Formas normales

Horst H. von Brand
vonbrand@inf.utfsm.cl

Departamento de Informática
Universidad Técnica Federico Santa María

Contenido

Formas normales

Para demostraciones y algoritmos que trabajan con gramáticas de contexto libre, la libertad de los lados derechos de producciones nos complica la existencia. Para simplificar el trabajo, se han definido *formas normales* de gramáticas de contexto libre, restricciones a las formas de las producciones. Su utilidad radica en que tenemos algoritmos que transforman una gramática de contexto libre cualquiera en la forma normal del caso. Las transformaciones mantienen la no ambigüedad y otras propiedades de interés.

La forma normal más usada es la de Chomsky, abreviada CNF por *Chomsky normal form*.

Forma normal de Chomsky

Definición

La gramática $G = (N, \Sigma, P, S)$ está en la *forma normal de Chomsky* si todas sus producciones son de una de las formas:

$$
\begin{array}{ll}
A \rightarrow BC & A, B, C \in N \\
A \rightarrow a & A \in N, a \in \Sigma
\end{array}
$$

Forma normal de Chomsky

Estrategia general

Podemos suponer que la gramática que nos dan no tiene símbolos ni producciones inútiles.
Eliminar terminales en lados derechos de más de un símbolo es fácil: basta definir no-terminales y producciones $X_a \to a$ para cada $a \in \Sigma$ y substituir X_a por a en todas las producciones con lado derecho de más de un símbolo.

Problemas producen producciones resultantes de lado derecho un único no-terminal (*producciones unitarias*) y de lado derecho de largo 3 o mayor.

Forma normal de Chomsky

Producciones unitarias

Debemos primero identificar los no-terminales que participan en producciones unitarias, formando cadenas $A_1 \Rightarrow A_2 \Rightarrow \cdots \Rightarrow A_n$. Identificar las cadenas de producciones unitarias es un algoritmo obvio.

Para cada cadena, si $A_i \Rightarrow^* A_j$, y las producciones no unitarias de A_j son $A_j \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$, reemplazar A_i por los distintos α_k. Finalmente, eliminar las producciones unitarias.

Forma normal de Chomsky

Lados derechos largos

Para eliminar lados derechos demasiado largos, creamos nuevos no-terminales $X_1, X_2, \ldots, X_{m-2}$ para la producción del caso, luego reemplazamos:

$$A \rightarrow B_1 B_2 \cdots B_m$$

por:

$$\begin{aligned} A &\rightarrow B_1 X_1 \\ X_1 &\rightarrow B_2 X_2 \\ &\vdots \\ X_{m-2} &\rightarrow B_{m-1} B_m \end{aligned}$$

Forma normal de Chomsky

Comentarios

Es claro que las operaciones anteriores no introducen ambigüedades.
La gramática resultante suele ser gigante.

Registrando el origen de las producciones es posible reconstruir la derivación en la gramática original de la en la gramática en la forma normal de Chomsky. Esto es relevante en algoritmos que la usan como paso intermedio.

El lema de bombeo

Teorema

Sea L un lenguaje de contexto libre. Entonces hay una constante K tal que toda palabra $\sigma \in L$ con $|\sigma| \geq K$ puede escribirse:

$$\sigma = uvwxy$$

donde $|vwx| \leq K$ y $vx \neq \varepsilon$ tal que para todo $k \geq 0$ es $uv^k wx^k y \in L$.

El lema de bombeo

Demostración

Podemos suponer que nos dan una gramática $G = (N, \Sigma, P, S)$ para L en forma normal de Chomsky.
El árbol de derivación en G de una palabra tiene forma de árbol binario (salvo las producciones $A \to a$ que finalmente dan terminales).

El lema de bombeo

Demostración

Un árbol binario de altura h tiene a lo más 2^h hojas, por lo que en la derivación de toda palabra σ de largo mayor a $2^{|N|+1}$ se repite algún no-terminal:

$$S \Rightarrow^* uAy \Rightarrow^+ uvAxy \Rightarrow^* uvwxy = \sigma$$

con $u, v, w, x, y \in \Sigma^*$. En el segundo tramo $A \Rightarrow^+ vAx$ al no haber producciones unitarias en forma normal de Chomsky garantiza que $vx \neq \varepsilon$. La observación sobre árboles binarios nos dice que lo que deriva del último no-terminal A que se repite (el vwx final) tiene largo a lo más $K = 2^{|N|+1} - 1$.

