Examen de Computabilidad y Complejidad
(CMC)
29 de enero de 2007

(I) CUESTIONES: (Justifique formalmente las respuestas)

1. ¿Es el lenguaje $\{x \in \{a,b\}^* \ / \ (x = uvw) \wedge (|u| = |v| = |w|) \wedge (|v|_a = 0) \}$ incontextual?

   Antes de aplicar el Lema de Iteración considérese la siguiente observación: Sean $t = a^j b^k a^m$ y $r = |t|/3$, si se da que $j > r \vee m > r$, entonces t no pertenece a este lenguaje. Para establecer este resultado supóngase que $j > r$, si r no es un número natural, entonces $|t|$ no es múltiplo de 3 y de modo inmediato se tiene que t no pertenece al lenguaje, en otro caso se tendrá que $j = r + p$, $p > 0$, y $t = a^r a^p b^k a^m = uvw$, con $|u| = |v| = |w|$, así el primer símbolo de v será una a y claramente t no pertenece al lenguaje ya que se exige que $|v|_a = 0$. De modo similar puede razonarse cuando $m > r$.

   Supóngase que el lenguaje es incontextual. Sea n la constante del Lema de Iteración. Considérese la palabra $z = a^n b^n a^n$. Claramente z es una palabra del lenguaje y $|z| \geq n$. Demostraremos que el lenguaje no es incontextual estableciendo que no existe ninguna factorización admisible de z en las condiciones del Lema de Iteración.

   Supóngase z factorizado de acuerdo al Lema de Iteración, debido a las restricciones cuantitativas del Lema no pueden iterarse simultáneamente a's del primer bloque y a´s del segundo, en consecuencia al iterar, en cualquier caso posible, con $i = 0$ se obtiene una palabra $s = a^j b^k a^m$ con $j = n \vee m = n$ y $n > |s|/3$ ya que $|s| < |z| = 3n$. Podemos ahora de modo inmediato concluir, en virtud de la observación anterior, que z no pertenece al lenguaje, por lo que no se cumple el Lema de Iteración y el lenguaje dado, en consecuencia, no es incontextual.

   (1.0 punto)

2. Sea L un lenguaje incontextual y sea a un símbolo de su alfabeto. Se define el lenguaje L_a como el lenguaje formado por todas las palabras de L que tienen al menos tres a consecutivas. ¿Es L_a incontextual?

   Sea $\Sigma = \{a_1,...,a_n\}$ el alfabeto de L. Para cualquier $a \in \Sigma$, se tiene que $\Sigma^*_a = (a_1 + ... + a_n)^* aaa (a_1 + ... + a_n)^*$. Así ya que que $L \subseteq \Sigma^*$ también $L_a \subseteq \Sigma^*_a$ y más concretamente $L_a = L \cap \Sigma^*_a$. Puesto que Σ^*_a se ha definido mediante una expresión regular es un lenguaje regular y al intersectarlo con un lenguaje incontextual el resultado es un lenguaje incontextual. Por tanto se concluye que L_a es un lenguaje incontextual.

   (1.0 punto)

3. Sea L un lenguaje, se define $P(L) = \{x \ / \ (\exists u,v)(x = uv \wedge vu \in L)\}$.
   I. Si L es un lenguaje recursivamente enumerable ¿lo es también P(L)?
   II. Si L es un lenguaje recursivo ¿lo es también P(L)?

   $\boxed{\text{I}}$. Si L es un lenguaje recursivamente enumerable, entonces también lo es P(L). Para establecerlo demostraremos que existe un generador de Turing que genera P(L). Puesto que L es recursivamente enumerable existe un generador de

Turing M tal que G(M) = L. A partir de M definiremos un generador M’ tal que G(M’) = P(L). M’ está definido para operar como sigue: utiliza el generador M para generar palabras de L; inicialmente opera hasta generar la primera, luego seguirá operando cuando reciba una señal de generación. Cada vez que M genera una palabra z ésta se aplica a un módulo F que factoriza secuencialmente la misma en todos los pares (u,v) tales que z = uv, generando seguidamente, a su vez, en cada caso, la palabra vu. Cuando F termina este proceso envía a M una señal de generación para que continúe con la tarea de generación de L. Así se tiene que G(M’) = P(L). Para llevar a cabo estas factorizaciones en el módulo F puede utilizarse una cinta con dos sectores: el inferior para z y el superior para llevar mediante una marca, ✓, el control de donde termina u y donde comieza v; por ejemplo, puede tomarse que v comience en la posición marcada. Esta marca se avanza una posición, si es posible, cada vez que se recibe una señal de generación. Puede comenzarse colocándola sobre el primer símbolo de z ($u = \lambda$ y $v = z$) y terminado en la celdilla, en blanco, siguiente al ultimo símbolo de z ($u = z$ y $v = \lambda$). Para el caso $z = \lambda$ se toma directamente como única factorización $u = v = \lambda$.

