Examen de Computabilidad y Complejidad

(CMC)

8 de julio de 2009

(I) CUESTIONES: (Justifique formalmente las respuestas)

1. ¿Es el lenguaje $\{a^n b^m c^k \ / \ (n \neq m) \vee (n \neq k)\}$ incontextual?

EL lenguaje dado puede descomponerse como $L \cup L'$, donde

$$L = \{a^n b^m c^k \ / \ n \neq m,\ k \geq 0\},\ \text{y}$$

$$L' = \{a^n b^m c^k \ / \ n \neq k,\ m \geq 0\}.$$

Nuevamente L puede descomponerse como L_1L_2, donde

$$L_1 = \{a^n b^m \ / \ n \neq m\} \text{ y } L_2 = \{c^k \ / \ k \geq 0\}.$$

Seguidamente obtendremos para cada uno de estos lenguajes una gramática incontextual que lo genere. A partir de esto, y tomando en cuenta que los lenguajes incontextuales son cerrados para la unión y la concatenación, tendremos establecido que el lenguaje dado es incontextual.

I) $L_1 = \{a^n b^m \ / \ n \neq m\}$. $n \neq m$ significa que $n = m + k$ o que $n + k = m$, para algún $k > 0$; en consecuencia se tiene que si $x \in L_1$, entonces

$$x = a^n b^{n+k} \quad \vee \quad x = a^{n+k} b^n,\ \text{con } n \geq 0 \text{ y } k > 0.$$

En consecuencia la gramática

$$S \rightarrow aSb \mid aA \mid Bb,\ \ A \rightarrow aA \mid \lambda,\ \ B \rightarrow Bb \mid \lambda,$$

genera a L_1.

II) $L_2 = \{c^k \ / \ k \geq 0\}$. Este lenguaje puede, de modo inmediato, generarse con la gramática

$$S \rightarrow cS \mid \lambda.$$

III) $L' = \{a^n b^m c^k \ / \ n \neq k,\ m \geq 0\}$. A partir de lo dicho en *I* podemos inmediatamente obtener la siguiente gramática para la generación de L'

$$S \rightarrow aSc \mid aA \mid Cc,\ \ A \rightarrow aA \mid B,\ \ C \rightarrow Cc \mid B,\ \ B \rightarrow bB \mid \lambda.$$

(1.0 punto)

2. ¿Es el lenguaje $\{x \in \{a,b\}^* \ / \ (|x|_a)! \leq |x|_b\}$ incontextual?

El lenguaje dado no es incontextual. Lo estableceremos aplicando operaciones de cierre y el lema de iteración.

Sea $L = \{x \in \{a,b\}^* \ / \ (|x|_a)! \leq |x|_b\} \cap a^*b^* = \{a^m b^k \ / \ m! \leq k\}$. Puesto que la intersección con lenguajes regulares es una operación de cierre en la clase de los lenguajes incontextuales se tiene que si el lenguaje dado fuera incontextual también lo sería L.

Seguidamente estableceremos que L no es incontextual viendo que incumple el lema de iteración. Supóngase que lo cumple y sea n la constante del lema; sea $z = a^n b^{n!}$ que debe poder factorizarse como $z = uvwxy$ en las condiciones ya conocidas del lema. Veamos que esto no es posible. Cualquier posible factorización se encuentra en uno de los siguientes casos:

I) Si $vx \in a^+$, entonces tomando una iteración $i > 1$ se obtiene una palabra $s = a^j b^{n!}$ donde $n < j$. En consecuencia $j! > n!$ y en por tanto $s \notin L$.

II) Si $vx \in b^+$, entonces tomando una iteración $i = 0$ se obtiene una palabra $s = a^n b^j$ donde $j < n!$. Así $s \notin L$.

