Propiedades de los Lenguajes Libres de Contexto

INAOE

Contenido

Forma Normal de Chomsky (CNF)

Queremos mostrar que todo CFL (sin ϵ) se genera por una CFG donde todas las producciones son de la forma:
$A \rightarrow BC$ o $A \rightarrow a$
Donde A, B, y C son variables, y a es un símbolo terminal. A esto se le conoce como CNF, y para llegar a ella debemos:

1. Eliminar símbolos no-útiles, aquellos que no aparecen en ninguna derivación $S \stackrel{*}{\Rightarrow} w$, para el símbolo de inicio S y la cadena de símbolos terminales w.
2. Eliminar las producciones-ϵ, es decir, producciones de la forma $A \rightarrow \epsilon$.
3. Eliminar producciones unitarias, es decir, producciones de la forma $A \rightarrow B$, donde A y B son variables.

Eliminando símbolos no-útiles

- Un símbolo X es *útil* para una gramática $G = (V, T, P, S)$, si hay una derivación: $S \stackrel{*}{\Rightarrow}_G \alpha X \beta \stackrel{*}{\Rightarrow}_G w$ para una cadena terminal w. A los símbolos que no son útiles se les denomina *inútiles*.
- Un símbolo X es *generador* si $X \stackrel{*}{\Rightarrow}_G w$, para alguna $w \in T^*$.
- Un símbolo X es *alcanzable* si $S \stackrel{*}{\Rightarrow}_G \alpha X \beta$, para algún $\alpha, \beta \subseteq (V \cup T)^*$.
- Un símbolo útil es generador y alcanzable.
- Cabe notar que si eliminamos a los símbolos no-generadores primero, y luego a los no-alcanzables, nos quedamos únicamente con símbolos útiles.

Ejemplo

- Sea G: $S \rightarrow AB|a, A \rightarrow b$.
- S y A son generadores, B no lo es. Si eliminamos B tenemos que eliminar $S \rightarrow AB$, dejando la gramática $S \rightarrow a, A \rightarrow b$. Ahora sólo S es alcanzable.
- Eliminando A y b nos deja con $S \rightarrow a$. Con el lenguaje $\{a\}$.
- El orden importa, de otra manera (para este ejemplo), si eliminamos primero los símbolos no-alcanzables, nos damos cuenta de que todos los símbolos son alcanzables.
- A partir de: $S \rightarrow AB|a, A \rightarrow b$. Si eliminamos los inalcanzables y después a B como no-generador, nos quedamos con $S \rightarrow a, A \rightarrow b$, que todavía contiene símbolos inútiles.

Teorema

Sea $G = (V, T, P, S)$ una CFG tal que $L(G) \neq \emptyset$. Sea $G_1 = (V_1, T_1, P_1, S)$ la gramática obtenida:

1. Eliminando todos los símbolos no-generadores y las producciones en las que ocurren. Sea la nueva gramática $G_2 = (V_2, T_2, P_2, S)$.
2. Eliminando de G_2 todos los símbolos no-alcanzables y las producciones en que ocurren.

G_1 no tiene símbolos inútiles, y $L(G_1) = L(G)$.

Cálculo de Símbolos Generadores y Alcanzables

- Necesitamos algoritmos para calcular los símbolos generadores y alcanzables de $G = (V, T, P, S)$.
- Los símbolos generadores $g(G)$ se calculan con el siguiente algoritmo de cerradura:
  1. **Base:** Todo símbolo de T es generador, se genera a sí mismo.
  2. **Inducción:** Suponemos que tenemos la producción $A \rightarrow \alpha$, y cada símbolo de α es generador. Entonces A es generador (esto incluye $\alpha = \epsilon$, las reglas que tienen a ϵ en el cuerpo son generadoras).

Ejemplo

- Sea G: $S \rightarrow AB|a, A \rightarrow b$
- Entonces, primero $g(G) = \{a, b\}$.
- Como $S \rightarrow a$ ponemos a S en $g(G)$, y porque $A \rightarrow b$ añadimos también a A, y eso es todo, el conjunto de símbolos generadores es $\{a, b, A, S\}$.

Teorema

- El algoritmo anterior encuentra todos y sólo los símbolos generadores de G.
- El conjunto de símbolos alcanzables $r(G)$ de $G = (V, T, P, S)$ se calcula con el siguiente algoritmo de cerradura:
  1. *Base*: $r(G) = \{S\}$, S es alcanzable.
  2. *Inducción*: Si la variable $A \in r(G)$ y $A \rightarrow \alpha \in P$ entonces se añaden todos los símbolos de α a $r(G)$.