$\boxed{\text{II}}$. Si L es un lenguaje recursivo, entonces también lo es P(L). Para establecerlo demostraremos que existe una máquina de Turing $M^{\bullet}$ que se detiene para cada entrada y que reconoce a P(L). Puesto que L es recursivo existe una máquina de Turing M° que se detiene para cada entrada con $L(M^{\circ})$ = L. La máquina $M^{\bullet}$ está definida para operar como sigue: cuando se le aplica una entrada z ésta, a su vez, se aplica internamente a un módulo de factorización F’ que, en condiciones similares al F, obtiene un par (u,v) con z = uv cada vez que, con excepción de la primera, se le envía una señal de factorización. Cuando se le suministra z obtiene automáticamente el primer par. Si recibe una señal de factorización y ya ha realizado todas las posible factorizaciones detiene todo el proceso y rechaza la entrada. Cuando genera un par (u,v) proporciona como salida la palabra t = vu que se le aplica seguidamente a la máquina M°, si ésta acepta t, entonces $M^{\bullet}$ acepta z, en otro caso, si rechaza t, se envía al módulo F’ una señal de factorización. Claramente, en estas condiciones, $M^{\bullet}$ se detiene para cada entrada y $L(M^{\bullet}) = P(L)$.

(2.0 puntos)

4. Sea Σ un alfabeto, para $x,y \in \Sigma$ se define la operación $\lozenge(x,y) = \{z \,/\, (\exists u,v,w)(x = uv \wedge y = vw \wedge z = uvw)\}$. Esta operación se extiende a lenguajes $L,L' \subseteq \Sigma^*$ de la manera habitual, esto es, $\lozenge(L,L') = \bigcup_{x\in L,\ y\in L'} \lozenge(x,y)$. Si L y L’ son lenguajes recursivos ¿lo es también $\lozenge(L,L')$?

   Si L y L’ son lenguajes recursivos, entonces también lo es $\lozenge(L,L')$. Para establecerlo demostraremos que existe una máquina de Turing $M^{\blacksquare}$ que se detiene para cada entrada y que reconoce a P(L). Pueso que L y L’ son recursivos existen dos máquinas de Turing M y M’ que se detienen para cada entrada y L(M) = L y L(M’) = L’. La máquina $M^{\blacksquare}$ está definida para operar como sigue: cuando se le aplica una entrada z se la pasa a un módulo de factorización T que la factoriza en ternas (u,v,w) tales que z = uvw cada vez que recibe una señal de factorización, excepto en la primera ocasión que lo hace automáticamente, y proporciona como salida las palabras x = uv e y = vw. Si al recibir la señal de factorización no hay ninguna nueva terna que obtener detiene todo el proceso y rechaza la entrada z. La salida x se le aplica ahora a la máquina M si ésta rechaza se envía una señal de factorización a T, si acepta se le aplica la palabra y a M' si ésta rechaza se envía una señal de factorización a T, en otro caso, si acepta, se acepta la entrada z. Para factorizar, el módulo T (sin entrar en todos los detalles precisos) puede utilizar una cinta con tres sectores: el inferior para z y los otros dos para albergar, en cada uno, una marca, ✓, que permita controlar donde se ubican los segmentos u, v y w. Así, por ejemplo, la marca del sector intermedio puede señalar la celdilla donde termina u y la del sector superior donde comienza w. Cada vez que recibe una señal de factorización mueve una posición, hacia la derecha, la marca del sector superior, salvo si ya ha sobrepasado el final de z en cuyo caso avanza, si es posible, una posición la marca del sector intermedio y sobre la celdilla contigua se coloca la marca del sector superior ($v = \lambda$). Inicialmente la marca intermedia se encuentra en la celdilla en blanco inmediatamente anterior al primer símbolo de z y la marca superior sobre el primer símbolo de z ($u = v = \lambda$ y $w = z$). En la última factorización la marca intermedia se encuentra sobre el último símbolo de z y la superior sobre la celdilla inmediatamente a la derecha ($u = z$ y $v = w = \lambda$). Si $z = \lambda$ sólo produce la factorización $u = v = w = \lambda$. Claramente, en estas condiciones, la máquina $M^{\blacksquare}$ se detiene para cada entrada y $L(M^{\blacksquare}) = \Diamond(L,L')$.

