# Examen de Computabilidad y Complejidad

(CMC)

16 de septiembre de 1997

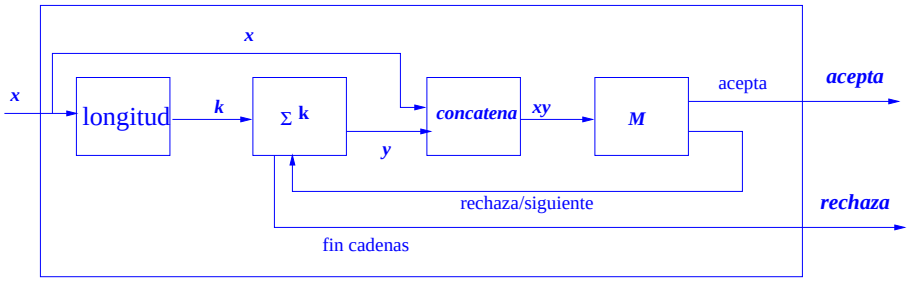
**(I) Cuestiones** (justifique formalmente las respuestas)

1. Sea $L$ un lenguaje recursivo sobre $\Sigma$ cuyas palabras poseen longitud par. Sea $L' = \{u \in \Sigma^* \mid \exists v \in \Sigma^*, uv \in L, |u| = |v|\}$. ¿ Es $L'$ recursivo ?

(1.5 ptos)

Solución

El lenguaje $L'$ es recursivo. Propondremos una máquina de Turing que acepte $L'$ y que garantice la parada ante cualquier cadena de entrada. Para el esquema que vamos a proponer, contaremos con un módulo *longitud* que, dada una cadena de entrada, devuelve como salida su longitud. Esto se puede hacer con una máquina de Turing que recorra la entrada y actualice un contador. De igual forma, contaremos con un módulo $\Sigma^k$ que genera todas las cadenas definidas sobre $\Sigma$ con longitud $k$. De nuevo, el anterior módulo se basa en un generador de cadenas en orden lexicográfico que puede ser realizado mediante una máquina de Turing multicinta. También contaremos con un módulo *concatena* basado en una máquina de Turing que, dadas dos cadenas de entrada proporciona como salida la concatenación de ambas. Por último, contaremos con una máquina de Turing $M$ que acepta $L$ y garantiza siempre su parada (ya que $L$ es recursivo). A partir de los anteriores módulos proponemos el siguiente esquema de una máquina de Turing que acepta $L'$ y que garantiza siempre la parada




El funcionamiento del anterior esquema se explica a continuación. Dada una cadena de entrada $x$, en primer lugar, se calcula su longitud $k$ mediante el módulo *longitud*. El valor $k$ se toma como entrada en el módulo $\Sigma^k$ que generará mediante petición todas las cadenas de $\Sigma$ con longitud igual a $k$. Cada una de las cadenas $y$ generada en el módulo anterior se concatena con la cadena de entrada $x$ en el módulo *concatena* que proporciona como salida la cadena $xy$. A continuación, se establece si la cadena $xy$ anterior pertenece o no a $L$ mediante la máquina $M$. En el caso de que $M$ acepte

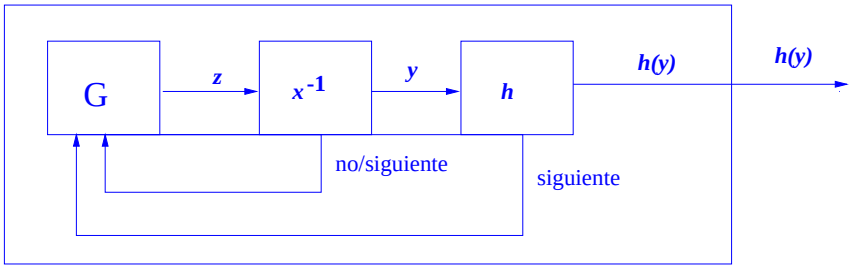
$xy$ entonces se acepta la cadena de entrada $x$ (ya que pertenece a $L'$). En el caso de que la cadena $xy$ no se acepte, se solicita una nueva cadena $y$ al módulo $\Sigma^k$ y se replica el proceso anterior. Si el módulo $\Sigma^k$ genera todas las cadenas de longitud $k$ y ninguna de ellas ha producido un resultado de aceptación en $M$ entonces se rechaza la cadena de entrada $x$ puesto que no pertenece a $L'$. El anterior esquema permite aceptar $L'$ y garantiza la parada ante cualquier cadena de entrada, por lo tanto $L'$ es recursivo.

