

Gramáticas independientes del contexto



Senén Barro Ameneiro, CiTIUS

@SenenBarro

Material elaborado fundamentalmente por
el profesor Manuel Mucientes Molina

Bibliografía

- J.E. Hopcroft, R. Motwani y J.D. Ullman.
"Teoría de Autómatas, Lenguajes y Computación", Addison Wesley, 2008.
 - Capítulo 5
- P. Linz, "An Introduction to Formal Languages and Automata", Jones and Bartlett Publishers, Inc., 2001.
 - Capítulo 5

GIC: lo que vamos a ver

- Lenguajes independientes del contexto (LIC)
- Gramáticas independientes del contexto (GIC)
 - implementación de analizadores sintácticos
 - descripción de formatos de documentos: XML
- Árboles sintácticos: representan gráficamente la estructura de la gramática
- Autómatas con pila

Definición de GIC

Las GIC están formadas por cuatro componentes:

- el conjunto finito de **símbolos no terminales (V)** o variables, que permiten representar subconjuntos del lenguaje o estados intermedios en la generación de las palabras del lenguaje
- el alfabeto de **símbolos terminales (T)**, que son los símbolos finales del lenguaje
- un conjunto finito de **producciones o reglas (P)**, que indican las transformaciones posibles desde los símbolos no terminales a las palabras del lenguaje
- el **símbolo inicial o axioma (S)** de la gramática (una de las variables), a partir del cual se obtiene cualquier palabra del lenguaje

Definición de GIC

Las **reglas de la** gramática están formadas por:

- una variable, cabeza de la producción,
- el símbolo de producción \rightarrow ,
- una cadena de cero o más símbolos terminales y no terminales, que son el cuerpo de la producción

Es decir, las producciones son de la forma $B \rightarrow x$, con $B \in V$ y $x \in (V \cup T)^*$

$$G = (V, T, P, S)$$

- ejemplo: $G_{\text{palíndromo}} = (\{S\}, \{0, 1\}, P, S)$, donde P son las producciones o reglas:

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

Distintas gramáticas

Las **distintas gramáticas** admiten **formas distintas para las producciones**:

Tipo 0, no restringida o recursivamente enumerables

$$\begin{aligned}x &\rightarrow y \\ x &\in (NT/T)^+ \\ y &\in (NT/T)^*\end{aligned}$$

Tipo 1, sensible al contexto

$$\begin{aligned}\alpha &\rightarrow \beta; |\alpha| \leq |\beta| \\ \alpha &= z_1 x z_2 \\ \beta &= z_1 y z_2 \\ z_1, z_2 &\in T^* \\ x &\in NT \\ y &\in (NT/T)^+\end{aligned}$$

Tipo 2, libre de contexto

$$\begin{aligned}x &\rightarrow y \\ x &\in NT \\ y &\in (NT/T)^*\end{aligned}$$

Notemos que se está usando NT (No Terminal) como V

Tipo 3, regular

$$\begin{aligned}\alpha &\rightarrow \beta \\ \alpha &\in NT \\ \beta &\in \begin{cases} aB \\ Ba \\ b \end{cases} \\ B &\in NT \\ a &\in T^+ \\ b &\in T^*\end{aligned}$$

Gramáticas regulares

Los lenguajes regulares, estudiados en capítulos anteriores, pueden asociarse a una **gramática de tipo 3 o regular**

- Gramáticas regulares:
 - lineales por la derecha
 - lineales por la izquierda
- Una gramática $G = (V, T, P, S)$ es lineal por la derecha si todas sus producciones son de la forma:
 - $A \rightarrow xB$
 - $A \rightarrow x$donde A y B pertenecen a V y x pertenece a T^*
- Una gramática $G = (V, T, P, S)$ es lineal por la izquierda si todas sus producciones son de la forma:
 - $A \rightarrow Bx$
 - $A \rightarrow x$

Ejemplo de gramática (GIC)

$G = (\{E, I\}, \{+, *, (,), a, b, 0, 1\}, P, E)$,
donde P es el conjunto de producciones:

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow (E)$
5. $I \rightarrow a$
6. $I \rightarrow b$
7. $I \rightarrow Ia$
8. $I \rightarrow Ib$
9. $I \rightarrow I0$
10. $I \rightarrow I1$

Derivaciones de una gramática

Sea $G = (V, T, P, S)$ y $\alpha A \beta$ una cadena de terminales y no terminales (variables), donde A está en V y α y β están en $(V \cup T)^*$.