(1.0 punto)

(II) PROBLEMAS

5. Desarrolle un módulo *Mathematica*, adecuadamente explicado, que reciba como entrada una gramática incontextual y retorne True si en la misma existen al menos dos producciones $A \to \alpha$ y $B \to \beta$ tales que $A \neq B$, $A \in \beta$ y $B \in \alpha$, retornando False en otro caso.

   Para la resolución de este ejercicio se desarrollará un módulo *Mathematica* tal como sigue. En primer lugar para cada símbolo auxiliar A se calculará aux(A) que está definido como el conjunto de todos los símbolos auxiliares que aperecen en sus consecuentes. Construyéndose una lista denominada relaciónAuxiliares cuyos elementos son, para cada símbolo auxiliar, listas de la forma {A, aux(A)}. Ahora examinar si la condición dada es cierta se traduce en buscar dos índices i, j tales que:

   - $i \neq j$
   - $1 \leq i, j \leq$ Length[relaciónAuxiliares]
   - relaciónAuxiliare[[i,1]] $\in$ relaciónAuxiliares[[j,2]] $\wedge$
     relaciónAuxiliare[[j,1]] $\in$ relaciónAuxiliares[[i,2]]

   Así el módulo queda como sigue.

```
P5[G_List] := Module[{relaciónAuxiliares = {}, eliminables = {},
                        aux, condición, auxiliares, auxiliares1, antecedente},
  (*Cálculo de la lista relaciónAuxiliares donde se relaciona cada símbolo auxiliar
    con el conjunto formado por los símbolos auxiliares de sus consecuentes.*)
```

```
For[i = 1, i ≤ Length[G[[3]]], i++,
  aux = {};
  For[j = 1, j ≤ Length[G[[3, i, 2]]], j++,
    aux = Union[aux, Intersection[G[[3, i, 2, j]], G[[1]]]]
    ];
   relaciónAuxiliares = Append[relaciónAuxiliares, {G[[3, i, 1]], aux}]
  ];
(*Examen de la condición pedida.*)
condición = False;
For[k = 1, k ≤ Length[relaciónAuxiliares] && ! condición, k++,
  antecedente = relaciónAuxiliares[[k, 1]];
  eliminables = Union[eliminables, {antecedente}];
  auxiliares = Complement[relaciónAuxiliares[[k, 2]], eliminables];
     For[m = 1, m ≤ Length[auxiliares] && ! condición, m++,
      auxiliares1 = Cases[ relaciónAuxiliares, {auxiliares[[m]], _}][[1, 2]];
      condición = MemberQ[auxiliares1, antecedente]
    ]
  ];
 Return[condición]
]
```

(2.0 puntos)

6. Dadas las gramáticas

   $G1 : S \to aSSa \mid bSa \mid \lambda$

   $G2 : S \to aSSb \mid aaS \mid SS \mid b$

   y la sustitución $f(a) = L(G1)L(G2)$ y $f(b) = L(G2)^r$, obténgase una gramática incontextual para el lenguaje $f((L(G1) - \{\lambda\})^+ \cup f(L(G2)))$.

   Para clarificar la resolución de este ejercicio puede definirse la sustitución h de modo que sea idéntica a la f, esto es: $h(a) = f(a)$ y $h(b) = f(b)$. Ahora $f((L(G1) - \{\lambda\})^+ \cup f(L(G2))) = h((L(G1) - \{\lambda\})^+ \cup f(L(G2)))$; con esta última notación resolveremos el ejercicio.

f(a): $S_1 \to S_2S_3$ $S_2 \to aS_2S_2a \mid bS_2a \mid \lambda$ $S_3 \to aS_3S_3b \mid aaS_3 \mid S_3S_3 \mid b$ f(b): $S_4 \to bS_4S_4a \mid S_4aa \mid S_4S_4 \mid b$ f(L(G2): $S_5 \to S_1S_5S_5S_4 \mid S_1S_1S_5 \mid S_5S_5 \mid S_4$

