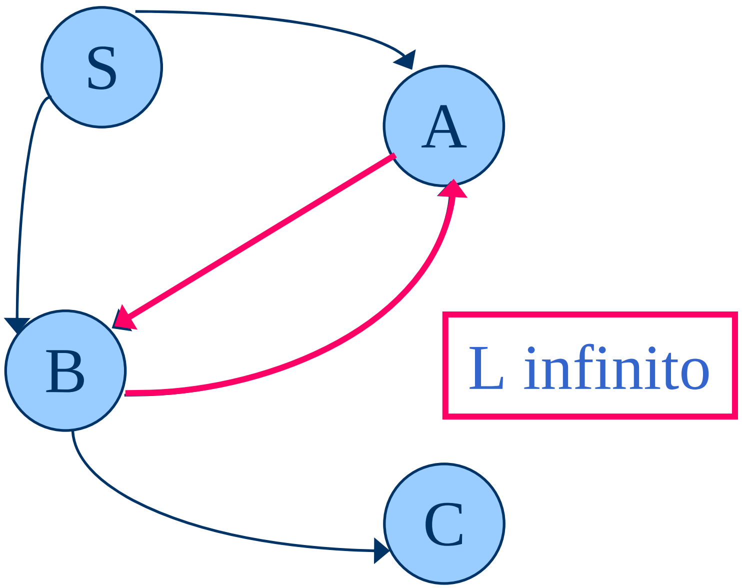
  P = {

  S ::= aB / aA

  A ::= abB

  B ::= bC / aA

  C ::= c }



Gramáticas Independientes del Contexto
Gramáticas Bien Formadas

- Transformación de una G dada en otra equivalente cuyas reglas de producción estén en un formato carente de imperfecciones:

1. **Limpieza de Gramáticas**
   1. Reglas Innecesarias
   2. Símbolos Inaccesibles
   3. Reglas Supérfluas
2. **Eliminación de símbolos no generativos**
3. **Eliminación de reglas no generativas**
4. **Eliminación de reglas de redenominación**

Gramáticas Independientes del Contexto
Gramáticas Bien Formadas

1. **Limpieza de Gramáticas**
   1. **Reglas Innecesarias**: las reglas **A::= A $\in$P** son innecesarias y hacen que G sea ambigua $\Rightarrow$ eliminarlas
   2. **Símbolos Inaccesibles:** sea U::= x $\in$P, donde U$\in \Sigma_N \neq$ S y no aparece en la parte derecha de ninguna otra regla de producción se dice que U es inaccesible.

      Todo símbolo U$\in \Sigma_N$ no inaccesible debe cumplir S $\rightarrow$ * xUy.

      Eliminación de símbolos inaccesibles:

      a) matriz booleana (alfonseca cap. 3)

      b) grafo análogo al de L(G2) = $\propto$. Los símbolos inaccesibles no serán alcanzables desde el axioma.

      Se eliminan así como las reglas que los contengan

Gramáticas Independientes del Contexto
Gramáticas Bien Formadas

1. **Limpieza de Gramáticas**
   2. **Símbolos Inaccesibles: ejemplo:**

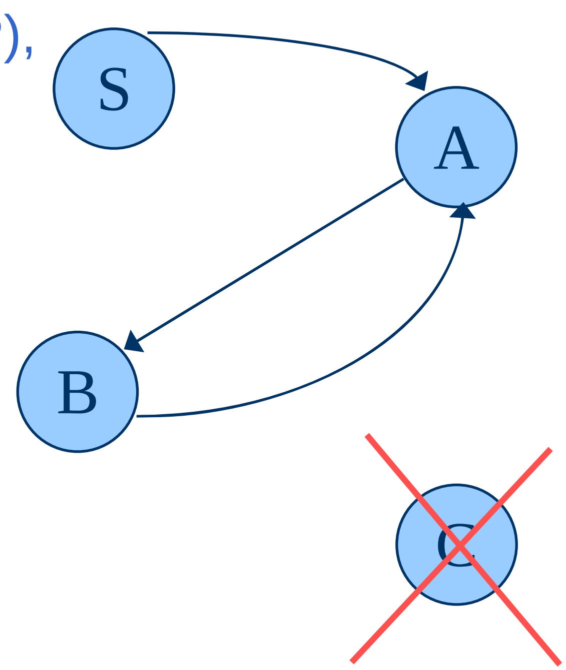
sea la G = ({a,b,c}, {S,A,B,~~C~~}, S, P),

donde P = {S::= aA

A ::= Bc

B::= bA

~~C::= c~~}



Gramáticas Independientes del Contexto Gramáticas Bien Formadas

1. **Limpieza de Gramáticas**
   3. **Reglas Supérfluas**: son aquellas que no contribuyen a la formación de palabras x $\in \Sigma_T^*$.

