Pauta de Corrección

Certamen Recuperativo
Introducción a la Informática Teórica

30 de julio de 2014

1. Las conclusiones del lema de bombeo para lenguajes regulares son que hay una constante N tal que si $\sigma \in L$ es de largo $|\sigma| \geq N$, puede escribirse $\sigma = \alpha\beta\gamma$ tal que $|\alpha\beta| \leq N$ con $\beta \neq \epsilon$ tal que para todo $k \in \mathbb{N}_0$ tenemos $\alpha\beta^k\gamma \in L$.

   Las conclusiones del lema de bombeo para lenguajes de contexto libre son que hay una constante N tal que si $\sigma \in L$ es de largo $|\sigma| \geq N$, puede escribirse $\sigma = uvwxy$ tal que $|vwx| \leq N$ con $vx \neq \epsilon$ tal que para todo $k \in \mathbb{N}_0$ tenemos $uv^k wx^k y \in L$.

   Si comparamos ambas conclusiones, vemos que las para lenguajes regulares corresponden al caso especial $\alpha = u$, $\beta = v$, $x = \epsilon$ y $\gamma = wy$. O sea, si se cumplen las conclusiones del primero se aplican las del segundo. Esto es simplemente una consecuncia de que los lenguajes regulares son de contexto libre.

Puntajes

Total		20
Plantear lema de bombeo, lenguajes regulares	5	
Plantear lema de bombeo, lenguajes de contexto libre	5	
Contrastarlas	10	

2. Toda palabra en L podemos escribirla como $\alpha\beta\gamma$, con $\alpha = \epsilon$, $\beta = a$ y γ el resto. Si $\sigma = a^m b^n c^n$, es $\gamma = a^{m-1} b^n c^n$. Podemos repetir β todas las veces que queramos, dando $\alpha\beta^k\gamma = a^{m-1+k} b^n c^n \in L$.

Puntajes

Total	25

3. La demostración es por contradicción. Supongamos que L sea de contexto libre. Entonces es cerrado respecto de homomorfismos, y dado el homomorfismo:

$$\begin{aligned} h(a) &= \epsilon \\ h(b) &= a \\ h(c) &= b \\ h(d) &= c \end{aligned}$$

con lo que $h(L) = \{a^n b^n c^n : n \geq 1\}$, que sabemos no es de contexto libre.

Puntajes

Total	25

4. 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

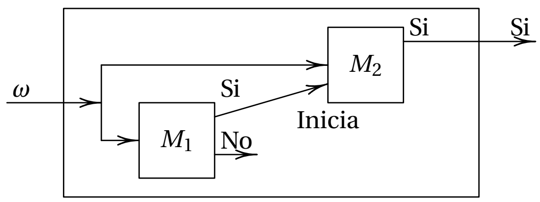

Figura 1: Construcción para intersección entre recursivo y recursivamente enumerable

   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 NP, sabemos que hay máquinas de Turing no 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

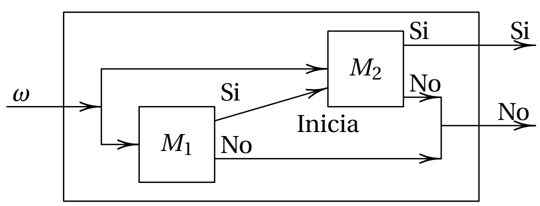

Figura 2: Construcción para intersección de lenguajes en NP

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

Puntajes

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

5. 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

6. 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	