4. Lenguajes y Gramáticas Formales

Grado Ingeniería Informática
Teoría de Autómatas y Lenguajes Formales

LENGUAJES FORMALES

Definiciones

Operaciones

Reglas de derivación

Definiciones

Símbolo

Entidad abstracta, no se define (análogo al punto en geometría). Son letras, dígitos, caracteres, etc. También es posible encontrar símbolos formados por varios caracteres, pej: IF, THEN, ELSE, ...

Alfabeto (Σ)

Conjunto **finito no vacío** de letras o símbolos.

Sea "a" una letra y Σ un alfabeto, si a pertenece a ese alfabeto $\Rightarrow a \in \Sigma$

Ejemplos:

Σ_1= {A, B, C, ...,Z}

Σ_2= {0, 1}

Σ_3= {IF, THEN, ELSE, BEGIN, END}

Definiciones

Palabra, cadena, tira: toda secuencia finita de símbolos del alfabeto.

Ejemplos:

palabras sobre Σ_1 JUAN, ISABEL, etc.

palabras sobre Σ_2 00011101

palabras sobre Σ_3 IFTHENELSEEND

Notación: las palabras se representan por letras minúsculas del final del alfabeto

(x, y, z), pej x= JUAN, y= IFTHENELSEEND

5

Definiciones

Longitud de palabra

Es el número de símbolos que componen una palabra

La longitud de la palabra x se representa por $|x|$

Ejemplos:

$|x|$ = $|$JUAN$|$ = 4

$|y|$ = $|$IFTHENELSEEND$|$ =13 (en Σ_1)

$|y|$ = $|$IFTHENELSEEND$|$ = 4 (en Σ_3)

Palabra vacía (λ)

Es aquella palabra cuya longitud es cero

Se representa por λ, $|\lambda|$ = 0

Sobre cualquier alfabeto es posible construir λ

Utilidad: es elemento neutro en muchas operaciones con palabras y lenguajes

6

Definiciones

Universo del discurso, W(Σ): Es el conjunto de todas las palabras que se pueden formar con los símbolos de un alfabeto Σ

- ✓ También se denomina Lenguaje Universal de Σ
- ✓ Se representa como W(Σ)
- ✓ Es un conjunto infinito
- ✓ Ejemplo: sea $\Sigma_4 = \{A\}$, $W(\Sigma_4) = \{\lambda, A, AA, AAA, \ldots\}$ con un número ∞ de palabras

COROLARIO:

$\forall\ \Sigma,\ \ \lambda \in W(\Sigma) \Rightarrow$ La palabra vacía pertenece a todos los lenguajes universales de todos los alfabetos posibles

7

Definiciones

Operaciones

Reglas de derivación

8

Operaciones con palabras

Operaciones con palabras sobre palabras de un universo del discurso dado

1 Concatenación de palabras

2 Potencia

3 Reflexión

1 Concatenación de palabras

sean dos palabras *x, y* tal que $x \in W(\Sigma)$, $y \in W(\Sigma)$, y sea

$$|x| = i = |x_1 x_2 \ldots x_i| \text{ e } |y| = j = |y_1 y_2 \ldots y_j|,$$

se llama **concatenación** de x con y, a:

$$x \cdot y = x_1 x_2 \ldots x_i \; y_1 y_2 \ldots y_j = \mathbf{z}, \text{ donde } z \in W(\Sigma)$$

1 Concatenación de palabras

Propiedades:

- ✓ Operación cerrada (o interna)
- ✓ Propiedad Asociativa
- ✓ Con elemento neutro.
- ✓ No conmutativa

Definiciones:

- ✓ Cabeza
- ✓ Cola
- ✓ Longitud de palabra

11

2 Potencia

Es la reducción de la concatenación a los casos que se refieren a una misma palabra

- ✓ potencia *i-ésima* de una palabra al resultado de concatenar esa palabra consigo misma i veces
- ✓ concatenación es asociativa $\Rightarrow$ no especificar el orden
- ✓ $x^i = x \cdot x \cdot x \ldots . x$ i veces
- ✓ $|x^i| = i \cdot |x|$
- ✓ se cumple:
  - $x^1 = x$
  - $x^{1+i} = x \cdot x^i = x^i \cdot x$ (i>0)
  - $x^{j+i} = x^j \cdot x^i = x^i \cdot x^j$ (i, j>0)
- ✓ Si se define $x^0 = \lambda$
  - (i, j>=0)

12

3 Reflexión de una palabra

Sea la palabra $x = a_1 \cdot a_2 \cdot a_3 \cdot \ldots \cdot a_n$,

se denomina **palabra refleja** de x,

$x^{-1} = a_n \cdot a_{n-1} \cdot \ldots \cdot a_2 \cdot a_1$,

formada por los mismos símbolos en distinto orden

$|x^{-1}| = |x|$

13

Operaciones con lenguajes

¿Qué es un Lenguaje?

Se denomina **Lenguaje sobre el alfabeto Σ** a todo subconjunto del lenguaje universal de Σ ($L \subset W(\Sigma)$)

i.e. a todo conjunto de palabras sobre un determinado Σ

i.e. a todo conjunto de palabras generado a partir del alfabeto Σ

14

Operaciones con lenguajes

Son lenguajes especiales:

ϕ = Lenguaje vacío, $\phi \subset W(\Sigma)$

$\{\lambda\}$ = Lenguaje de la palabra vacía

ϕ y $\{\lambda\}$ se diferencian en el número de palabras (cardinalidad) que los forman $C(\phi) = 0$ mientras que $C(\{\lambda\})=1$ se parecen en que ϕ y $\{\lambda\}$ son lenguajes sobre cualquier alfabeto

Un alfabeto es uno de los lenguajes generados por el mismo:

$\Sigma \subset W(\Sigma)$, por ejemplo el chino

Operaciones con lenguajes

Operaciones con lenguajes sobre un un alfabeto dado Σ

1. Unión de lenguajes
2. Concatenación de lenguajes
3. **Binoide libre**
4. Potencia de un lenguaje
5. **Clausura o cierre positivo de un lenguaje**
6. **Iteración, clausura o cierre de un lenguaje**
7. **Reflexión de lenguajes**

1 Unión de lenguajes

Sean L1 y L2 definidos sobre el mismo alfabeto Σ, L1, L2 $\subset W(\Sigma)$, se llama **unión** de dos lenguajes, L1, L2 y se representa por L1$\cup$ L2 al lenguaje así definido:

$$L1 \cup L2 = \{ x / x \in L1 \text{ ó } x \in L2 \}$$

Es el conjunto formado indistintamente por palabras de uno u otro de los dos lenguajes (equivale a la suma)

$$L1 + L2 = L1 \cup L2$$

17

2 Concatenación de lenguajes

Sean L_1 y L_2 definidos sobre el mismo alfabeto, L_1, $L_2 \subset W(\Sigma)$, se llama **concatenación o producto** de dos lenguajes, L_1 y L_2 , y se representa por $L_1 \cdot L_2$ al lenguaje así definido:

$$L_1 \cdot L_2 = \{ xy / x \in L_1 \text{ AND } y \in L_2 \}$$

- ✓ Es el conjunto de palabras formado por la concatenación de palabras de L_1 con palabras de L_2
- ✓ Definición válida para lenguajes con algún elemento.
- ✓ Y con el lenguaje vacío: $\phi \cdot L = L \cdot \phi = \phi$

18

2 Concatenación de lenguajes

Propiedades

- ✓ Operación cerrada
- ✓ Propiedad Asociativa
- ✓ Con elemento neutro
- ✓ Propiedad distributiva respecto a la unión.