Ejemplo

- Sea $G : S \rightarrow AB|a, A \rightarrow b$
- Entonces, primero $r(G) = \{S\}$.
- Con base en la primera producción añadimos $\{A, B, a\}$ a $r(G)$.
- Con base en la segunda producción añadimos $\{b\}$ a $r(G)$ y eso es todo.

Teorema: El algoritmo anterior encuentra todos y solo los símbolos alcanzables de G.

Eliminando Producciones-ϵ

- Aunque las producciones ϵ son convenientes, no son esenciales. Si L es CF, entonces $L - \{\epsilon\}$ tiene una CFG sin producciones ϵ.
- La estrategia consiste en descubrir cuáles variables son *nulificables*.
- Se dice que la variable A es *nulificable* si $A \stackrel{*}{\Rightarrow} \epsilon$.
- Sea A nulificable, entonces en todas las producciones en donde A aparece en el cuerpo, digamos $B \rightarrow CAD$, creamos dos versiones de la producción, una sin A, $B \rightarrow CD$ y otra con A, $B \rightarrow CAD$. Si utilizamos la producción con A no permitimos que A derive a ϵ.

Algortimo

- El siguiente algoritmo calcula $n(G)$, el conjunto de símbolos nulificables de una gramática $G = (V, T, P, S)$ como sigue:
  1. *Base*: $n(G) = \{A : A \rightarrow \epsilon \in P\}$
  2. *Inducción*: Si $\{C_1, C_2, \ldots, C_k\} \subseteq n(G)$ y $A \rightarrow C_1, C_2, \ldots, C_k \in P$, entonces $n(G) = n(G) \cup \{A\}$.
  3. Nota, cada C_i debe ser una variable para ser nulificable, entonces se consideran sólo las producciones con cuerpos conformados de variables.

Teorema

- En cualquier gramática G, los únicos símbolos nulificables son las variables encontradas por el algoritmo anterior.
- Una vez que conocemos los símbolos nulificables, podemos transformar G en G_1 como sigue:
  1. Para cada $A \rightarrow X_1X_2 \ldots X_k \in P$ con $m \leq k$ símbolos nulificables, reemplazar por 2^m reglas, una con cada sub-lista de los símbolos nulificables ausentes.
  2. Excepción: Si $m = k$ no añadimos la regla donde borramos todos los m símbolos nulificables.
  3. Borrar todas las reglas de la forma $A \rightarrow \epsilon$.

Ejemplo

- Considere la siguiente gramática:
  $S \rightarrow AB$
  $A \rightarrow aAA|\epsilon$
  $B \rightarrow bBB|\epsilon$
- A y B son nulificables porque tienen a ϵ en el cuerpo de una de sus producciones.
- S también es nulificable, porque $S \rightarrow AB$ tiene puros símbolos nulificables.
- Ahora para construir las nuevas producciones sin ϵ, consideremos la primera: $S \rightarrow AB$.
- De aqui construimos las producciones con y sin los símbolos nulificables (y sin eliminar todas):
  $S \rightarrow AB|A|B$.

Ejemplo (cont.)

- Para $A \rightarrow aAA$, hacemos algo parecido y nos queda: $A \rightarrow aAA|aA|aA|a$, que como hay dos iguales podemos elimiar una.
- Finalmente para B es parecido, por lo que la gramática final queda como:
  $S \rightarrow AB|A|B$
  $A \rightarrow aAA|aA|a$
  $B \rightarrow bBB|bB|b$
- La gramática anterior no cambia el lenguaje, excepto que ϵ ya no está presente.

Teorema

- Si la gramática G_1 se construye a partir de G con la construcción anterior para eliminar producciones ϵ, entonces $L(G_1) = L(G) - \{\epsilon\}$.

Eliminando Producciones Unitarias

- $A \rightarrow B$ es una producción unitaria, cuando A y B son variables. Las producciones unitarias se pueden eliminar.
- Veamos la gramática:
  $I \rightarrow a|b|Ia|Ib|I0|I1$
  $F \rightarrow I|(E)$
  $T \rightarrow F|T * F$
  $E \rightarrow T|E + T$
  tiene las producciones unitarias $E \rightarrow T$, $T \rightarrow F$, y $F \rightarrow I$.
- Podemos expandir T en la producción $E \rightarrow T$ y obtener: $E \rightarrow F|T * F$.
- Expandiendo $E \rightarrow F$ nos da: $E \rightarrow I|(E)$.
- Finalmente expandemos $E \rightarrow I$ y obtenemos: $E \rightarrow a|b|Ia|Ib|I0|I1|(E)|T * F|E + T$

Eliminando Producciones Unitarias

- El método de expansión trabaja siempre y cuando no haya ciclos en las reglas, por ejemplo en: $A \rightarrow B, B \rightarrow C, C \rightarrow A$.
- Para calcular $u(G)$, el conjunto de todos los pares unitarios de $G = (V, T, P, S)$ utilizamos el siguiente algoritmo de cerradura.