III) Si $v \in a^+b^+$ o $x \in a^+b^+$, entonces tomando una iteración $i > 1$ se obtiene una palabra $s \notin a^*b^*$ y en consecuencia $s \notin L$.
IV) Si $v \in a^+$ y $x \in b^+$, sean $p = |v|$ y $q = |x|$, entonces tomando la iteración $i = 2$, se obtiene la palabra $s = a^{n+p}b^{n!+q}$. Ahora se tiene que: $(n+p)! \geq (n+1)!$, ya que $p \geq 1$, y $(n+1)! = n!(n+1) = n!n + n! > n! + q$, ya que $1 \leq q < n$; así, puesto que $(n+p)! > n! + q$, se tiene que $s \notin L$.

Puesto que no existe ninguna otra factorización en las condiciones exigidas por el lema se concluye que L no es incontextual y en consecuencia el lenguaje dado tampoco lo es.

(1.0 punto)
(2.0

3. Sea Σ un alfabeto y $L \subseteq \Sigma^*$, se define
$P(L) = \{x \in \Sigma^* \ / \ (\exists y \in \Sigma^*)((xy \in L) \wedge (|x| = |y|))\}$.
I. Si L es recursivo ¿lo es también P(L)? *(0.75 puntos)*
II. Si L es recursivamente enumerable ¿lo es también P(L)? *(0.75 puntos)*

I. Si L es recursivo, entonces también lo es P(L); para su demostración construiremos una MT M■ que lo reconozca y que se detenga para cada entrada. Puesto que L es recursivo existe una MT M que lo reconoce y que se detiene para cada entrada. La máquina M■ opera como sigue. Cuando se le aplica una entrada x, arranca un generador canónico G_x de todas las palabras de longitud $|x|$ de Σ^*, este generador comienza generando la primera palabra en el orden canónico de longitud $|x|$ y seguidamente continuará la generación, de palabra en palabra, cada vez que reciba una señal de control ✓. Cuando se le solicite mediante esta señal que genere la siguiente palabra y ya no haya más indicará a la máquina M■ que se detenga rechazando. Cada vez que G_x genere una palabra y, ésta se concatenará a x formando la palabra $z = xy$ que se aplicará a la máquina M; si M acepta, entonces M■ se detendrá aceptando. Si M rechaza se enviará a G_x una señal ✓. En estas condiciones la máquina M■, por construcción, se detiene para cada entrada y reconoce a P(L).

II. Si L es recursivamente enumerable, entonces P(L) también lo es; para su demostración construiremos un generador de Turing **M** que lo genere. Puesto que L es recursivamente enumerable existe un generador de Turing M que lo genera. El generador **M** opera como sigue. Comienza arrancando el generador M que actuará, controlado por una señal de continuación ✓, de modo similar al generador G_x del apartado anterior. Cada vez que M genera una palabra z ésta, si se puede (longitud par), se divide en dos mitades, y se enumera la primera en la cinta de salida, enviándose en cualquier caso (tanto si se puede dividir como si no) la señal ✓ al generador M. La división de z, en xy con $|x| = |y|$, puede realizarse utilizando una cinta infinita en ambos sentidos con dos sectores uno para z y el otro para marcar conveniente con dos marcas (×, ⊠), unificando, sus posiciones iniciales con sus correspondientes posiciones finales. Así, por construcción, G(**M**) = P(L).

4. Sea Σ un alfabeto y $L \subseteq \Sigma^*$, se define la función $g: \Sigma^* \longrightarrow \Sigma^*$ definida como sigue: 1) $g(\lambda) = \lambda$, y 2) $\forall x \in \Sigma^*, \forall a \in \Sigma, \ g(xa) = g(x)a^{|x|}$.
I. Si L es recursivo ¿lo es también $g^{-1}(L)$? *(0.75 puntos)*
II. Si L es recursivamente enumerable ¿lo es también $g^{-1}(L)$? *(0.75 puntos)*