3 Binoide libre

- ✓ La concatenación (monoide) de lenguajes y la unión (monoide) de lenguajes constituyen un **binoide**
- ✓ Los símbolos de Σ se pueden considerar conjuntos de una sola palabra
- ✓ Con Σ, la unión y la concatenación se puede formar cualquier lenguaje sobre dicho Σ. Excepto ϕ y $\{\lambda\}$.

El alfabeto Σ es un conjunto de generadores para el conjunto L $\Rightarrow$ **L es el BINOIDE LIBRE (operaciones U y •) generado por Σ**

4 Potencia de un lenguaje

- ✓ Es la reducción de la concatenación a los casos que se refieren a un mismo lenguaje
- ✓ potencia *i-ésima* de un lenguaje al resultado de concatenar ese lenguaje consigo mismo *i* veces
- ✓ concatenación es asociativa ⇒ no especificar el orden
- ✓ $L^i = L \cdot L \cdot L \cdot \ldots \cdot L$; *i* veces
- ✓ Se define $L^1 = L$
- ✓ Se cumple:
  - $L^{1+i} = L \cdot L^i = L^i \cdot L$ (i>0)
  - $L^{j+i} = L^i \cdot L^j$ (i, j >0)
- ✓ Si se define $L^0 = \{ \lambda \}$, entonces (i≥0) (i, j ≥ 0)

21

5 Clausura o cierre positivo

Se representa como L^+ y es el lenguaje obtenido uniendo el lenguaje L con todas sus potencias posibles **excepto L^0**

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

Ninguna clausura positiva contiene a λ, salvo si λ€L

Como Σ es un lenguaje sobre Σ, la clausura positiva de Σ será:

$$\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i = W(\Sigma) - \{\lambda\}$$

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

22

6 Iteración, clausura o cierre

Se representa como L^* y es el lenguaje obtenido uniendo el lenguaje L con todas sus potencias posibles.

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Toda clausura contiene a λ

* es el operador unario de Kleene

✓ Se cumple:

$$L^* = L^+ \cup \{\lambda\}$$

$$L^+ = L^* \cdot L = L \cdot L^*$$

✓ Como Σ es un lenguaje sobre Σ, se le puede aplicar el *:

$$\Sigma^* = W(\Sigma)$$ → $\boxed{\text{El lenguaje universal es } \Sigma^*}$

7 Reflexión de lenguaje

Se llama lenguaje reflejo o inverso de L y se representa por L^{-1} al lenguaje:

$$L^{-1} = \{x^{-1} \mid x \in L\}$$

es decir, al lenguaje formado por todas las palabras reflejas de L

- Definiciones
- Operaciones
- Reglas de derivación

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Producciones

Sea Σ un alfabeto

- ✓ Se llama producción a un **par ordenado** (*x* , *y*) donde *x* , $y \in \Sigma^*$

  $x \in \Sigma^*$, es la parte izquierda de la producción e

  $y \in \Sigma^*$, la parte derecha de la producción

- ✓ Se representa como: $x ::= y$

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Derivación directa

- ✓ Sea Σ un alfabeto
- ✓ Sea (*x, y*) una producción sobre palabras de Σ, x::=y
- ✓ Sean v y w dos palabras sobre Σ (i.e v, w ∈ Σ*)

Se dice

"w es derivación directa de v" ó
"v produce directamente w" $v \to w$
"w se reduce directamente a v"

si ∃ dos palabras z , u ∈ Σ* tales que $v = z \cdot x \cdot u$ y $w = z \cdot y \cdot u$

COROLARIO

Si x ::= y es una producción sobre Σ $\Rightarrow x \to y$
(una regla de escritura es una derivación directa)

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

27

Derivación directa

Ejemplos

- ✓ Sea Σ el alfabeto castellano de las letras mayúsculas y ME::=BA una producción sobre Σ

  CABALLO es derivación directa de CAMELLO

  (CAMELLO produce directamente CABALLO)

- ✓ Y con la producción CA::=PE sobre Σ

  PERA es derivación directa de CARA

En el castellano no son así las cosas, existen las raíces.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

28

Derivación directa

En un conjunto de producciones

- ✓ Sea Σ un alfabeto y P un conjunto de producciones sobre Σ
- ✓ Sean v y w dos palabras sobre Σ, v, w ∈Σ*

Se dice que

$$\left.\begin{array}{l}\text{“w es derivación directa de v”}\\ \text{“v produce directamente w”}\\ \text{“w se reduce directamente a v”}\end{array}\right\} v \to w$$

si ∃ dos palabras z, u ∈Σ* tales que v = z·x·u y w = z·y·u y se cumple **(x::=y) ∈ P**

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Derivación

- ✓ Sea Σ un alfabeto y P un conjunto de producciones sobre Σ
- ✓ Sean *v* y *w* dos palabras sobre Σ, *v*, *w* ∈Σ*

Se dice que

$$\left.\begin{array}{l}\text{“w es derivación de v”}\\ \text{“v produce w”}\\ \text{“w se reduce a v”}\end{array}\right\} v \overset{+}{\to} w$$

si ∃ una secuencia finita de palabras, $u_0, u_1, u_2, \ldots u_n$ (n>0) ∈Σ*

tales que $v = u_0$

$$\left.\begin{array}{l}u_0 \to u_1\\ u_1 \to u_2\\ \ldots\ldots\\ u_{n-1} \to u_n\\ w = u_n\end{array}\right\} \textbf{Derivación de longitud n}$$

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Relación de Thue

- ✓ Sea Σ un alfabeto y P un conjunto de producciones sobre Σ
- ✓ Sean v y w dos palabras sobre Σ, es decir v, w $\in\Sigma^*$

Se dice que existe una relación de Thue entre v y w y se representa por **v *→ w** si se verifica que:

$$\left.\begin{array}{l}\mathbf{v} \overset{+}{\rightarrow} \mathbf{w} \quad \text{ó} \\ \mathbf{v} = \mathbf{w}\end{array}\right\}$$ **o ∃ una derivación de longitud *n* o son iguales**

Propiedades

- ✓ Reflexiva
- ✓ Transitiva
- ✓ Simétrica (en general NO se cumple)

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

31

GRAMÁTICAS FORMALES

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

32

- Definiciones
- Jerarquía de Chomsky
- Árboles de derivación

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

¿Qué es una gramática?

