Curso:
Teoría de la Computación.
Unidad 2, Sesión 8: Complejidad computacional (2)

Instituto de Computación, Facultad de Ingeniería
Universidad de la República, Montevideo, Uruguay

dictado semestre 2 - 2009

Nota: **Volviendo a las reducciones.**

Las reducciones polinomiales verifican las siguientes propiedades (demostrables fácilmente a partir de la definición):

Si A se reduce polinomialmente a B (notación $A < B$), y $B < C$, entonces $A < C$.

Si $A < B$ y $B \in P$, entonces $A \in P$. Si $A \notin P$, entonces $B \notin P$.

Si $A < B$ y $B < A$, entonces ambos problemas son "igualmente difíciles".

Nota: **Máquina de Turing universal eficiente.** De la misma manera que vimos que existe una máquina de Turing universal, es posible demostrar (no incluido aquí) que hay una máquina de Turing universal eficiente, en el siguiente sentido:

Propiedad: Existe una Máquina de Turing U tal que, tomando como entrada la descripción $< M >$ de una máquina de Turing y la secuencia x, se comporta como M en la entrada x; es decir, si $M(x)$ acepta entonces $U(< M >, x)$ acepta, si $M(x)$ rechaza entonces $U(< M >, x)$ rechaza, y si $M(x)$ no para entonces $U(< M >, x)$ tampoco para. Además, si $M(x)$ para en t pasos, entonces entonces $U(< M >, x)$ para en $O(| < M > |^{O(1)} \times t)$ pasos.

Nota: **Relación entre máquinas determinísticas y no-determinísticas.**

Propiedad: El problema de, dado como entrada una descripción $< M >$ de una Máquina de Turing no determinística, una secuencia x y un entero $t > n$, decidir si M acepta x en t pasos, es resoluble por una máquina determinística que corre en tiempo $| < M > |^{O(1)} 2^{O(t)}$.

Nota: **Máquina de Turing no determinística universal eficiente.**

Propiedad: Existe una Máquina de Turing NU tal que, tomando como entrada la descripción $< M >$ de una máquina de Turing no determinística y la secuencia x, se comporta como M en la entrada x; es decir, si $M(x)$ tiene una computación que finaliza en la aceptación en t pasos, entonces $NU(< M >, x)$ tiene una computación que finaliza en la aceptación en $O(| < M > |^{O(1)} t)$ pasos; y si no hay computaciones de $M(x)$ que terminen en aceptación, entonces no hay computaciones de $NU(< M >, x)$ que finalicen en aceptación.

Nota: **Como mostrar que un problema es $\mathcal{NP}$ -completo .**

Sea un problema U que pertenece a $\mathcal{NP}$, y candidato a estar en la clase $\mathcal{NP}$ -completo.

Hay esencialmente dos maneras de probarlo: sea demostrar que para cualquier problema L en $\mathcal{NP}$, $L < U$ (por lo tanto, U es al menos tan difícil como cualquier otro en $\mathcal{NP}$; o mostrar un problema L en $\mathcal{NP}$-completo, tal que $L < U$ (por lo tanto, U es al menos tan difícil como L, que es $\mathcal{NP}$-completo).

La segunda manera es la más usual, pero requiere encontrar "un primer problema" en $\mathcal{NP}$-completo,. ♣

Nota: **Un ejemplo de un problema $\mathcal{NP}$ -completo .**

Sea el siguiente problema de decisión, llamado U: la entrada es una 4-upla (M, x, t, l), donde M es una máquina de Turing, $x \in \{0,1\}^*$ es una secuencia de entrada, y t y l son enteros codificados en unario; el problema consiste en determinar si existe un $y \in \{0,1\}^*$ tal que $|y| < l$, y tal que $M(x, y)$ para aceptando la entrada en a lo sumo t pasos.

