Ciencias computacionales

Propedéutico: Teoría de Autómatas y Lenguajes Formales

Propiedades de los lenguajes libres de contexto

Contents

1 Formas normales de las gramáticas libres de contexto

El objetivo de esta sección es mostrar que cualquier CFL sin el símbolo ϵ es generado por una CFG en la que todas las producciones son de la forma $A \to BC$ o $A \to a$, donde A, B y C son variables y a es una terminal. Esta es llamada la forma normal de Chomsky. Para llegar a la forma es necesario realizar los siguientes pasos:

- Se deben eliminar los símbolos innecesarios. Dichos variables o terminales que no aparecen en alguna derivación de una cadena terminal empezando por el símbolo inicial.
- Se eliminan las producciones ϵ. Dichas producciones son de la forma $A \to \epsilon$ para alguna variable A.
- Las producciones unitarias deben ser eliminadas. Las producciones unitarias son de la forma $A \to B$ para alguna variable A y B.

1.1 Eliminación de símbolos innecesarios

Un símbolo X es *útil* para una gramática $G = (V, T, P, S)$, si hay una derivación: $S \overset{*}{\Rightarrow}_G \alpha X \beta \overset{*}{\Rightarrow}_G w$ para una cadena terminal w. A los símbolos que no son útiles se les denomina *inútiles*. Un símbolo X es *generador* si $X \overset{*}{\Rightarrow}_G w$, para alguna $w \in T^*$. Un símbolo X es *alcanzable* si $S \overset{*}{\Rightarrow}_G \alpha X \beta$, 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 1 *Sea G:* $S \to AB|a, A \to b$.

S y A son generadores, B no lo es. Si eliminamos B tenemos que eliminar $S \to AB$, dejando la gramática $S \to a, A \to b$. Ahora sólo S es alcanzable. Eliminando A y b nos deja con $S \to a$. 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: $S \to AB|a, A \to b$. Después eliminamos B como no-generador, y nos quedamos con $S \to a, A \to b$, que todavía contiene símbolos inútiles.

Teorema 1 *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* $L(G_1) = L(G)$.

Prueba: Primero probamos que G_1 no tiene símbolos inútiles:
Si $X \in (V_1 \cup T_1)$. Así $X \overset{*}{\Rightarrow} w$ en G_1, para algún $w \in T^*$. Además, cada símbolo utilizado en esta derivación también es generador. Así que $X \overset{*}{\Rightarrow} w$ en G_2 también.

Como X no se eliminó en el paso 2, hay una α y una β, tal que $S \overset{*}{\Rightarrow} \alpha X \beta$ en G_2. Aún más, cada símbolo utilizado en esta derivación también es alcanzable, así $S \overset{*}{\Rightarrow} \alpha X \beta$ en G_1.

Sabemos que cada símbolo en $\alpha X \beta$ es alcanzable y que están en $V_2 \cup T_2$, entonces cada uno de ellos es generador en G_2.

La derivación terminal $\alpha X \beta \overset{*}{\Rightarrow} xwy$ en G_2 solo involucra símbolos que son alcanzables desde S, porque son alcanzados por símbolos en $\alpha X \beta$. De este modo, la derivación terminal también es una derivación de G_1, i.e., $S \Rightarrow \alpha X \beta \overset{*}{\Rightarrow} xwy$ en G_1.
Ahora mostramos que $L(G_1) = L(G)$.

Como sólo eliminamos símbolos y producciones de G a G_1 ($P_1 \subseteq P$), entonces tenemos que $L(G_1) \subseteq L(G)$.

Entonces, sea $w \in L(G)$. Así $S \stackrel{*}{\Rightarrow}_G w$. Cada símbolo en esta derivación es evidentemente alcanzable y generador, entonces esta es también una derivación de G_1. Así $w \in L(G_1)$.

1.2 Cálculo de los 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 $A \to \alpha$, 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 2 *Sea G:* $S \to AB|a, A \to b$. *Entonces, primero* $g(G) = \{a, b\}$. *Como* $S \to a$ *ponemos a* S *en* $g(G)$, *y porque* $A \to b$ *añadimos también a* A, *y eso es todo, el conjunto de símbolos generadores es* $\{a, b, A, S\}$.

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 $A \to \alpha \in P$ entonces se añaden todos los símbolos de α a $r(G)$.

Prueba: Se mostrara con una inducción sobre la fase en la cual un símbolo X se añade a $g(G)$ y que X es en verdad generador. Entonces suponemos que X es generador. De este modo $X \stackrel{*}{\Rightarrow}_G w$, para alguna $w \in T^*$. Ahora probamos por inducción sobre esta derivación que $X \in g(G)$.

Base: Cero pasos. Entonces X es terminal y se añade en la base del algoritmo de cerradura.

Inducción: La derivación toma $n > 0$ pasos. Sea la primera producción utilizada $X \to \alpha$. Entonces: $X \Rightarrow \alpha \stackrel{*}{\Rightarrow} w$ y $\alpha \stackrel{*}{\Rightarrow} w$ en menos de n pasos y por la hipótesis de inducción $\alpha \in g(G)$. De la parte inductiva de algoritmo se concluye que $X \in g(G)$ porque $X \to \alpha$.

