Pauta de Corrección

Certamen Recuperativo
Introducción a la Informática Teórica

30 de julio de 2014

1. Por turno.

   a) Supongamos un lenguaje recursivamente enumerable L, por lo que hay una máquina de Turing M que lo acepta. Podemos construir una máquina de Turing no determinista que haga lo siguiente:

   - Copia un tramo de la entrada (elegido no deterministamente) a un segunda cinta. Si no queda entrada, acepta.
   - Simula M sobre ese tramo de la entrada en la segunda cinta.
   - Si M acepta, borra la segunda cinta y vuelve a comenzar.

   Es claro que esta construcción acepta L^*. Como la clase de lenguajes acpetados por máquinas de Turing deterministas y no deterministas es la misma, resulta lo solicitado.

   b) Suponemos dados L_1, un lenguaje recursivo, y L_2, un lenguaje recursivamente enumerable. Podemos suponer $L_R = \mathscr{L}(M_1)$, y $L_2 = \mathscr{L}(M_2)$, con M_1 y M_2 máquinas de Turing, tales que M_1 siempre se detiene. La figura 1 ilustra la construcción: Se corre M_1 sobre una copia de la entrada, si acepta se inicia el proceso de M_2 sobre

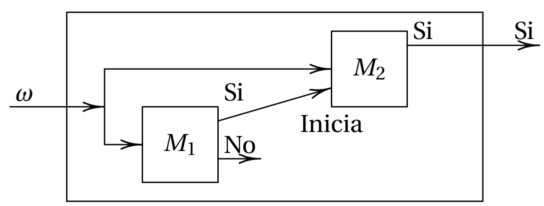

Figura 1: Construcción para intersección entre recursivo y recursivamente enumerable

   otra copia. Si M_1 rechaza, eso simplemente se ignora (la construcción fuerza un ciclo).

   c) Puede usarse una idea similar a la anterior. Si L_1 y L_2 están en P, sabemos que hay máquinas de Turing deterministas M_1 y M_2 que aceptan o rechazan los lenguajes en tiempos acotados respectivamente por polinomios $p_1(|\omega|)$ y $p_2(|\omega|)$. La construcción dada por la figura 2 muestra que el tiempo total está acotado por

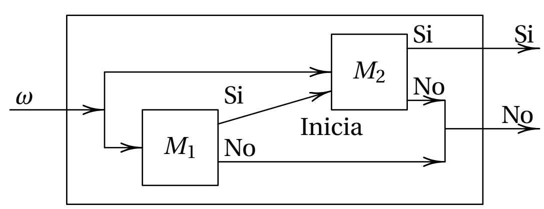

Figura 2: Construcción para intersección de lenguajes en P

   $p_1(|\omega|) + p_2(|\omega|)$, que claramente es un polinomio en $|\omega|$.

Puntajes

Total		25
a) Construcción/explicación	10	
b) Construcción/explicación	7	
c) Construcción/explicación	8	

2. Este lenguaje no es de contexto libre, cosa que demostramos por contradicción. Supongamos que L es de contexto libre. Entonces también es de contexto libre:

$$L' = L \cap \mathscr{L}(a^* b^* c^*) = \{a^{6n} b^{3n} c^{2n} : \, n \geq 0\}$$

ya que la intersección entre un lenguaje de contexto libre y uno regular es de contexto libre. Demostramos que L' no es de contexto libre por contradicción. Supongamos que L' es de contexto libre, y sea N la constante del lema de bombeo de lenguajes de contexto libre. Consideremos:

$$\sigma = a^{6N} b^{3N} c^{2N}$$

Entonces $|\sigma| = 11N \geq N$, por lo que por el lema podemos escribir:

$$\sigma = uvwxy$$

con $|vx| > 0$, $|vwx| \leq N$ tales que para todo $k \in \mathbb{N}_0$:

$$uv^k wx^k y \in L'$$

Es claro que v ni x pueden estar formados por más de un símbolo, ya que de de caso contrario la palabra bombeada no estaría en $\mathscr{L}(a^* b^* c^*)$. Pero al repetir v y x, a lo más aumenta el número de dos de los símbolos, y no se preserva la relación entre los tres. Esto contradice al lema de bombeo, L' no es de contexto libre, y tampoco lo es L.

Como todo lenguaje regular es de contexto libre, L tampoco es regular.