Es inmediato ver que U pertenece a $\mathcal{NP}$. Se puede definir un procedimiento V_U que dada una entrada (M, x, t, l, y) para aceptando la entrada sí y sólo sí $|y| < l$ y $M(x, y)$ para aceptando en a lo sumo t pasos; como l y t están codificados en unario, la cantidad de pasos es entonces polinómica (en realidad lineal) en el tamaño de la entrada.

Tenemos que ver que cualquier otro problema de a $\mathcal{NP}$ puede reducirse a este. En particular, sea L un problema de decisión $\mathcal{NP}$ cualquiera; entonces (por definición de $\mathcal{NP}$) existe un algoritmo V_L, y polinomios T_L y p_L, tales que $x \in L$ sí y sólo sí existe y, $|y| \leq p_L(|x|)$ tal que $V_L(x, y)$ acep'ta en tiempo máximo $T_L(|x| + |y|)$.

Entonces podemos reducir L a U de esta forma: mapeamos x a la instancia $f(x) = (V_L, x, T_L(|x| + p_L(|x|)), p_L(|x|))$ (este mapeo se puede hacer en tiempo polinomial). Aplicando las definiciones, vemos que $x \in L$ sí y sólo sí $f(x) \in U$, por lo tanto tenemos la reducción.

♣

Nota: **Otro problema $\mathcal{NP}$ -completo - "SAT" .**

El problema de "Conjunctive Normal Form satisfiability" (conocido por "SAT") es el problema prototipo en la clase $\mathcal{NP}$ -completo, y ha sido muy utilizado para reducirlo a otros problemas de manera de demostrar que son $\mathcal{NP}$-completos. El enunciado de "SAT" es el siguiente: dado un conjunto de variables booleanas $x_1, x_2, \ldots, x_n$ y una fórmula booleana ϕ, dada en forma normal conjuntiva, el problema consiste en saber si existe una forma de asignar valores (booleanos) a las variables de manera que la fórmula sea satisfecha (sea verdadera para esos valores).

Una fórmula está en forma normal conjuntiva sí es una secuencia de operaciones AND, cada operación AND involucrando una cláusula (es decir, un conjunto de literales - variables o negaciones de variables - unidos por operaciones OR). Un ejemplo es $(x1ORx2)AND(x1ORNOTx3)AND(NOTx2ORx3)$; esta fórmula es satisfacible por ejemplo por los valores $x_1 = TRUE, x2 = FALSE, x3 = FALSE$ (entre otros).

Uno de los primeros resultados importantes de la teoría de la complejidad es el "Teorema de Cook", demostrando que "SAT" es $\mathcal{NP}$-completo. También se ha demostrado que "3-SAT" (la versión de SAT donde las cláusulas constan a lo sumo de 3 literales) es también $\mathcal{NP}$-completo.

Nota: **Clases de complejidad P y $\mathcal{N}P$ (cont).**

Si un problema de $\mathcal{N}P$ -completo perteneciera a la clase P, entonces cualquier problema en $\mathcal{N}P$ podría ser resuelto eficientemente (y las clases $\mathcal{N}P$ y P coincidirian) . Recíprocamente, si se demostrara que existe al menos un problema en $\mathcal{N}P$ que no está en P, entonces ningún problema de $\mathcal{N}P$-completo podría serlo.

Este problema está abierto, y se conoce por el problema de determinar si $\mathcal{N}P$. Es quizás el problema abierto de la informática teórica de mayor trascencencia. Recientemente ha adquirido también notoriedad en el público en general por haber sido incluído en una lista de siete problemas abiertos de gran trascencencia en las matemáticas para los que el Clay Institute ha establecido un premio de un millón de dólares por su solución (ver http://www.claymath.org/millennium/).

La mayor parte de la comunidad informática sospecha que $P \neq \mathcal{N}P$, pero aún no se han encontrado técnicas suficientemente potentes para demostrarlo.

En general la técnica más empleada hasta el momento para probar la separación de clases es la diagonalización (de manera similar a como se utiliza para demostrar no-computabilidad), pero esta técnica no ha sido útil para el problema precedente.
