Pauta de Corrección
Segundo Certamen
Introducción a la Informática Teórica
Informática Teórica

26 de septiembre de 2015

1. Primeramente, un *problema* es un lenguaje L, y una *instancia* del problema es determinar si $\omega \in L$.

   a) Un problema está en NP si hay una máquina de Turing no determinista que acepta L en un número de pasos acotado por un polinomio en $|\omega|$. Más operativamente, dada una solución propuesta al problema hay un algoritmo que se ejecuta en tiempo acotado por un polinomio en $|\omega|$ que verifica la solución.

   b) Un problema es NP-duro si hay una reducción polinomial (vea el punto (1*c*) de todos los problemas en NP a él.

   c) Dados problemas P_1 y P_2, decimos que P_1 se reduce polinomialmente a P_2 (se anota $P_1 \leq_p P_2$) si toda instancia de P_1 (vale decir, la pregunta si $\omega \in P_1$) puede traducirse mediante un algoritmo en una instancia ω' de P_2 en tiempo acotado por un polinomio en $|\omega|$. En pedante: Hay una máquina de Turing determinista que siempre se detiene, que toma ω en una cinta y entrega ω' en otra, y que en el proceso da un número de pasos acotado por un polinomio en $|\omega|$.

   d) Un problema se dice *no decidible* si no hay una máquina de Turing determinista que siempre se detiene y que acepta o rechaza. Esto es lo mismo que decir que el lenguaje del caso no es recursivo.

Puntajes

Total	20
a) Construcción/explicación	5
b) Construcción/explicación	5
c) Construcción/explicación	5
d) Construcción/explicación	5

2. Por turno.

   a) Como en gramáticas sensibles al contexto el lado derecho de las producciones nunca es más corto que el lado izquierdo, podemos generar sistemáticamente las formas sentenciales de la gramática, deteniendo la generación al hallar σ o una forma sentencial más larga. Esto revisa un número finito de posibilidades, por lo que el proceso se detiene siempre.

   b) Demostramos en clase que si L es recursivamente enumerable y también lo es $\overline{L}$, L y $\overline{L}$ son recursivos. Sabemos que hay lenguajes recursivamente enumerables que no son recursivos, por ejemplo el lenguaje universal L_u. Con esto $\overline{L}_u$ no es recursivamente enumerable. En consecuencia, no son cerrados respecto de complemento.

   c) Suponemos dados lenguajes recursivos L_1 y L_2. Podemos suponer $L_1 = \mathscr{L}(M_1)$, y $L_2 = \mathscr{L}(M_2)$, con M_1 y M_2 máquinas de Turing que siempre se detienen. Podemos construir una máquina de Turing que siempre se detiene mediante la siguiente estrategia, dada una palabra ω:

   - Nodeterministamente divida $\omega = \omega_1\omega_2$. Esto se hace copiando un trozo de la entrada en una nueva cinta para ω_1, copiando el resto a otra cinta para ω_2.
   - Ejecute M_1 con entrada ω_1. Rechaze si M_1 rechaza.
   - Ejecute M_2 con entrada ω_2. Rechaze si M_2 rechaza, acepte en caso contratio.

   La figura 1 ilustra la construcción. La división nodeterminista siempre termina (o se puede lograr el mismo

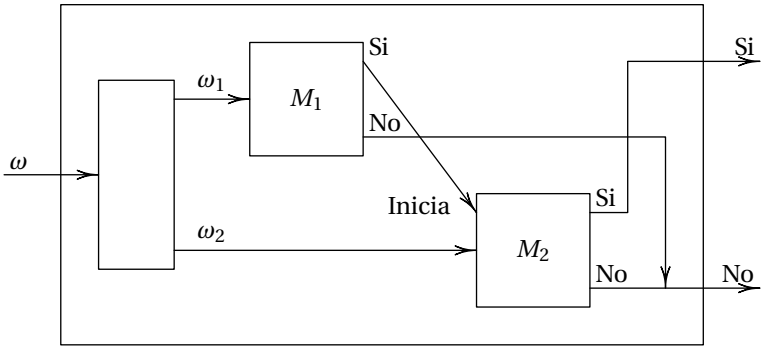

Figura 1: Construcción para concatenación

   efecto probando sistemáticamente todas las divisiones de ω).

