Pauta de Corrección

Segundo Certamen
Introducción a la Informática Teórica

14 de julio de 2012

1. Primero algunas precisiones:

   - Un *problema* consta de un conjunto infinito de instancias, cada una de las cuales tiene respuesta si o no. Los modelamos como lenguajes, donde el string σ codifica una instancia del problema, y $\sigma \in \mathscr{L}$ es que la respuesta es afirmativa.
   - Un problema pertenece a la clase $\mathscr{NP}$ si una máquina de Turing no determinista toma un número de pasos acotado por algún polinomio en el tamaño de la entrada σ para aceptar o rechazar el string.

   a) Un problema P se dice $\mathscr{NP}$-duro si hay una reducción polinomial (ver punto (g)) de todos los problemas en $\mathscr{NP}$ a él.

   Informalmente, poder resolver eficientemente un problema $\mathscr{NP}$-duro significa poder resolver eficientemente todos los problemas en $\mathscr{NP}$, por lo que estos son problemas a lo menos tan difíciles de resolver como los en $\mathscr{NP}$.

   b) Un problema P se dice $\mathscr{NP}$-completo si es $\mathscr{NP}$-duro y pertenece a $\mathscr{NP}$.

   Informalmente, si alguno de estos puede resolverse eficientemente, hay solución eficiente para todos los problemas en $\mathscr{NP}$; son los problemas más difíciles que pertenecen a $\mathscr{NP}$.

   c) Un lenguaje se llama recursivo si es aceptado por una máquina de Turing que siempre se detiene.

   Informalmente, hay un algoritmo que responde la consulta. Puede demorarse muchísimo en dar la respuesta; pero basta tener la paciencia necesaria, la respuesta garantizadamente se entregará en un plazo finito.

   d) Un autómata de pila, o PDA (*Push Down Automaton*) $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ consta de:

   Q**:** Conjunto finito de *estados*

   Σ**:** Alfabeto de entrada

   Γ**:** Alfabeto de pila (stack)

   δ**:** Función de transición, $\delta : Q \times (\{\epsilon\} \cup \Sigma) \times \Gamma \to 2^{Q \times \Gamma^*}$ (los subconjuntos considerados son finitos)

   q_0**:** Estado inicial, $q_0 \in Q$

   Z_0**:** Pila inicial, $Z_0 \in \Gamma$

   F**:** Estados finales, $F \subseteq Q$

   La idea es que si $(p, \gamma) \in \delta(q, x, Z$, si M está en el estado q y en el tope de su pila tiene Z, al leer x de la entrada (si $x = \epsilon$ no hace nada con la entrada, si $x \in \Sigma$ consume ese símbolo) pasa al estado p y reemplaza el símbolo Z del tope de la pila por el string γ. El autómata parte al comienzo del string de entrada en el estado q_0 y con únicamente Z_0 en la pila. Si hay manera de que M consuma toda la entrada (puede llegar a una situación desde la cual no hay movidas posibles) y está en un estado final, se dice que M acepta. Esto define el *lenguaje aceptado por estado final*, $\mathscr{L}(M)$. Alternativamente, podemos decir que M acepta si después de consumir el string completo termina con la pila vacía, independiente del estado en que esté, definiendo el *lenguaje aceptado por pila vacía*, $\mathscr{N}(M)$. De interesar este último caso, el conjunto de estados finales es irrelevante, y convencionalmente se pone $F = \emptyset$.

   Informalmente, es un autómata finito con memoria adicional que se administra como una pila, tomando en cuenta el símbolo en el tope de la pila para definir las transiciones posibles y reemplazando el tope de la pila en cada paso.

e) Un autómata no determinista es uno que puede entrar en una situación desde la cual hay varias movidas legales. Un autómata determinista, en cambio, tiene todas sus opciones predeterminadas.

f) Un lenguaje es regular si puede representarse mediante una expresión regular. Demostramos que esto es equivalente a que sea aceptado por un autómata finito (determinista o no determinista).

g) Dados problemas P y Q, una *reducción* es un algoritmo (formalmente, una máquina de Turing determinista que siempre termina) que traduce cada instancia p de P en una instancia q de Q, de forma que si traduce $p \rightsquigarrow q$ entonces las respuestas a p y q son la misma (verdadero o falso en ambos casos). Una *reducción polinomial* es una reducción que toma tiempo acotado por un polinomio en el tamaño de la entrada.

No se requieren respuestas formales, una descripción informal razonablemente precisa es suficiente.