$L(G1) - \{\lambda\}$: $S_6 \to aS_6S_6a \mid aS_6a \mid aa \mid bS_6a \mid ba$ $(L(G1) - \{\lambda\})^+$: $S_7 \to S_6S_7 \mid S_6$

$L(G1) - \{\lambda\})^+ \cup f(L(G2)$: $S \to S_7 \mid S_5$

h(a): $S'_1 \rightarrow S'_2S'_3$ $S'_2 \rightarrow aS'_2S'_2a \mid bS'_2a \mid \lambda$ $S'_3 \rightarrow aS'_3S'_3b \mid aaS'_3 \mid S'_3S'_3 \mid b$ h(b): $S'_4 \rightarrow bS'_4S'_4a \mid S'_4aa \mid S'_4S'_4 \mid b$

Finalmente, una gramática para el lenguaje $h((L(G1) - \{\lambda\})^+ \cup f(L(G2)))$ es la que sigue.

$S \rightarrow S_7 \mid S_5$
$S_7 \rightarrow S_6S_7 \mid S_6$
$S_6 \rightarrow S'_1S_6S_6S'_1 \mid S'_1S_6S'_1 \mid S'_1S'_1 \mid S'_4S_6S'_1 \mid S'_4S'_1$
$S_5 \rightarrow S_1S_5S_5S_4 \mid S_1S_1S_5 \mid S_5S_5 \mid S_4$
$S_1 \rightarrow S_2S_3$
$S_2 \rightarrow S'_1S_2S_2S'_1 \mid S'_4S_2S'_1 \mid \lambda$
$S_3 \rightarrow S'_1S_3S_3S'_4 \mid S'_1S'_1S_3 \mid S_3S_3 \mid S'_4$
$S_4 \rightarrow S'_4S_4S_4S'_1 \mid S_4S'_1S'_1 \mid S_4S_4 \mid S'_4$
$S'_1 \rightarrow S'_2S'_3$
$S'_2 \rightarrow aS'_2S'_2a \mid bS'_2a \mid \lambda$
$S'_3 \rightarrow aS'_3S'_3b \mid aaS'_3 \mid S'_3S'_3 \mid b$
$S'_4 \rightarrow bS'_4S'_4a \mid S'_4aa \mid S'_4S'_4 \mid b$

(1.5 puntos)

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

$S \rightarrow ASA \mid AC \mid SE \mid B$
$A \rightarrow Ba \mid aA \mid \lambda$
$B \rightarrow bSA \mid bD \mid BB \mid \lambda$
$C \rightarrow EDc \mid Fc \mid CC$
$D \rightarrow Dc \mid ccF \mid EF \mid AF \mid a$
$E \rightarrow cEab \mid Ea \mid aCc$
$F \rightarrow ESE \mid EF$

Eliminación de símbolos no generativos:

$S \rightarrow ASA \mid B$
$A \rightarrow \lambda \mid aA \mid Ba$
$B \rightarrow \lambda \mid bD \mid BB \mid bSA$
$D \rightarrow a \mid Dc$

Todos los símbolos son alcanzables.

Eliminación de producciones lambda:

Símbolos anulables: S, A, B.
$S \rightarrow A \mid B \mid S \mid AA \mid AS \mid SA \mid ASA$
$A \rightarrow a \mid aA \mid Ba$
$B \rightarrow b \mid B \mid bA \mid bD \mid bS \mid BB \mid bSA$
$D \rightarrow a \mid Dc$

Eliminación de producciones unitarias:

$S \rightarrow a \mid b \mid aA \mid AA \mid AS \mid bA \mid bD \mid bS \mid Ba \mid BB \mid SA \mid ASA \mid bSA$

$A \rightarrow a \mid aA \mid Ba$
$B \rightarrow b \mid bA \mid bD \mid bS \mid BB \mid bSA$
$D \rightarrow a \quad Dc$

Todos los símbolos son útiles.

Forma Normal de Chomsky:

$S \rightarrow a \mid b \mid AA \mid AS \mid AC_1 \mid BB \mid BC_a \mid SA \mid C_aA \mid C_bA \mid C_bD \mid C_bS \mid C_bC_1$
$A \rightarrow a \mid C_aA \mid BC_a$
$B \rightarrow b \mid BB \mid C_bA \mid C_bD \mid C_bS \mid C_bC_1$
$D \rightarrow a \mid DC_c$
$C_1 \rightarrow SA$
$C_a \rightarrow a$
$C_b \rightarrow b$
$C_c \rightarrow c$

(1.5 puntos)