Modelos de Computación: Relación de problemas 1

David Cabezas Berrido

Ejercicio 13. Sí que existe un procedimitendo algorítmico para comprobar si dos homomorfismos $f, g : A^* \to B^*$ son iguales, puesto que basta comprobar que $f|_A = g|_A$. Para comprobar esto sabemos que existe un algoritmo porque A es finito, obviamente esta condición es necesaria, probemos que es suficiente ($f|_A = g|_A \implies f = g$).

Tomemos $x \in A^*$, será de la forma $x = a_1 a_2 \dots a_m$ para algún $m \geq 0$ y con $a_i \in A \ \forall i = 1, \dots, m$. Usando la condición y que f y g son homomorfismos tenemos

$$f(x) = f(a_1 a_2 \dots a_m) = f(a_1) f(a_2) \dots f(a_m) = g(a_1) g(a_2) \dots g(a_m) = g(a_1 a_2 \dots a_m) = g(x)$$

como queríamos.

Ejercicio 16. La gramática $G = (\{S, A\}, \{a, b\}, P, S)$ donde
$P = \{S \to abAS, abA \to baab, S \to a, A \to b\}$ genera el lenguaje

$$L = \{u_1 u_2 \dots u_n a : n \geq 0, u_i \in \{baab, abb\} \ \forall i = 1, \dots, n\}$$

La idea es que una S, siempre produce a la derecha otra S o una a, luego cualquier palabra generada debe terminar en a. Para generar las secuencias $baab$ y abb a la izquierda de S se deriva $S \implies abAS \implies baabS$ ó $S \implies abAS \implies abbS$. Como no hay más reglas de derivación, estas son las únicas secuencias que se pueden generar.

Ejercicio 17. La gramática $G = (V, T, P, S)$ donde:

- $V = \{< numero >, < digito >\}$
- $T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $S =< numero >$
- $P = \{< numero > \to < numero >< digito >,$
  $< numero > \to < digito >,$
  $< digito > \to 0|1|2|3|4|5|6|7|8|9\}$

genera el lenguaje de los números naturales. Es claro que cualquier palabra generada va a ser una secuencia de dígitos correspondiente a cualquier número natural. Además podemos generar cualquier secuencia de dígitos de derecha a izquierda, luego cualquier número natural es una palabra del lenguaje generado.

Ejercicio 18. La gramática $G = (\{A, S\}, \{a, b\}, P, S)$ donde $P = \{S \to aS, S \to aA, A \to bA, A \to b\}$ genera el lenguaje $L = \{a^n b^m : n, m \geq 1\}$. Partiendo desde S puedo generar a la izquierda todas las a que quiera hasta generar una A a la derecha (se genera una a como mínimo), desde A puedo generar todas las b que quiera a la izquierda (a continuación de las a) hasta finalizar (por lo menos se genera una b).

Ejercicio 19. Sobre alfabeto $\{a, b, c\}$, una gramática lineal por la derecha o independiente del contexto que genere el lenguaje L será $G = (\{S, X, Y\}, \{a, b, c\}, P, S)$ donde

- L es el lenguaje de las palabras que no contienen dos símbolos b consecutivos: $P = \{S \to aS|bX|cS|\varepsilon,\ X \to aS|cS|\varepsilon\}$.

- L es el lenguaje de las palabras que contienen dos símbolos b consecutivos: $P = \{S \to aS|bX|cS,\ X \to aS|bY|cS,\ Y \to aY|bY|cY|\varepsilon\}$

- L es el lenguaje de las palabras que contienen un número impar de símbolos c: $P = \{S \to aS|bS|cX,\ X \to aX|bX|cS|\varepsilon\}$

- L es el lenguaje de las palabras que no contienen el mismo número de símbolos b que de símbolos c:

  Definiré dos gramáticas, la que genera las palabras que contienen más símbolos b que símbolos c y la que genera las palabras que contienen más símbolos c que símbolos b.

  $$P_b = \{S_b \to aS_b|bY|cXX,\ X \to YX|bY,\ Y \to YY|a|b|cX|Xc|\varepsilon\}$$

  Se puede generar cualquier secuencia, pero cada vez que se añade una c aparecen X que sólo desaparecen añadiendo una b. Del mismo modo definimos

  $$P_c = \{S_c \to aS_c|bX'X'|cY',\ X' \to Y'X'|cY',\ Y' \to Y'Y'|a|bX'|X'b|c|\varepsilon\}$$

  Cualquier palabra generada por estas reglas tendrá más símbolos b que símbolos c o viceversa, pero siempre tendrá distinto número de símbolos b y c. Por tanto, sólo tenemos que unir ambas gramáticas: $P = \{S \to S_b|S_c\} \cup P_b \cup P_c$