Puntajes

Total	30
a)	4
b)	4
c)	5
d)	4
e)	4
f)	4
g)	5

2. Por turno:

   a) Este lenguaje no es regular. Supongamos que lo fuera, y sea N la constante del lema de bombeo para lenguajes regulares. elegimos $\sigma = a^N bc^{N+1}$, $|\sigma| = 2N+2 \geq N$. Al dividir $\sigma = \alpha\beta\gamma$ con $|\alpha\beta| \leq N$, vemos que β está formado enteramente por a y al dejarlo fuera se rompe el balance entre a, b y c. Esto contradice al lema de bombeo, y el lenguaje $\mathscr{L}_1$ no es regular.

   O podemos usar el que los lenguajes regulares son cerrados respecto de substitución y homomorfismos inversos para llegar a una contradicción. Si $\mathscr{L}_1$ fuera regular, lo sería el lenguaje que se obtiene al substituir $a \mapsto a$, $b \mapsto a$, $c \mapsto b$ en $\mathscr{L}_1$, que es $\mathscr{L}' = \{a^{2n}b^{2n} : n \geq 1\}$. A su vez, con el homomorfismo definido por $h(a) = aa$ y $h(b) = bb$, vemos que $h^{-1}(\mathscr{L}') = \{a^n b^n : n \geq 1\}$ debiera ser regular, pero ese lenguaje sabemos que no es regular. En consecuencia, tampoco puede serlo $\mathscr{L}_1$.

   Por el otro lado, la gramática de contexto libre siguiente genera $\mathscr{L}_1$:

   $$S \to aSc \mid aAc$$
   $$A \to bAc \mid bc$$

   Cada vez que se usa la primera producción para S se agregan una a y una c, al usar la segunda se asegura que hay al menos una a y una c; al usar la primera producción para A se agrega una b y una c, la segunda asegura que haya al menos una b con la correspondiente c.

   En resumen, $\mathscr{L}_1$ es de contexto libre, pero no regular.

   b) El lenguaje $\mathscr{L}_2$ puede describirse mediante la expresión regular $a^+(bb)^+(ccc)^+$, con lo que es regular. Al ser regular, es de contexto libre.

   c) Vimos en clase que $\mathscr{L}_3$ no es regular. Podemos adaptar el mismo argumento para demostrar que tampoco es de contexto libre: Supongamos que $\mathscr{L}_3$ es de contexto libre, con lo es aplicable el lema de bombeo para lenguajes de contexto libre. Sea N la constante del lema, elegimos $\sigma = a^{N^2} \in \mathscr{L}_3$. El lema de bombeo asegura que podemos dividir $\sigma = uvwxy$ tales que $0 < |vx| \leq N$. de forma que $uv^k wx^k y \in \mathscr{L}_3$ para todo $k \in \mathbb{N}_0$. Pero como σ está formado únicamente por a, basta considerar los largos:

   $$|uv^k wx^k y| = N^2 + (k-1)|vx|$$

   Con esto y las condiciones sobre $|vx|$ del lema tenemos al elegir $k = 2$:

   $$N^2 < |uv^2 wx^2 y| \leq N^2 + N < (N+1)^2$$

   El largo no es un cuadrado perfecto, con lo que $uv^2 wx^2 y \notin \mathscr{L}_3$, una contradicción. En consecuencia, $\mathscr{L}_3$ no es de contexto libre, y tampoco regular (ya sea por lo visto en clase, o porque si fuera regular sería de contexto libre).

Puntajes

Total		20
a)		10
No es regular	5	
Es de contexto libre	5	
b)		3
c)		7

3. Por turno, aunque las partes son independientes:

   a) Informalmente, podemos usar una cinta adicional para representar la pila del PDA; y M' sólo avanza en la cinta de entrada, sobreescribiendo cada símbolo consigo mismo.

   Es $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ el PDA dado, y construimos una máquina de Turing con dos cintas que acepta $\mathscr{L}(M)$. En la primera cinta usa el alfabeto $\Sigma \cup \{B\}$ (el blanco es obligatorio para máquinas de Turing; incidentalmente ayuda a detectar el final del string), en la segunda cinta usa $\Gamma \cup \{B\}$. Describiremos las movidas de M' mediante

   $$\delta' \colon Q' \times (\Sigma \cup \{B\}) \times (\Gamma \cup \{B\}) \to \text{conjuntos de } Q' \times (\Sigma \cup \{B\}) \times \{L, R, N\} \times (\Gamma \cup \{B\}) \times \{L, R, N\}$$

   Dado el estado y lo que ve en las dos cintas (entrada y pila) elige un elemento del conjunto, que determina el nuevo estado, lo a escribir en la cinta de entrada y el movimiento del caso, lo a escribir en la cinta de la pila y su respectivo movimiento. Acá L, R, N representan mover el cabezal respectivo a la izquierda, derecha o no moverlo, respectivamente. Mantenemos el cabezal en la cinta de entrada en el siguiente símbolo a leer, y en la cinta de la pila en el tope.