Algoritmo

- *Base*: (A, A) es un par unitario para cualquier variable A. Esto es, $A \stackrel{*}{\Rightarrow} A$ en cero pasos. $u(G) = \{(A, A) : A \in V\}$.
- *Inducción*: Suponemos que $(A, B) \in u(G)$ y que $B \rightarrow C \in P$ donde C es una variable. Entonces añadimos (A, C) a $u(G)$.

Teorema: El algoritmo anterior encuentra todos y solo los pares unitarios de una CFG G.

Algoritmo

Para eliminar producciones unitarias, procedemos de la siguiente manera. Dada $G = (V, T, P, S)$, podemos construir $G_1 = (V, T, P_1, S)$:

1. Encontrando todos 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 en P.
3. Note que es posible tener $A = B$; de esta manera, P_1 contiene todas las producciones unitarias en P.
4. $P_1 = \{A \rightarrow \alpha : \alpha \notin V, B \rightarrow \alpha \in P, (A, B) \in u(G)\}$

Ejemplo

A partir de la gramática:

- $I \rightarrow a|b|Ia|Ib|I0|I1$
- $F \rightarrow I|(E)$
- $T \rightarrow F|T * F$
- $E \rightarrow T|E + T$

Creamos un nuevo conjunto de producciones usando el primer elemento del par como cabeza y todos los cuerpos no unitarios del segundo elemento del par como cuerpos de las producciones:

Ejemplo (cont.)

Par	Producción
(E, E)	$E \rightarrow E + T$
(E, T)	$E \rightarrow T * F$
(E, F)	$E \rightarrow (E)$
(E, I)	$E \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(T, T)	$T \rightarrow T * F$
(T, F)	$T \rightarrow (E)$
(T, I)	$T \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(F, F)	$F \rightarrow (E)$
(F, I)	$F \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(I, I)	$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Ejemplo (cont.)

Eliminamos las producciones unitarias. La gramática resultante es equivalente a la original.

$E \rightarrow E + T | T * F | (E) | a | b | Ia | Ib | I0 | I1$
$T \rightarrow T * F | (E) | a | b | Ia | Ib | I0 | I1$
$F \rightarrow (E) | a | b | Ia | Ib | I0 | I1$
$I \rightarrow a | b | Ia | Ib | I0 | I1$

Resumen

Para “limpiar” una gramática podemos:

1. Eliminar producciones-ϵ
2. Eliminar producciones unitarias
3. Eliminar símbolos inútiles

en este orden.

Forma Normal de Chomsky, CNF

Forma Normal de Chomsky
Eliminando Producciones-ϵ
Eliminando Producciones Unitarias
Forma Normal de Chomsky, CNF
Pumping Lemma para CFL
Propiedades de Cerradura de los CFLs

- Ahora se mostrará que cada CFL no vacío sin ϵ tiene una gramática G sin símbolos inútiles, de tal manera que cada producción tenga la forma: $A \rightarrow BC$, donde $\{A, B, C\} \subseteq T$, o $A \rightarrow \alpha$, donde $A \in V$, y $\alpha \in T$.
- Para lograr esto, iniciamos con alguna gramática para el CFL, y:
  1. "Limpiamos la gramática".
  2. Hacemos que todos los cuerpos de longitud 2 o más consistan solo de variables.
  3. Dividimos los cuerpos de longitud 3 o más en una cascada de producciones con cuerpos de dos variables.