Ejemplo 3 *Sea* $G : S \to AB|a, A \to b$. *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.*

1.3 Eliminación de 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 $B \to CAD$, creamos dos versiones de la producción, una sin $A, B \to CD$ y otra con $A, B \to CAD$. Si utilizamos la producción con A no permitimos que A derive a ϵ.

El siguiente algoritmo calcula $n(G)$, el conjunto de símbolos nulificables de una gramática $G = (V, T, P, S)$ como sigue:

1. *Base*: $n(G) = \{A : A \to \epsilon \in P\}$

2. *Inducción*: Si $\{C_1, C_2, \ldots, C_k\} \subseteq n(G)$ y $A \to C_1, C_2, \ldots, C_k \in P$, 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.

En cualquier gramática G, los únicos símbolos nulificables son las variables encontradas por el algoritmo anterior.

Prueba: La inducción es sencilla en ambas direcciones.
Base: Un paso. $A \to \epsilon$ debe ser una producción y se encuentra en la parte base del algoritmo.

Inducción: Suponemos que $A \stackrel{*}{\Rightarrow} \epsilon$ en n pasos donde $n > 1$. El primer paso se ve como $A \Rightarrow C_1C_2 \ldots C_k \stackrel{*}{\Rightarrow} \epsilon$, donde cada C_i deriva ϵ en una secuencia de menos de n pasos.

Por la HI, cada C_i es descubierta por el algoritmo como nulificable. Entonces, por el paso de inducción, se descubre que A es nulificable, por la producción $A \to C_1C_2 \ldots C_k$.

Una vez que conocemos los símbolos nulificables, podemos transformar G en G_1 como sigue:

1. Para cada $A \to X_1X_2 \ldots X_k \in P$ 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 $A \to \epsilon$.

Ejemplo 4 *Considere la siguiente gramática:*
$S \to AB$
$A \to aAA|\epsilon$
$B \to bBB|\epsilon$

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

Para $A \to aAA$, hacemos algo parecido y nos queda: $A \to aAA|aA|aA|a$, que como hay dos iguales podemos eliminar una. Finalmente para B es parecido, por lo que la gramática final queda como:
$S \to AB|A|B$
$A \to aAA|aA|a$
$B \to bBB|bB|b$

La gramática anterior no cambia el lenguaje, excepto que ϵ ya no está presente.

Teorema 2 *Si la gramática G_1 se construye a partir de G con la construcción anterior para eliminar producciones ϵ, entonces $L(G_1) = L(G) - \{\epsilon\}$.*

Prueba: Se probará el enunciado más fuerte:
$A \stackrel{*}{\Rightarrow} w$ en G_1 sí y solo si $w \neq \epsilon$ y $A \stackrel{*}{\Rightarrow} w$ en G. *Dirección* (*sólo si*): Suponemos que $A \stackrel{*}{\Rightarrow} w$ en G_1. Entonces claramente $w \neq \epsilon$, porque G_1 no tiene producciones-ϵ.

Ahora mostraremos por una inducción sobre la longitud de la derivación que $A \stackrel{*}{\Rightarrow} w$ también en G.
Base: Un paso. Entonces existe $A \to w$ en G_1. A partir de la construcción de G_1 tenemos que existe $A \to \alpha$ en G, donde α es w más algunas variables nulificables esparcidas. Entonces: $A \Rightarrow \alpha \stackrel{*}{\Rightarrow} w$ en G.

Inducción: La derivación toma $n > 1$ pasos. Entonces $A \Rightarrow X_1X_2 \ldots X_k \stackrel{*}{\Rightarrow} w$ en G_1. y la primera derivación debió venir de una producción:
$A \to Y_1Y_2 \ldots Y_m$ donde $m \geq k$, algunas Y_i's son X_j's y los otros son símbolos nulificables de G.

Más aún, $w = w_1 w_2 \dots w_k$, y $X_i \stackrel{*}{\Rightarrow} w_i$ en G_1 en menos de n pasos.
Por la hipótesis de inducción tenemos que $X_i \stackrel{*}{\Rightarrow} w_i$ en G. Ahora tenemos:
$A \Rightarrow_G Y_1 Y_2 \dots Y_m \Rightarrow^*_G X_1 X_2 \dots X_k \Rightarrow^*_G w_1 w_2 \dots w_k = w$

Dirección (*si*): Sea $A \Rightarrow^*_G w$, y $w \neq \epsilon$. Se mostrará por inducción de la longitud de la derivación que $A \stackrel{*}{\Rightarrow} w$ en G_1. *Base*: La longitud es uno. Entonces $A \to w$ esta en G, y como $w \neq \epsilon$ la regla también esta en G_1.
Inducción: La derivación toma $n > 1$ pasos. Entonces esto se ve como: $A \Rightarrow_G Y_1 Y_2 \dots Y_m \Rightarrow^*_G w$. Ahora $w = w_1 w_2 \dots w_m$, y $Yi \Rightarrow^*_G w_i$ en menos de n pasos. Sean $X_1 X_2 \dots X_k$ aquellos Y_j's en orden, tal que $w_j \neq \epsilon$. Entonces $A \to X_1 X_2 \dots X_k$ es una regla de G_1.