Sea $A \rightarrow \gamma$ una producción de G . Entonces:

$$\alpha A \beta \Rightarrow_G \alpha \gamma \beta$$

Una **derivación de una sentencia** ω es la **secuencia** de sustituciones de no terminales que, partiendo del símbolo inicial S , produce como resultado ω .

Derivaciones de una gramática

Ejemplo de derivación de: "a * (a + b00)"

- Derivación más a la izquierda:

$$\begin{aligned} E &\Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow \\ &a * (E) \Rightarrow a * (E + E) \Rightarrow a * (I + E) \Rightarrow a * (a + E) \Rightarrow \\ &a * (a + I) \Rightarrow a * (a + I0) \Rightarrow a * (a + I00) \Rightarrow a * (a + b00) \end{aligned}$$

- Derivación más a la derecha:

$$\begin{aligned} E &\underset{md}{\Rightarrow} E * E \underset{md}{\Rightarrow} E * (E) \underset{md}{\Rightarrow} E * (E + E) \underset{md}{\Rightarrow} \\ &E * (E + I) \underset{md}{\Rightarrow} E * (E + I0) \underset{md}{\Rightarrow} E * (E + I00) \underset{md}{\Rightarrow} E * (E + b00) \underset{md}{\Rightarrow} \\ &E * (I + b00) \underset{md}{\Rightarrow} E * (a + b00) \underset{md}{\Rightarrow} I * (a + b00) \underset{md}{\Rightarrow} a * (a + b00) \end{aligned}$$

Lenguaje de una gramática

Si $G = (V, T, P, S)$ es una GIC, el **lenguaje de G** será:

$$L(G) = \{w \text{ que están en } T^* \mid S \xRightarrow[G]{*} w\}$$

- Si $G = (V, T, P, S)$ es una GIC, cualquier cadena $\alpha \in (V \cup T)^*$, tal que $S \xRightarrow{*} \alpha$ es una **forma sentencial**
- El lenguaje $L(G)$ está formado por las formas sentenciales que están en T^* y se denominan **sentencias**

Ejemplos de formas sentenciales:

$$E \Rightarrow E * E \Rightarrow E * (E) \Rightarrow E * (E + E) \Rightarrow E * (I + E)$$

$$\underset{mi}{E} \Rightarrow \underset{mi}{E * E} \Rightarrow \underset{mi}{I * E} \Rightarrow a * E$$

$$\underset{md}{E} \Rightarrow \underset{md}{E * E} \Rightarrow \underset{md}{E * (E)} \Rightarrow \underset{md}{E * (E + E)}$$

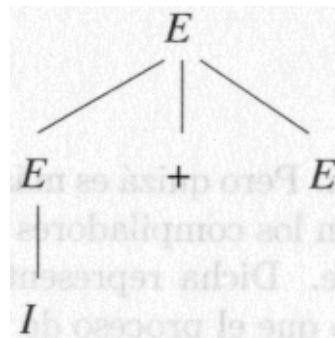
Árboles de derivación

Representación de las derivaciones en forma de árbol

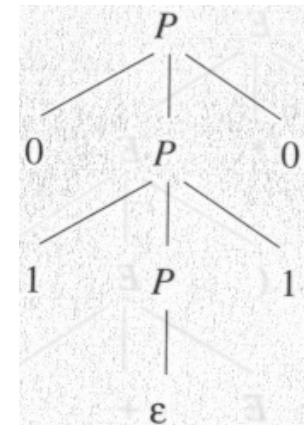
- Sea $G = (V, T, P, S)$. El árbol de derivación para G tendrá las siguientes características:
 - cada nodo interior está etiquetado con una variable
 - cada hoja está etiquetada con una variable, un terminal o ϵ . Si es ϵ , tiene que ser el único hijo de su nodo progenitor
 - si un nodo interior está etiquetado con A y sus hijos están etiquetados con X_1, X_2, \dots, X_k (de izquierda a derecha), entonces $A \rightarrow X_1 X_2 \dots X_k$ es una producción de P