Una gramática describe la estructura de las frases y de las palabras de un lenguaje y se aplica por igual a:

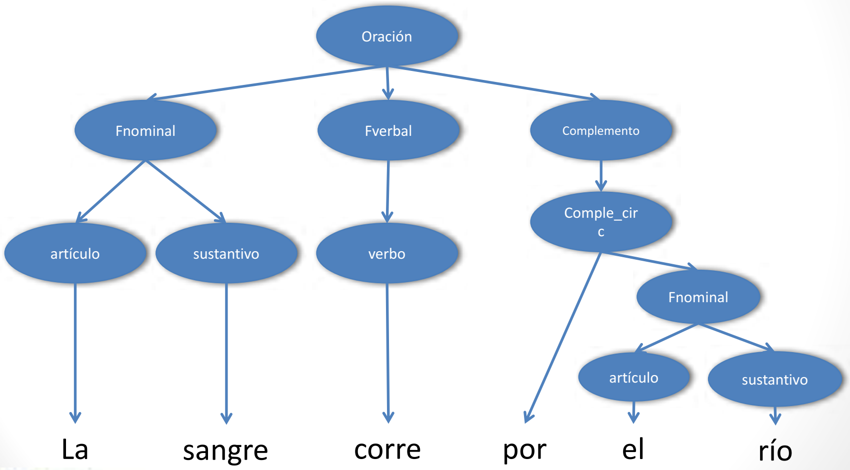
- ✓ Las lenguas naturales humanas
- ✓ Lenguajes de programación.

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

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramática del Castellano

Una gramática del castellano como diagrama sintáctico



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

35

Reglas de producción

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

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

36

Definición

Cómo se determina (define) una gramática:
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: Alfabeto de símbolos No Terminales.** 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 $\in \Sigma_N$.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Definición

Σ_T: Alfabeto de símbolos terminales

Σ_N: Alfabeto de símbolos NO terminales

Axioma

G = ({0,1}, {N,C}, N, P)

P = {N::= NC,
N::=C,
C::= 0,
C ::=1}

Producciones

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Definición

Se cumple entre los alfabetos de G:

$\Sigma_T \cup \Sigma_N = \Sigma$ 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,D}, N,P)
P = { N::= ND | D ; D::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}

Forma sentencial

Sea una *G*, sea *x* $\in \Sigma^*$, es decir sea x $\in (\Sigma_T \cup \Sigma_N)^*$,

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

$$S \overset{*}{\longrightarrow} x$$

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

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

Lenguaje asociado a una gramática

Sea la gramática:

$G_1 = (\Sigma_T, \Sigma_N, S, P)$

Se llama:

lenguaje asociado a G1
lenguaje generado por G1
o lenguaje descrito por G1

al **conjunto de todas las sentencias (palabras) generadas** por G1, es decir:

$L(G_1) = \{x \mid S \overset{*}{\rightarrow} x, x \in \Sigma^*_T\}$

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

41

Recursividad

Sea *G*,

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

  $U \overset{+}{\rightarrow} x\,U\,y$

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

  $U ::= x\,U\,y$
- ✓ Si un lenguaje es infinito, la gramática que lo representa tiene que ser recursiva

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

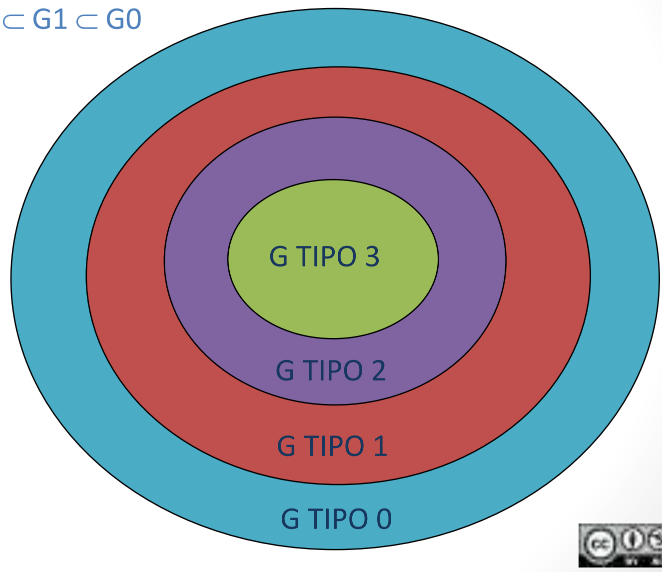
42

- Definiciones
- Jerarquía de Chomsky
- Árboles de derivación

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

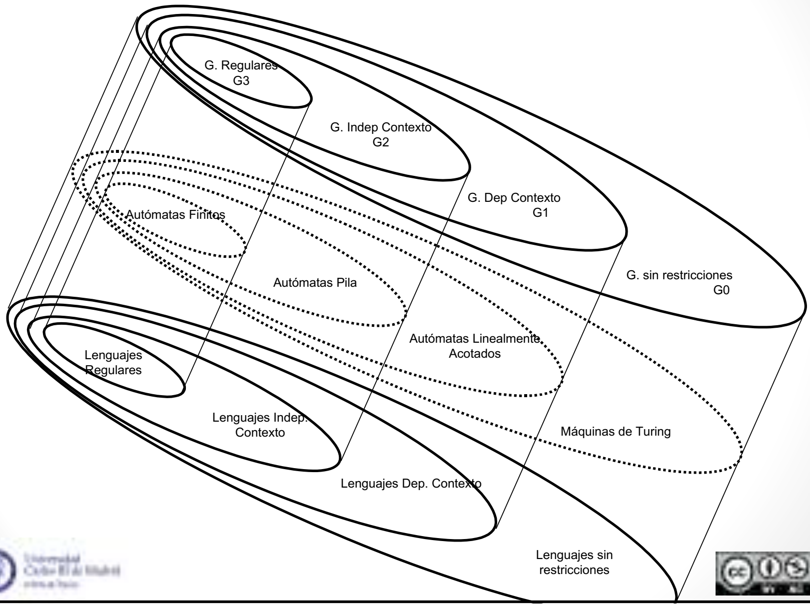
Jerarquía de Chomsky

$G3 \subset G2 \subset G1 \subset G0$



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Máquinas, Lenguajes y Algoritmos



45

Tipo 0

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

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,A), S, P}, donde:
P={S→0S0A, SA→AS, 0A→1}

Con estructura de frases:
(xAy ::= xvy) $\in$ P, donde:
x,y $\in \Sigma^*$, A $\in \Sigma_N$, v $\in \Sigma^*$
En xAy ::= xvy cuando v=λ,
xAy ::= xy reglas compresoras

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

46

Tipo 0

- ✓ 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)
P = {A::= aABC / abC
CB::= BC
bB::=bb
bC::=b}

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

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

47

Tipo 1

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

G1
Sensibles al Contexto

xAy ::= xvy

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

Ejemplo:
G= ({a,b,c}, {A,B},A, P},
P={A ::= Aaa | a,
aA ::= aB | a,
aCCc ::= aAaCc,
C ::= c}

Los Lenguajes representados por una gramática de Tipo 1 se llaman ***lenguajes dependientes del contexto*** o lenguajes sensibles al contexto (se puede cambiar A por v, siempre en el contexto x...y)

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

48

Tipo 1

Los lenguajes que son representados por G1 se llaman **Lenguajes sensibles al contexto**

donde

$$\lambda \in L(G1) \text{ Sii } (S::=\lambda) \in P$$

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}

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Tipo 1. No decrecimiento

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:

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

La longitud de la parte derecha 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

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Tipo 2

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

G2
De Contexto Libre

A ::= v

$v \in \Sigma^*$
$A \in \Sigma_N$

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

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

Ejemplo:
G= ({0,1,2,3,4,5,6,7,8,9},{N,D},N, P)
P={N $\rightarrow$ DN | D,
D $\rightarrow$ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}

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

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Tipo 2

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 $A ::= \lambda$ $(A \neq S)$

Si $\lambda \in$ L(G2) → **puede aplicarse el algoritmo de eliminación reglas NO generativas.**

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Tipo 3

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

G3
Gramáticas Regulares

G3 Lineales por la Izda.
A::= a
A::= V a
S::= λ

G3 Lineales por la Drcha.
A::= a
A::= a V
S::= λ

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

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

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)

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Tipo 3

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 $\lambda \in$ L(G3).