Ahora $X_1 X_2 \dots X_k \Rightarrow^*_G w$, las únicas Y_j's no presentes entre las X's son las que derivan ϵ. Cada $X_j / Y_j \Rightarrow^*_G w_j$ en menos de n pasos, entonces, por la hipótesis de inducción tenemos que si $w \neq \epsilon$ entonces $Y_j \stackrel{*}{\Rightarrow} w_j$ en G_1. Así $A \Rightarrow X_1 X_2 \dots X_k \stackrel{*}{\Rightarrow} w$ en G_1. la demanda del teorema ahora sigue del enunciado: $A \stackrel{*}{\Rightarrow} w$ en G_1 sí y solo si $w \neq \epsilon$ y $A \stackrel{*}{\Rightarrow} w$ en G (presentado en la prueba del teorema 7.9) al elegir $A = S$.

1.4 Eliminación de producciones unitarias

$A \to B$ es una producción unitaria, cuando A y B son variables. Las producciones unitarias se pueden eliminar.

Veamos la gramática:
$I \to a|b|Ia|Ib|I0|I1$
$F \to I|(E)$
$T \to F|T * F$
$E \to T|E + T$
tiene las producciones unitarias $E \to T, T \to F$ y $F \to I$.

Podemos expandir T en la producción $E \to T$ y obtener: $E \to F|T * F$. Expandiendo $E \to F$ nos da: $E \to I|(E)$. Finalmente expandemos $E \to I$ y obtenemos: $E \to a|b|Ia|Ib|I0|I1|(E)|T * F|E + T$.

El método de expansión trabaja siempre y cuando no haya ciclos en las reglas, por ejemplo en: $A \to B, B \to C, C \to A$.

Para calcular $u(G)$, el conjunto de todos los pares unitarios de $G = (V, T, P, S)$ utilizamos el siguiente algoritmo de cerradura.

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 $B \to C \in P$ donde C es una variable. Entonces añadimos (A, C) a $u(G)$.

Teorema 3 *El algoritmo anterior encuentra todos y solo los pares unitarios de una CFG G.*

Prueba: En una dirección, hacemos una inducción sobre el orden en que se descubren los pares, si encontramos que (A, B) es un par uitario, entonces $A \Rightarrow^*_G B$ utilizando únicamente producciones unitarias (prueba omitida).

En la otra dirección, suponemos que $A \Rightarrow^*_G B$ usando únicamente producciones unitarias. Podemos mostrar por inducción de la longitud de la derivación que encontraremos el par (A, B).

Base: Cero pasos. Entonces $A = B$, y añadimos el par (A, B) en la base.

Inducción: Suponemos que $A \stackrel{*}{\Rightarrow} B$ en n pasos, para alguna $n > 0$, cada paso consiste en aplicar una producción unitaria. La derivación luce como: $A \stackrel{*}{\Rightarrow} C \Rightarrow B$. $A \stackrel{*}{\Rightarrow} C$ toma $n - 1$ pasos, y por la hipótesis

inductiva, descubrimos el par (A, C).

La parte inductiva del algoritmo combina el par (A, C) con la producción $C \to B$ para inferir el par (A, B).

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 $A \to \alpha$, donde $B \to \alpha$ 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. $P_1 = \{A \to \alpha : \alpha \notin V, B \to \alpha \in P, (A, B) \in u(G)\}$

Ejemplo 5 *A partir de la gramática:*

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

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:

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

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

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

En resumen para "limpiar" una gramática podemos:

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

en este orden.

1.5 Forma normal de Chomsky

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: $A \to BC$, donde $\{A, B, C\} \subseteq T$, o $A \to \alpha$, 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.

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 $A \to a$.

Para el paso 3, por cada regla de la forma $A \to B_1 B_2 \dots B_k$, $k \geq 3$, introducimos variables nuevas $C_1, C_2, \dots C_{k-2}$, y reemplazamos la regla con:

$A \to B_1 C_1$
$C_1 \to B_2 C_2$
$\dots$
$C_{k-3} \to B_{k-2} C_{k-2}$
$C_{k-2} \to B_{k-1} B_k$

Ejemplo 6 *Iniciamos con la gramática (el paso 1 ya está hecho):*

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

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

Al reemplazar obtenemos la gramática:
$E \to EPT|TMF|LER|a|b|IA|IB|IZ|IO$
$T \to TPE|LEL|a|b|IA|IB|IZ|IO$
$F \to LER|a|b|IA|IB|IZ|IO$
$I \to a|b|IA|IB|IZ|IO$
$A \to a, B \to b, Z \to 0, O \to 1$
$P \to +, M \to *, L \to (, R \to)$

Para el paso 3, reemplazamos:

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

La gramática CNF final es:

- $E \to EC_1|TC_2|LC_3|a|b|IA|IB|IZ|IO$

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

1.6 Ejercicios

Ejercicio 1 *Encuentre una gramática equivalente a la siguiente sin símbolos inútiles:*
$S \to AB|CA$
$A \to a$
$B \to BC|AB$
$C \to aB|b$

Ejercicio 2 *Con la siguiente gramática:*
$S \to ASB|\epsilon$
$A \to aAS|a$
$B \to SbS|A|bb$

- *Elimine producciones* ϵ.
- *Elimine cualquier producción unitaria en la gramática resultante.*
- *Elimine cualquier símbolo inútil en la gramática resultante.*
- *Convierta a la forma normal de Chomsky.*

Ejercicio 3 *Repita el Ejercicio 2 con la siguiente gramática:*
$S \to 0A0|1B1|BB$
$A \to C$
$B \to S|A$
$C \to S|\epsilon$

Ejercicio 4 *Repita el Ejercicio 2 con la siguiente gramática:*
$S \to AAA|B$
$A \to aA|B$
$B \to \epsilon$

2 Lema de bombeo para los lenguajes libres de contexto

El lema de bombeo dice que en una cadena lo suficientemente larga de un CFL, es posible encontrar a lo mas dos subcadenas que pueden ser *bombeadas* de forma conjunta. Esto es, podemos repetir ambas cadenas i veces, para cualquier entero i y la cadena resultante se encontrará en el lenguaje.

Se puede contrastar con el lema de bombeo para lenguajes regulares que dice que solo podemos encontrar una pequeña cadena para *bombear*.

2.1 El tamaño de los analizadores sintácticos de árboles

Teorema 4 *Suponga que tiene un árbol de derivación de acuerdo a la forma normal de Chomsky de la gramática $G = (V, T, P, S)$ y suponga que la franja del árbol es un terminal w. Si la longitud del camino más largo es n, entonces $|w| \leq 2^{n-1}$.*

Prueba Se deja al lector hacer la prueba en inducción sobre n.

2.2 Declaración del lema de bombeo

El lema de bombeo para lenguajes libres de contexto es muy parecido al lema para lenguajes regulares, sin embargo cada cadena z en el CFL L en cinco partes y se *bombean* la segunda y cuarta parte en conjunto.

Sea L un CFL. Existe una constante n tal que si z es cualquier cadena de L de forma que $|z|$ es al menos n, entonces podemos escribir $z = uvwxy$ de acuerdo a las siguientes condiciones:

1. $|vwx| \leq n$. Es decir, la parte media no es tan larga.
2. $vx \neq \epsilon$. Dado que v y x son las piezas a ser *bombeadas*, la condición dice que al menos una de las cadenas a ser *bombeadas* no es vacía.
3. Para todo $i \geq 0$, uv^iwx^iy se encuentra en L. Es decir, las dos cadenas v y x pueden ser *bombeadas* un numero indefinido de veces, incluido 0 veces, y la cadena resultante seguirá siendo un miembro de L.

2.3 Ejercicios

Ejercicio 5 *Use el lema de bombeo para mostrar que los siguientes lenguajes no son libres de contexto.*

- $\{a^ib^jc^k | i < j < k\}$
- $\{a^nb^nc^i | i \leq n\}$
- $\{0^p | p \text{ es un primo }\}$
- $\{a^nb^nc^i | n \leq i \leq 2n\}$

3 Propiedades de cerradura para los lenguajes libres de contexto

En esta sección se consideran algunas de las operaciones de los lenguajes libres de contexto que garantizan la producción de una CFL. Primero introducimos una operación llamada sustitución, en la cual se reemplaza cada símbolo en las cadenas de un lenguaje por otro lenguaje.

3.1 Sustituciones

Sea Σ un alfabeto, suponga que para cada símbolo a en Σ, seleccionamos un lenguaje L_a. Estos lenguajes pueden ser de cualquier alfabeto, no necesariamente de Σ ni tienen que ser los mismos. Esta selección de lenguajes define una función s (una sustitución) en Σ, y nos vamos a referir a L_a como $s(a)$ para cada símbolo de a.

Si $w = a_1a_2 \cdots a_n$ es una cadena en Σ^*, entonces $s(w)$ es un lenguaje de todas las cadenas $x_1x_2 \cdots x_n$ tal que la cadena x_i se encuentra en el lenguaje $s(a_i)$, para $i = 1, 2, \cdots, n$. De tal forma que $s(w)$ es la concatenación de los lenguajes $s(a_1)s(a_2) \cdots s(a_n)$.