- Ejemplos:

$$E \xRightarrow{*} I + E$$



$$P \xRightarrow{*} 0110$$



Resultado de un árbol de derivación

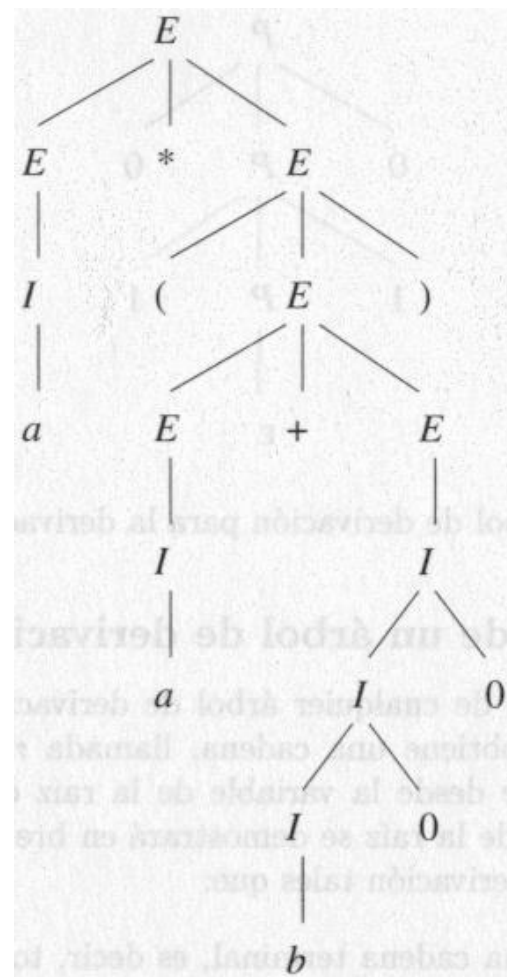
Concatenando la hojas de un árbol desde la izquierda se obtiene la cadena resultado del árbol, que se deriva desde la raíz

Cadenas del lenguaje:

- los nodos hoja son terminales
- la raíz está etiquetada con el símbolo inicial

$$E \stackrel{*}{\vdash} a * (a + b00)$$

1.	$E \rightarrow I$
2.	$E \rightarrow E + E$
3.	$E \rightarrow E * E$
4.	$E \rightarrow (E)$
5.	$I \rightarrow a$
6.	$I \rightarrow b$
7.	$I \rightarrow Ia$
8.	$I \rightarrow Ib$
9.	$I \rightarrow I0$
10.	$I \rightarrow I1$



Aplicaciones de las GIC

Descripción de lenguajes de programación

- Análisis sintáctico

Lenguajes de marcado

- HTML (HyperText Markup Language)

1. $Car \rightarrow a \mid A \mid \dots$
2. $Texto \rightarrow \varepsilon \mid Car \ Texto$
3. $Doc \rightarrow \varepsilon \mid Elemento \ Doc$
4. $Elemento \rightarrow Texto \mid$
 $\quad \quad \quad \ Doc \ \mid$
 $\quad \quad \quad <P> \ Doc \mid$
 $\quad \quad \quad \ Lista \ \mid \dots$
5. $ListItem \rightarrow \ Doc$
6. $Lista \rightarrow \varepsilon \mid ListItem \ Lista$

Figura 5.13. Parte de una gramática de HTML.

Aplicaciones de las GIC

Lenguajes de marcado: XML (eXtensible Markup Language)

- El formato se especifica con un DTD (Document Type Definition)
 - Uso de GIC para describir las etiquetas permitidas y su forma de anidarse

```
<!DOCTYPE PcSpecs [  
  <!ELEMENT PCS (PC*)>  
  <!ELEMENT PC (MODELO, PRECIO, PROCESADOR, RAM, DISCO+)>  
  <!ELEMENT MODELO (#PCDATA)>  
  <!ELEMENT PRECIO (#PCDATA)>  
  <!ELEMENT PROCESADOR (FABRICANTE, MODELO, VELOCIDAD)>  
  <!ELEMENT FABRICANTE (#PCDATA)>  
  <!ELEMENT MODELO (#PCDATA)>  
  <!ELEMENT VELOCIDAD (#PCDATA)>  
  <!ELEMENT RAM (#PCDATA)>  
  <!ELEMENT DISCO (DISCODURO | CD | DVD)>  
  <!ELEMENT DISCODURO (FABRICANTE, MODELO, TAMAÑO)>  
  <!ELEMENT TAMAÑO (#PCDATA)>  
  <!ELEMENT CD (VELOCIDAD)>  
  <!ELEMENT DVD (VELOCIDAD)>  
>
```

Figura 5.14. Una DTD para computadoras personales.