2. Sea $L$ un lenguaje recursivamente enumerable sobre $\Sigma$ y $x \in \Sigma^*$ una palabra fija. Sea $h : \Sigma^* \to \Delta^*$ un homomorfismo. ¿ Es $h(Lx^{-1})$ recursivamente enumerable ?

(1.5 ptos)

Solución

$h(Lx^{-1})$ es recursivamente enumerable. Recordemos, en primer lugar, que $Lx^{-1} = \{y \in \Sigma^* : yx \in L\}$. Dado que $L$ es recursivamente enumerable, existe una máquina de Turing $G$ que genera $L$. Propondremos el esquema de una máquina de Turing que genere $h(Lx^{-1})$. Contaremos con un módulo $x^{-1}$ que, dada una cadena de entrada, establece si la cadena tiene algún sufijo que coincida con $x$ y, en caso afirmativo, elimine el sufijo dando el resto de la cadena como salida (esto se puede hacer trivialmente con una máquina de Turing multicinta). También contaremos con un módulo $h$ que, tomando como entrada una cadena sobre $\Sigma$, nos devuelva como salida el homomorfismo $h$ aplicado sobre la cadena de entrada. Esto, de nuevo, se puede hacer con una máquina de Turing multicinta que, a medida que lee la entrada, escribe las imágenes de cada símbolo en la cinta de salida. A partir de los anteriores módulos proponemos el siguiente esquema de generación de $h(Lx^{-1})$




El funcionamiento del anterior esquema se explica a continuación. El módulo $G$ genera las cadenas de $L$. La cadena $z$ generada por $G$ y perteneciente a $L$ se proporciona como entrada al módulo $x^{-1}$ que establece si $z$ finaliza con el sufijo $x$. En caso afirmativo, $x^{-1}$ proporciona como salida la cadena $y$ de forma que $z = yx$. En caso negativo, se solicita otra cadena al módulo $G$. La anterior cadena $y$ se proporciona como entrada al módulo $h$ que emite como salida la cadena $h(y)$ que se escribe en la cinta de salida del esquema general. Una vez escrita la cadena $h(y)$ como salida, se solicita otra cadena al módulo $G$ y se vuelve a empezar todo el proceso descrito. Dado que el esquema planteado genera el lenguaje $h(Lx^{-1})$, podemos concluir que éste es un lenguaje recursivamente enumerable.

3. ¿ Es $L = \{a^i b^j c^{max(i,j)} \mid i, j \geq 1\}$ incontextual ?

(1.5 ptos)

Solución

El lenguaje $L$ no es incontextual. Haremos la demostración mediante el lema de bombeo. Sea $n$ la constante del lema y tomemos la cadena $z = a^n b^n c^n$ que pertenece

a $L$ y supera en longitud a la constante $n$. Dado que, de acuerdo con el lema de bombeo, podemos factorizar $z = uvwxy$ estudiaremos todos los posibles casos de localización de las subcadenas $v$ y $x$.

(a) Tomemos $v$ y $x$ formadas sólo por símbolos $a$ con $|vx| = j \geq 1$. Si tomamos un valor de $i = 2$ podemos formar la cadena $uvvwxxy = a^{n+j}b^nc^n$ que no pertenece al lenguaje ya que la cantidad de símbolos $c$ no es el máximo entre $n + j$ y $n$.

(b) Tomemos $v$ y $x$ formadas sólo por símbolos $b$ con $|vx| = j \geq 1$. Si tomamos un valor de $i = 2$ podemos formar la cadena $uvvwxxy = a^nb^{n+j}c^n$ que no pertenece al lenguaje ya que la cantidad de símbolos $c$ no es el máximo entre $n$ y $n + j$.

(c) Tomemos $v$ y $x$ formadas sólo por símbolos $c$ con $|vx| = j \geq 1$. Si tomamos un valor de $i = 0$ podemos formar la cadena $uwy = a^nb^nc^{n-j}$ que no pertenece al lenguaje ya que la cantidad de símbolos $c$ no es $n$ (que es el máximo número de símbolos $a$ y $b$).

(d) Tomemos $vx$ formada por símbolos $a$ y $b$ con $|vx|_a = j \geq 1$ y $|vx|_b = k \geq 1$. Si tomamos un valor de $i = 0$ podemos formar la cadena $uwy = a^{n-j}b^{n-k}c^n$ que no pertenece al lenguaje ya que la cantidad de símbolos $c$ no es el máximo entre $n - j$ y $n - k$.