Ejemplo 7 *Sea $\Sigma = \{0, 1\}, s(0) = \{a^nb^n : n \geq 1\}, s(1) = \{aa, bb\}$. Sea $w = 01$. Entonces $s(w) = s(0)s(1) = \{a^nb^naa : n \geq 1\} \cup \{a^nb^{n+2} : n \geq 1\}$. Si $L = \{0\}^*$, entonces $s(L) = (s(0))^* = \{a^{n_1}b^{n_1}a^{n_2}b^{n^2} \ldots a^{n_k}b^{n_k} : k \geq 0, n_i \geq 1\}$*

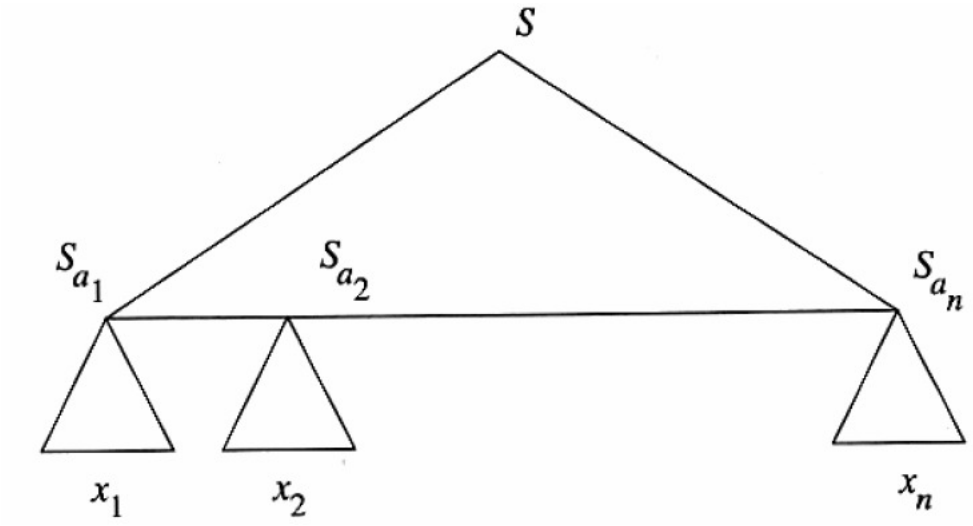

Figure 1: Árbol de derivación.

Teorema 5 *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.*

Para demostrar el Teorema 5, iniciamos con las gramáticas: $G = (V, \Sigma, P, S)$ para L, y $G = (V_a, T_a, P_a, S_a)$ para cada $s(a)$. Construimos $G = (V', T', P', S')$, donde: $V' = (U_{a\in\Sigma} T_a) \cup V, T' = U_{a\in\Sigma} T_a, P' = U_{a\in\Sigma} P_a$ más las producciones de P con cada a en un cuerpo reemplazado con el símbolo S_a.

Ahora debemos mostrar que $L(G') = s(L)$. Sea $w \in s(L)$. Entonces $\exists x = a_1 a_2 \dots a_n$ en L, y $\exists x_i \in s(a_i)$, tal que $w = x_1 x_2 \dots x_n$. Un árbol de derivación en G' se muestra en la Figura 1.

Así podemos generar $S_{a_1} S_{a_2} \dots S_{a_n}$ en G' y de ahí generamos $x_1 x_2 \dots x_n = w$. De este modo $w \in L(G')$. Después, sea $w \in L(G')$. Entonces el árbol de parseo para w debe verse como el de arriba.

Ahora borramos los sub-árboles que cuelgan. Ahora se tiene la producción: $S_{a_1} S_{a_2} \dots S_{a_n}$, donde $a_1 a_2 \dots a_n \in L(G)$. Ahora w es también igual a $s(a_1 a_2 \dots a_n)$, lo cual esta en $S(L)$.

3.2 Aplicaciones del teorema de sustitución

Teorema 6 *Si tenemos uno o más CFL's, también son CFL el resultados de hacer: (i) unión, (ii) concatenación, (iii) Cerradura de Kleene, (iv) cerradura positiva +, (v) inversión, (vi) homomorfismo, (vii) homomorfismo inverso.*

Prueba:

(i) : Sean L_1 y L_2 CFL's, sea $L = \{1, 2\}$, y $s(1) = L_1, s(2) = L_2$. Entonces $L_1 \cup L_2 = s(L)$.

(ii) : Aquí elegimos $L = \{1, 2\}$ y s como antes. Entonces $L_1 \cdot L_2 = s(L)$.

(iii) : Suponemos que L_1 es CF. Sea $L = 1*, s(1) = L_1$. Ahora $L_1^* = s(L)$. Prueba similar para $+$.

(iv) : Sea L_1 un CFL sobre Σ, y h un homomorfismo sobre Σ. Entonces definimos s por $a \mapsto \{h(a)\}$

Entonces $h(L) = s(L)$ ■.