Ver algoritmo eliminación reglas NO generativas.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Jerarquía de Chomsky. Resumen

$G3 \subset G2 \subset G1 \subset G0$

G0. No Restringidas o con Estructura de Frase

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

Única restricción: **λ::= v ∉P**

Con Estructura De Frases:
(xAy::=xvy) ∈ P, donde: x, y ∈ Σ^*, A ∈ Σ_N, v ∈ Σ^*
Si v=λ → xAy ::= xy reglas compresoras

G1. Sensibles al Contexto

xAy ::= xvy

x, y ∈ Σ^*, A ∈ Σ_N

v ∈ Σ^+ → No permite reglas **compresoras**
Excepción: **(S::= λ)** ∈ **P**

G2. De Contexto Libre

A ::= v

v ∈ Σ^*
A ∈ Σ_N

(S::= λ) ∈**P** y **(A::= λ)** ∉**P**

G3. Gramáticas Regulares

G3 Lineales por la Izda.	G3 Lineales por la Drcha.
A::= a	A::= a
A::= V a	A::= a V
S::= λ	S::= λ

a ∈ Σ_T
A,V ∈ Σ_N

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

55

Gramáticas equivalentes

Dos gramáticas son equivalentes si representan el mismo lenguaje.

Dada una gramática lineal por la derecha cualquiera, existe otra lineal por la izquierda equivalente y viceversa.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

56

Gramáticas equivalentes

ALGORITMO: 3 PASOS.

PASO 1.

Construir una gramática equivalente que no sea recursiva en el axioma (axioma inducido):

1. se añade un nuevo símbolo en el alfabeto Σ_N, B
2. $\forall$ S::= x, donde $x \in \Sigma^+$, se añade una regla B::= x
3. Se transforman las reglas A::= a S (que desaparecen) en reglas del tipo A::= a B.
4. Las reglas tipo S::= λ no se ven afectadas por este algoritmo.

Nota: las reglas S ::= x, $x \in \Sigma^+$, no desaparecen

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas equivalentes

ALGORITMO: 3 PASOS.

PASO 1. Quitar el axioma inducido:

Construir una gramática equivalente que no sea recursiva en el axioma:

G1 = ({a,b}, {S, A}, S, P)

P = { S ::= bA,
A ::= aS | a }

PASO 1

G2 = ({a,b}, {S, A, B}, S, P)

P = { S ::= bA,
A ::= aB | a
B ::= bA}

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas equivalentes

ALGORITMO: 3 PASOS.

PASO 2.

Construcción de un grafo dirigido a partir de la gramática.

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

b) cada $A::=a\ B \in P$ $\longrightarrow$ $A \xrightarrow{a} B$

c) cada $A::=a \in P$ $\longrightarrow$ $A \xrightarrow{a} \lambda$

d) si $S::= \lambda \in P$ $\longrightarrow$ $S \longrightarrow \lambda$

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas equivalentes

ALGORITMO: 3 PASOS.

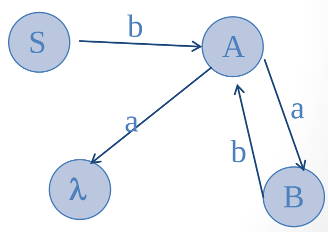
PASO 2.

Construcción de un grafo dirigido a partir de la gramática.

G2 = ({a,b}, {S, A, B}, S, P}

P = { S ::= bA,
A ::= aB | a
B ::= bA}

paso 2



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas equivalentes

ALGORITMO: 3 PASOS.

PASO 3.

Se intercambian las etiquetas y se construye un nuevo grafo.

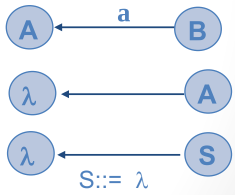
Transformación del grafo dirigido anterior:

a) intercambiar etiquetas de S y λ

b) invertir sentido de todos arcos

c) deshacer el camino y generar las nuevas reglas

$\Rightarrow$ interpretar el grafo para obtener la G3LI equivalente



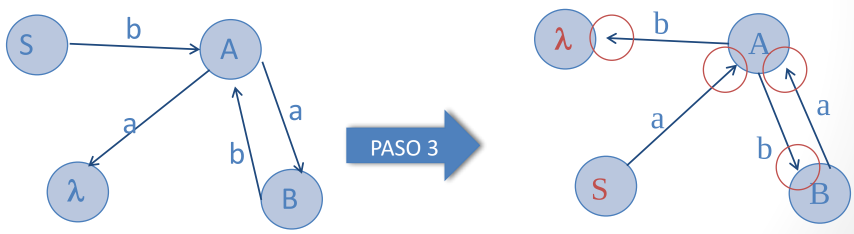
A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas equivalentes

ALGORITMO: 3 PASOS.

PASO 3.

Se intercambian las etiquetas y se construye un nuevo grafo.

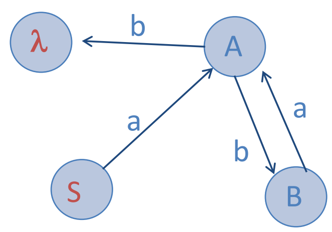


A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas equivalentes

ALGORITMO: 3 PASOS.

A partir del grafo, se obtiene la gramática equivalente a izquierdas.



G3 = ({a,b}, {S, A, B}, S, P}

P = { S ::= Aa,
A ::= Bb | b
B ::= Aa}

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

63

Gramáticas equivalentes

ALGORITMO: 3 PASOS.

Comprobemos que las gramáticas son equivalentes...

