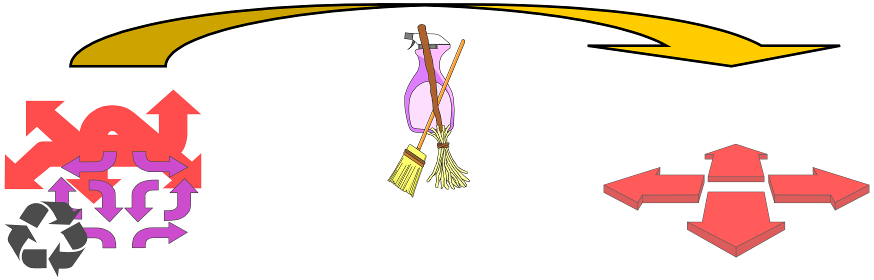
Limpieza de gramáticas y formas normales



César Ignacio García Osorio
Área de Lenguajes y Sistemas Informáticos
Universidad de Burgos

Adecuación de gramáticas

- Eliminación de símbolos no terminables.
- Eliminación de símbolos no alcanzables.
- Eliminación de producciones no generativas.
- Eliminación recursividad izda.
- Factorización izquierda.
- Forma normal Chomsky.
- Lema de bombeo para lenguajes independientes del contexto.
- Algoritmo Cocke-Younger-Kasami (CYK)
- Forma normal de Greibach.

Introducción

- La definición de gramática independiente del contexto proporciona escaso control sobre la clase de producciones permitidas.
- Podemos tener una gramática libre de contexto (*GLC* o *CFG*) con árboles de derivación incontrolablemente tupidos o inútilmente profundos y delgados.
- Sería interesante establecer las restricciones necesarias para que las producciones se formen de manera que el árbol de derivación resultante no sea necesariamente complejo o inútilmente sencillo.
- Se pretende encontrar una forma normal para las gramáticas, el primer paso es "limpiar" las gramáticas.

Eliminación de símbolos no terminables

- **Símbolo no terminable (*SNT*)**: No terminal para el que no es posible construir una derivación que lo convierta en una cadena de terminales ($SNT=\{A\in N : \neg \exists A \Rightarrow^* w \text{ con } w\in\Sigma^*\}$).
- $G=(\Sigma,N,P,S) \longrightarrow G'=(\Sigma,N',P',S)$ con $L(G)=L(G')$ y sin *SNT*

1. Inicializar N' con todos los no terminales A para los que $A\rightarrow w$, es una producción de G, con $w\in\Sigma^*$
2. Inicializar P' con todas las producciones $A\rightarrow w$ para las cuales $A\in N'$ y $w\in\Sigma^*$
3. Repetir hasta que no se puedan añadir más no terminales a N'

   Añadir a N' todos los no terminales A para los cuales $A\rightarrow w$, para algún $w\rightarrow(N'\cup\Sigma)^*$ que sea una producción de P y añadirla a P'

Eliminación de símbolos no alcanzables

- **Símbolo no alcanzable (*SNA*)**: Aquel para el que no es posible llegar partiendo del axioma de la gramática ($SNA=\{A\in N : \neg\exists S\Rightarrow^* wAv$ con $w,v\in(\Sigma\cup N)^*\}$).
- $G=(\Sigma,N,P,S) \longrightarrow G'=(\Sigma',N',P',S)$ con $L(G)=L(G')$ y sin *SNA*

1. Inicializar N' de forma que contenga el símbolo inicial *S*, e inicializar P' y Σ' a $\varnothing$
2. Repetir hasta que no se puedan añadir nuevas producciones

   Para $A\in N'$, si $A\rightarrow w$ es una producción de *P*, entonces:
   1. Introducir $A\rightarrow w$ en P'
   2. Para todo no terminal *B* de *w*, introducir *B* en N'
   3. Para todo terminal σ de *w*, introducir Σ'

Eliminación de producciones epsilon

- Primero se necesita identificar los no terminales anulables.
- **Símbolo anulable**: Se dice que *A* es anulable si es posible derivar a partir de él la cadena vacía ($anulables=\{A\in N : A\Rightarrow^* \varepsilon\}$).
- $G=(\Sigma,N,P,S) \longrightarrow$ conjunto A de no terminales anulables.