Teorema 7 *Si L es CF, entonces también en L^R.*

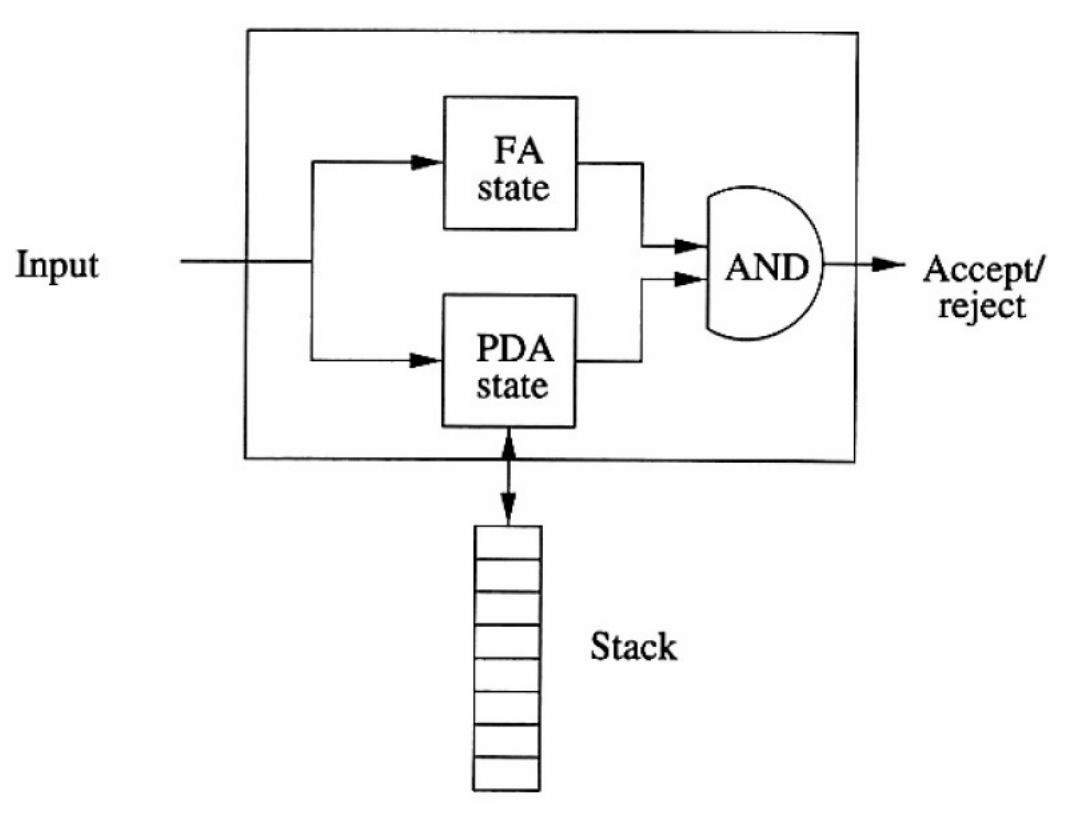

Figure 2: PDA.

Prueba: Suponemos que L es generado por $G = (V, T, P, S)$. Construimos $G^R = (V, T, P^R, S)$, donde: $P^R = \{A \to \alpha^R : A \to \alpha \in P\}$. Mostrar (en casa) por inducción sobre las longitudes de las derivaciones en G (para una dirección) y en G^R (para la otra dirección) que $(L(G))^R = L(G^R)$.

Sea $L_1 = \{0^n 1^n 2^i : n \geq 1, i \geq 1\}$. L_1 es CF con la gramática:
$S \to AB$
$A \to 0A1|01$
$B \to 2B|2$

Además, $L_2 = \{0^i 1^n 2^n : n \geq 1, i \geq 1\}$ es CF con la gramática
$S \to AB$
$A \to 0A|0$
$B \to 1B2|12$

Sin embargo, $L_1 \cap L_2 = \{0^n 1^n 2^n : n \geq 1\}$ lo cual no es CF ■.

Teorema 8 *Si L, L_1, L_2 son CFL's, y R es lenguaje regular, entonces $L \cap R$, $L \setminus R$ son CFL y $\overline{L}$, $L_1 \setminus L_2$ no son necesariamente CFL's.*

Prueba: Sea L aceptado por el PDA: $P = (Q_P, \Sigma, \Gamma, \delta_P, q_P, Z_0, F_P)$, por el estado final, y R aceptado por el DFA: $A = (Q_A, \Sigma, \Gamma, \delta_A, q_A, Z_0, F_A)$

Construiremos un PDA para $L \cap R$ de acuerdo a la Figura 2.

Formalmente, definimos: $P = (Q_P \times Q_A, \Sigma, \Gamma, \delta, (q_P, q_A), Z_0, F_P \times F_A)$ donde: $\delta((q, p), a, X) = \{((r, \hat{\delta}_A(p, a)), \gamma) : (r, \gamma) \in \delta_P(q, a, X)\}$.

Se deja al lector la demostración por por inducción de $\overset{*}{\vdash}$, para P y para P' que: $(q_P, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \gamma)$ en P sí y solo si $((q_P, q_A), w, Z_0) \overset{*}{\vdash} ((q, \hat{\delta}(p_A, w)), \epsilon, \gamma)$ en P'.

Teorema 9 *Sean L, L_1, L_2 CFL's y R regular. Entonces:*

1. *L R es CFL*
2. *$\overline{L}$ no es necesariamente CFL*
3. *L_1 L_2 no es necesariamente CFL*

Prueba:

1. $\overline{R}$ es regular, $L \cap \overline{R}$ es regular, y $L \cap \overline{R} = L$ R.
2. Si $\overline{L}$ siempre fue CF, se tiene que dar que: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ siempre sea CF.
3. Note que Σ^* es CF, de este modo, si L_1 L_2 siempre fue CF, entonces también lo sería $\Sigma *$ $L = \overline{L}$.

