2.13. Ambigüedad y forma normal de Chomsky

Definición (Derivación ambigua)**.** Sea G una GLC. Un string $w \in \mathcal{L}(G)$ es *derivado ambiguamente por G* si posee dos o más derivaciones por la izquierda distintas; equivalentemente, si posee dos o más árboles de derivación distintos.

Definición (Gramática ambigua)**.** Una GLC G es *ambigua* si deriva ambiguamente algún string de su lenguaje. Una gramática que no es ambigua se llama *no ambigua*.

2.14. Forma normal de Chomsky

Definición (Forma normal de Chomsky)**.** Una GLC $G = (V, \Sigma, R, S)$ está en *forma normal de Chomsky* si la variable inicial aparece sólo al lado izquierdo de las reglas y todas las reglas tienen una de las formas siguientes:

- $A \to BC$,
- $A \to a$,
- $S \to \varepsilon$,

donde $A, B, C \in V$, $a \in \Sigma$, y la última regla aparece solamente cuando $\varepsilon \in \mathcal{L}(G)$.

Teorema. Toda GLC posee una GLC equivalente en forma normal de Chomsky.

Para transformar una gramática a forma normal de Chomsky se procede de la siguiente manera:

1. si es necesario, se introduce una nueva variable inicial que no aparece al lado derecho de ninguna regla;
2. se eliminan las reglas ε, conservando $S \to \varepsilon$ cuando corresponde;
3. se eliminan las reglas unitarias $A \to B$;
4. se aíslan los terminales que aparecen en lados derechos de largo al menos dos; y
5. se binarizan los lados derechos que contienen tres o más símbolos.

Ejercicios

1. Consideremos los tokens $\{\texttt{(}, \texttt{)}, \texttt{[}, \texttt{]}, \texttt{t}\}$. Acá `t` representa "texto" y los demás tokens representan paréntesis y corchetes.

   a) Diseñe una GLC y un AP que generen exactamente los strings de tokens que representan expresiones con paréntesis y corchetes bien balanceados.

b) ¿Es ambigua la GLC que diseñó? Justifique su respuesta.

c) Si diseño una GLC ambigua, diseñe una GLC no ambigua que genere el mismo lenguaje.

d) Diseñe una GLC y un AP para las expresiones con paréntesis y corchetes bien balanceados que además contengan exactamente un token `t` sin paréntesis ni corchetes.

e) Transforme alguna de las GLC que diseñó a forma normal de Chomsky.

2. Sea $\Sigma = \{0, 1, \mathtt{f}\}$. Representaremos árboles binarios con hojas etiquetadas con 0 o 1 y nodos internos etiquetados con `f` en texto plano como sigue:

   - 0 y 1 son representaciones válidas.
   - Si x e y son representaciones válidas, entonces $\mathtt{f}xy$ también lo es.

   Llamaremos por $\mathcal{T}$ al lenguaje de todas las representaciones válidas de árboles binarios.

   a) Diseñe una GLC (con tres reglas y una variable) y un AP que acepten $\mathcal{T}$.

   b) Use su gramática anterior para construir una GLC para $\mathcal{F} = (\mathcal{T})^*$, el lenguaje de todas las representaciones válidas de bosques binarios. ¿Es ambigua la GLC que diseñó?

   c) Para $u \in \Sigma^*$, sus *posiciones pendientes* $p\colon \Sigma^* \to \mathbb{Z}$ se definen como

   $$p(u) = 1 + |u|_{\mathtt{f}} - |u|_0 - |u|_1.$$

   Pruebe que $u \in \mathcal{T}$ si y sólo si $p(u) = 0$ y $p(v) \geq 1$ para todo prefijo v propio de u. ¿Qué significa intuitivamente esta condición?

   d) Muestre que si $x, y \in \mathcal{T}$ e y es un prefijo de x, entonces $x = y$. Concluya que toda palabra $w \in \mathcal{F}$ tiene una única factorización $w = t_1 t_2 \dots t_k$ con $t_i \in \mathcal{T}$ para todo $i = 1, \dots, k$.

   e) Use lo anterior para construir una GLC no ambigua para $\mathcal{F}$.

   f) Convierta la GLC no ambigua que diseñó a forma normal de Chomsky.

3. El objetivo de esta pregunta es diseñar un algoritmo eficiente que dada una GLC G y un string w, determine si $w \in \mathcal{L}(G)$. Dicho algoritmo se conoce como *algoritmo CYK* en honor a sus inventores Cocke, Younger y Kasami.

   Sea $G = (V, \Sigma, R, S)$ una GLC **en forma normal de Chomsky** y sea $w \in \Sigma^*$ un string de largo n. Para $1 \leq i \leq j \leq n$, defina la tabla

   $$T[i, j] = \{A \in V : A \overset{*}{\Rightarrow} w_i w_{i+1} \cdots w_j\}.$$

   Es decir, $T[i, j]$ contiene las variables que pueden generar el substring $w_i \cdots w_j$.

   a) Suponga que ya calculó toda la tabla T. Explique cómo decidir, en base a la tabla T, si $w \in \mathcal{L}(G)$.

   b) ¿Cómo calcularía $T[i, i]$ para todo $i = 1, \dots, n$? ¿Cuánto tiempo toma calcular toda la diagonal de la tabla?

   c) Fije $i < j$ y suponga que ya conocemos $T[i, k]$ y $T[k+1, j]$ para todo $k = i, i+1, \dots, j-1$. Explique cómo calcular $T[i, j]$ usando esta información y las reglas de G. ¿Cúanto tiempo toma calcular $T[i, j]$?

d) Use los incisos anteriores para describir un algoritmo de programación dinámica que decida si $w \in \mathcal{L}(G)$.

e) Muestre que su algoritmo corre en tiempo $\mathcal{O}(n^3|R|)$.