   Necesitamos inicializar la cinta que representa la pila, para lo que agregamos un nuevo estado inicial, llamémosle p_0, y para todo $x \in \Sigma \cup \{B\}$ hacemos:

   $$\delta'(p_0, x, B) = \{(q_0, x, N, Z_0, N)\}$$

   Como la máquina de Turing sólo escribe un símbolo a la vez, y el PDA sobreescribe el tope de su pila con un string, se requiere simular esta acción. Si la movida de M para $a \in \Sigma$ es $(q_2, \epsilon) \in \delta(q_1, a, Z)$, basta sobreescribir el tope del stack con B y mover a la izquierda, o sea debe ser

   $$(q_2, a, R, B, L) \in \delta'(q_1, a, Z)$$

   Para $(q_2, \epsilon) \in \delta(q_1, \epsilon, Z)$ habrá que mantener quieto el cabezal de la cinta de entrada para cada $x \in \Sigma \cup \{B\}$

   $$(q_2, x, N, B, L) \in \delta(q_1, x, Z)$$

   Si la movida de M es sobreescribir el tope de la pila con un único símbolo, o sea $(q_2, Z_1) \in \delta(q_1, a, Z)$ basta con:

   $$(q_2, a, R, Z_1, N) \in \delta'(q_1, a, Z)$$

   Si no lee de la entrada, o sea para $(q_2, Z_1) \in \delta(q_1, \epsilon, Z)$, con todo $x \in \Sigma \cup \{B\}$ hacemos:

   $$(q_2, x, N, Z_1, N) \in \delta'(q_1, x, Z)$$

   Para movidas que substituyen el tope de la pila por varios símbolos, como $(q_2, Z_1 Z_2 \dots Z_n) \in \delta(q_1, a, Z)$ inventamos nuevos estados que anotaremos

   $$[q_1 Z_1 Z_2 \dots Z_{i-1} \colon Z_i Z_{i+1} \dots Z_n q_2]$$

   para representar que partimos de q_1 y ya agregamos $Z_1 Z_2 \dots Z_{i-1}$, queda por agregar $Z_i Z_{i+1} \dots Z_n$ a la pila, y luego ir al estado q_2. Registramos el origen para evitar confusiones que mezclen partes de movidas diferentes del PDA. Iniciamos el proceso con:

   $$([q_1 Z_1 \colon Z_2 \dots Z_n q_2], a, R, Z_1, R) \in \delta'(q_1, a, Z)$$

   Luego para todo $x \in \Sigma \cup \{B\}$ y $2 \le i \le n$ sobreescribimos el primer espacio de la pila:

   $$\delta'([q_1, Z_1 Z_2 \dots Z_{i-1} \colon Z_i \dots Z_n q_2], x, B) = \{([q_1, Z_1 Z_2 \dots Z_i \colon Z_{i+1} \dots Z_n q_2], x, N, Z_i, R)\}$$

Para todo $x \in \Sigma \cup \{B\}$ rematamos con

$$\delta'([q_1 Z_1 \dots Z_{n-1} : Z_n q_2], x, B) = \{(q_2, x, N, Z_n, N)\}$$

Si la movida a simular es $q_2, Z_1 Z_2 \dots Z_n) \in \delta(q_1, \epsilon, Z)$, la primera movida cambia ligeramente, para todo $x \in \Sigma \cup \{B\}$ debe ser

$$([q_1 Z_1 : Z_2 \dots Z_n q_2], x, N, Z_1, R) \in \delta'(q_1, x, Z)$$

Queda el problema de finalizar el proceso. Detectamos el fin de la entrada cuando M' lee B en su entrada, y en tal caso hacemos que M' vaya a su estado final p_f siempre que M esté en uno de sus estados finales. Vale decir, para $q_f \in F$ y todo $Z \in \Gamma$ definimos:

$$\delta'(q_f, B, Z) = \{(p_f, B, N, Z, N)\}$$

b) Demostramos en clase que todo lenguaje de contexto libre es aceptado por un PDA M. Suponiendo que la construcción solicitada en el punto anterior es posible, tenemos una máquina de Turing que acepta $\mathscr{L}(M)$, con lo que los lenguajes de contexto libre son aceptados por máquinas de Turing y por tanto son recursivamente enumerables.