3.3 Homomorfismo inverso

Sea $h : \Sigma \to \Theta^*$ un homomorfismo. Sea $L \subseteq \Theta^*$, y definimos: $h^{-1}(L) = \{w \in \Sigma^* : h(w) \in L\}$ ahora tenemos:

Teorema 10 *Sea L un CFL, y h un homomorfismo. Entonces $h^{-1}(L)$ es un CFL.*

Prueba: Sea L aceptado por el PDA: $P = (Q, \Theta, \Gamma, \delta, q_0, Z_0, F)$, construimos un nuevo PDA $P' = (Q', \Theta, \Gamma, \delta', (q_0, \epsilon), Z_0, F \times \{\epsilon\})$ donde: $Q' = \{(q, x) : q \in Q, x \in suffix(h(a)), a \in \Sigma\}$
$\delta'((q, \epsilon), a, X) = \{((q, h(a)), X) : \epsilon \neq a \in T \cup \{\epsilon\}, q \in Q, X \in \Gamma\}$

Se deja al lector mostrar mediante inducción que: $(q_0, h(w), Z_0) \overset{*}{\vdash} (p, \epsilon, \gamma)$ in P si y solo si $((q_0, \epsilon), w, Z_0) \overset{*}{\vdash} ((p, \epsilon), \epsilon, \gamma)$ en P'.

3.4 Ejercicios

Ejercicio 6 *Considere los siguiente dos lenguajes:*
$L_1 = \{a^n b^{2n} c^m | n, m \geq 0\}$
$L_2 = \{a^n b^m c^{2m} | n, m \geq 0\}$

- *Muestre que cada uno de estos lenguajes es libre de contexto dando gramáticas para cada caso.*
- *Es ¿$L_1 \cap L_2$ un CFL? Justifique su respuesta.*

4 Propiedades de decisión de los lenguajes libres de contexto

4.1 Complejidad de conversión entre gramáticas libres de contexto y autómatas de pila deterministas

El tamaño de la entrada es n. Donde n es el tamaño total de la entrada CFG o PDA. Lo siguiente trabaja en tiempo $O(n)$:

1. Convertir una CFG a un PDA
2. Convertir un PDA de "estado final" a un PDA de "pila vacía"
3. Convertir un PDA de "pila vacía" a un PDA de "estado final"

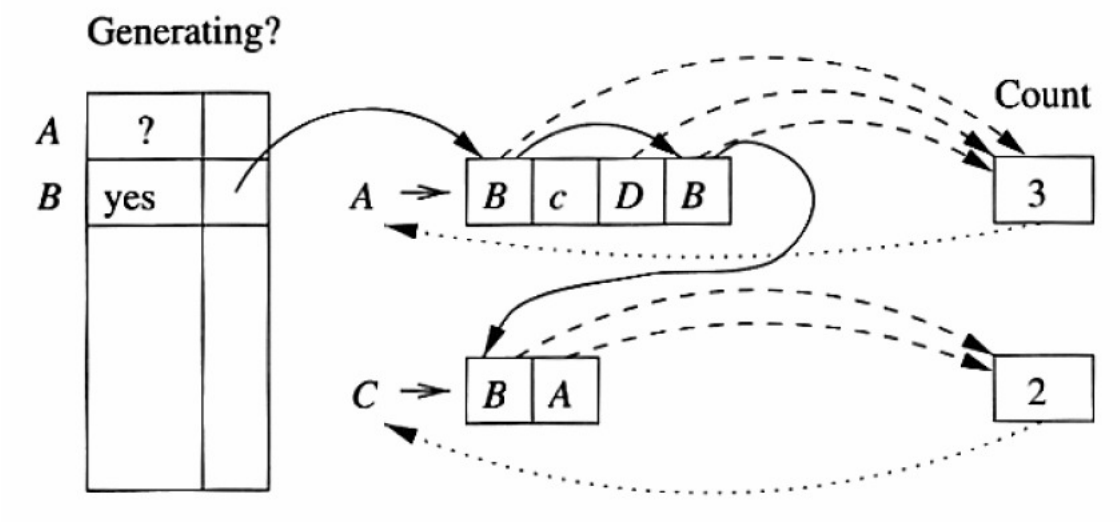

Figure 3: Prueba de vacío.

Para convertir un PDA a una CFG se tiene formalmente, sea $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ un PDA. Definimos $G = (V, \Sigma, R, S)$, donde:

$$\begin{aligned} V = & \quad \{[pXq] : \{p, q\} \subseteq Q, X \in \Gamma\} \cup \{S\} \\ R = & \quad \{S \to [q_0 Z_0 p] : p \in Q\} \cup \{[qXr_k] \to a[rY_1r_1] \dots [r_{k-1}Y_k r_k] : \\ & \quad a \in \Sigma \cup \{\epsilon\}, \{r_1, \dots, r_k\} \subseteq Q, (r, Y_1 Y_2 \dots Y_k) \in \delta(q, a, X)\} \end{aligned}$$

cuando mucho n^3 variables de la forma $[pXq]$.

