Examen de Computabilidad y Complejidad

(CMC)

6 de febrero de 2008

(I) CUESTIONES: (Justifique formalmente las respuestas)

1. Sea Σ un alfabeto con más de un símbolo. ¿Es el lenguaje

$$\{x \in \Sigma^* / (\exists u,z \in \Sigma^*)((x = uyz) \wedge (y = zu))\}$$

incontextual?

Estableceremos que este lenguaje no es incontextual demostrando, en última instancia, que no cumple con el Lema de Iteración. Podemos transformar el lenguaje dado de la manera que sigue:

$$\{x \in \Sigma^* / (\exists u,z \in \Sigma^*)((x = uyz) \wedge (y = zu))\} = \{x \in \Sigma^* / (\exists u,z \in \Sigma^*)((x = uzuz))\} =$$
$$\{x \in \Sigma^* / (\exists t \in \Sigma^*)((x = tt))\} = \{t^2 / t \in \Sigma^*\}.$$

Sean dos símbolos a, b $\in \Sigma$, a $\neq$ b, si ahora este lenguaje lo intersectamos con el lenguaje regular a*b*a*b* obtenemos como resultado el lenguaje $\{a^n b^m a^n b^m / n, m \geq 0\}$ que ya sabemos que no es incontextual porque así se ha demostrado en clase utilizando el Lema de Iteración. Por tanto, a partir de las propiedades de cierre de los lenguajes incontextuales, tampoco es incontextual el lenguaje dado.

(1.0 punto)

2. ¿Es el lenguaje $\{xy / x \in \{0,1\}^* \wedge (y = 2^n) \wedge (n = |x|_0)\}$ incontextual? .

El lenguaje dado, que para abreviar lo denominaremos L en lo que sigue, es incontextual porque es generado por la siguiente gramática incontextual

$$G: S \rightarrow 1S \mid 0S2 \mid \lambda$$

Seguidamente demostramos que L = L(G). Sea $\Sigma = \{0,1,2\}$.

I) $L(G) \subseteq L$. Lo demostramos por inducción sobre el número n de pasos de las derivaciones.

- n = 1 (caso base): Si $S \Rightarrow z \in \Sigma^*$, entonces necesariamente $z = \lambda$ que pertenece a L.
- n = m + 1 (caso general): Por hipótesis de inducción asumimos que si $S \Rightarrow^m w \in \Sigma^*$, entonces $z \in L$. Sea ahora que $S \Rightarrow^{m+1} z \in \Sigma^*$, se tiene que la primera producción utilizada en la derivación es $S \rightarrow 1S$ o $S \rightarrow 0S2$, así tenemos

$$S \Rightarrow 1S \Rightarrow^m z \in \Sigma^* \quad \text{o} \quad S \Rightarrow 0S2 \Rightarrow^m z \in \Sigma^*,$$

donde $S \Rightarrow^m w \in \Sigma^*$. En consecuencia z = 1w o z = 0w2 con $w \in L$. Así w = xy e $y = 2^k$ con $k = |x|_0$, por tanto también $z \in L$ ya que en el primer caso z = 1xy ($k = |1x|_0$ e $y = 2^k$) y en el segundo $0xy2 = 0x2^k2 = 0x2^{k+1}$ ($k + 1 = |0x|_0$), cumpliéndose en ambos casos la condición de pertenencia a L.

II) $L \subseteq L(G)$. Sea $z \in L$, así z = xy donde $y = 2^k$ con $k = |x|_0$. Sea $x = a_1a_2.....a_n$, donde cada $a_i \in \{0,1\}$. Veamos que $S \Rightarrow^+ z$. Si $z = \lambda$, entonces $S \Rightarrow \lambda = z$ utilizando la producción $S \rightarrow \lambda$. En otro caso comenzamos a derivar z secuencialmente de a_1 a a_n utilizando para cada a_i la producción $S \rightarrow 1S$ si a_i es 1 o la producción $S \rightarrow 0S2$ si a_i es 0 y finalmente la producción $S \rightarrow \lambda$. Por tanto, según este esquema derivativo, tendremos $S \Rightarrow^+ a_1a_2.....a_n2^k = xy = z \in L(G)$.

(1.0 punto)

3. Sea Σ un alfabeto y sea $\leq$ una relación de orden canónico definida sobre Σ^*. Sean dos lenguajes $L, L' \subseteq \Sigma^*$, se define la operación $\mu(L,L')$ del modo que sigue:

$$\mu(L,L') = \{x \in L / (\exists y \in L')(y \leq x)\}.$$

2.1) Si L y L′ son lenguajes recursivamente enumerables, entonces ¿lo es también $\mu(L,L')$?

2.2) Si L y L′ son lenguajes recursivos, entonces ¿lo es también $\mu(L,L')$?

