TEORÍA DE LA COMPUTACIÓN Y VERIFICACIÓN DE PROGRAMAS
Licenciatura en Informática.
Plantel Docente: R. Rosenfeld, J. Irazábal, L. Mendoza, I. Rosenfeld.

Comentario. El examen dura 3 horas, es a libro abierto, y tiene 3 partes con 2 ejercicios cada una (calcular una hora para computabilidad, una hora para complejidad computacional, y una hora para verificación de programas). Resolver ejercicios de las 3 partes, no desarrollar nada ya hecho en clase, y ser <u>breves y claros</u> (p.ej., describir las máquinas de Turing con el nivel de detalle utilizado en clase).

PARTE 1. COMPUTABILIDAD

1. Probar que el conjunto RE es cerrado con respecto a la operación $\cap$ (intersección), es decir que para todo par de lenguajes L_1 y L_2 de RE, se cumple que $L_1 \cap L_2 \in$ RE.

2. Sea L_1 un lenguaje generado con los símbolos del alfabeto Σ_1. Sea f: $\Sigma_1 \longrightarrow \Sigma_2^*$ una función total computable, consistente en asignar a cada símbolo de Σ_1 una cadena de Σ_2^*. Y sea L_2 el lenguaje formado por las cadenas obtenidas de la aplicación de la función f sobre las cadenas de L_1. Es decir que dado $L_1 = \{w_1, w_2, \ldots\}$, el lenguaje L_2 es $\{f(w_1), f(w_2), \ldots\}$, entendiendo que f(w), con $w = w_1 w_2 \ldots w_n$, significa $f(w_1)f(w_2)\ldots f(w_n)$. Por ejemplo, si abc $\in L_1$, f(a) = 1, f(b) = 00, y f(c) = 101, entonces f(abc) = 100101 $\in L_2$. Probar que si $L_1 \in$ R, entonces $L_2 \in$ RE. <u>Ayuda</u>: Construir una MT que reconozca L_2.

PARTE 2. COMPLEJIDAD COMPUTACIONAL

3. El problema K-CLIQUE consiste en determinar si un grafo no orientado incluye un clique de tamaño K, es decir un subgrafo completo de K vértices. El lenguaje que lo representa es K-CLIQUE = {(G, K) | G es un grafo no orientado, K > 0 es un número entero, y G incluye un clique de tamaño K}. Probar que 3-CLIQUE $\in$ P, es decir, se puede determinar en tiempo determinístico polinomial si un grafo no orientado incluye un triángulo.

4. El problema PARTICIÓN consiste en determinar si un conjunto finito de números naturales se puede partir en dos conjuntos C_1 y C_2, tales que la suma de los números de C_1 sea igual a la suma de los números de C_2. El lenguaje que lo representa es PARTICIÓN = $\{\{x_1, \ldots, x_m\} \mid \{x_1, \ldots, x_m\}$ se puede partir en $\{y_1, \ldots, y_{k1}\}$ y $\{z_1, \ldots, z_{k2}\}$, tales que $\Sigma_{i=1,k1}\, y_i = \Sigma_{i=1,k2}\, z_i\}$. Por su parte, el problema MOCHILA consiste en determinar si dado un conjunto finito de números naturales, existe un subconjunto del mismo tal que la suma de sus números sea igual a un determinado valor N. El lenguaje que lo representa es MOCHILA = $\{(\{a_1, \ldots, a_m\}, N) \mid$ existe $\{b_1, \ldots, b_k\} \subseteq \{a_1, \ldots, a_m\}$ tal que $\Sigma_{i=1,k}\, b_i = N\}$. Probar que existe una reducción polinomial de PARTICIÓN a MOCHILA. <u>Ayuda</u>: Notar que si C $\in$ PARTICIÓN y los números de C suman N, entonces tanto los números de C_1 como los de C_2 suman N/2.

PARTE 3. VERIFICACIÓN DE PROGRAMAS

5. La sintaxis de un determinado lenguaje de programación L se define de la siguiente manera. Un programa S pertenece a L si y sólo si S tiene la forma:
   S :: skip | x := e | S_1 ; S_2 | case of $B_1 \longrightarrow S_1$; $B_2 \longrightarrow S_2$; otherwise $\longrightarrow S_3$ end, con:
   e :: n | x | $e_1 + e_2$
   B :: true | false | $e_1 = e_2$ | $\neg$ B, siendo:
   e una expresión entera, *n* una constante entera, x una variable entera, y B una expresión booleana. Sea var(S) la expresión que denota el conjunto de las variables que aparecen en el programa S. Definir por inducción el conjunto var(S).

6. Probar empleando el método de verificación H:
   a. {p} while true do skip od {q}
   b. {p} while ¬q do skip od {q}