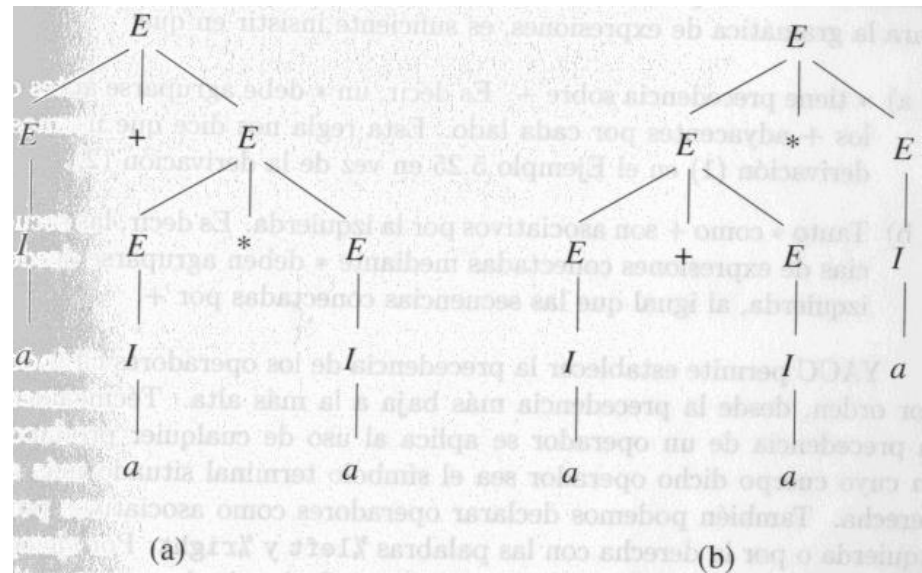
```
<PCS>  
  <PC>  
    <MODELO>4560</MODELO>  
    <PRECIO>$2295</PRECIO>  
    <PROCESADOR>  
      <FABRICANTE>Intel</FABRICANTE>  
      <MODELO>Pentium</MODELO>  
      <VELOCIDAD>800MHz</VELOCIDAD>  
    </PROCESADOR>  
    <RAM>256</RAM>  
    <DISCO><DISCODURO>  
      <FABRICANTE>Maxtor</FABRICANTE>  
      <MODELO>Diamond</MODELO>  
      <TAMAÑO>30.5Gb</TAMAÑO>  
    </DISCODURO></DISCO>  
    <DISCO><CD>  
      <VELOCIDAD>32x</VELOCIDAD>  
    </CD></DISCO>  
  </PC>  
<PC>  
  ...  
</PC>  
</PCS>
```

Figura 5.15. Parte de un documento que obedece a la estructura de la DTD

Ambigüedad

Una GIC $G = (V, T, P, S)$ es ambigua si existe al menos una cadena w en T^* para la que podemos encontrar dos árboles de derivación distintos con la raíz etiquetada con S y cuyo resultado es w

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow (E)$
5. $I \rightarrow a$
6. $I \rightarrow b$
7. $I \rightarrow Ia$
8. $I \rightarrow Ib$
9. $I \rightarrow I0$
10. $I \rightarrow I1$



La existencia de derivaciones diferentes para una cadena no supone un defecto en la gramática; *la existencia de árboles de derivación diferentes sí supone un problema*

$$E \rightarrow E + E \rightarrow I + E \rightarrow a + E \rightarrow a + I \rightarrow a + b$$

$$E \rightarrow E + E \rightarrow E + I \rightarrow E + b \rightarrow I + b \rightarrow a + b$$

Problemas

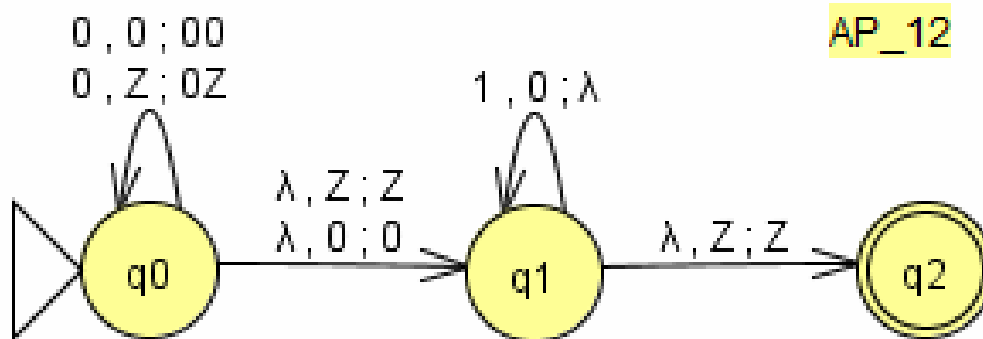
Diseñar la GIC que genere el lenguaje:

$$L = \{0^n 1^n \mid n \geq 0\}$$

$$L = \{\lambda, 01, 0011, 000111\ldots\}$$

$$G = (\{S\}, \{0,1\}, P, S)$$

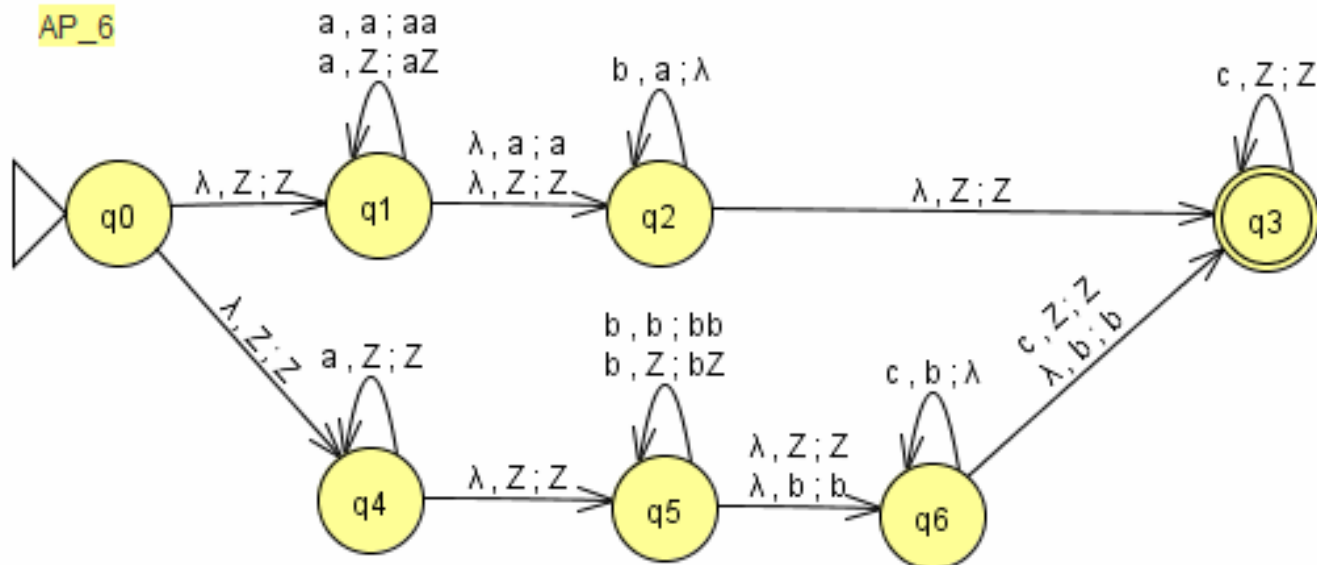
$$P = \{S \rightarrow \lambda \mid 0S1\}$$



Problemas

Diseñar la GIC que genere el lenguaje:

$$L = \{a^i b^j c^k / i = j \text{ ó } j \neq k, \text{ con } i, j, k \geq 0\}$$



Problemas

1. Diseñar la GIC que genere el lenguaje:

$$L = \{0^n 1^n \mid n \geq 1\}$$

1. Diseñar la GIC que genere el lenguaje:

$$L = \{a^i b^j \mid 2i = j; i, j > 0\}$$

1. Diseñar la GIC que genere el lenguaje:

$$L = \{a^i b^j c^k \mid i \neq j \text{ ó } j \neq k\}$$

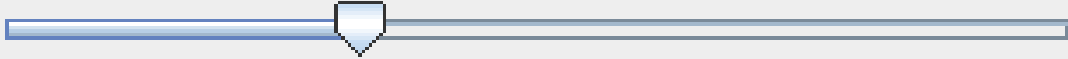
1. Diseñar la GIC que genere el lenguaje:

$$L = \{(a+b+c)^* \mid N(a)+N(b) > N(c)\}$$

Problemas

Diseñar la GIC que genere el lenguaje:

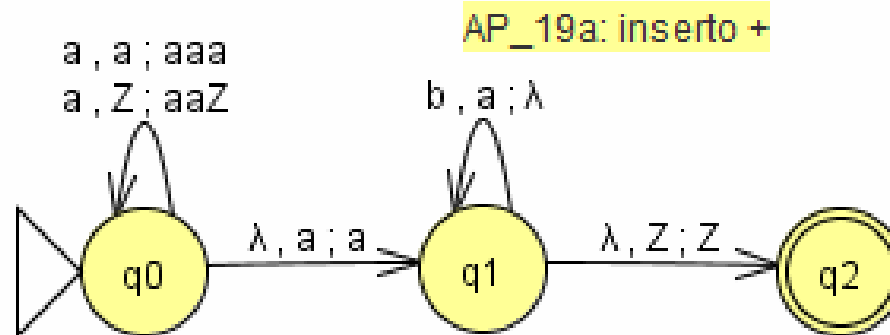
$$L = \{0^n 1^n \mid n \geq 1\}$$

Table Text Size		
		
LHS		RHS
S	→	0S1
S	→	01

Problemas

Diseñar la GIC que genere el lenguaje:

$$L = \{a^i b^j \mid 2i = j; i, j > 0\}$$



Problemas

Diseñar la GIC que genere el lenguaje:

$$L = \{a^i b^j c^k / i \neq j \text{ ó } j \neq k\}$$

Table Text Size		
LHS		RHS
S	→	XC
S	→	AY
X	→	aXb
X	→	aA
X	→	bB
Y	→	bYc
Y	→	bB
Y	→	cC
A	→	aA
A	→	λ
B	→	bB
B	→	λ
C	→	cC
C	→	λ

Problemas

Diseñar la GIC que genere el lenguaje:

$$L = \{(a+b+c)^* \mid N(a)+N(b)>N(c)\}$$

Table Text Size		
LHS		RHS
Z	→	SaS
Z	→	SbS
S	→	bScS
S	→	aScS
S	→	cSaS
S	→	cSbS
S	→	aS
S	→	bS
S	→	λ

Formas normales para GIC

- Las gramáticas en **formas normales** pueden generar todos los LIC
 - Forma Normal de Chomsky
 - Forma Normal de Greibach
- Las gramáticas en formas normales reducen la complejidad para la obtención de las derivaciones
- Para obtener una gramática en forma normal es necesario realizar una serie de transformaciones que no modifican el lenguaje generado:
 - eliminación de producciones ε
 - eliminación de producciones unitarias (reglas de encadenamiento)
 - eliminación de símbolos inútiles

Formas normales para GIC

Eliminación de producciones ε

- Una variable es anulable si $A \Rightarrow^* \varepsilon$
- Algoritmo. Sea $G = (V, T, P, S)$ una GIC. Encontraremos todos los símbolos anulables de G mediante el siguiente algoritmo:
 - Base: si $A \rightarrow \varepsilon$ es una producción de G , A es anulable
 - Paso inductivo: si existe una producción $B \rightarrow C_1 C_2 \dots C_k$ en la que las $C_i, i=1, \dots, k$ son anulables, B es anulable

Formas normales para GIC

Eliminación de producciones ε

- Construcción de una gramática sin producciones ε :
 - sea $G = (V, T, P, S)$ una GIC
 - se determinan todos los símbolos anulables de G
 - se construye la gramática $G_1 = (V, T, P_1, S)$ donde P_1 se determina como sigue:
 - para cada producción $A \rightarrow X_1 X_2 \dots X_k$ donde $k \geq 1$, supongamos que m de los k símbolos son anulables
 - la nueva gramática tendrá 2^m versiones de esta producción (las X_i anulables estarán presentes o ausentes en todas las combinaciones posibles)
 - si, $m = k$ no se incluirá el caso de todas las X_i ausentes
 - además, las producciones de P de la forma $A \rightarrow \varepsilon$ no estarán en P_1
 - si ε forma parte del lenguaje, se añade la producción $S \rightarrow \varepsilon$

Formas normales para GIC

Eliminación de producciones ε

Ejemplos:

- Dada la gramática $S \rightarrow ABC$, $A \rightarrow aAA \mid \varepsilon$, $B \rightarrow bBC \mid \varepsilon$, $C \rightarrow bc \mid \varepsilon$, construir la gramática sin producciones ε
- Dada la gramática $S \rightarrow aS \mid AB \mid AC$, $A \rightarrow aA \mid \varepsilon$, $B \rightarrow bB \mid bS$, $C \rightarrow cC \mid \varepsilon$, construir la gramática sin producciones ε

Formas normales para GIC

Eliminación de producciones unitarias

- Producción unitaria: $A \rightarrow B$, donde A y B son variables
 - añaden pasos adicionales en las derivaciones
- Pares unitarios: pares de variables A y B tales que $A \Rightarrow^* B$, usando una secuencia que sólo hace uso de producciones unitarias
- Obtención de los pares unitarios (A, B)
 - Base: (A, A) es un par unitario para toda variable A
 - Paso inductivo: sea (A, B) un par unitario, y $B \rightarrow C$ una producción donde C es una variable. Entonces (A, C) es un par unitario

Formas normales para GIC

Eliminación de producciones unitarias

Ejemplo: determinar los pares unitarios de la gramática

$$\begin{array}{lcl} I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\ F & \rightarrow & I \mid (E) \\ T & \rightarrow & F \mid T * F \\ E & \rightarrow & T \mid E + T \end{array}$$