Recuérdese que $g^{-1}(L) = \{x \in \Sigma^* \,/\, g(x) \in L\}$, de este modo : $x \in g^{-1}(L) \Leftrightarrow g(x) \in L$. Así puesto que la pertenencia de x a $g^{-1}(L)$ se reduce a la pertenencia de g(x) a L, lo primero que abordaremos es la construcción de una MT $\mathcal{G}$, con cintas infinitas en ambos sentidos, que calcule la función g. Esta máquina dispone de dos cintas: una, de entrada, en la que se proporciona `x` y otra, de resultado, en la que escribe g(x). La primera es una cinta con tres sectores. En uno de ellos se encuentra x siendo los otros dos de control y que denominaremos en lo que sigue `c1` y `c2`. Cuando se le propociona una palabra `x`, si $x \in \Sigma \cup \{\lambda\}$, entonces no se escribe nada en la cinta de resultado ya que en este caso $g(x) = \lambda$; en otro caso el cabezal se ubica sobre la celdilla correspondiente al último símbolo de `x` y marca `c1` con ✓, transitando a un estado modificado finitamente que tenga, en un campo, el símbolo correspondiente de `x` y desplazándose a la celdilla anterior en la que marca `c2` con ✗, ahora mientras en la celdilla marcada con ✗ exista un símbolo de `x` copia el símbolo marcado con ✓, codificado en el estado, en la cinta de resultado y desplaza los dos cabezales una posición a la izquierda, repitiéndose, en estas condiciones, mientras se pueda, este paso. Cuando no se dé la condición anterior se repite todo este proceso desplazando la marca ✓ una posición a la izquierda y desplazando la marca ✗ a la celdilla a su izquierda. Procediéndose de este modo mientras la celdilla marcada con ✓ contenga un símbolo de `x`.

I. Si L es recursivo, entonces $g^{-1}(L)$ también lo es. Sea M una máquina que se detiene para cada entrada y que reconoce a L. Sea $\mathcal{M}$ la MT que opera como sigue cuando se le aplica una entrada x utiliza a $\mathcal{G}$ para calcular g(x) que seguidamente aplica a M dejándole la aceptación o rechazo. De este modo $\mathcal{M}$ se detiene para cada entrada y reconoce el lenguaje $g^{-1}(L)$.

II. Si L es un lenguaje recursivamente enumerable, entonces $g^{-1}(L)$ también lo es. Se procede como en el caso anterior utilizando una MT M tal que L(M) = L.

(II) PROBLEMAS

5. Desarrolle un módulo *Mathematica*, adecuadamente explicado, que reciba como entrada una gramática incontextual G y retorne `True` si cada uno de los consecuentes α de sus producciones es de una de las formas siguientes: α = `AbB` o α = `AB` o α = `b`, donde `A` y `B` son símbolos auxiliares, y `b` puede ser un símbolo terminal o λ; retornando en otro caso `False`.

A patir del formato definido en el enunciado se tiene la siguiente clasificación en cuanto a la validez de los consecuentes:

a) $|\alpha| = 0$: *válido*.
b) $|\alpha| = 1$: *válido* $\Leftrightarrow \alpha = a$, a $\in$`Terminales`.
c) $|\alpha| = 2$: *válido* $\Leftrightarrow \alpha = AB$, A,B $\in$ `Auxiliares`.
d) $|\alpha| = 3$: *válido* $\Leftrightarrow \alpha = AaB$, A,B $\in$ `Auxiliares`, a $\in$`Terminales`.
e) $|\alpha| > 3$: *inválido*.

Así podemos establecer:

válido = $(|\alpha| \le 3) \wedge [(|\alpha| = 0) \vee (|\alpha| = 1 \wedge \alpha = a;\ a \in$ `Terminales`$) \vee (|\alpha|=2 \wedge \alpha = AB;\ A,B \in$ `Auxiliares`$) \vee (|\alpha|=3 \wedge \alpha = AaB;\ A,B \in$ `Auxiliares`, $a \in$ `Terminales`$)]$