Puntajes

Total	25
a) Construcción/explicación	8
b) Construcción/explicación	8
c) Construcción/explicación	9

3. Cada punto en turno.

   a) Un lenguaje L está en P si podemos decidir en tiempo polinomial en el largo de σ si $\sigma \in L$. Pero entonces podemos decidir en tiempo polinomial si $\sigma \notin L$. Concluimos que si $L \in \mathsf{coP}$ entonces $L \in \mathsf{P}$, o sea $\mathsf{P} = \mathsf{coP}$.

   b) Sabemos que $\mathsf{P} \subseteq \mathsf{NP}$. Dado $L \in \mathsf{P}$, si podemos determinar en tiempo polinomial que $\sigma \in L$ podemos determinar en tiempo polinomial que $\sigma \in \overline{L}$, lo que es simplemente un caso particular de determinar en tiempo polinomial en una máquina no determinista. Vale decir, si $L \in \mathsf{P}$ entonces $L \in \mathsf{coNP}$. Uniendo las anteriores, tenemos $L \in \mathsf{NP}$ y $L \in \mathsf{coNP}$, con lo que $P \subseteq \mathsf{NP} \cap \mathsf{coNP}$.

   c) Determinar si la fórmula Φ está en $\mathsf{TAUTOLOGY}$ es lo mismo que determinar si $\overline{\Phi}$ es siempre falsa, que es exactamente el complemento de SAT. Como SAT es NP-completo, está en NP; concluimos que $\mathsf{TAUTOLOGY}$ está en coNP.

Puntajes

Total		30
a)		8
Qué significa $L \in \mathsf{P}$	3	
Corresponde a $\overline{L} \in \mathsf{P}$	3	
Concluir $\mathsf{P} = \mathsf{coP}$	2	
b)		8
Sabemos $\mathsf{P} \subseteq \mathsf{NP}$	2	
Demostrar $L \in \mathsf{P}$ implica $L \in \mathsf{coNP}$	3	
$P \subseteq \mathsf{NP}$ y $P \subseteq \mathsf{coNP}$ es $P \subseteq \mathsf{NP} \cap \mathsf{coNP}$	3	
c)		14
$\Phi \in \mathsf{TAUTOLOGY}$ es equivalente a $\overline{\Phi} \in \overline{\mathsf{SAT}}$	5	
SAT es NP-completo, está en NP	5	
Conclusión por la reducción anterior	4	

4. Veamos primero la pista. Sea el grafo $G = (V, E)$, con conjunto máximo de vértices independientes S. No hay arcos entre vértices de S, y no hay otros vértices que no sean adyacentes a alguno en S. O sea, los vértices en $C = V \setminus S$ están en todos los arcos de G, C es un VERTEX COVER de G.

   Para demostrar que VERTEX COVER es NP-completo debemos demostrar que es NP-duro y que está en NP.

   Primero demostramos que VERTEX COVER está en NP. Dado el grafo $G = (V, E)$ y k, podemos elegir un conjunto de vértices $C \subseteq V$ con $|C| = k$, y luego revisamos los arcos E para verificar que todos incluyan un vértice de C. El tiempo total que esto demanda es lineal en el número de vértices para generar C. Suponiendo que C se almacena como una lista, revisar un arco tomará tiempo $O(|C|)$, y revisar todos los arcos tiempo $O(|E| \cdot |V|)$, que es simplemente $O(|G|^2)$, lo que es polinomial. Concluimos que VERTEX COVER está en NP.

   Para demostrar que VERTEX COVER es NP-completo damos una reducción polinomial de INDEPENDENT SET a VERTEX COVER. Por la pista, determinar si $G = (V, E)$ tiene un conjunto independiente de tamaño k es determinar si G tiene una cobertura de vértices de tamaño $|V| - k$. La traducción consiste simplemente en contar los vértices de G y restarle k, proceso que claramente toma tiempo polinomial en el tamaño de G. Como logramos reducir INDEPENDENT SET $\leq_p$ VERTEX COVER, VERTEX COVER es NP-duro.

   Como VERTEX COVER es NP-duro y está en NP, es NP-completo.

Puntajes

Total		35
Demostrar la pista	7	
Demostrar VERTEX COVER $\in$ NP	8	
Reducir INDEPENDENT SET $\leq_p$ VERTEX COVER	15	
NP-duro y en NP es NP-completo	5	