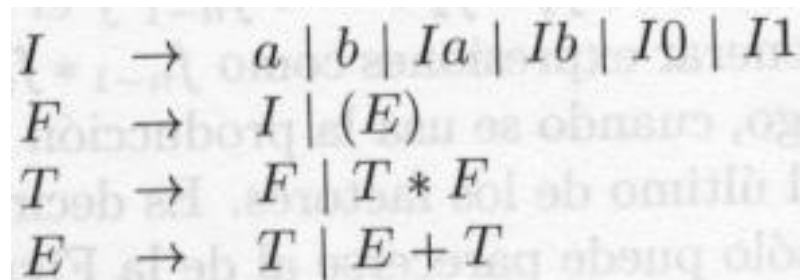
Formas normales para GIC

Eliminación de producciones unitarias

- Eliminación de las producciones unitarias: dada la GIC $G = (V, T, P, S)$, se construye la GIC

$G_1 = (V, T, P_1, S)$ como sigue:

1. encontramos los pares unitarios de G
 2. para cada par unitario (A, B) , añadimos a P_1 todas las producciones $A \rightarrow \alpha$, donde $B \rightarrow \alpha$ es una producción no unitaria de P
- Ejemplo: eliminar las producciones unitarias de la siguiente gramática



$$\begin{array}{lcl} I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\ F & \rightarrow & I \mid (E) \\ T & \rightarrow & F \mid T * F \\ E & \rightarrow & T \mid E + T \end{array}$$

Formas normales para GIC

Eliminación de símbolos inútiles

- Un símbolo X es útil para una gramática $G = (V, T, P, S)$ si existe alguna derivación de la forma $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$, donde w está en T^*
- X es **generador** si $X \Rightarrow^* w$ (w está en T^*)
 - todo símbolo terminal es generador
- X es **alcanzable** si existe una derivación $S \Rightarrow^* \alpha X \beta$ para algún α y β
- **Todo símbolo útil es generador y alcanzable**
- Eliminación de símbolos inútiles
 1. eliminamos los símbolos no generadores
 2. eliminamos los símbolos no alcanzables

Cálculo de símbolos generadores

Algoritmo: sea $G = (V, T, P, S)$ una gramática. Para calcular los símbolos generadores de G se lleva a cabo la siguiente inducción:

- Base: todo símbolo de T es generador
- Paso inductivo: dada una producción $A \rightarrow a$, donde todo símbolo de a es generador, entonces A es generador (se incluye el caso $a = \varepsilon$)

Cálculo de símbolos alcanzables

Algoritmo: sea $G = (V, T, P, S)$ una gramática. Para calcular los símbolos alcanzables de G se lleva a cabo la siguiente inducción:

- Base: S es alcanzable
- Paso inductivo: dada una variable A alcanzable, para todas las producciones cuya cabeza es A , todos los símbolos de los cuerpos de dichas producciones serán alcanzables

Problemas

Dada la gramática siguiente, eliminar producciones ε y unitarias y símbolos inútiles:

$$S \rightarrow AC \mid BS \mid B$$

$$A \rightarrow aA \mid aF$$

$$B \rightarrow CF \mid b$$

$$C \rightarrow cC \mid D$$

$$D \rightarrow aD \mid BD \mid C$$

$$E \rightarrow aA \mid BSA$$

$$F \rightarrow bB \mid b$$

Problemas

S	→	AC
S	→	BS
S	→	cF
S	→	b
D	→	BD
E	→	BSA
C	→	BD
D	→	cC
C	→	aD
F	→	b
F	→	bB
E	→	aA
D	→	aD
C	→	cC
B	→	b
B	→	cF
A	→	aF
A	→	aA

Hay tres producciones unitarias:

(S,B), (C,D) y (D,C),

que han de

reescribirse para

eliminarlas, quedando

como se ve en la

figura

Problemas

S	\rightarrow	BS
S	\rightarrow	cF
S	\rightarrow	b
B	\rightarrow	cF
B	\rightarrow	b
F	\rightarrow	bB
F	\rightarrow	b

Después de eliminar las producciones unitarias y los símbolos inútiles, eliminando los que no son generadores y los que no son alcanzables, queda el conjunto de producciones de la figura

Resumen de las transformaciones

Recordemos: para convertir una GIC G en una GIC equivalente que no tiene símbolos inútiles, producciones ε o producciones unitarias, es necesario realizar los siguientes pasos en el orden establecido:

1. eliminar las producciones ε
2. eliminar las producciones unitarias
3. eliminar los símbolos inútiles

