4.14. Algoritmos de decisión para GIC

En esta sección consideraremos problemas de decisión para GIC, similares a los problemas para autómatas presentados en la sección 3.6. Dada una propiedad $\mathcal{P}$, referente a gramáticas independientes del contexto, un problema de decisión para $\mathcal{P}$ consiste en buscar un algoritmo, aplicable a una GIC arbitraria G, que responda SI o NO a la pregunta: ¿satisface G la propiedad $\mathcal{P}$? Los algoritmos vistos en el presente capítulo (para encontrar las variables terminables, las alcanzables, las anulables, etc) son frecuentemente útiles en el diseño de algoritmos de decisión más complejos.

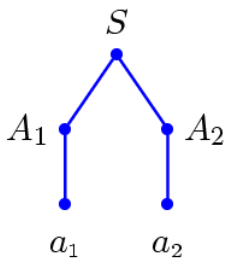
Problema 1 (Problema de la vacuidad). *Dada una gramática $G = (V, \Sigma, S, P)$, ¿es $L(G) \neq \varnothing$?*

Algoritmo de decisión: ejecutar el algoritmo para determinar el conjunto **TERM** de variables terminables. $L(G) \neq \varnothing$ si y sólo si $S \in$ **TERM**.

Problema 2 (Problema de la pertenencia). *Dada una gramática $G = (V, \Sigma, S, P)$ y una cadena $w \in \Sigma^*$, ¿se tiene $w \in L(G)$?*

Para resolver este problema primero convertimos G a la forma FNC, con variable inicial no recursiva, siguiendo el procedimiento de la sección 4.10.

A partir de una GIC G en FNC podemos diseñar un algoritmo bastante ineficiente para decidir si $w \in L(G)$: se encuentran *todas* las posibles derivaciones a izquierda (o los árboles de derivación) que generen cadenas de longitud $n = |w|$. Más específicamente, las cadenas de longitud 1 se pueden derivar únicamente con producciones de la forma $S \to a$. Las cadenas de longitud 2 sólo tienen árboles de derivación de la forma:



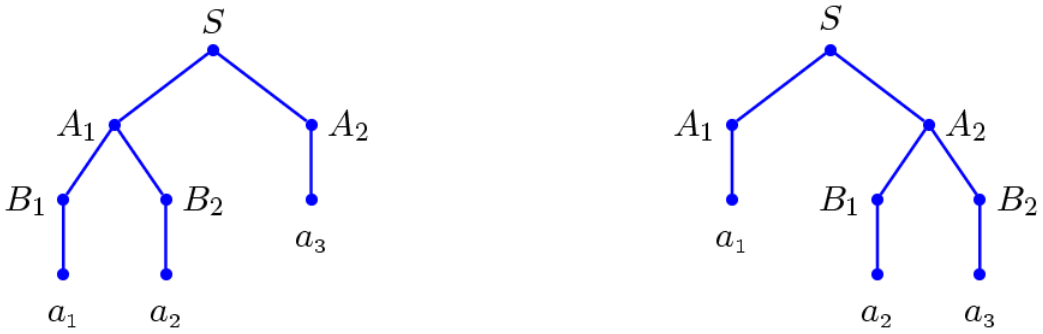
en los que aparecen exactamente 3 variables. Para derivar cadenas de longitud 3 sólo se puede proceder de dos formas:

$$S \Rightarrow A_1A_2 \Rightarrow B_1B_1A_2 \overset{3}{\Longrightarrow} a_1a_2a_3,$$

ó

$$S \Rightarrow A_1A_2 \Rightarrow a_1A_2 \Rightarrow a_1B_1B_2 \overset{2}{\Longrightarrow} a_1a_2a_3.$$

Los árboles de estas derivaciones son:



Cada uno de estos árboles tiene exactamente 5 nodos etiquetados con variables. La situación general es la siguiente: un árbol de derivación de una cadena de longitud n tiene exactamente $2n-1$ nodos etiquetados con variables. Puesto que la raíz del árbol es S y S no es recursiva, en un árbol de derivación de una cadena de longitud n hay exactamente $2n-2$ nodos interiores etiquetados con variables. La demostración general puede hacerse por inducción sobre n y la dejamos como ejercicio para el lector.

Por consiguiente, para determinar si una cadena dada de longitud n es o no generada por G, consideramos todos los posibles árboles de derivación con $2n-2$ variables interiores. Este algoritmo es ineficiente porque si G tiene k variables, hay que chequear no menos de k^{2n-2} árboles (esto sin contar las posibilidades para las hojas). Por consiguiente, el procedimiento tiene complejidad exponencial con respecto al tamaño de la entrada.