Forma Normal de Chomsky, CNF

- Para el paso 2, por cada terminal *a* que aparece en un cuerpo de longitud ≥ 2, creamos una nueva variable, A, y reemplazamos a *a* por A en todos los cuerpos. Después añadimos una nueva regla $A \rightarrow a$.
- Para el paso 3, por cada regla de la forma $A \rightarrow B_1 B_2 \ldots B_k$, $k \geq 3$, introducimos variables nuevas $C_1, C_2, \ldots C_{k-2}$, y reemplazamos la regla con:
  $A \rightarrow B_1 C_1$
  $C_1 \rightarrow B_2 C_2$
  $\ldots$
  $C_{k-3} \rightarrow B_{k-2} C_{k-2}$
  $C_{k-2} \rightarrow B_{k-1} B_k$

Ejemplo

Iniciamos con la gramática (el paso 1 ya está hecho):

- $E \rightarrow E + T|T * F|(E)|a|b|Ia|Ib|I0|I1$
- $T \rightarrow T * E|(E)|a|b|Ia|Ib|I0|I1$
- $F \rightarrow (E)|a|b|Ia|Ib|I0|I1$
- $I \rightarrow a|b|Ia|Ib|I0|I1$

Para el paso 2, introducimos nuevas variables y nos quedan las siguientes reglas:
$A \rightarrow a, B \rightarrow b, Z \rightarrow 0, O \rightarrow 1$
$P \rightarrow +, M \rightarrow *, L \rightarrow (, R \rightarrow)$

Ejemplo (cont.)

y al reemplazar obtenemos la gramática:

$E \rightarrow EPT|TMF|LER|a|b|IA|IB|IZ|IO$

$T \rightarrow TPE|LEL|a|b|IA|IB|IZ|IO$

$F \rightarrow LER|a|b|IA|IB|IZ|IO$

$I \rightarrow a|b|IA|IB|IZ|IO$

$A \rightarrow a, B \rightarrow b, Z \rightarrow 0, O \rightarrow 1$

$P \rightarrow +, M \rightarrow *, L \rightarrow (, R \rightarrow)$

Ejemplo (cont.)

Para el paso 3, reemplazamos:

- $E \rightarrow EPT$ por $E \rightarrow EC_1, C_1 \rightarrow PT$
- $E \rightarrow TMF, T \rightarrow TMF$ por $E \rightarrow TC_2, T \rightarrow TC_2, C_2 \rightarrow MF$
- $E \rightarrow LER, T \rightarrow LER, F \rightarrow LER$ por $E \rightarrow LC_3, T \rightarrow LC_3, F \rightarrow LC_3, C_3 \rightarrow ER$

Ejemplo (cont.)

La gramática CNF final es:

- $E \rightarrow EC_1|TC_2|LC_3|a|b|IA|IB|IZ|IO$
- $T \rightarrow TC_2|LC_3|a|b|IA|IB|IZ|IO$
- $F \rightarrow LC_3|a|b|IA|IB|IZ|IO$
- $I \rightarrow a|b|IA|IB|IZ|IO$
- $C_1 \rightarrow PT, C_2 \rightarrow MF, C_3 \rightarrow ER$
- $A \rightarrow a, B \rightarrow b, Z \rightarrow 0, O \rightarrow 1$
- $P \rightarrow +, M \rightarrow *, L \rightarrow (, R \rightarrow)$

Ejemplo

- Encuentre una gramática equivalente para:
  $S \rightarrow AB|CA$
  $A \rightarrow a$
  $B \rightarrow BC|AB$
  $C \rightarrow aB|b$
  sin símbolos inútiles
- Con la gramática:
  $S \rightarrow ASB|\epsilon$
  $A \rightarrow aAS|a$
  $B \rightarrow SbS|A|bb$
  Eliminar: (a) producciones ϵ, (b) producciones unitarias, (c) símbolos inútiles, y (d) ponerla en CNF

Solución al 1

- A y C son generadoras por que tienen producciones con terminales. Como S tienen una producción con puros generadores (CA), entonces también es generador. B no es generador, por lo que si lo eliminamos y todas las producciones donde aparece y nos quedamos solo con los alcanzables, la gramática queda:
  $S \rightarrow CA$
  $A \rightarrow a$
  $C \rightarrow b$

Solución al 2

- Solo S es nulificable, por lo que tenemos que ponerla y quitarla en cada lugar donde ocurre:
  $S \rightarrow ASB|AB$
  $A \rightarrow aAS|aA|a$
  $B \rightarrow SbS|Sb|bS|b|A|bb$
- La única producción unitaria es: $B \rightarrow A$ por lo que la reemplazamos directamente:
  $S \rightarrow ASB|AB$
  $A \rightarrow aAS|aA|a$
  $B \rightarrow SbS|Sb|bS|b|aAS|aA|a|bb$
- A y B generan símbolos terminales, y por lo tanto también S, por lo que no hay símbolos inútiles

Solución al 2 (cont.)