Formal Normal de Chomsky

Todo LIC no vacío tiene una gramática G en la que todas las producciones tienen una de las formas siguientes:

- $A \rightarrow BC$, donde A, B, C son variables
- $A \rightarrow a$, donde A es una variable y a un símbolo terminal
- $S \rightarrow \varepsilon$

Se dice que G está en Forma Normal de Chomsky (FNC)

- Una cadena de longitud n se analiza en $2n-1$ pasos
- El árbol de derivación es binario y su profundidad máxima es n
- Se usa como algoritmo el método CYK

Formal Normal de Chomsky

Ejemplos:

- Gramática: $S \rightarrow AX_1, X_1 \rightarrow BX_2, X_2 \rightarrow CX_3, X_3 \rightarrow DE, A \rightarrow a, B \rightarrow b, C \rightarrow c, D \rightarrow d, E \rightarrow e$. Cadena: abcde
- Gramática: $S \rightarrow SS \mid AB \mid CD, A \rightarrow a, B \rightarrow b, C \rightarrow c, D \rightarrow d$. Cadena: abcdab

Formal Normal de Chomsky

Para transformar una gramática a FNC:

- no puede tener producciones ε , producciones unitarias, ni símbolos inútiles
- las producciones de esa gramática tienen la forma $S \rightarrow \varepsilon$, $A \rightarrow a$ (ya están en FNC) o bien un cuerpo de longitud dos o más. Habrá que:
 - a) conseguir que en los cuerpos de longitud dos o más sólo aparezcan variables
 - b) descomponer los cuerpos de longitud tres o más en una cascada de producciones en cuyos cuerpos sólo aparezcan dos variables

Formal Normal de Chomsky

Construcción para (a):

- para cada símbolo terminal a que aparece en un cuerpo de longitud dos o más, se crea una nueva variable A
- esta variable sólo tendrá la producción $A \rightarrow a$
- se sustituyen las apariciones de a por A siempre que aparezca en un cuerpo de longitud mayor o igual que dos

Construcción para (b):

- se descomponen las producciones de la forma $A \rightarrow B_1 B_2 \dots B_k$ para $k \geq 3$, en un grupo de producciones con dos variables en cada cuerpo
- se introducen $k-2$ variables nuevas, C_1, C_2, \dots, C_{k-2}
- se reemplaza la producción original por las $k-1$ producciones
 $A \rightarrow B_1 C_1, C_1 \rightarrow B_2 C_2, \dots, C_{k-3} \rightarrow B_{k-2} C_{k-2}, C_{k-2} \rightarrow B_{k-1} B_k$

Formal Normal de Chomsky

Ejemplo: convertir la gramática en FNC

$$\begin{aligned} E &\rightarrow E + T \mid T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid IO \mid I1 \\ T &\rightarrow T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid IO \mid I1 \\ F &\rightarrow (E) \mid a \mid b \mid Ia \mid Ib \mid IO \mid I1 \\ I &\rightarrow a \mid b \mid Ia \mid Ib \mid IO \mid I1 \end{aligned}$$
$$\begin{aligned} E &\rightarrow EC_1 \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\ T &\rightarrow TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\ F &\rightarrow LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\ I &\rightarrow a \mid b \mid IA \mid IB \mid IZ \mid IO \\ A &\rightarrow a \\ B &\rightarrow b \\ Z &\rightarrow 0 \\ O &\rightarrow 1 \\ P &\rightarrow + \\ M &\rightarrow * \\ L &\rightarrow (\\ R &\rightarrow) \\ C_1 &\rightarrow PT \\ C_2 &\rightarrow MF \\ C_3 &\rightarrow ER \end{aligned}$$

Formal Normal de Greibach

Todo LIC no vacío es $L(G)$ para alguna gramática G cuyas producciones tienen la forma: $A \rightarrow a\alpha$, donde a es un símbolo terminal y α una cadena de cero o más variables

El uso de una producción introduce un símbolo terminal en una forma sentencial

- una cadena de longitud n tiene una derivación de n pasos
- Un analizador sintáctico descendente parará a profundidad n
- Nunca habrá recursividad por la izquierda

Problemas

1. Encontrar una gramática equivalente
a: $S \rightarrow AB \mid CA, A \rightarrow a, B \rightarrow BC \mid AB,$
 $C \rightarrow aB \mid b$, sin símbolos inútiles
2. Dada la gramática $S \rightarrow ASB \mid \varepsilon, A \rightarrow$
 $aAS \mid a, B \rightarrow SbS \mid A \mid bb$, obtener la
gramática equivalente en FNC