Veamos que la operación binaria propuesta μ es de cierre en ambos casos. Lo estableceremos mediante dos caminos diferentes: uno, utilizando propiedades de cierre y el otro, asociando las correspondientes máquinas de Turing.

Utilizando las propiedades de cierre procedemos como sigue. Si $L' = \varnothing$, entonces $\mu(L,L') = \varnothing$, que es tanto un lenguaje recursivo como recursivamente enumerable. Si $L' \neq \varnothing$, entonces sea $\min_{\leq}(L')$ el mínimo elemento de L' de acuerdo al orden canónico dado $\leq$. Sea $\Theta = \{x \in \Sigma^* / x < \min_{\leq}(L')\}$ que claramente es un lenguaje finito, en consecuencia es también un lenguaje recursivo. Así su complementario $\Theta^c = \Sigma^* - \Theta = \{x \in \Sigma^* / \min_{\leq}(L') \leq x\}$ también es recursivo y, por tanto, también recursivamente enumerable. Ahora puede verse que $\mu(L,L') = L \cap \Theta^c$, puesto que

$$x \in \mu(L,L') \Leftrightarrow (x \in L \wedge \min_{\leq}(L') \leq x) \Leftrightarrow x \in L \cap \Theta^c,$$

en consecuencia puesto que los lenguajes recursivos, y también los recursivamente enumerables, son cerrados para la intersección se tiene que la operación $\mu(L,L')$ es de cierre tanto en la clase de los lenguajes reursivos como en la clase de los lenguajes recursivamente enumerables.

Otro manera de resolver la cuestión es definir las máquinas de Turing correspondientes en cada caso.

- Para el caso *2.1* sea M una máquina de Turing tal que L(M) = L, y sea M' un generador de Turing tal que G(M') = L'. Definimos seguidamente una máquina de Turing M_μ tal que $L(M_\mu) = \mu(L,L')$. Esta máquina opera como sigue. Cuando se le suministra una palabra de entrada `x` ésta se aplica a la máquina M y sólo en el caso de que ésta acepte se continúa arrancando el generador M' que estará controlado mediante una señal de continuación. Este generador cuando se arranca comienza el proceso generador y cuando genera la primera palabra se suspende hasta que recibe la señal de continuación en cuyo momento continuará el proceso generativo y seguirá así operando, mientras sea necesario, de acuerdo a este esquema. Cada vez que se genera una palabra `y` se comprueba si o no $y \leq x$; si es que **sí** se termina el proceso aceptando `x`, si es que **no** se envía al generador la señal de continuación. El test $y \leq x$ se lleva a cabo en un módulo que siempre responde **si** o **no** y opera como sigue: se examina en primer lugar si $y = x$ si son iguales, entonces se retorna **sí**, en otro caso se arranca un generador canónico de Σ^* asociado al orden $\leq$ y se ve que palabra aparece primero, si lo hace `y`, entonces $y < x$ y se retorna **sí**; en otro caso, si primero aparece `x`, entonces $y > x$ y se retorna **no**. En consecuencia la máquina M_μ así definida cumple que $L(M_\mu) = \mu(L,L')$ y, por tanto, $\mu(L,L')$ es un lenguaje recursivamente enumerable.
- Para el caso *2.2* sean A y A' dos máquinas de Turing que se detienen para cada entrada y que además cumplen que L(A) = L y L(A') = L'. Definimos seguidamente una máquina de Turing A_μ que se detiene para cada entrada y tal que $L(A_\mu) = \mu(L,L')$. Esta máquina opera como sigue. Cuando se le suminisra una palabra de entrada `x` ésta se aplica a la máquina A, si ésta rechaza la palabra `x` se rechaza, y si ésta acepta se continúa arrancando un generador canónico de Σ^* con las mismas caracterísiticas que en el caso anterior. Cada vez que genera una palabra `y` se aplica a A', si esta máquina acepta `y` entonces se acepta `x`, si rechaza e $y \neq x$ se envía al generador la señal de continuación; si $y = x$ entonces se termina rechazando `x`.

(2.0 puntos)

4. Se define la operación $P:\{a,b\}^* \longrightarrow \wp(\{a,b\}^*)$ del modo que sigue:

   I. $P(\lambda) = \{\lambda\}$
   II. $P(x) = a^*x$, si $x = by$
   III. $P(x) = b^*x$, si $x = ay$

¿Es la operación `P` de cierre dentro de la clase de los lenguajes recursivos?
Veamos que la operación P es de cierre en la clase de los lenguajes recursivos. Un modo inmediato de hacerlo es observar que $P(L) = (L \cap \{\lambda\}) \cup \{a\}^*(L \cap \{b\}\Sigma^*) \cup \{b\}^*(L \cap \{a\}\Sigma^*)$ y que al ser $\{\lambda\}$, $\{b\}\Sigma^*$, $\{a\}\Sigma^*$, $\{a\}^*$ y $\{b\}^*$ lenguajes recursivos (de hecho son regulares y todos los lenguajes regulares son lenguajes recursivos, ya que cada autómata finito determinista puede simularse, de modo directo, mediante una máquina de Turing cuyo cabezal nunca retrocede y que en consecuencia se detiene para cada entrada), y ser éstos cerrados para la unión, la intersección y la concatenación resulta que P(L) también es recursivo.