Otra forma de ver esto es ir generando sistemáticamente todas las derivaciones en la gramática de contexto libre G de forma similar a como simulamos una máquina de Turing no determinista mediante una determinista, copiando la forma sentencial resultante a la cinta de salida si sólo contiene terminales.

Puntajes

Total	20
a) Explicación informal razonablemente completa [1]	10
b)	10

[1] Si hace la construcción formal correcta, el puntaje es:

$$120 - (P_1 + P_{2b} + P_3 + P_5 + P_5) + 5$$

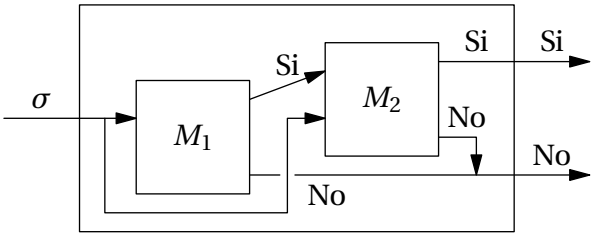

Figura 1: Intersección entre lenguajes recursivos

4. Sabemos que los lenguajes regulares son aceptados por DFA. Dado el DFA $M = (Q, \Sigma, \delta, q_0, F)$ podemos construir una máquina de Turing que acepte $\mathscr{L}(M)$, básicamente sólo avanzando y aceptando si al primer B el estado es final.

   Formalmente, construimos la máquina de Turing determinista $M' = (Q', \Sigma, \Sigma \cup \{B\}, \delta', q_0', B, F')$, donde $Q' = Q \cup \{q_f'\}$ y:

$$\delta'(q, a) = \{(\delta(q, a), a, R)\}$$

$$\delta'(q, B) = \begin{cases} \{(q_f, B, R)\} & \text{si } q \in F \\ \emptyset & \text{caso contrario} \end{cases}$$

   Como sólo avanza hasta hallar B, M' siempre se detiene.

   Esto demuestra que los lenguajes regulares son recursivos, y sabemos que la intersección entre lenguajes recursivos es recursiva, como indica la figura 1. La idea de la construcción acá es que dadas máquinas de Turing que siempre se detienen M_1 y M_2 para los lenguajes $\mathscr{L}_1$ y $\mathscr{L}_2$, respectivamente, construimos una nueva máquina que:

   - Copia la entrada σ a una nueva cinta
   - Simula M_1 sobre la entrada original. Si M_1 no acepta, no acepta. En caso de aceptar M_1, simula M_2 sobre la copia guardada de σ, y responde lo que responda M_2.

   En consecuencia, la intersección entre un lenguaje regular y uno recursivo es recursiva.

Puntajes

Total	20
Lenguajes regulares son recursivos	7
Intersección entre lenguajes recursivos es recursiva	10
Conclusión	3

5. Sabemos de lo visto en clases que si $\mathscr{L}$ es recursivo, entonces $\overline{\mathscr{L}}$ es recursivo. Si $\mathscr{L}$ es recursivamente enumerable, y $\overline{\mathscr{L}}$ es recursivamente enumerable, entonces $\mathscr{L}$ es recursivo (y también lo es $\overline{\mathscr{L}}$). Si $\mathscr{L}$ es recursivamente enumerable pero no recursivo, entonces $\overline{\mathscr{L}}$ no es recursivamente enumerable. Además, todo lenguaje recursivo es recursivamente enumerable.

   En consecuencia:

   a) Si $\mathscr{L}$ es recursivo, demostramos en clase que $\overline{\mathscr{L}}$ es recursivo.

   b) Si $\mathscr{L}$ es recursivamente enumerable, su complemento puede ser recursivo (en cuyo caso $\mathscr{L}$ también es recursivo) o el complemento no es ni siquiera recursivamente enumerable (si $\mathscr{L}$ es recursivamente enumerable pero no es recursivo).

   c) Acá $\overline{\mathscr{L}}$ no puede ser recursivamente enumerable (si lo fuera, $\mathscr{L}$ sería recursivo).

   d) Sabemos que hay casos en que $\overline{\mathscr{L}}$ es recursivamente enumerable pero no recursivo, o puede ser que $\overline{\mathscr{L}}$ tampoco sea recursivamente enumerable.

   No se requieren respuestas formales, una explicación informal razonablemente precisa es suficiente.

Puntajes

Total	30
a)	4
b)	4
c)	5
d)	4