- Introducir variabes y producciones $C \rightarrow a$ y $D \rightarrow b$ y ponerla en todos los cuerpos que no tienen un solo símbolo terminal
  $S \rightarrow ASB|AB$
  $A \rightarrow CAS|CA|a$
  $B \rightarrow SDS|SD|DS|b|CAS|CA|a|DD$
  $C \rightarrow a$
  $D \rightarrow b$
- Para las producciones con más de 3 símbolos se introducen nuevas variables:
  $S \rightarrow AE|AB$
  $A \rightarrow CF|CA|a$
  $B \rightarrow SG|SD|DS|b|CF|CA|a|DD$
  $C \rightarrow a$, $D \rightarrow b$
  $E \rightarrow SB$, $F \rightarrow AS$
  $G \rightarrow DS$

Pumping Lemma para CFL

- Existe un equivalente del *Pumping Lemma* para CFL
- Sea L un CFL. Existe una constante n tal que si z es cualquier cadena de L tal que $|z|$ es al menos n, podemos escribir $z = uvwxy$ tal que:
  1. $|vwx| \leq n$
  2. $vx \neq \epsilon$
  3. Para toda $i \geq 0, uv^iwx^iy$ está en L

Pumping Lemma para CFL

- Es muy parecido al de CFL solo que ahora tenemos que encontrar a dos subcadenas que podamos repetir indefinidamente
- La idea viene de que toda CFL la podemos representar como CNF y generar árboles de parseo binarios
- Mientras que los CFL pueden servir para dos grupos de caracteres, no sirven para tres (e.g., $\{0^n 1^n 2^n | n \geq 1\}$)
- Los CFL tampoco sirven si tenemos pares de símbolos intercalados (e.g., $\{0^i 1^j 2^i 3^j | i \geq 1, i \geq 1\}$)
- Tampoco pueden verificar la igualdad de dos cadenas arbitrariamente largas (e.g., $ww | w \in \{0, 1\}^*\}$)

Ejemplos

- $\{0^n 1^n 2^2 | n \geq 1\}$ Idea: $z = uvwxy$, $|vwx| \leq n$ (lemma), *vwx* no puede tener al mismo tiempo 0's y 2's porque el último 0 y el primer 2 están separados por $n + 1$ pocisiones. Si *vwx* no tiene 2's (o 0's) con el *pumping* lo desbalanceamos.
- $\{0^i 1^j 2^i 3^j | i \geq 1, i \geq 1\}$. Con un argumento similar podemos tener que la cadena no tiene todos los símbolos y desbalancear la cadena.
- $ww | w \in \{0, 1\}^*\}$. En este caso, podemos poner como ejemplo: $0^n 1^n 0^n 1^n$ la cual es una cadena *ww* y mostrar tomando las ideas de los ejemplos anteriores que podemos generar una cadena que no está en *L*.

Ejemplo

- Mostrar que $\{a^i b^j C^k | i < j < k\}$ no es CFL
- Tomando a $z = a^n b^{n+1} C^{n+2}$ y $|vwx| \leq$. Si *vwx* no tiene *c*'s entonces podemos generar más *a*s y *b*s que *c*s. Si *vwx* tiene una *c*, entonces no podría tener una *a* porque la longitud está limitada a *n*. Esto quiere decir que *uwy* (sin las partes que se repiten) tiene *n a*'s, pero no más que $2n + 2$ *b*s y *c*s juntas con lo cual no es posible que se tengan al mismo tiempo más *b*s que *a*s y más *c*s que *b*s.
- Probar que: $\{a^n b^n C^i | i \leq n\}$ no es CFL.

Propiedades de Cerradura de los CFL's

- Existen varias propiedades de los CFL, una de las más importantes es la de substitución.
- **Substitución**: Sea Σ un alfabeto y supongamos que para cada símbolo $a \in \Sigma$ definimos un lenguaje arbitrario L_a.
- Estos lenguajes definen una función s o una substitución.
- Si $w = a_1 a_2 \ldots a_n$ es una cadena en Σ^*, $s(w)$ es la concatenación de los lenguajes $s(a_1)s(a_2)\ldots s(a_n)$.