(1.0 punto)

(II) PROBLEMAS

5. Desarrolle un módulo *Mathematica*, adecuadamente explicado, que reciba como entrada una gramática incontextual y retorne `True` si existen al menos dos producciones diferentes con el mismo consecuente, retornando `False` en otro caso.
La condición dada puede expresarse formalmente como:
$(\exists A,B \in \text{Auxiliares})(\exists \alpha \in (\text{Auxiliares} \cup \text{Terminales})^*)$
$((A \neq B) \wedge (A \rightarrow \alpha \in \text{Producciones}) \wedge (B \rightarrow \alpha \in \text{Producciones}))$
que, equivalentemente, con gramáticas codificadas puede expresarse como
$(\exists i,j \in \{1,..,\text{Length}[G[[3]]]\})((i \neq j) \wedge (G[[3,i,2]] \cap G[[3,j,2]] \neq \varnothing))$.
Por tanto, algorítmicamente, la condición anterior puede examinarse utilizando dos bucles anidados tal como se hace en el siguiente programa *Mathematica* que implementa lo anteriormente expuesto.

```
P5[G_List] := Module[{producciones, encontrado, n},
  encontrado = False;
  producciones = G[[3]];
  n = Length[producciones];
  For[i = 1, i < n && ! encontrado, i++,
    For[j = i + 1, j ≤ n && ! encontrado, j++,
      encontrado = Intersection[producciones[[i, 2]],
                                producciones[[j, 2]]] ≠ {}
    ]
  ];
  Return[encontrado]
]
```

(2.0 puntos)

6. Dadas las gramáticas
$G1 : S \rightarrow aSb \mid Sba \mid \lambda$ $\qquad$ $G2 : S \rightarrow aSb \mid bbSS \mid a \mid \lambda$
$G3 : S \rightarrow SaS \mid SbS \mid SS \mid \lambda$
y la sustitución $f(a) = (L(G2)^r)^*$, $f(b) = L(G2) \cup L(G3)$. Obténgase una gramática incontextual para el lenguaje $(L(G1) - \{\lambda\})^+ f(L(G1))$.

f(a): $A \rightarrow XA \mid \lambda$
$X \rightarrow bXa \mid XXbb \mid a \mid \lambda$

f(b): $B \rightarrow Y \mid Z$
$Y \rightarrow aYb \mid bbYY \mid a \mid \lambda$
$Z \rightarrow ZaZ \mid ZbZ \mid ZZ \mid \lambda$

L(G1) – {λ}: C → aCb | ab | Cba | ba
(LG1) – {λ})$^+$: D → CD | C

f(L(G1)): E → AEB | EBA | λ

(LG1) – {λ})$^+$ f(L(G1)): S → DE

(1.5 puntos)

7. Dada la gramática G obtenga una gramática G' simplificada y en Forma Normal de Chomsky con L(G') = L(G) – {λ}.

S → AS | SS | B
B → A | bbSa | AS | λ
D → CaC | DDa
A → aAb | aS | bAb
C → aCA | DD | EEa
E → a | λ

Eliminación de los Símbolos Inútiles

Símbolos Productivos: S, A, B, C, D, E
Símbolos Alcanzables: S, A, B, a, b

S → AS | SS | B
A → aAb | aS | bAb
B → A | bbSa | AS | λ

Eliminación de las Producciones Lambda

Símbolos Anulables: S, B

S → AS | A | SS | S | B
A → aAb | aS | a | bAb
B → A | bbSa | bba | AS

Eliminación de las Producciones Unitarias

S → AS | SS | aAb | aS | a | bAb | bbSa | bba
A → aAb | aS | a | bAb
B → aAb | aS | a | bAb | bbSa | bba | AS

Símbolos útiles: S, A, ya que el símbolo B es inalcanzable.

S → AS | SS | aAb | aS | a | bAb | bbSa | bba
A → aAb | aS | a | bAb

Forma Normal de Chomsky

Etapa I

S → AS | SS | XAY | XS | a | YAY | YYSX | YYX
A → XAY | XS | a | YAY
X → a
Y → b

Etapa II

S → AS | SS | XU | XS | a | YU | YV | YZ
A → XU | XS | a | YU
U → AY
V → YW
W → SX
Z → YX
X → a
Y → b

(1.5 puntos)