G1 = ({a,b}, {S, A}, S, P}

P = { S ::= bA,
A ::= aS | a }

G3 = ({a,b}, {S, A, B}, S, P}

P = { S ::= Aa,
A ::= Bb | b
B ::= Aa}

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

64

Definiciones

Jerarquía de Chomsky

Árboles de derivación

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Árboles de derivación

- ✓ A las derivaciones de las *G tipo 1, 2 y 3* les corresponde un árbol de derivación equivalente, también llamado

  "árbol sintáctico" ó
  "parse tree"

- ✓ Representa las producciones aplicadas durante la generación de **una sentencia**, es decir, su estructura de acuerdo con la G

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Árboles de derivación

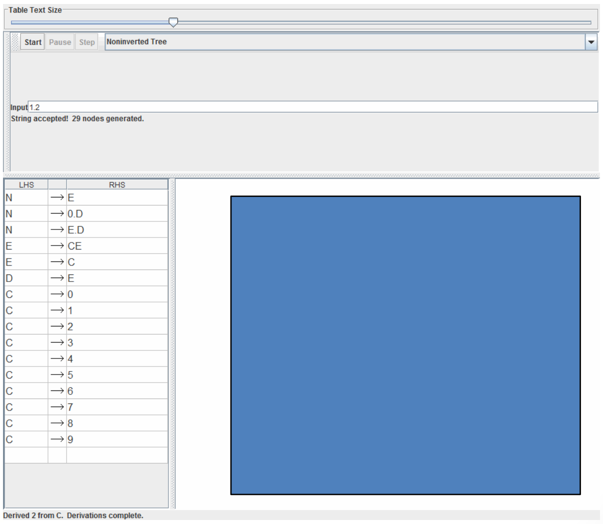
Es un árbol ***ordenado*** y ***etiquetado*** que se construye:

- ✓ La raíz se denota por el axioma de G (S habitualmente)
- ✓ 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 izq. 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:

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

67

Árboles de derivación



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

68

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 **ascendente** 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. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

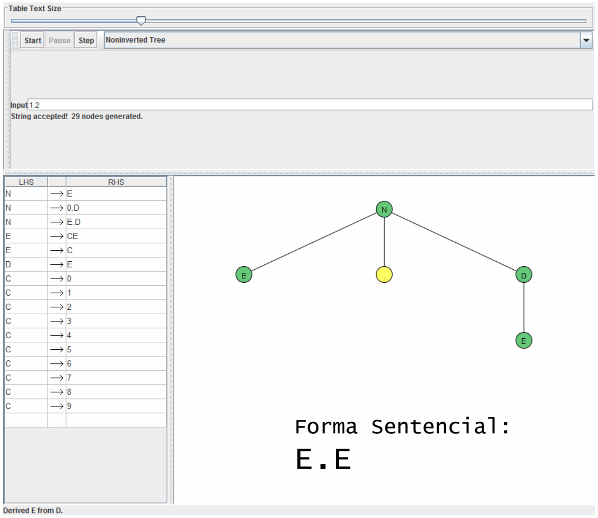
Árboles de derivación

- ✓ A lo largo del proceso de construcción del árbol, los nodos finales de cada paso sucesivo, leídos de izqda. 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 izqda. a dcha. nos dan la ***sentencia*** generada por la derivación

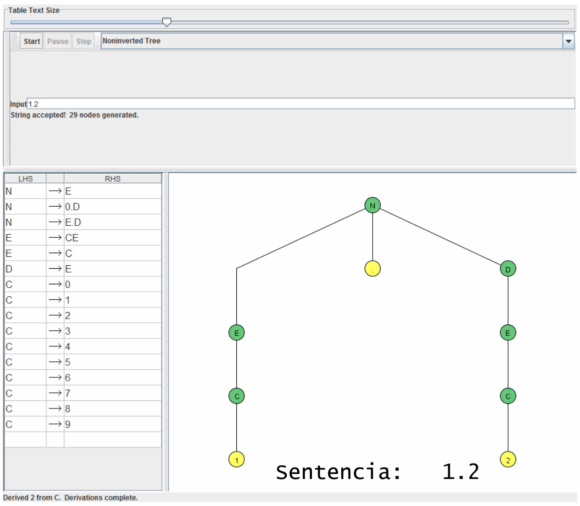
A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Árboles de derivación



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Árboles de derivación



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

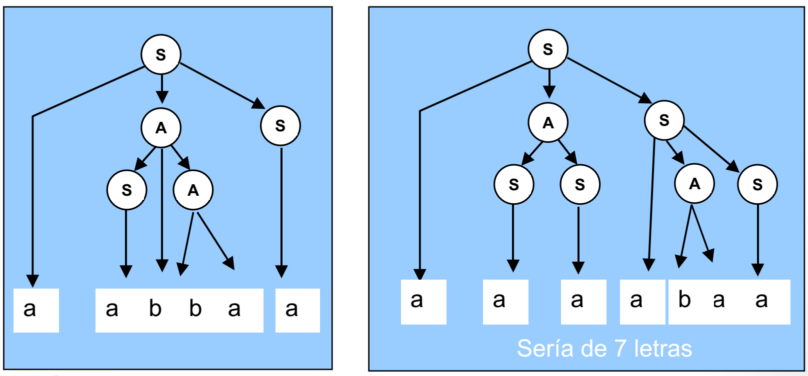
Árboles de derivación

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.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Árboles de derivación

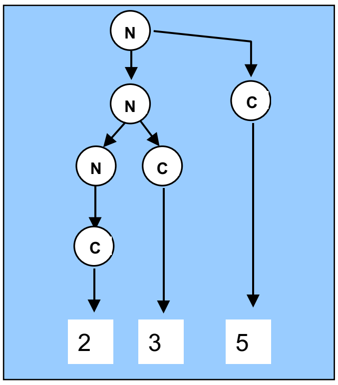
Dada la G = ({a,b}, {A,S}, S, {S::= aAS | a, A::= SbA | SS | ba}). Hallar un árbol de derivación para una sentencia de 6 letras.



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Árboles de derivación

Dada la G = ({0,1,2,3,4,5,6,7,8,9}, {N,C}, N, {N ::= NC | C, C ::= 0|1|2|3|4|5|6|7|8|9}). Hallar un árbol de derivación para N → 235

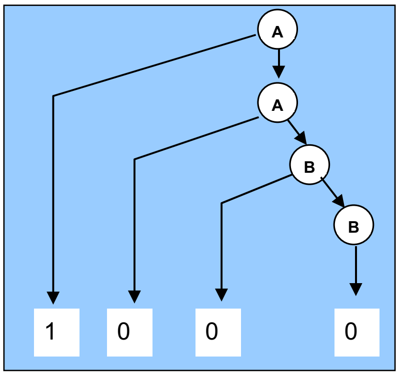


A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

75

Árboles de derivación

Dada la G = ({0,1}, {A,B}, A, {A::= 1A | 0B, B::= 0B | 0}) una de cuyas derivaciones válidas es: A → 1A → 10B → 100B → 1000. Hallar un árbol de derivación para A → 1000



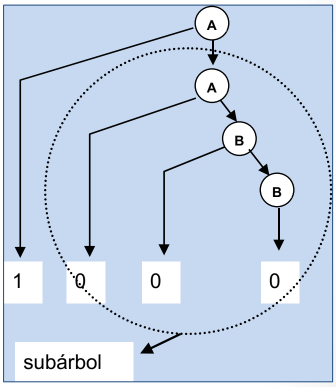
A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

76

Subárboles de derivación

Dado un árbol A correspondiente a una derivación, se llama **subárbol de A** al árbol

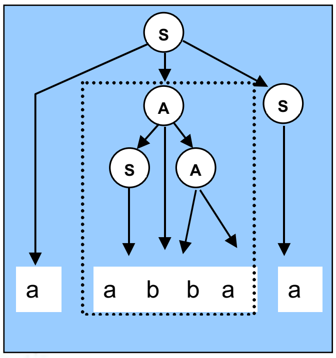
- ✓ cuya raíz es un nodo cualquiera de A,
- ✓ cuyos nodos son todos los descendientes de la raíz del subárbol en A
- ✓ y cuyas ramas son todas las que unen dichos nodos entre si en A.