1. Inicializar A con todos los no terminales *A* para los cuales existe una producción ε, $A\rightarrow\varepsilon$
2. Repetir hasta que no se añadan más no terminales a A :

   Si $B\rightarrow w$ para algún $w\in(N\cup\Sigma)^*$ y todos los símbolos de *w* están en A, añadir *B* a A.

- Ahora se sustituyen las producciones $B\rightarrow X_1\ldots X_n$ por todas las formas posibles que se obtienen al considerar los sucesivos *Xi* que sean anulables (si hay *m* X_i anulables, 2^m-1 formas; $B\rightarrow\varepsilon$ no se considera).

Eliminación de producciones no generativas

- **Producción no generativa (*PNG*)**: Son las producciones de la forma $A\rightarrow B$. Se llaman también producciones unitarias.
- Para $A\in N$ se define

  $Unitario(A)=\{B\in N \mid A\Rightarrow^* B$ usando sólo producciones unitarias$\}$
- $G=(\Sigma,N,P,S) \longrightarrow G'=(\Sigma,N,P',S)$ con $L(G)=L(G')$ y sin *PNG*

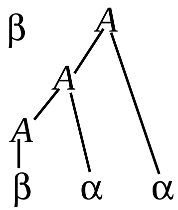
1. Inicializar P' de forma que contenga todos los elementos de *P*
2. Para cada *A* , obtener el conjunto *Unitario*(*A*)
3. Para cada *A* para el cual $Unitario(A)\neq\{A\}$

   Para cada $B\in Unitario(A)$ y para cada producción no unitaria $B\rightarrow w$ de *P* añadir $A\rightarrow w$ a P'
4. Eliminar todas las producciones unitarias que haya en P'

Eliminación de recursividad 1

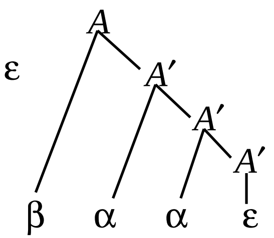
- **Recursividad directa**
- Una gramática *G* es recursiva a izquierdas si existe un no terminal *A* para el cual $A\Rightarrow^+ A\alpha$.

$A\rightarrow A\alpha \mid \beta$



$A\rightarrow\beta A'$

$A'\rightarrow\alpha A' \mid \varepsilon$



- Si hay más de dos producciones
  - Estratificación: $A\rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$
  - Reescritura: $A\rightarrow\beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A'$

    $A'\rightarrow\alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A' \mid \varepsilon$

Eliminación de recursividad 2

- **Recursividad indirecta**
- La recursividad se alcanza a través de varias producciones.

  $A \to B\alpha \mid \alpha'$ $\quad B \to C\beta \mid \beta'$ $\quad C \to A\gamma \mid \gamma'$
- $G=(\Sigma, N, P, S)$ sin producciones ε y sin ciclos
  $\to G'=(\Sigma, N', P', S)$ no recursiva con $L(G)=L(G')$

1. Indexar los no terminales: $A_1, A_2, ..., A_n$
2. **for** i:=1 **to** n **do**
   for j:=1 **to** i-1 **do**
   a. sustituir cada producción de la forma $A_i \to A_j\gamma$ por las producciones $A_i \to \delta_1\gamma \mid \delta_2\gamma \mid ... \mid \delta_k\gamma$ donde $A_j \to \delta_1 \mid \delta_2 \mid ... \mid \delta_k$ son todas las producciones actuales de A_j
   b. eliminar la recursividad directa por la izquierda de A_i

Factorización por la izquierda

- $G=(\Sigma, N, P, S) \to G'=(\Sigma, N, P', S)$ factorizada con $L(G)=L(G')$

1. Para cada no terminal A, encontrar el prefijo α más largo común a dos o más de sus alternativas
2. Si $\alpha \neq \varepsilon$ (es decir, existe un prefijo) transformar

   $A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \ldots \mid \alpha\beta_n \mid \gamma$

   (γ: alternativas que no comienzan por α)

   en $A \to \alpha A' \mid \gamma$

   $A' \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$

- **Idea**: cuando en el análisis descendente no este claro cuál de dos producciones alternativas utilizar para ampliar un no terminal A, se podrán reescribir las producciones de A para retrasar la decisión hasta haber visto suficiente entrada.