      Todo símbolo no supérfluo debe cumplir U $\rightarrow^+$t, tal que t $\in \Sigma_T^*$

      algoritmo: es un algoritmo recursivo de marcado

      a) marcar los NT para los que existe una regla U ::= x donde x $\in \Sigma^*$ (es una cadena de T o λ, o en pasadas sucesivas contiene NT marcados)

      b) si todos los NT están marcados $\Rightarrow$ no existen símbolos supérfluos y fin

      c) si la última vez que se pasó por el paso a) se marcó un NT, volver al paso a).

      d) todo A $\in \Sigma_{NT}$ no marcado es supérfluo.

Gramáticas Independientes del Contexto
Gramáticas Bien Formadas

1. **Limpieza de Gramáticas**
   3. **Reglas Supérfluas**:

      ejemplo: sea G= ({e,f}, {S,A,B,C,D}, S, P)

      donde P= {

      S::= Be

      A::= Ae /e

      B::= Ce / Af

      C::= Cf

      D::= f}

1ª pasada:

D::= f y A::= e

2ª pasada:

B::= Af

3ª pasada:

S::= Be

Gramáticas Independientes del Contexto Gramáticas Bien Formadas

1. **Limpieza de Gramáticas**
   3. **Reglas Supérfluas**:

ejemplo: sea G= ({e,f}, {S,A,B,~~C~~,D}, S, P)

donde P= {

S::= Be

A::= Ae /e

B::= ~~C~~e / Af

C::~~=~~Cf

D::= f}

1ª pasada:

D::= f y A::= e

2ª pasada:

B::= Af

3ª pasada:

S::= Be

Sin marcar: C, que se elimina, así como sus reglas

Gramáticas Independientes del Contexto
Gramáticas Bien Formadas

2. **Eliminación de símbolos no generativos:**

   Sea G2 = $(\Sigma_T, \Sigma_N, S, P)$, $\forall \mathbf{A} \in \Sigma_{\mathbf{N}}$ construiremos la gramática G(A), donde A es el axioma. Si **L(G(A))=** $\phi \Rightarrow$ **A es símbolo no generativo** y se puede eliminar, así como todas las reglas que lo contengan, obteniéndose otra G2 equivalente.

Gramáticas Independientes del Contexto
Gramáticas Bien Formadas

3. **Eliminación de reglas no generativas:** son $A::= \lambda$ $(A \neq S)$

- Si $\lambda \in$ **L(G)**: se añade $S::= \lambda$ **y** $\forall\ A \in \Sigma_{Nt}$ $(A::= \lambda\ A \neq S)$ y $\forall$ regla de G de la forma $B::=xAy$, añadiremos en P' una regla de la forma $B::= xy$, excepto en el caso en que $x=y=\lambda$

Gramáticas Independientes del Contexto Gramáticas Bien Formadas

4. Eliminación de reglas de redenominación: son reglas del tipo A::=B

algoritmo:

a) la G' equivalente contiene todas las reglas excepto las tipo

A::= B

b) $\forall A \in \Sigma_{NT} \mid A \rightarrow^{*} B$ en G y $\forall$ (B::= x) $\in P$ donde $x \notin \Sigma_{NT} \Rightarrow$

P' = P + {A::= x}

Gramáticas Independientes del Contexto
Gramáticas Bien Formadas

- **G = ({0,1}, {S,A,B,C},S, P),**

  donde P = {S::= AB / 0S1 / A / C,A ::= 0AB / B::= B1 / λ}

  - **No hay innecesarias**

  - **Inaccesibles**

Gramáticas Independientes del Contexto
Gramáticas Bien Formadas

- **G = ({0,1}, {S,A,B,C},S, P),**
  donde P = {S::= AB / 0S1 / A / C,A ::= 0AB / B::= B1 / λ}
  - **No hay innecesarias**
  - **Inaccesibles**
  - **R. Supérfluas y S. No generativos**
  - **R. No generativas**
  - **R. De redenominación**

Gramáticas Independientes del Contexto
Gramáticas Bien Formadas

- **G = ({0,1}, {S,A,B,C},S, P),**

  donde P = {S::= AB / 0S1 / A / ~~C~~,A ::= 0AB / B::= B1 / λ}

  - **No hay innecesarias**
  - **Inaccesibles: no hay**
  - **R. Supérfluas y S. No generativos:**
  - **R. No generativas: se quedan así las**

  P' = {S::= A / B / 0S1/λ, A::= 0AB/ 0A /0B /0, B::=B1/1**}**

  - **R. De redenominación**

Hay 2:

S::= A / B, se sustituyen por sus partes derechas:

$\boxed{\text{S::= 0AB/ 0A /0B /0/ B1/1/ 0S1}}$

Gramáticas Independientes del Contexto
Gramáticas Bien Formadas

- **G = ({0,1}, {S,A,B,C},S, P),**

  donde P = {S::= AB / 0S1 / A / C,A ::= 0AB / B::= B1 / λ}

- **G = ({0,1}, {S,A,B},S, P),**

  donde P = {S::= 0AB/ 0A /0B /0/ B1/1/ 0S1/ λ,

  A::= 0AB/ 0A /0B /0, B::=B1/1}

Es la gramática bien formada equivalente

Gramáticas Independientes del Contexto
Formas Normales

- Son notaciones que se aplican a las G2:

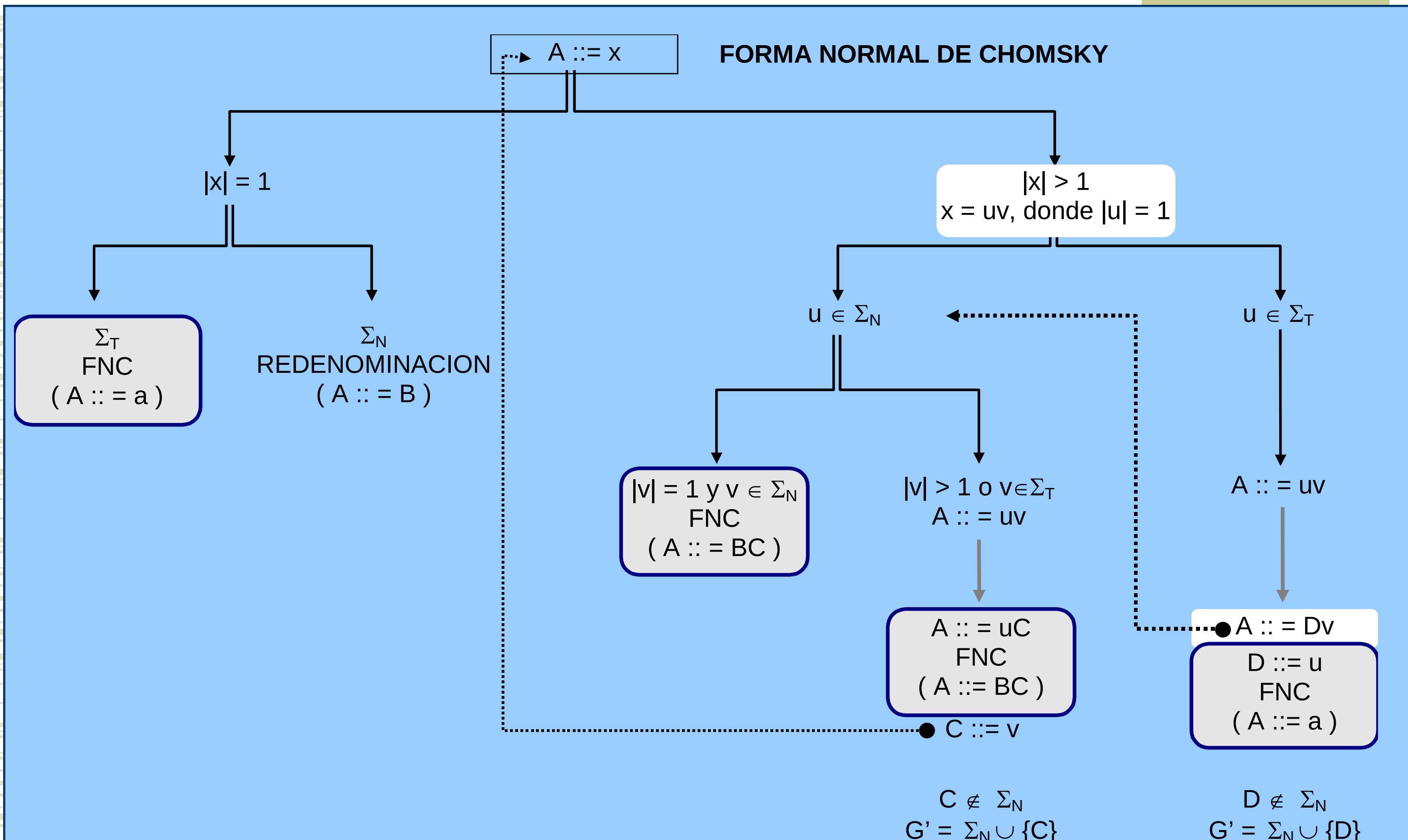
  Afectan a la forma de las reglas de producción

- Son dos las que se va a estudiar:

  Forma Normal de Chomsky

  Forma Normal de Greibach

Gramáticas Independientes del Contexto
Forma Normal de Chomsky



Gramáticas Independientes del Contexto Forma Normal de Chomsky

- Ejercicio: Alfonseca pg 212: sea la G = ({0,1}, {S,A,B},S, P),

  donde P = {S::= AB / 0S1 / 0AB / 0B / 0A / 0 / B1 / 1 / λ

  A ::= 0AB / 0B / 0A / 0

  B::= B1 / 1}

  Hallar la gramática en FNC equivalente

Gramáticas Independientes del Contexto
Forma Normal de Greibach

- **FNG** es una notación muy interesante para algunos reconocimientos sintácticos. En ella todas las reglas tienen la parte derecha comenzando con un terminal seguido opcionalmente de uno o varios NT
- TEOREMA: todo **L de contexto libre sin** λ puede ser generado por una G2 en la que todas las reglas sean de la forma:
  - $\mathbf{A \rightarrow a\alpha}$ donde $A \in \Sigma_{NT}$ $a \in \Sigma_T$ $\alpha \in \Sigma_{NT}^*$
  - Si $\lambda \in L$ habrá que añadir **S::=** λ
- TERⲞREMA: toda G2 puede reducirse a otra G2 equivalente sin reglas recursivas a izquierdas

Gramáticas Independientes del Contexto
Forma Normal de Greibach

- **FNG:** para transformar una G2 en su equivalente en forma normal de Greibach:

1. Eliminar la recursividad a izquierdas
2. Aplicar el algoritmo de transformación a FNG, verificando en cada paso que no aparezcan nuevas reglas recursivas a izquierdas y si aparecen, eliminándolas con el paso 1

Gramáticas Independientes del Contexto
Forma Normal de Greibach

1. Eliminar la recursividad a izquierdas: en un paso (estudiar en varios pasos, Isasi Martínez y Borrajo, pg 24):

   sea G = ({α,β}, {A}, A, P), donde P = {A::= A α / β}

$A \rightarrow \beta$

$A \rightarrow A\,\alpha \rightarrow \beta\,\alpha$

$A \rightarrow A\,\alpha \rightarrow A\,\alpha\,\alpha \rightarrow \beta\,\alpha\,\alpha$

$A \rightarrow A\,\alpha \rightarrow A\,\alpha\,\alpha \rightarrow \ldots \rightarrow \beta \underbrace{\ldots\ldots\,\alpha\,\alpha}$

A::= β

A::= βX

X::= α X

X::= α

Gramáticas Independientes del Contexto
Forma Normal de Greibach

1. Eliminar la recursividad a izquierdas, resumiendo, sería:

sea G = ($\{\alpha_1, \alpha_2, \beta_1, \beta_2\}$, {A}, A, P), donde P = {A::= A α_1/ Aα_2 / β_1/ β_2}

Quedaría:

$$\boxed{\begin{array}{l} A ::= \beta_1 / \beta_2 / \beta_1 X / \beta_2 X \\ X ::= \alpha_1 / \alpha_2 / \alpha_1 X / \alpha_2 X \end{array}}$$

Gramáticas Independientes del Contexto
Forma Normal de Greibach

2. transformación de G2 bien formada sin RI a FNG:

2.1 Establecer una relación de orden parcial en Σ_{NT}

$\Sigma_{NT} = \{A_1, A_2, ..., A_n\}$ basándose en: si Ai $\rightarrow$ Aj α, Ai precederá a Aj.

Cuando hay reglas "contradictorias" usar una de ellas para el orden y mirar el resto para ver que conviene más

2.2 Se clasifican las reglas en 3 grupos:

Grupo 1: Ai $\rightarrow$ a α ,donde a $\in \Sigma_T$ y $\alpha \in \Sigma_{NT}{}^*$

Grupo 2: Ai $\rightarrow$ Aj α donde Ai precede a Aj en el conjunto Σ_{NT} ordenado

Grupo 3: Ak $\rightarrow$ Ai α donde Ai precede a Ak en el conjunto Σ_{NT} ordenado

2.3 Se transforman las reglas de grupo 3 $\rightarrow$ grupo 2 $\rightarrow$ grupo 1: FNG

Ak $\rightarrow$ Ai α se sustituye **Ai** por la parte dcha de todas las reglas que tienen **Ai** como parte izda

Hacer lo mismo con las de grupo 2

Gramáticas Independientes del Contexto
Forma Normal de Greibach

2. transformación de G2 bien formada sin RI a FNG:

2. 4. Cuando todas las reglas son de grupo 1, la G está en FNG a falta de **eliminar los símbolos terminales no situados en la cabecera de la parte derecha**

$\boxed{\mathbf{A} \rightarrow \alpha\ \ \mathbf{a}\,\beta}$ **donde a**$\in\Sigma_{\mathbf{T}}$ **y** $\alpha \neq \lambda$

$\downarrow$

$$\boxed{\begin{array}{l} \mathbf{A} \rightarrow \alpha\ \ \mathbf{B}\,\beta \\ \mathbf{B} \rightarrow \mathbf{a} \end{array}}$$