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**

Ambigüedad

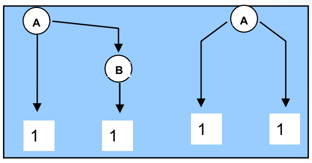
- ✓ Concepto relacionado con el de árbol de derivación:
- ✓ Si una sentencia puede obtenerse en una G por medio de dos o más árboles de derivación diferentes, **la sentencia es ambigua**
- ✓ Una **G** es **ambigua** si contiene **al menos una sentencia ambigua**

Ambigüedad

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 inherentemente ambiguo***: si todas las gramáticas que lo generan son ambiguas.

Ambigüedad

- Aunque una G sea ambigua, es posible que el lenguaje que describe no sea ambiguo [Floyd 1962] ⇒ es posible encontrar una G equivalente que no lo sea
- Existen lenguajes para los que NO es posible encontrar G no ambiguas ⇒ 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.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Ambigüedad

Lenguajes Inherentemente Ambiguos: para los que NO es posible encontrar G no ambiguas

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,B}, A, {A::= 1B / 11, B::= 1})

G’ = ({1}, {A}, A, {A::=11}) --> Gramática NO ambigua

L(G)=L(G’)

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

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**"

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

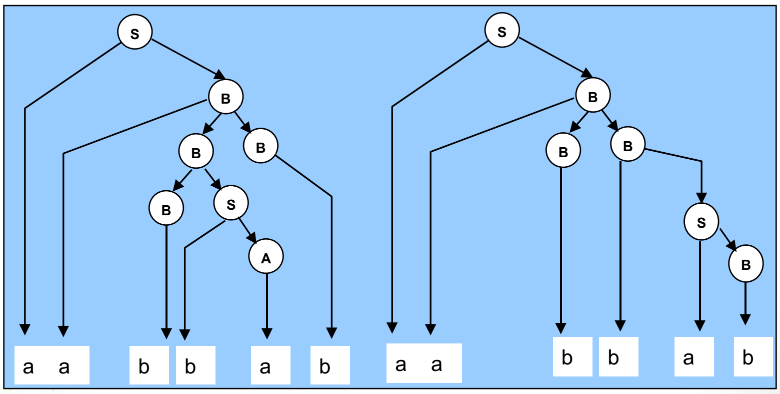
Ambigüedad

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**"



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Segunda parte

GRAMÁTICAS FORMALES

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

- Gramáticas Tipo 2
- Gramáticas bien formadas
- Formas normales

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas Independientes del Contexto

Son las gramáticas de tipo 2 en la jerarquía de Chomsky.

Relevancia

- ✓ son las empleadas en la Definición de Lenguajes de Programación y en la compilación de los mismos

Compilador
- analizador → aceptador de la G
  - explorador (AF)
  - reconocedor (AP)
- generador de código → Traductor

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

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

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas Independientes del Contexto

1. Dada una G, el Lenguaje que genera, ¿es vacío o no?:

Sea **G2**, m = C(Σ_{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(Σ_{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

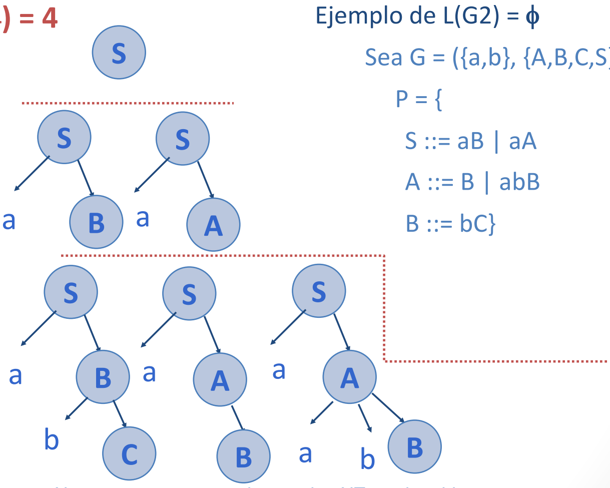
A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

89

Gramáticas Independientes del Contexto

m=C(Σ_{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

Ejemplo de L(G2) = ϕ

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

P = {

S ::= aB | aA

A ::= B | abB

B ::= bC}

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

90

Gramáticas Independientes del Contexto

2. Si L(G2) es no vacío, comprobar si L(G2) = ∝

- ✓ Se construye un grafo cuyos nodos están etiquetados con los símbolos de (Σ_{NT}) mediante el algoritmo:
  - a) si $\exists$ una producción $A ::= \alpha\ B\ \beta$, 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) = ∝ si existen ciclos accesibles desde el axioma que corresponden a derivaciones de la forma $A \rightarrow^{+} \alpha\ A\ \beta$, donde $|\alpha| + |\beta| > 0$ (que no sean λ las dos a la vez).

L(G2) ≠ ∝ si no hay ciclos en el grafo

Gramáticas Independientes del Contexto

Ejemplo de **L(G2) = ∝**

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

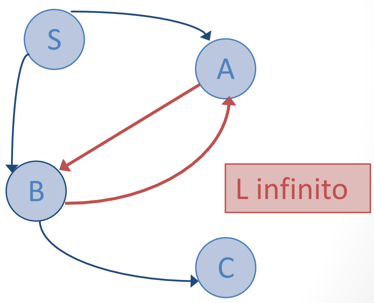
P = {

S ::= aB / aA

A ::= abB

B ::= bC / aA

C ::= c }



Gramáticas Tipo 2

Gramáticas bien formadas

Formas normales

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

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**
   a. Reglas Innecesarias
   b. Símbolos Inaccesibles
   c. Reglas Supérfluas
2. **Eliminación de símbolos no generativos**
3. **Eliminación de reglas no generativas**
4. **Eliminación de reglas de redenominación**

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

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**
   a. Reglas Innecesarias
   b. Símbolos Inaccesibles
   c. Reglas Superfluas
2. **Eliminación de símbolos no generativos**
3. **Eliminación de reglas no generativas**
4. **Eliminación de reglas de redenominación**

Gramática Reducida

Gramática Limpia

Gramática Bien Formada

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas Bien Formadas

1. **Limpieza de Gramáticas**
   a. **Reglas Innecesarias**: las reglas **A::= A** $\in$**P** son innecesarias y hacen que G sea ambigua $\Rightarrow$ eliminarlas

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas Bien Formadas

1. **Limpieza de Gramáticas**
   b. **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 $S \overset{*}{\rightarrow} xUy$.

   Eliminación de símbolos inaccesibles:

   1. Hacer una lista con todos los símbolos de la gramática (T y NT)
   2. Marcar el axioma de la gramática.
   3. Dado *xUy ::= xuy* → Marcar todos los símbolos que aparecen en la cadena *u* de la parte derecha.
   4. Si en el paso anterior se ha marcado algún símbolo, se repite de nuevo dicho paso teniendo en cuenta los símbolos marcados. En caso contrario, fin del algoritmo.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

97

Gramáticas Bien Formadas

1. **Limpieza de Gramáticas**
   b. **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}

1. {a, b, c, S, A, B, C}
2. {a, b, c, S, A, B, C}
3. {a, b, c, S, A, B, C}
4. {a, b, c, S, A, B, C}
5. {a, b, c, S, A, B, C}
6. {a, b, c, S, A, B, C}

Sin cambios. Fin algoritmo

C es un símbolo INACCESIBLE!!