Si $(r, Y1Y2 \dots Yk) \in \delta(q, a, X)$, tendremos $O(n^n)$ reglas de la forma: $[qXr_k] \to a[rY_1r_1] \dots [r_{k-1}Y_k r_k]$. Introduciendo $k-2$ estados nuevos podemos modificar el PDA para hacer un *push* de a lo más un símbolo por transición. Ahora, k será ≤ 2 para todas las reglas. La longitud total de todas las transiciones es todavía $O(n)$.

Ahora, cada transición genera a lo más n^2 producciones. El tamaño total (y tiempo para calcular) la gramática es entonces $O(n^3)$.

4.2 Tiempo de ejecución para conversión a la forma normal de Chomsky

1. Calcular $r(G)$ y $g(G)$ y eliminar símbolos no-útiles toma tiempo $O(n)$. Esto se mostrará pronto.
2. El tamaño de $u(G)$ y la gramática resultante con producciones P_1 es $O(n^2)$
3. Arreglando que los cuerpos consistan solo de variables lleva $O(n)$
4. Dividir los cuerpos lleva $O(n)$

Por otra parte, la eliminación de los símbolos nulificables puede hacer que la nueva gramática tenga un tamaño de $O(2^n)$. La mala noticia es evitable: Dividir los cuerpos primero antes de la eliminación de símbolos nulificables. La conversión a CNF lleva $O(n^2)$.

4.3 Prueba de vacío de los lenguajes libres de contexto

$L(G)$ es no-vacío si el símbolo de inicio S es generador. Una implementación ingenua sobre $g(G)$ tome tiempo $O(n^2)$. $g(G)$ se puede calcular en tiempo $O(n)$ como sigue (ver Figura 3).

La creación e inicialización del arreglo lleva $O(n)$. La creación e inicialización de las ligas y contadores lleva $O(n)$. Cuando un contador va a cero, tenemos que:

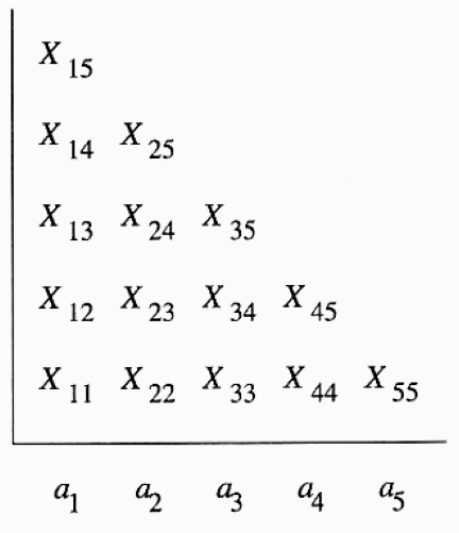

Figure 4: Algoritmo para pertenencia.

1. Encontrar la variable cabeza A, verificar si ya existe "yes" en el arreglo, y si no, ponerlo en la cola lleva $O(1)$ por producción. Total $O(n)$
2. Seguir las ligas para A, y decrementar los contadores. Toma un tiempo de $O(n)$.

El tiempo total es $O(n)$.

4.4 Prueba de pertenencia a un lenguaje libre de contexto

- Usar el algoritmo CYK (construye tabla triangular)
- En el primer renglón pone producciones tipo $A \to a$
- En el resto pone producciones que tengan una parte del prefijo de la cadena seguida de el resto de la cadena. Ver Figura 4

Ejercicio 7 *Probar que la siguiente gramática genera: baaba. Ver Figura 5*
$S \to AB|BC$
$A \to BA|a$
$B \to CC|b$
$C \to AB|a$

4.5 Vista previa a problemas de los lenguajes libres de contexto indecidibles

Los siguientes son problemas no-decidibles:

1. Es una CFG G dada ambigua?
2. Es un CFL dado inherentemente ambiguo?
3. Es la intersección de dos CFL's vacía?
4. Son dos CFL's el mismo?
5. Es un CFL dado universal (igual a Σ^*)?

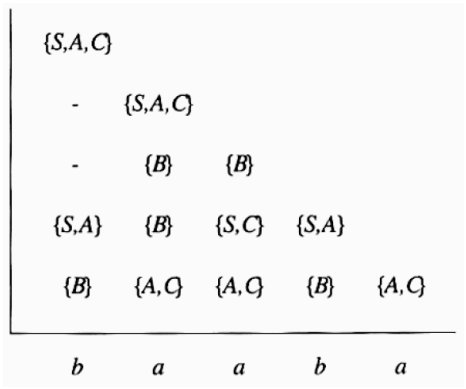

Figure 5: Gramática *baaba*.

4.6 Ejercicios

Ejercicio 8 *Escriba algoritmos para decidir lo siguiente:*

- *¿Es $L(G)$ finito, para un CFG G dado?*
- *¿Contine $L(G)$ al menos 100 cadenas, para un CFG G dado?*
- *Dado un CFG G y una de sus variables A, existe una forma sentencial en la cual A es el primer símbolo. Nota: Recuerde que es posible para A a aparecer primero en medio de una forma sentencial pero despues para todos los símbolos a la izquierda se deriva a ϵ*