Para resolver el problema de la pertenencia hay un algoritmo muy eficiente (su complejidad es polinomial) en el que se usa la llamada *programación dinámica* o *tabulación dinámica*, técnica para llenar tablas progresivamente, re-utilizando información previamente obtenida. El algoritmo que presentaremos se denomina *algoritmo CYK* (nombre que corresponde a las iniciales de los investigadores Cocke, Younger y Kasami). El algoritmo (exhibido en la página siguiente) tiene como entrada una GIC G en FNC y una cadena de n terminales $w = a_1a_2\cdots a_n$; se aplica llenando una tabla de n filas (una por cada terminal de la entrada w) y n columnas. X_{ij} es el conjunto de variables de las que se puede derivar la subcadena de w cuyo primer símbolo está en la posición i y cuya longitud es j. O sea,

$$X_{ij} = \text{conjunto de variables } A \text{ tales que } A \overset{+}{\Longrightarrow} a_ia_{i+1}\cdots a_{i+j-1}.$$

Al determinar los conjuntos X_{ij} se obtienen las posibles maneras de derivar subcadenas de w que permitan construir una derivación de la cadena completa w. La tabla se llena por columnas, de arriba hacia abajo; la primera columna ($j = 1$) corresponde a las subcadenas de longitud 1, la segunda columna ($j = 2$) corresponde a las subcadenas de longitud 2, y así sucesivamente. La última columna ($j = n$) corresponde a la única subcadena de longitud n que tiene w, que es la propia cadena w. Se tendrá que $w \in L(G)$ si y sólo si $S \in X_{1n}$.

Algoritmo CYK

ENTRADA:

Gramática G en FNC y cadena de n terminales $w = a_1 a_2 \cdots a_n$.

INICIALIZAR:

$j = 1$. Para cada i, $1 \leq i \leq n$,

$X_{ij} = X_{i1} :=$ conjunto de variables A tales que $A \to a_i$ es una producción de G.

REPETIR:

$j := j + 1$. Para cada i, $1 \leq i \leq n - j + 1$,

$X_{ij} :=$ conjunto de variables A tales que $A \to BC$ es una producción de G, con $B \in X_{ik}$ y $C \in X_{i+k,j-k}$, considerando todos los k tales que $1 \leq k < j - 1$.

HASTA: $j = n$.

SALIDA: $w \in L(G)$ si y sólo si $S \in X_{1n}$.

Ejemplo Vamos a aplicar el algoritmo CYK a la gramática:

$$G: \quad \begin{cases} S \to BA \mid AC \\ A \to CC \mid b \\ B \to AB \mid a \\ C \to BA \mid a \end{cases}$$

y a la cadena $w = bbab$. Se trata de determinar si $w \in L(G)$ o no. La tabla obtenida al hallar los X_{ij}, $1 \leq i, j \leq 4$, es la siguiente:

		$j=1$	$j=2$	$j=3$	$j=4$
b	$i=1$	$\{A\}$	—	$\{B\}$	$\{S,C\}$
b	$i=2$	$\{A\}$	$\{B,S\}$	$\{S,C\}$	
a	$i=3$	$\{B,C\}$	$\{S,C\}$		
b	$i=1$	$\{A\}$			

A continuación se indica de manera detallada cómo se obtuvo la tabla anterior, columna por columna:

$j=1$. Se obtiene directamente de las producciones de G.

$j=2$. Para X_{12} se buscan cuerpos de producciones en $X_{11}X_{21} = \{A\}\{A\} = \{A\}$. Así que $X_{12} = \{\ \}$.
Para X_{22} se buscan cuerpos de producciones en $X_{21}X_{31} = \{A\}\{B,C\} = \{AB, AC\}$. Así que $X_{22} = \{B, S\}$.
Para X_{23} se buscan cuerpos de producciones en $X_{31}X_{41} = \{B,C\}\{A\} = \{BA, CA\}$. Así que $X_{23} = \{S, C\}$.

$j=3$. Para X_{13} se buscan cuerpos de producciones en $X_{11}X_{22} \cup X_{12}X_{31} = \{A\}\{B,S\} \cup \{\ \} = \{AB, AS\}$. Así que $X_{13} = \{B\}$.
Para X_{23} se buscan cuerpos de producciones en $X_{21}X_{32} \cup X_{22}X_{41} = \{A\}\{S,C\} \cup \{B,S\}\{A\} = \{AS, AC\} \cup \{BA, SA\}$. Así que $X_{23} = \{S, C\}$.

$j=4$. Para X_{14} se buscan cuerpos de producciones en $X_{11}X_{23} \cup X_{12}X_{32} \cup X_{13}X_{41} = \{A\}\{S,C\} \cup \{\ \} \cup \{B\}\{A\} = \{AS, AC\} \cup \{BA\}$. Así que $X_{14} = \{S, C\}$.

Puesto que la variable S pertenece al conjunto X_{14}, se concluye que la cadena $w = bbab$ es generada por G.

Consideremos ahora la entrada $w = baaba$, de longitud 5. Al hallar los X_{ij}, $1 \leq i, j \leq 5$, se obtiene la tabla siguiente. Como $S \in X_{15}$, se concluye que $w \in L(G)$.

		$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
b	$i=1$	$\{A\}$	$\{B,S\}$	—	—	$\{S,B,C\}$
a	$i=2$	$\{B,C\}$	$\{A\}$	$\{A\}$	$\{S,B,C\}$	
a	$i=3$	$\{B,C\}$	$\{S,C\}$	$\{A\}$		
b	$i=4$	$\{A\}$	$\{B,S\}$			
a	$i=5$	$\{B,C\}$				

Al procesar la entrada $w = aaba$ se obtiene la tabla siguiente. Como S no pertenece al conjunto X_{14}, se deduce que w no es generada por G.