El lema de bombeo

Demostración

Si la derivación anterior es válida en G, también lo son para todo $k \geq 0$:

$$S \Rightarrow^* uAy \Rightarrow^* uv^k A x^k y \Rightarrow^* uv^k w x^k y$$

(Podemos simplemente saltarnos el paso $A \Rightarrow^* vAx$, dando el caso $k = 0$; o repetirlo las veces que deseemos, para $k \geq 1$). □

Un lenguaje no de contexto libre

Al igual que el lema de bombeo para lenguajes regulares, el principal uso es demostrar que ciertos lenguajes no son de contexto libre, usando esencialmente la misma estrategia.
(Tomar la salida fácil de decir que $\{\varepsilon\}$ no es de contexto libre se considera trampa.)

Un lenguaje no de contexto libre

Teorema

El lenguaje $L = \{a^n b^n c^n : n \geq 1\}$ *no es de contexto libre.*

Un lenguaje no de contexto libre

Demostración

Por contradicción.
Supongamos que L es de contexto libre. Sea N la constante del lema de bombeo, consideremos $\sigma = a^N b^N c^N$, con lo que $|\sigma| = 3N \geq N$. Por el lema, podemos escribir $\sigma = uvwxy$, con $|vwx| \leq N$ y $vx \neq \varepsilon$ tal que $uv^k wx^k y \in L$ para todo $k \geq 0$. Pero como vwx es corto, puede contener a lo más dos tipos de símbolo. Al bombear, aumenta a lo más el número de dos tipos de símbolos, el resultado no puede estar en L, contradicción. $\square$

Un lenguaje sensible al contexto

Teorema

El lenguaje $L = \{a^n b^n c^n : n \geq 1\}$ *es sensible al contexto.*

Un lenguaje sensible al contexto

Diseño de la gramática

Para demostrar esto, exhibimos una gramática sensible al contexto que genera este lenguaje.

La idea es que hay dos puntos activos (no-terminales A y B), donde A genera una nueva a y un mensaje M que viaja hacia B. Cuando B recibe el mensaje, agrega b y c. Para terminar, A se transforma en un mensaje de fin F, que hace que B dé una última ronda de bc.

Un lenguaje sensible al contexto

Gramática

$S \rightarrow AB$	Partida
$A \rightarrow aAM$	Mensaje de nuevo a
$Mb \rightarrow bM$	Mensaje viaja
$MB \rightarrow bBc$	Mensaje recibido
$A \rightarrow aF$	Mensaje de último a
$Fb \rightarrow bF$	Mensaje viaja
$FB \rightarrow bc$	Mensaje recibido

Un lenguaje sensible al contexto

Derivación ejemplo

$$
\begin{array}{ll}
\underline{S} & S \rightarrow AB \\
\quad \Rightarrow \underline{A}B & A \rightarrow aAM \\
\quad \Rightarrow aA\underline{MB} & MB \rightarrow bBc \\
\quad \Rightarrow a\underline{A}bBc & A \rightarrow aAM \\
\quad \Rightarrow aaA\underline{Mb}Bc & Mb \rightarrow bM \\
\quad \Rightarrow aaAb\underline{MB}c & MB \rightarrow bBc \\
\quad \Rightarrow aa\underline{A}bbBcc & A \rightarrow aF \\
\quad \Rightarrow aaa\underline{Fb}bBcc & Fb \rightarrow bF \\
\quad \Rightarrow aaab\underline{Fb}Bcc & Fb \rightarrow bF \\
\quad \Rightarrow aaabb\underline{FB}cc & FB \rightarrow bc
\end{array}
$$

Un lenguaje sensible al contexto

Gramática

Es claro que la gramática es sensible al contexto y hace lo prometido: se crea un nuevo M por cada a que se agrega, solo la producción $MB \to bBc$ permite eliminar un M (agregando un b y un c); solo usando $A \to aF$ podemos eliminar A, solo $FB \to bc$ permite eliminar B y F.

Revisando lo anterior, cada a da lugar a una b y una c. El lenguaje generado realmente es el indicado.

Jerarquía de Chomsky

Esto justifica otro peldaño de la jerarquía de Chomsky.

Antes que se entusiasmen, queda de ejercicio diseñar una gramática similar que genera $\{a^n b^n c^n d^n \colon n \geq 1\}$. El paso siguiente es bastante más pesado. Pero ese es un tema futuro.

Resumen

- Justificamos las formas normales de gramáticas de contexto libre.
- Definimos la forma normal de Chomsky, y esbozamos un algoritmo para transformar una gramática de contexto libre cualquiera a la forma normal de Chomsky.
- La forma normal de Chomsky permite demostrar en forma simple el lema de bombeo de lenguajes de contexto libre.
- Exhibimos un lenguaje que es sensible al contexto y no de contexto libre, justificando otro peldaño de la jerarquía de Chomsky.