Ejemplo

- $\Sigma = \{0,1\}$ y $s(0) = \{a^n b^n : n \geq 1\}, s(1) = \{aa, bb\}$
- Sea $w = 01$. Entonces $s(w) = s(0)s(1) = \{a^n b^n aa : n \geq 1\} \cup \{a^n b^{n+2} : n \geq 1\}$.
- Si $L = L(0^*)$, entonces $s(L) = (s(0))^* = \{a^{n_1} b^{n_1} a^{n_2} b^{n_2} \ldots a^{n_k} b^{n_k} : k \geq 0, n_i \geq 1\}$

Teoremas

- **Teoremas**: Sea L un CFL sobre Σ, y s una substitución, tal que $s(a)$ sea un CFL, $\forall a \in \Sigma$. Entonces $s(L)$ es un CFL.
- **Teoremas**: Si tenemos uno o más CFL's, también son CFL el resultado de hacer: (i) unión, (ii) concatenación, (iii) Cerradura de Kleene, (iv) cerradura positiva $+$, (v) inversión, (vi) homomorfismo, y (vii) homomorfismo inverso.
- **Teorema**: Si L, L_1, L_2 son CFL's, y R es lenguaje regular, entonces $L \cap R$, $L \setminus R$ son CFL pero $\overline{L}$, $L_1 \setminus L_2$, $L_1 \cap L_2$, $L_1 \setminus L_2$ no son necesariamente CFL's.

Probar membresía en un CFL

- Para probar si una cadena w es parte del lenguaje (L) de un CGF (i.e., $w \in L$), usar el algoritmo CYK
- El eje x corresponde a la cadena $w = a_1, a_2, a_3, \ldots, a_n$
- En el primer renglón pone producciones tipo $A \rightarrow a$
- En el resto (e.g., X_{ij}) se ponen las variables (e.g., A) tales que tengan una producción (e.g., $A \rightarrow BC$) donde B está en X_{ik} (parte del prefijo) y C está en X_{kj} (resto de la cadena), $i < k < j$.

Algoritmo CYK

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
a_1	a_2	a_3	a_4	a_5

Ejemplo

- Probar que la siguiente gramática genera: *baaba*
  $S \rightarrow AB|BC$
  $A \rightarrow BA|a$
  $B \rightarrow CC|b$
  $C \rightarrow AB|a$

$b : \{B\}$
$a : \{A, C\}$

$ba : \{B\}\{A, C\} = \{BA, BC\} : \{S, A\}$
$aa : \{A, C\}\{A, C\} = \{AA, AC, CA, CC\} : \{B\}$
$ab : \{A, C\}\{B\} = \{AB, CB\} : \{S, C\}$
$ba : \{B\}\{A, C\} = \{BA, BC\} : \{S, A\}$

Ejemplo

$bab : b/ab$ o $ba/b : \{B\}\{B\} \cup \{S, A\}\{A, C\} =$
$\{BB, SA, SC, AA, AC\} : \emptyset$
$aab : a/ab$ o $aa/b : \{A, C\}\{S, C\} \cup \{B\}\{B\} =$
$\{AS, AC, CS, CC, BB\} : \{B\}$
$aba : a/ba$ o $ab/a : \{A, C\}\{S, A\} \cup \{S, C\}\{A, C\} =$
$\{AS, AA, CS, CA, SA, SC, CA, CC\} : \{B\}$

$baab : b/aab$ o ba/ab o $baa/b : \{B\}\{B\} \cup$
$\{S, A\}\{S, C\} \cup \emptyset\{B\} = \{BB, SS, SC, AS, AC\} : \emptyset$
$aaba : a/aba$ o aa/ba o $aab/a : \{A, C\}\{B\} \cup$
$\{B\}\{S, A\} \cup \{B\}\{A, C\} = \{AB, AC, BS, BA, BA, BC\} : \{S, A, C\}$

Ejemplo

$baaba$: $b/aaba$ o ba/aba o baa/ba o $baab/a$:
$\{B\}\{S,A,C\} \cup \{S,A\}\{B\} \cup \emptyset \cup \emptyset =$
$\{BS, BA, BC, SB, AB\}$: $\{S, A, C\}$

{S,A,C}				
-	{S,A,C}			
-	{B}	{B}		
{S,A}	{B}	{S,C}	{S,A}	
{B}	{A,C}	{A,C}	{B}	{A,C}
b	*a*	*a*	*b*	*a*

Problemas no-decidibles

En el siguiente tema vamos a ver que hay problemas que no los podemos resolver con una computadora (no-decidibles). Por ejemplo:

1. Es una CFG G dada ambígua?
2. Es un CFL dado inherentemente ambíguo?
3. Es la intersección de dos CFLs vacía?
4. Son dos CFLs iguales?
5. Es un CFL dado igual a Σ^* donde Σ es el alfabeto de ese lenguaje?