		$j=1$	$j=2$	$j=3$	$j=4$
a	$i=1$	$\{B,C\}$	$\{A\}$	$\{B\}$	—
a	$i=2$	$\{B,C\}$	—	—	
b	$i=3$	$\{B\}$	—		
a	$i=4$	$\{B,C\}$			

Problema 3 (Problema de la infinitud). *Dada una gramática $G = (V, \Sigma, S, P)$, ¿es $L(G)$ infinito?*

El lema de bombeo sirve para establecer un criterio que permite resolver este problema (de manera análoga a lo que sucede en el caso de los lenguajes regulares). El criterio aparece en el siguiente teorema.

4.14.1 Teorema. *Sea $G = (V, \Sigma, S, P)$ una gramática en FNC, con variable inicial no recursiva, tal que $L(G) = L$, y sea $k = |V|$ = número de variables de G. El lenguaje L es infinito si y solo si contiene una cadena z tal que $2^k < |z| \leq 2^{k+1}$.*

Demostración: Si $z \in L$ y $2^k < |z| \leq 2^{k+1}$, entonces por la demostración del lema de bombeo, z se puede descomponer como $z = uvwxy$, donde $|vwx| \leq 2^k$, $v \neq \lambda$. L posee infinitas cadenas: uv^iwx^iy para todo $i \geq 0$.

Recíprocamente, si L es infinito, existe $z \in L$ con $|z| \geq 2^k$. Por la demostración del lema de bombeo, $w = uvwxy$ donde $|vwx| \leq 2^k$; además, $v \neq \lambda$ ó $x \neq \lambda$. Si $|z| \leq 2^{k+1}$, la demostración termina. Si $|z| > 2^{k+1} = 2^k + 2^k$, puesto que $|z| = |uy| + |vwx|$, se tendrá

$$|uwy| \geq |uy| = |z| - |vwx| \geq |z| - 2^k > 2^k.$$

De nuevo, si $|uwy| < 2^{k+1}$, la demostración termina; en caso contrario, se prosigue de esta forma hasta encontrar una cadena en L cuya longitud ℓ satisfaga $2^k < \ell \leq 2^{k+1}$. $\square$

Utilizando el Teorema 4.14.1 podemos ahora presentar un algoritmo de decisión para el problema de la infinitud:

1. Convertir la gramática G dada a una gramática equivalente G' en Forma Normal de Chomsky.
2. Aplicar el algoritmo CYK a G', con cada una de las cadenas $|z|$ cuya longitud ℓ satisfaga $2^k < \ell \leq 2^{k+1}$, siendo k el número de variables de G'. L es infinito si y sólo si alguna de las cadenas examinadas está en $L(G')$.

Obsérvese que este algoritmo tiene de complejidad exponencial ya que el número de cadenas z tales que $2^k < |z| \leq 2^{k+1}$ es m^{2^k}, donde m es el número de terminales en la gramática dada.

✎ Hay muchos problemas referentes a gramáticas que son *indecidibles*; para estos problemas no existen algoritmos de decisión. Entre ellos mencionamos:

1. Dada una gramática G, ¿es G ambigua?
2. Dada una gramática G, ¿genera G *todas* las cadenas de terminales?, es decir, ¿$L(G) = \Sigma^*$?
3. Dadas dos gramáticas G_1 y G_2, ¿generan G_1 y G_2 el mismo lenguaje?, es decir, ¿$L(G_1) = L(G_2)$?

Ejercicios de la sección 4.14

1. Sea $G = (V, \Sigma, S, P)$ una gramática dada. Encontrar algoritmos para los siguientes problemas de decisión:
   (i) ¿Hay en $L(G)$ alguna cadena de longitud 1250?
   (ii) ¿Hay en $L(G)$ alguna cadena de longitud mayor que 1250?
   (iii) ¿Hay en $L(G)$ por lo menos 1250 cadenas?
2. Encontrar un algoritmo para el siguiente problema de decisión: dada una gramática $G = (V, \Sigma, S, P)$ y una variable $A \in V$, ¿es A recursiva?, es decir, ¿existe una derivación de la forma $A \overset{+}{\Longrightarrow} uAv$, con $u, v \in (V \cup \Sigma)^*$?

3. Encontrar un algoritmo para el siguiente problema de decisión: dado un lenguaje finito L y una GIC G, ¿se tiene $L \subseteq L(G)$?

4. Sea G la gramática

$$G: \quad \begin{cases} S \to BA \mid AB \\ A \to CA \mid a \\ B \to BB \mid b \\ C \to BA \mid c \end{cases}$$

Ejecutar el algoritmo CYK para determinar si las siguientes cadenas w son o no generadas por G:

(i) $w = bca$.
(ii) $w = acbc$.
(iii) $w = cabb$.
(iv) $w = bbbaa$.