Forma normal de Chomsky 1

- Una *GLC* se dice que esta en *forma normal de Chomsky* si no contiene producciones ε y si todas las producciones son de la forma $A \to a$ para $a \in \Sigma$, o de la forma $A \to BC$, donde B y C son no terminales.
- Así se consigue que el árbol de derivación sea un árbol binario.
- Si G es una *GLC* y $\varepsilon \notin L(G)$, G puede ser transformada en una gramática en forma normal de Chomsky.
- Para ello, **primero** se eliminan todas las producciones ε, los símbolos inútiles (*SNT* o *SNA*) y las producciones unitarias de G.
- Una vez realizado lo anterior, si $A \to w$ es una producción de G, se puede asegurar que $|w| \geq 1$. Es más, si $|w|=1$ entonces w es un símbolo terminal de Σ, puesto que no hay producciones unitarias.
- Por otro lado, si $|w|>1$, entonces w puede contener tanto terminales como no terminales.

Forma normal de Chomsky 2

- **Ahora** transformaremos G convirtiendo tales w en cadenas que contengan sólo no terminales.
- Para cada producción $A \to w$ con $w=X_1X_2...X_n$, si X_i es un símbolo terminal σ, sustituimos X_i por un nuevo no terminal C_σ y añadimos la producción. $C_\sigma \to \sigma$
- La **última** etapa consiste en eliminar las cadenas con más de dos no terminales consecutivos en los lados derechos de las producciones. Para ello , si $A \to B_1B_2...B_n$, es una producción con $n \geq 2$, la reemplazaremos por n-1 producciones:

$$\begin{aligned} A &\to B_1D_1 \\ D_1 &\to B_1D_2 \\ &\vdots \\ D_{n-2} &\to B_{n-1}B_n \end{aligned}$$

Lema de bombeo

- **Lema**: Sea L un lenguaje independiente del contexto que no contiene ε. Entonces existe un entero k para el cual, si $z\in L$ y $|z|>k$, entonces z se puede volver a escribir como $z=uvwxy$ con las propiedades siguientes:
  1. $|vwx|\leq k$
  2. Bien v, bien x no es ε (es decir $|vx| \geq 1$)
  3. $uv^iwx^iy\in L$ para todo $i\geq 0$
- Se puede usar este lema para demostrar que un lenguaje no es independiente del contexto.
- También se puede usar para demostrar que un lenguaje independiente del contexto es infinito.

Algoritmo CYK (1)

- Sea $G=(\Sigma,N,P,S)$ una gramática independiente del contexto que no tiene producciones ε y que *está en forma normal de Chomsky*. Sea x una cadena de Σ^*. Se puede determinar, para cada $A\in N$ y para cada subcadena w de x, si $A\Rightarrow^* w$.
- Si w_{ij} es una subcadena de x de longitud j que empieza en i:
  - Si $j=1$, $|w_{ij}|=1$ y si existe algún no terminal para el cual $A\Rightarrow^* w_{ij}$ es porque existe la producción $A\rightarrow w_{ij}$ (como P es finito es fácil encontrar la producción),
  - Supongamos que $j>1$ y que el teorema se cumple para subcadenas de longitud menor que j. Si existe la derivación $A\Rightarrow^* w_{ij}$ el primer paso tiene que ser de la forma $A\rightarrow BC$ y para algún k entre 1 y j-1 se tiene $B\Rightarrow^* w_{ik}$ y $C\Rightarrow^* w_{i+k,j-k}$, como w_{ik} y $w_{i+k,j-kj}$ tiene longitud menor que j se puede aplicar la hipótesis de inducción.

Algoritmo CYK (2)

- El algoritmo de CYK se enuncia como sigue:

1. Para cada i=1,2, ..., n, sea

$$N_{i1}=\{A: A\rightarrow w_{i1}\}$$

   Es decir, N_{i1} es el conjunto de todos los no terminales que producen el i-ésimo símbolo de x.

2. Para j=2,3, ..., n, hacer lo siguiente:

   Para i=1,2, ..., n-j+1, hacer lo siguiente:

   a. Inicializar $N_{ij}=\emptyset$.

   b. Para k=1,2, ..., j-1, añadir a N_{ij} todos los no terminales A para los cuales $A\rightarrow BC$, con $B\in N_{ik}$ y $B\in N_{i+k,j-k}$.

3. Si $S\in N_{1n}$, entonces $x\in L(G)$.

Forma normal de Greibach

- Una gramática independiente del contexto está en *forma normal de Greibach* (*FNG*) si todas las producciones son dela forma $A\rightarrow a\alpha$, donde a es un símbolo terminal y $\alpha\in(\Sigma\cup N)^*$.