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

98

Gramáticas Bien Formadas

1. **Limpieza de Gramáticas**

   c. **Reglas Superfluas**: son aquellas que no contribuyen a la formación de palabras x $\in \Sigma_T^*$.

   Estas reglas contiene algún **símbolo No Terminal no generativo**.

   Todo símbolo no superfluo debe cumplir U $+\!\!\to$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 superfluos 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 superfluo.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

99

Gramáticas Bien Formadas

1. **Limpieza de Gramáticas**

   c. **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

Es una cadena de Terminales

2ª pasada:

B::= Af

Los NoTerminales están marcados.

3ª pasada:

S::= Be

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

100

Gramáticas Bien Formadas

1. **Limpieza de Gramáticas**
   c. **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

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

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

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**
   a. Reglas Innecesarias
   b. Símbolos Inaccesibles
   c. Reglas Superfluas
2. **Eliminación de símbolos no generativos**
3. **Eliminación de reglas no generativas**
4. **Eliminación de reglas de redenominación**

Gramática Reducida

Gramática Limpia

Gramática Bien Formada

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas Bien Formadas

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

   Sea G2 = (Σ_T, Σ_N, S, P), **$\forall$A$\in\Sigma_N$** 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.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas Bien Formadas

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

   Algoritmo:

   1. Eliminar de la Gramática las regla de la forma U::= λ
   2. Por cada regla de la gramática donde U aparezca en la parte dcha., V::=xUy, se añade la regla V::=xy (a menos que ya existe).
   3. Repetir 2. hasta que no quede ninguna regla de la forma U::= λ, o que sólo quede S::= λ.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas Bien Formadas

3. **Eliminación de reglas no generativas:** son A::= λ (A≠S)

Ejemplo:

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

Pr = { S ::= PQ | aSb | P

P ::= aPQ | a

Q ::= Qb | λ }

Regla NO generativa:
Q ::= λ

ELIMINARLA

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas Bien Formadas

3. **Eliminación de reglas no generativas:** son A::= λ (A≠S)

Ejemplo:

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

Pr = { S ::= PQ | aSb | P

P ::= aPQ | a

Q ::= Qb | λ }

Eliminar la Regla
No generativa

G' = ({a, b}, {S, P, Q}, S, Pr)

Pr = { S ::= PQ | aSb | P

P ::= aPQ | a | aP

Q ::= Qb | b }

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

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**
   a. Reglas Innecesarias
   b. Símbolos Inaccesibles
   c. Reglas Supérfluas
2. **Eliminación de símbolos no generativos**
3. **Eliminación de reglas no generativas**
4. **Eliminación de reglas de redenominación**

Gramática Reducida

Gramática Limpia

Gramática Bien Formada

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas Bien Formadas

4. Eliminación de reglas de redenominación: son reglas del tipo A::=B (donde A ≠ B)

Algoritmo

1. Eliminar de la Gramática las regla de la forma U::= V
2. Por cada regla de la forma V::=x, añadir U::=x (si No Existe).
3. Repetir 2. hasta que no quede ninguna regla de redenominación.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas Bien Formadas

4. **Eliminación de reglas de redenominación:** son reglas del tipo A::=B (donde A ≠ B)

Ejemplo:

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

Pr = { S ::= PQ | aSb | P

P ::= aPQ | aP | a

Q ::= Qb | b }

Regla de Redenominación: S ::= P

ELIMINARLA

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Gramáticas Bien Formadas

4. **Eliminación de reglas de redenominación:** son reglas del tipo A::=B (donde A ≠ B)

Ejemplo:

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

Pr = { S ::= PQ | aSb | P

P ::= aPQ | aP | a

Q ::= Qb | b }

Eliminar la Redenominación

G' = ({a, b}, {S, P, Q}, S, Pr)

Pr = { S ::= PQ | aSb | aPQ | aP | a

P ::= aPQ | aP | a

Q ::= Qb | b }

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

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**
   a. Reglas Innecesarias
   b. Símbolos Inaccesibles
   c. Reglas Supérfluas
2. **Eliminación de símbolos no generativos**
3. **Eliminación de reglas no generativas**
4. **Eliminación de reglas de redenominación**

Gramática Reducida

Gramática Limpia

Gramática Bien Formada

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

- Gramáticas Tipo 2
- Gramáticas bien formadas
- Formas normales

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

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

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Chomsky

Una gramática bien formada está en Forma Normal de Chomsky (FNC) si las partes derechas de todas sus reglas de producción tienen a lo sumo dos símbolos, y cuando son dos, ambos son No Terminales.

Ejemplo: La siguiente gramática está en FNC:

S ::= λ | a | NN

N ::= NT | AB

A ::a

B::= b

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

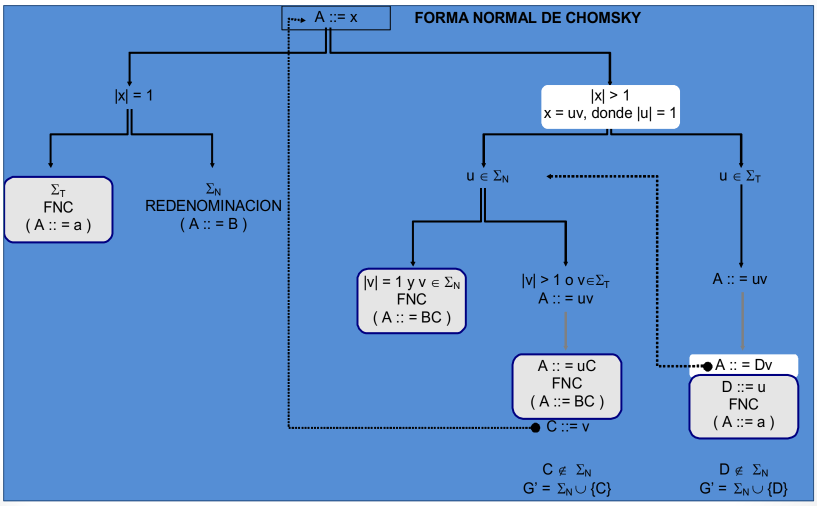
Forma Normal de Chomsky

A partir de cualquier gramática de Tipo 2, se puede construir otra equivalente que está en FNC.

