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

29 de noviembre de 2014

1. 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 que L no es de contexto libre	10
Como L no es de contexto libre, no es regular	5

2. 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

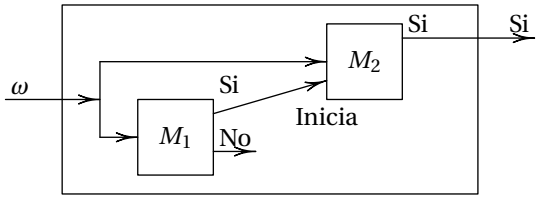

Figura 1: Construcción para intersección entre recursivo y recursivamente enumerable

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

   c) 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.

Puntajes

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

3. Reducimos el problema de correspondencia de Post a determinar si un lenguaje de contexto libre contiene palíndromos.

   De la instancia dada en la pista construimos la gramática:

   $$S \to \alpha_1 S \beta_1 \mid \cdots \mid \alpha_n S \beta_n \mid c$$

   El lenguaje generado contiene palíndromos si y solo si la instancia del problema de correpondencia de Post tiene solución.

   Como sabemos que el problema de correspondencia de Post no es decidible, no es decidible determinar si este lenguaje contiene palíndromos.

Puntajes

Total	15
Reducir PCP a contener palíndromos	6
Construcción de la gramática	6
Conclusión	3

4. 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		

5. La demostración es por contradicción. Si contáramos con un mecanismo para determinar si dos programas tienen la misma salida, bastaría usarlo con un programa cualquiera y el clásico del listado 1 para resolver el problema de si escribe ‘‘`Hola, mundo!`’’. Pero sabemos que esto es imposible.

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        printf("Hola, mundo!\n");

        exit(EXIT_SUCCESS);
}
```

Listado 1: Un clásico

Puntajes

Total	20
Demostración por contradicción	8
Reducir de ‘‘`Hola, mundo!`’’	8
Conclusión	4

6. El problema más cercano es CLIQUE (¿Contiene el grafo G una clique de tamaño k, o sea, tiene un subgrafo isomorfo a K_k?). Podemos reducir CLIQUE a SUBGRAPH ISOMORPHISM de la siguiente forma: Dados el grafo $G = (V, E)$ y k, verificamos si $k \leq |V|$, de no ser así la respuesta es no. Construimos K_k (significa agregar k vértices y $k(k-1)/2$ arcos, esto toma tiempo $O(|V|^2)$, polinomial en el tamaño de los datos de entrada), y entregamos $\langle G, K_k \rangle$ a SUBGRAPH ISOMORPHISM.

   Es claro que la instancia así construida tiene la misma respuesta que el problema original, lo que demuestra que SUBGRAPH ISOMORPHISM es NP-duro.

   Para demostrar que SUBGRAPH ISOMORPHISM está en NP, basta elegir para cada vértice de G' un vértice de G que es la supuesta contraparte, y luego verificar que los arcos de G' conectan los vértices en G. Suponiendo que los grafos se representan mediante matrices de adyacencia, esto toma tiempo lineal en el tamaño de los datos originales.

   Concluimos que SUBGRAPH ISOMORPHISM es NP-duro y está en NP, o sea es NP-completo.

Puntajes

Total	25
Explicar reducción, ambos la misma respuesta	8
Argüir que la reducción es polinomial	4
Concluir NP-duro	3
Esbozar solución nodeterminista polinomial	5
NP-duro y en NP es NP-completo	5