Puntajes

Total		15
Demostrar qye L no es de contexto libre	12	
Como L no es de contexto libre, no es regular	3	

3. El lenguaje es regular, cosa que demostraremos construyendo un NFA que acepta el mismo lenguaje dando su función de transición δ_N. Supongamos la máquina de Turing de una vía $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$. Supongamos que la máquina de Turing de una vía llega a un casillero con símbolo a en el estado q. Si entra en un ciclo sobreescribiendo ese casillero sin avanzar, nunca lo abandona, y $\delta_N(q, a) \supseteq \varnothing$. Si después de un número finito de pasos avanza en el estado p, el contenido final de ese casillero no importa (jamás se vuelve a visitar), así que $\delta_N(q, a) \ni p$. Si una vez alcanzado la secuencia de B después de la entrada, este NFA puede aceptar (vale decir, desde el estado q tenemos que $\delta_N(q, B^n) \in F$ para algún $n \in \mathbb{N}_0$) decimos que $q \in F_N$. Esto, junto con los datos de la máquina de Turing da el NFA $M' = (Q, \Sigma, \delta_N, q_0, F_N)$ que acepta el mismo lenguaje.

Puntajes

Total	15
Llevar a NFA	15

4. Como en las gramáticas sensibles al contexto al aplicar una producción la forma sentencial se alarga o se mantiene del mismo largo, podemos generar sistemáticamente las formas sentenciales alcanzables desde el símbolo de partida hasta hallar la palabra buscada o agotar las de su largo.

Puntajes

Total		10
Explicación/algoritmo informal	10	

Puntajes

5.

Total	10
Explicación/algoritmo informal	10

6. Sabemos que la intersección entre lenguajes de contexto libre y lenguajes regulares es de contexto libre. Podemos construir un PDA que acepta $L_= = \{\#a = \#b\}$ (por ejemplo partiendo de la gramática dada en la pregunta), y con él y el DFA M dado podemos construir un PDA M_I que acepta $\mathscr{L}(M) \cap L_=$. Dado M_I podemos construir una gramática de contexto libre G_I, y determinar si $L_I = \varnothing$ (básicamente, determinando si su símbolo de partida es útil). La respuesta a la pregunta original es si $L_I \neq \varnothing$ (este es un ejemplo de reducción).

Puntajes

Total	20
Construcción y argumento que son algoritmos	20

7. Vamos por turno.

   a) Si $C_a = (x \vee y_1 \vee \ldots \vee y_r)$ y $C_b = (\overline{x} \vee z_1 \vee \ldots \vee z_s)$ aparecen en ϕ, entonces $C_a \wedge C_b$ son satisfechas con x verdadero si ninguno de los y_i es verdadero y con $\overline{x}$ verdadero si ninguno de los z_i es verdadero, que es lo mismo que satisfacer $(y_1 \vee \ldots \vee y_r \vee z_1 \vee \ldots \vee z_s)$. Para incluir la cláusula $()$ estamos resolviendo cláusulas (x) y $(\overline{x})$, que corresponden a la subfórmula $x \wedge \overline{x}$, siempre falsa. Vale decir, esta técnica no puede cambiar la satisfabilidad de ϕ, y es sana.

      Cada vez que se aplica un paso de resolución el número de cláusulas disminuye, por lo que el proceso no puede continuar en forma indefinida. Si no aparece $()$ no hay contradicciones (podemos asignar valores a los literales restantes a gusto), así que es completa.

   b) El problema 2SAT corresponde a la satisfabilidad de fórmulas en las que cada cláusula tiene exactamente dos literales. Un paso de reducción en tal caso disminuye el número de cláusulas, y reemplaza $(x \vee y) \wedge (\overline{x} \vee z)$ por $(y \vee z)$. Si hay n cláusulas, se efectúan a lo más n pasos de resolución, y el procesamiento en cada paso (revisar para cada cláusula si alguna de las restantes comparte un literal negado, reescribir la colección de cláusulas) incluso con algoritmos ingenuos (usar listas y revisar secuencialmente) tomará tiempo proporcional a n, para un total de $O(n^2)$, que es un polinomio. Al haber un algoritmo polinomial, el problame está en P.

Puntajes

Total			30
a)		15	
Demostrar sanidad	7		
Demostrar completitud	8		
b)		15	
Esbozar algoritmo	10		
Argüir que es polinomial	5		

8. Podemos reducir SAT a DOUBLE SAT agregando variables x e y, y las cláusulas $(x \vee \overline{y})$ y $(\overline{x} \vee y)$, que pueden satisfacerse con x e y ambos verdaderos o ambos falsos, por lo que

$$\phi \wedge (x \vee \overline{y}) \wedge (\overline{x} \vee y)$$

puede satisfacerse de dos (o más) maneras si y solo si ϕ puede satisfacerse. Esto demuestra que DOUBLE SAT es NP-duro. Para demostrar que está en NP, basta adivinar dos juegos de valores de verdad, verificar que son diferentes (claramente $O(n)$ si ϕ es de largo n) y evaluar la expresión para ambos es también $O(n)$, lo que da un algoritmo no-determinista polinomial, y está en NP. Como es NP-duro y está en NP, es NP-completo.

Puntajes

Total		25
Explicar reducción, ambos la misma respuesta	10	
Argüir que la reducción es polinomial	3	
Concluir NP-duro	5	
Esbozar solución nodeterminista polinomial	5	
NP-duro y en NPes NP-completo	3	