TEORÍA DE LA COMPUTACIÓN Y VERIFICACIÓN DE PROGRAMAS

TP 4 - Misceláneas de Computabilidad y Complejidad (clases 8 y 9)

Ejercicio 1. El Teorema de Rice establece una manera muy sencilla para determinar que un lenguaje de códigos de MT no es recursivo: Sea Π un conjunto no vacío de lenguajes recursivamente numerables más chico que RE. Y sea L = {<M> | L(M) ∈ Π}. Entonces L ∉ R.
a) Justificar (en palabras, intuitivamente) por qué es lógico que el lenguaje L no sea recursivo. Ayuda: ¿cuál es la manera natural de comprobar que L(M) ∈ Π?

Para comprobar de manera natural que L(M) ∈ Π debo verificar que la MT M acepte solo las palabras de algún L ∈ Π, por lo que seria logico decir que L ∉ R ya que una MT que acepta L debería iterar sobre todas las palabras para verificar lo anteriormente dicho.

b) Probar, recurriendo al Teorema de Rice, que L = {<M> | L(M) es finito} ∉ R.

Sea Π el conjunto de los lenguajes finitos podriamos reescribir L = {<M> | L(M) es finito} de la forma L = {<M> | L(M) ∈ Π} y, mediante el Teorema de Rice podríamos decir que L ∉ R.

c) Probar que L = {<M> | M tiene un número par de estados} ∈ R. ¿Por qué esto no contradice el Teorema de Rice?

Debido a que solo puede haber una cantidad finita de estados de M, la MT que acepte el lenguaje L solo deberá recorrer el input M contando los estados y aceptando si la cantidad es par, rechazando de otra forma. A lo sumo hara |Q|+1 pasos, |Q| para recorrer y 1 para aceptar.
No contradice el teorema de Rice debido a que L no especifica el lenguaje de M necesario para aceptar.

Ejercicio 2. Las visiones de reconocedoras y generadoras de lenguajes de las MT son equivalentes en el siguiente sentido: si existe una MT que acepta un lenguaje, entonces existe otra que lo genera, y viceversa.
a) Dada una MT que acepta un lenguaje, explicar (en palabras, informalmente) cómo se puede construir otra que lo genere.

Para hacerlo se puede ir guardando en otra cinta el resultado parcial del procesamiento de w por la MT, de forma que al aceptar el lenguaje se encuentre el resultado final en tal cinta.

b) Dada una MT que genera un lenguaje, explicar (en palabras, informalmente) cómo se puede construir otra que lo reconozca.

Dada una MT que genera un lenguaje, para construir otra que lo reconozca basta con ver si esta maquina género una solución, si lo hizo entonces acepto, en caso contrario rechazo.

Ejercicio 3. Una forma alternativa de definir que una función S(n) es espacio-construible es que existe una MT que a partir de cualquier cadena w trabaja en espacio S(|w|). Probar que si S(n) es espacio-construible, entonces existe una MT que a partir de toda cadena w genera S(|w|) símbolos X.

Si S(n) es espacio construible entonces existe MT que a partir de w trabaja en S(|w|), de esta forma, como la MT trabaja en espacio deterministico S(|w|) entonces, si esta MT escribe un símbolo en cada paso, terminará generando S(|w|) simbolos.

Ejercicio 4. Sea FCLIQUE el problema de búsqueda del clique máximo de un grafo. Probar que existe una Cook-reducción de CLIQUE a FCLIQUE.

Se probara que CLIQUE $\leq^{c}$ FCLIQUE, en otras palabras, existe una MT $M^{FCLIQUE}$ que trabaja en tiempo poly(n) y decide CLIQUE.
Dado un grafo G y un número K, la siguiente MT con oraculo $M^{FCLIQUE}$ hará:

1. Invoco al oráculo FLCLIQUE con G. Si el oráculo responde no, entonces la MT responde no (no existe clique).
2. Si el oráculo responde un clique de G, entonces si la cantidad de vértices del clique es mayor o igual a K responde si.

$M^{FCLIQUE}$ trabaja en poly(n), los pasos 1 y 2 también, el paso 2 tiene O(K).

Ejercicio 5. Sea C(φ) una función que calcula el número de asignaciones de valores de verdad que satisfacen una fórmula booleana sin cuantificadores φ. Probar que si A es una aproximación polinomial de C que cumple: $(\forall x)\,(C(x)/D \leq A(x) \leq C(x).D)$, con $D \geq 1$, entonces se cumple P = NP.