(e) Tomemos $vx$ formada por símbolos $b$ y $c$ con $|vx|_b = j \geq 1$ y $|vx|_c = k \geq 1$. Si tomamos un valor de $i = 0$ podemos formar la cadena $uwy = a^nb^{n-j}c^{n-k}$ que no pertenece al lenguaje ya que la cantidad de símbolos $c$ no es el máximo entre $n$ y $n - j$.

Dado que no se pueden plantear más casos de localización de las subcadenas $v$ y $x$ y, en todos los casos estudiados, no se cumple el lema de bombeo, podemos concluir que $L$ no es incontextual.

4. Demuestre que todo lenguaje incontextual sobre $\Sigma$ que no contenga la cadena vacía es generado por una gramática $G = (N, \Sigma, P, S)$ cuyas reglas son todas de la forma $A \to \gamma a$ con $A \in N$, $\gamma \in N^*$, $a \in \Sigma$.

(1.5 ptos)

Solución

Para demostrar el enunciado de la cuestión, partiremos del hecho de que todo lenguaje incontextual que no contiene la cadena vacía puede ser generado por una gramática incontextual en Forma Normal de Greibach, es decir, con las producciones de la forma $A \to a\gamma$ con $A \in N$, $\gamma \in N^*$, $a \in \Sigma$.

Sea $L$ un lenguaje incontextual que no contiene la cadena vacía y partiremos de una gramática $G_1$ totalmente simplificada que genera a $L$. A partir de $G_1$ obtenemos una gramática $G_2$ que genera el reverso del lenguaje generado por $G_1$, es decir $L(G_2) = (L(G_1))^r = L^r$. A partir de $G_2$, obtenemos una gramática equivalente en Forma Normal de Greibach $G_3$. Es decir, $L(G_3) = L(G_2) = L^r$. Obsérvese que en $G_3$ todas las producciones son de la forma $A \to a\gamma$ con $A \in N$, $\gamma \in N^*$, $a \in \Sigma$. Por último, obtenemos una gramática $G_4$ para el lenguaje reverso de $L(G_3)$ de acuerdo con la construcción vista en clase. Al aplicar la citada construcción, todas las producciones de $G_4$ son de la forma $A \to \gamma a$ con $A \in N$, $\gamma \in N^*$, $a \in \Sigma$. Además, $L(G_4) = (L(G_3))^r = (L^r)^r = L$. Por lo tanto, dado el lenguaje $L$ siempre podemos encontrar la gramática que lo genere con las producciones en la forma predeterminada en el enunciado.

**(II) PROBLEMAS:**

5. Sea $G$ la gramática:

$$S \to AB \mid a$$
$$A \to aBS \mid b$$
$$B \to Sba \mid \lambda$$

Sea $h$ el homomorfismo definido por $h(a) = ab$ y $h(b) = a$. Dar una gramática $G'$ tal que $L(G') = (h(L(G)) \cup (L(G))^r)^*$.

(2 ptos)

Solución

Proporcionamos, en primer lugar, una gramática para $h(L(G))$

$S_h \to A_h B_h \mid ab$

$A_h \to abB_h S_h \mid a$

$B_h \to S_h aab \mid \lambda$

A continuación, una gramática para $(L(G))^r$

$S \to BA \mid a$

$A \to SBa \mid b$

$B \to abS \mid \lambda$

La gramática para $h(L(G)) \cup (L(G))^r$ queda definida por las siguientes reglas

$S_1 \to S_h \mid S$

$S_h \to A_h B_h \mid ab$

$A_h \to abB_h S_h \mid a$

$B_h \to S_h aab \mid \lambda$

$S \to BA \mid a$

$A \to SBa \mid b$

$B \to abS \mid \lambda$

Por último, la gramática $G'$ que se solicita en el enunciado y que genera $(h(L(G)) \cup (L(G))^r)^*$ queda definida por las siguientes reglas, donde $S'$ es el axioma

$S' \to S_1 S' \mid \lambda$

$S_1 \to S_h \mid S$

$S_h \to A_h B_h \mid ab$

$A_h \to abB_h S_h \mid a$

$B_h \to S_h aab \mid \lambda$

$S \to BA \mid a$

$A \to SBa \mid b$

$B \to abS \mid \lambda$