Para ello, se deben sustituir del siguiente modo las reglas cuya parte derecha tiene más de 2 símbolos por varias reglas que tengan en la parte derecha dos símbolos no terminales o un solo símbolo terminal:

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Chomsky



A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Chomsky

EJEMPLO:

G = ({a, b, c}, {A, B, C, D, E, F}, A, P)
P =
B::=B
C::=C
A::=ABC
D::=a
E::=FbB
F::=EaA
A::=bA
B::=Ba
A::=aBb
B::=bAa
A::=a
B::=a
C::=c
C::=Cb

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Chomsky

Bien Formar la Gramática

G = ({a, b, c}, {A, B, C, D, E, F}, A, P)
P =
B::=B ←Innecesaria
C::=C ←Innecesaria
A::=ABC
D::=a ←Inaccesible
E::=FbB ←Innaccesible
F::=EaA ←Innaccesible
A::=bA
B::=Ba
A::=aBb
B::=bAa
A::=a
B::=a
C::=c
C::=Cb

Gramática Bien Formada:
G = ({a, b, c}, {A, B, C}, A, P)
P =
A::=a
A::=aBb
A::=ABC
A::=bA
B::=a
B::=Ba
B::=bAa
C::=c
C::=Cb

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Chomsky

Gramática Bien Formada:
G = ({a, b, c}, {A, B, C, D, E, F}, A, P)
P =

A::=a	Sí (FNC)
A::=aBb	No (FNC)
A::=ABC	No
A::=bA	No
B::=a	Sí
B::=Ba	No
B::=bAa	No
C::=c	Sí
C::=Cb	No

Transformación a FNC (quedan tabuladas las nuevas reglas generadas en FNC)

Tratamiento de A::=aBb
A::=aBb
    D::=a
A::=DBb
E::=Bb
    A::=DE
    F::=b
    E::=BF

Tratamiento de A::=ABC
A::=ABC
    G::=BC
    A::=AG

Tratamiento de A::=bA
A::=bA
    A::=FA

Tratamiento de B::=Ba
B::=Ba
    B::=BD

Tratamiento de B::=bAa
B::=bAa
B::=FAa
H::=Aa
B::=FH
H::=AD

Tratamiento de C::=Cb
C::=Cb
    C::=CF

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

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

1. TEOREMA: todo **L de contexto libre sin** λ puede ser generado por una G2 en la que todas las reglas sean de la forma:
   - **$A \rightarrow a\alpha$** donde $A \in \Sigma_{NT}$,$a \in \Sigma_T$ y $\alpha \in \Sigma_{NT}^*$
   - Si $\lambda \in L$ habrá que añadir **S::=** λ
2. TEOREMA: toda G2 puede reducirse a otra G2 equivalente sin reglas recursivas a izquierdas

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Greibach

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

1. Limpiar y formar bien. 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

EJEMPLO:
G = ({a,b}, {S}, S, P), donde P = {S::=aSb | SS | λ}

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Greibach

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

   sea G = ({α_1, α_2, β_1, β_2}, {A}, A, P),

   donde P = {A::= A α_1| Aα_2 | β_1| β_2}

   Quedaría:

   A::= β_1 | β_2 | β_1X | β_2X
   X::= α_1| α_2 |α_1X| α_2X

Eliminar la recursividad a izquierdas:
S ::= aSb | SS | λ → (se transforma en) → S ::= aSb | aSbX | λ y X ::= SX | S

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

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

A ::= αBβ (A sería nº 1 y B nº 2). α es una cadena de 0 o más terminales y β una cadena de 0 o más símbolos.

B ::= δCγ (B sería nº 2 y C nº 3). δ es una cadena de 0 o más terminales y γ una cadena de 0 o más símbolos.

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Greibach

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

2.2 Se clasifican las reglas en 3 grupos:

Grupo 1: Ai $\rightarrow$ a α ,donde a $\in \Sigma_T$ y $\alpha \in \Sigma^*$

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

Hacer lo mismo con las de grupo 2

Ordenar los no terminales: {X, S} y clasificar las reglas según este orden:
S ::= aSb (G1, aunque hay que quitar el terminal que está detrás del no terminal)
S ::= aSbX (G1, aunque hay que quitar el terminal que está detrás del no terminal)
S ::= λ (G1)
X ::= SX (G2)
X ::= S (G2)

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Greibach

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

2.3 Se transforman las reglas de grupo 3 → grupo 2 → grupo 1: FNG

Ak → 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

Ordenar los no terminales: {X, S} y clasificar las reglas según este orden:
S ::= aSb (G1, aunque hay que quitar el terminal que está detrás del no terminal)
S ::= aSbX (G1, aunque hay que quitar el terminal que está detrás del no terminal)
S ::= λ (G1)
X ::= SX (G2)
X ::= S (G2)

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Greibach

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

2.3 Se transforman las reglas de grupo 3 → grupo 2 → grupo 1: FNG

Ak → Ai α se sustituye **Ai** por la parte dcha. de todas las reglas que tienen **Ai** como parte izda.

Es decir, Sustituir el primer símbolo NT de la parte dcha. de cada regla del grupo 3 por las partes dchas. de todas las reglas (en cualquier grupo) donde dicho primer símbolo aparezca como parte izquierda. Hacer esto hasta que hayamos conseguido que todas las reglas del grupo 3 hayan sido transformadas en reglas de otros grupos. Si en este proceso aparecen reglas recursivas a izquierdas, transformarlas.

Hacer lo mismo con las de grupo 2

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Greibach

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

2.3 Se transforman las reglas de grupo 3 → grupo 2 → grupo 1: FNG

Ak → 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

Transformamos las de grupo G2 en grupo G1:

X ::= SX →S ::= aSb→ X ::= aSbX (G1, aunque hay que quitar el terminal que está detrás del no terminal)

→S ::= aSbX→ X ::= aSbXX (G1, aunque hay que quitar el terminal que está detrás del no terminal)

X ::= S →S ::= aSb→ X ::= aSb (G1, aunque hay que quitar el terminal que está detrás del no terminal)

→S ::= aSbX→ X ::= aSbX (G1, aunque hay que quitar el terminal que está detrás del no terminal)

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

127

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.**

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

↓

$A \rightarrow \alpha\ B\,\beta$
$B \rightarrow a$

Es decir, transformar los símbolos terminales no situados en cabeza de parte derecha, asignándoles nuevos NT que deriven en ellos (o NT que ya existen que sólo deriven en ellos).

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

128

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.**

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

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

Introducimos un nuevo símbolo para quitar la a:

B ::= a

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Forma Normal de Greibach

EJEMPLO

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

P = {S::=aSb | SS | λ}

La Gramática en FNG queda:

G' = ({a,b}, {S,X,B}S, P'})

P' = {
S ::= aSB | aSBX | λ
X ::= aSBX | aSBXX | aSB
B ::= b
}

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso

Bibliografía

- Libro Básico 1 Bibliografía (AAM). Enrique Alfonseca Cubero, Manuel Alfonseca Cubero, Roberto Moriyón Salomón. Teoría de autómatas y lenguajes formales. McGraw-Hill (2007). Capítulo 5

- Libro Básico 2 Bibliografía (HMU). John E. Hopcroft, Rajeev Motwani, Jeffrey D.Ullman. Introducción a la teoría de autómatas, lenguajes y computación (3ª edición). Ed, Pearson Addison Wesley.

- Libro Básico 4 Bibliografía (AAM). Manuel Alfonseca, Justo Sancho, Miguel Martínez Orga. Teoría de lenguajes, gramáticas y autómatas. Publicaciones R.A.E.C. 1997 Capítulo 3

A. Sanchis, A. Ledezma, J. Iglesias, B. García, J. Alonso