Dado C(φ) una función que calcula el número de asignaciones de valores de verdad que satisfacen una fórmula booleana sin cuantificadores φ, mediante una Cook-reducción $SAT \leq^{c} C$ demostrare que C es tan o más difícil que SAT, por lo tanto que es NP-completo.
Para esto creare la siguiente MT M que dado φ con oráculo M^{c} resuelve SAT en tiempo poly(n):

1. Copia el input φ en la cinta de pregunta P
2. Invoca al oráculo C
3. Acepta sii el oráculo en el paso siguiente devuelve un número mayor a 0.

El paso 1 consume tiempo lineal y los pasos 2 y 3 son de tiempo constante, por lo que M trabaja en tiempo poly(n).
$SAT \in NP-completo \Rightarrow C \in NP-completo$

$(\forall x)\,(C(x)/D \leq A(x) \leq C(x).D) \Rightarrow$
$(\forall x)\,(C(x) \leq A(x).D \leq C(x).D) \Rightarrow$
$(\forall x)\,(C(x) \leq A(x).D/D \leq C(x)) \Rightarrow$
$(\forall x)\,(C(x) \leq A(x) \leq C(x)) \Rightarrow$

$(\forall x)\,(C(x) = A(x))$

Habiendo llegado a esto, dado que $(\forall x)\,(C(x) = A(x))$ como $C \in NP - completo$ y A se resuelve en tiempo poly(n) entonces estaría diciendo que C se resuelve en tiempo poly(n) por medio de A por lo que P = NP.

Ejercicio 6. Probar las inclusiones $P \subseteq RP \subseteq NP$.

Hipótesis:
Dado un lenguaje L, si $L \varepsilon P$ entonces $L \varepsilon RP$.
Demostración:
$L \varepsilon P$ entonces L puede ser resuelto por una MT M en tiempo poly(n).
Construiremos una PT M' tipo RP que en el primer paso haga una selección aleatoria de camino que solo posea un camino y luego simula la MT M. De la siguiente forma si M acepta entonces M' acepta y si M rechaza M' también rechaza. Por lo tanto $L \varepsilon RP$.
Por lo que $P \subseteq RP$.

Hipótesis:
Dado un lenguaje L, si $L \varepsilon RP$ entonces $L \varepsilon NP$.
Demostración:
$L \varepsilon RP$ entonces L puede ser resuelto por una PT M tipo RP.
Construiremos una MTN M' que en el paso de selección aleatoria de caminos tome todos los caminos no deterministicamente. De la siguiente manera si M rechaza en todos sus caminos, entonces M' rechaza también en todos sus caminos y si M acepta, entonces la mitad de sus caminos aceptan, por lo que al menos en un camino de M' se acepta, por lo tanto M' también acepta. Por lo tanto $L \varepsilon NP$.
Por lo que $RP \subseteq NP$.

Ejercicio 7. La clase probabilística PP ya fue definida en clase de la siguiente manera: L ∈ PP sii existe una MTN que: (a) trabaja en tiempo poly(n), y (b) a todo w ∈ L lo acepta en más de la mitad de sus computaciones, mientras que a todo w ∉ L lo rechaza en al menos la mitad de sus computaciones. Probar que $NP \subseteq PP$. Ayuda: Una MTN M acepta w si lo acepta en al menos una computación, y lo rechaza si lo rechaza en todas las computaciones. Pensar cómo construir a partir de M una MTN M' que para los casos en que w ∈ L(M) lo acepte en más de la mitad de sus computaciones, y para los casos en que w ∉ L(M) lo rechace en al menos la mitad de sus computaciones.

Hipótesis:
Dado un lenguaje L, si $L \varepsilon NP$ entonces $L \varepsilon PP$.
Demostración:
$L \varepsilon NP$ entonces L puede ser resuelto por una MTN M en tiempo poly(n).
Construiremos una PT M' tipo PP que en caso de que en algún camino acepte, acepta en ese camino, en caso de que rechace, acepta o rechaza en ese camino con %50 de probabilidad. De la siguiente manera si no se acepto en ningun camino, la mitad de las veces se habrá aceptado y la mitad se habrá rechazado, M rechaza y M' también rechaza,

pero si se aceptó en algun camino, se habrá aceptado en más de la mitad de las veces, M acepta y M' acepta. Por lo tanto $L \varepsilon PP$.
Por lo que $NP \subseteq PP$.