Todo esto queda plasmado en el siguiente programa.

```
P[G_List] := Module[
    {aux, ter, alfa, prod, long, i, j, válido, n, cons, nn},
    aux = G[[1]];
    ter = G[[2]];
    prod = G[[3]];
    válido = True;
    n = Length[prod];
    For[i = 1, válido && (i ≤ n), i++,
      cons = prod[[i, 2]];
      nn = Length[cons];
      For[j = 1, válido && (j ≤ nn), j++,
        alfa = cons[[j]];
        long = Length[alfa];
        válido = (long ≤ 3)  && ((long == 0)              ||
                (long == 1 && MemberQ[ter, alfa[[1]]])    ||
                (long == 2 && MemberQ[aux, alfa[[1]] &&
                              MemberQ[aux, alfa[[2]]])    ||
                (long == 3 && MemberQ[aux, alfa[[1]] &&
                              MemberQ[ter, alfa[[2]] &&
                              MemberQ[aux, alfa[[3]]]))
        ]
      ];
    Return[válido]
    ]
```

(2.0 puntos)

6. Dadas las gramáticas

   $G\ : S \to AC, \qquad A \to aA \mid bA \mid \lambda, \qquad C \to cC \mid \lambda,$ y

   $G' : S \to BA, \qquad A \to aA \mid \lambda, \qquad B \to bB \mid cB \mid \lambda,$

   y la sustituciones f y g definidas como sigue: $f(a) = \{a\}$, $f(b) = \{b\}$, $f(c) = L(G')$; y $g(a) = L(G)$, $g(b) = \{b\}$ y $g(c) = \{c\}$.

   Obténgase una gramática incontextual para el lenguaje $f(L(G)^+) \cup g(L(G')^r)$.

   Aplicando estrictamente la teoría (sin introducir ninguna simplificación) se tiene:

   f(a): $S_1 \to a$

   f(b): $S_2 \to b$

   f(c): $S_3 \to B_3A_3 \qquad A_3 \to aA_3 \mid \lambda \qquad B_3 \to bB_3 \mid cB_3 \mid \lambda$

   g(a): $S_4 \to A_4C_4 \qquad A_4 \to aA_4 \mid bA_4 \mid \lambda \qquad C_4 \to cC_4 \mid \lambda$

   g(b): $S_5 \to b$

   g(c): $S_6 \to c$

   $f(L(G)^+)$: $S_7 \to S_8S_7 \mid S_8$

   $S_8 \to A_8C_8 \qquad A_8 \to S_1A_8 \mid S_2A_8 \mid \lambda \qquad C_8 \to S_3C_8 \mid \lambda$

   $g(L(G)^r)$: $S_9 \to A_9B_9, \qquad A_9 \to A_9S_4 \mid \lambda \qquad B_9 \to B_9S_5 \mid B_9S_6 \mid \lambda$

   $S \to S_7 \mid S_9$

(1.5 puntos)

7. Dada la gramática G obtenga una gramática G' simplificada y en Forma Normal de Chomsky con $L(G') = L(G) - \{\lambda\}$.

   $S \to ASA \mid CS \mid BD \mid DAE \mid AbC \qquad A \to ba \mid aAb \mid \lambda$

B → BB | EE | Ba

C → ab | AAA

D → aa | ESSE | SB | λ

E → EB | EbE | aB

Eliminación de los símbolos inútiles

Eliminación de los símbolos no generativos: B,E

S → ASA | CS | AbC

A → ba | aAb | λ

C → ab | AAA

D → aa | ESSE | SB | λ

Eliminación de los símbolos no alcanzables: D

S → ASA | CS | AbC

A → ba | aAb | λ

C → ab | AAA

Eliminación de las producciones-λ

Símbolos anulables: A, C

S → ASA | AS | SA | S | CS | AbC | bC | Ab | b

A→ ba | aAb | ab

C → ab | AAA| AA | A

Eliminación de las producciones unitarias

S → ASA | AS | SA | CS | AbC | bC | Ab | b

A→ ba | aAb | ab

C → ab | AAA| AA | ba | aAb

Todos los símbolos son útiles

Forma normal de Chomsky

S → ASA | AS | SA | CS | AZC | ZC | AZ | b

A→ ZY | YAZ | YZ

C → YZ | AAA| AA | ZY | YAZ

Y → a

Z → b

S → AX | AS | SA | CS | AV | ZC | AZ | b

A→ ZY | YU | YZ

C → YZ | AT| AA | ZY | YU

Y → a

Z → b

X → SA

V → ZC

U → AZ

T → AA

(1.5 puntos)