Sobre la clase NP y los certificados suscintos

Un lenguaje L que pertenece a NP goza de la propiedad de que cada una de sus cadenas w tiene al menos un *certificado suscinto* z que atestigua su pertenencia a L. Se define que z es un certificado suscinto de w si:

- z es una cadena de tamaño polinomial con respecto al tamaño de w.
- El predicado R(w, z), que expresa que w es una cadena de L (y así que z es un certificado suscinto de w), se decide en tiempo determinístico polinomial.

La propiedad se puede expresar así: L pertenece a NP si y sólo si:

$$L = \{w \mid \exists z: |z| \leq p(|w|) \wedge R(w, z)\}$$

siendo p un polinomio y R un predicado de dos argumentos que se decide en tiempo determinístico polinomial.

Dicho de otro modo: para toda instancia positiva de un problema de NP existe al menos un certificado suscinto, una prueba corta, de que es positiva. Podemos no saber cómo encontrar el certificado eficientemente, pero seguro que existe. Por ejemplo, en el caso del problema del circuito de Hamilton, un grafo con un circuito de Hamilton tiene un certificado suscinto: en este caso es el mismo circuito de Hamilton (es decir, una permutación del conjunto de vértices que cumple la propiedad de constituir un circuito de Hamilton). Otro ejemplo lo constituye el problema de primalidad: todo número primo N tiene un certificado suscinto Y de su primalidad (en este caso, Y es un número, de tamaño polinomial respecto del tamaño de N, que permite verificar "fácilmente" que N es primo -basado en un teorema de Fermat-).

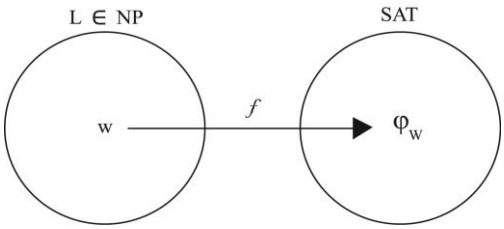
Esta propiedad de la clase NP justifica su riqueza. Muchísimos problemas comparten la propiedad de que sus posibles soluciones son a lo sumo polinomialmente más grandes que sus instancias, y de que se puede decidir eficientemente si una posible solución es efectivamente una solución. Asimismo, el concepto refuerza la conjetura $P \neq NP$: intuitivamente es más fácil probar que una posible solución es efectivamente una solución (lo que se requiere para demostrar la pertenencia a NP), en comparación con construir una solución (lo que se requiere para demostrar la pertenencia a P).

Teorema. El problema SAT es NP-completo.

Sea SAT = {φ | φ es una fórmula booleana satisfactible} el lenguaje que representa el problema SAT. El algoritmo determinístico natural para reconocer SAT, sobre una fórmula de m variables, consiste en probar en el peor caso 2^m asignaciones de valores de verdad, y por lo tanto tarda $O(2^n)$. La prueba de que SAT ∈ NP queda como ejercicio. En lo que sigue demostramos que SAT es NP-difícil, probando que todo lenguaje de NP se reduce polinomialmente a él.

Definición de la función de reducción.

Dado un lenguaje L de NP, tal que M es una MTN que lo reconoce en tiempo polinomial p(n), la idea es transformar toda cadena w en una determinada fórmula booleana φ_w, de modo tal que si alguna computación de M acepta w, entonces existe alguna asignación de valores de verdad que satisface φ_w, y si en cambio todas las computaciones de M rechazan w, entonces no existe ninguna asignación que satisface φ_w. La figura siguiente ilustra esta idea:



La función de reducción f genera una fórmula φ_w que consiste en la conjunción de cuatro subfórmulas, que enseguida describimos. Antes caben las siguientes aclaraciones:

- Una computación de M a partir de w, se representa mediante una cadena $\#\beta_0\#\beta_1\#\beta_2\ldots\#\beta_{p(n)}$, siendo n = |w|, y β_k la configuración k-ésima de la computación. Si β_k es una configuración final, se hace $\beta_k = \ldots = \beta_{p(n)}$. Se asume, sin perder generalidad, que M tiene una sola cinta.
- En cada β_k, el estado corriente, el símbolo corriente y la selección no determinísitica del próximo paso (que es un número natural entre 1 y K, si K es el grado de la relación de transición de M), se agrupan en un solo símbolo compuesto. Así, los símbolos para

representar una computación de M varían en un alfabeto Ψ formado por ternas con un estado de Q_M, un símbolo de $\Gamma_M \cup \{\#\}$ y un número entre 1 y K, o bien con un blanco, un símbolo de $\Gamma_M \cup \{\#\}$ y otro blanco.

- Como M trabaja en tiempo p(n), entonces no se necesitan más que p(n) símbolos para representar cada β_k. Por lo mismo, la representación de una computación tiene p(n) + 1 configuraciones, y por lo tanto $(p(n) + 1)^2$ símbolos incluyendo los símbolos #.
- Por cada uno de los $(p(n) + 1)^2$ símbolos se va a crear en la fórmula una variable booleana c_{ix}, con i variando entre 0 y $(p(n) + 1)^2 - 1$, y el subíndice x variando en el alfabeto Ψ. Que c_{ix} sea verdadera (falsa) va a representar que el i-ésimo símbolo es (no es) x.
- Para simplificar la escritura, las fórmulas $c_{1x} \wedge c_{2x} \wedge \ldots$, y $c_{1x} \vee c_{2x} \vee \ldots$, se abrevian con $\bigwedge_i c_{ix}$ y $\bigvee_i c_{ix}$, respectivamente. El rango de i se establece en cada caso, mientras que x siempre varía en el alfabeto Ψ.

Se define $\varphi_w = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$, tal que:

La subfórmula φ_1 establece que en una misma posición i no pueden haber dos símbolos distintos x e y del alfabeto Ψ: $\varphi_1 = \bigwedge_i [(\bigvee_x c_{ix}) \wedge \neg(\bigvee_{x \neq y} c_{ix} \wedge c_{iy})]$

con $i = 0, \ldots, (p(n) + 1)^2 - 1$.

La subfórmula φ_2 describe la configuración inicial β_0 de M con entrada w, y se expresa a su vez mediante una conjunción de la forma:

$\varphi_2 = \varphi_{21} \wedge \varphi_{22} \wedge \varphi_{23} \wedge \varphi_{24}$, con:

$\varphi_{21} = c_{0,\#} \wedge c_{p(n)+1,\#}$

$\varphi_{22} = c_{1,y1} \vee c_{1,y2} \vee \ldots \vee c_{1,ym}$

donde los y_i son todos los símbolos compuestos de Ψ que representan el estado inicial q_0 de M, el primer símbolo de w, y un número posible de próximo paso de M (variando entre 1 y K).

$\varphi_{23} = c_{2,w2} \wedge c_{3,w3} \wedge \ldots \wedge c_{n,wn}$

$\varphi_{24} = c_{n+1,B} \wedge c_{n+2,B} \wedge \ldots \wedge c_{p(n),B}$

La subfórmula φ_3 establece que la última configuración tiene un símbolo compuesto con el estado final q_A: $\varphi_3 = \bigvee_i (\bigvee_x c_{ix})$

tal que $i = p(n).(p(n) + 1) + 1, \ldots, (p(n) + 1)^2 - 1$, y el símbolo x de Ψ incluye el estado q_A. Por último, la subfórmula φ_4 indica que β_k es una configuración siguiente posible de β_{k-1}, con $k = 1, \ldots, p(n)$, de acuerdo a la selección no determinística del próximo paso especificada en β_{k-1} y la relación de transición de M:

$$\varphi_4 = \bigwedge_i \left(\bigvee_{v,x,y,z} c_{i-p(n)-2,v} \wedge c_{i-p(n)-1,x} \wedge c_{i-p(n),y} \wedge c_{iz}\right)$$

tal que $i = p(n) + 2, \ldots, (p(n) + 1)^2 - 1$, y los símbolos v, x, y, z, de Ψ cumplen el predicado S(v, x, y, z), el cual es verdadero si y sólo si z puede aparecer en la posición i de una configuración, estando v, x, y, en las posiciones i – 1, i, i + 1, de la configuración anterior (se debe tener en cuenta particularmente que dos configuraciones consecutivas β_{k-1} y β_k son iguales cuando β_{k-1} tiene un estado final).

La función f es total y pertenece a FP.

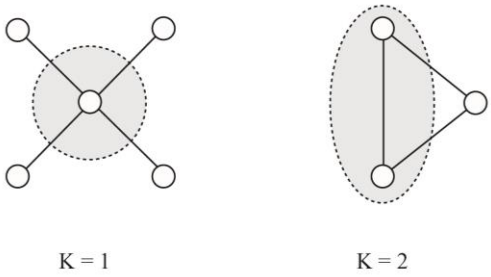
Existe una MTD M_f que a partir de una entrada w (y una MTN M), genera una fórmula booleana φ_w tal como la hemos descripto recién. Además, claramente M_f genera φ_w en tiempo determinístico $O(p(n)^2)$.

Se cumple $w \in L$ si y sólo si $\varphi_w \in SAT$.

a. Si w es una cadena de L, entonces existe una computación C de M que la acepta. Por cómo se construye φ_w, asignando valores de verdad a φ_w consistentemente con C se obtiene una evaluación verdadera de la misma, lo que significa que φ_w pertenece a SAT.
b. Si φ_w es una fórmula de SAT, entonces existe una asignación A de valores de verdad que la satisface. Por cómo se construye φ_w, generando las distintas configuraciones β_i especificadas anteriormente consistentemente con A se obtiene la representación de una computación de M que acepta una cadena w, lo que significa que w pertenece a L.

Teorema. El problema del cubrimiento de vértices es NP-completo

Sea VC = {(G, K) | G es un grafo que tiene un cubrimiento de vértices de tamaño K} el lenguaje que representa el problema del cubrimiento de vértices. Dado un grafo G = (V, E), entonces V' $\subseteq$ V es un cubrimiento de vértices de tamaño K de G, si |V'| = K, e incluye al menos un vértice de todos los arcos de G. La figura siguiente muestra dos cubrimientos de vértices, de tamaños K = 1 y K = 2:



K = 1 K = 2

La prueba de que VC está en NP queda como ejercicio. Para probar que es NP-difícil, definimos una reducción polinomial de 3-SAT a VC.

Definición de la función de reducción.

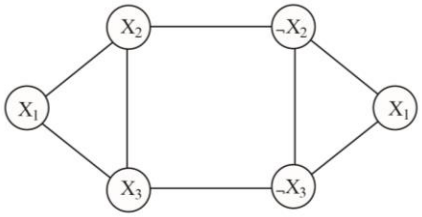
Dada una fórmula booleana φ en la forma normal conjuntiva y con tres literales por cláusula, se define

$$f(\varphi) = (G, 2C)$$

tal que C es la cantidad de cláusulas de φ, y G es un grafo que se construye del siguiente modo:

- Por cada literal de φ se crea un vértice en G.
- Todo par de vértices de G creados a partir de dos literales de una misma cláusula de φ, se unen por un arco. A este enlace lo denominamos de *tipo 1*, como así también a los triángulos resultantes.
- Todo par de vértices de G creados a partir de dos literales x_i y $\neg x_i$ de φ, también se unen por un arco. A este enlace lo denominamos de *tipo 2*.

Por ejemplo, la figura siguiente muestra el grafo que se genera a partir de la fórmula booleana $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$:



Queda como ejercicio probar que la función de reducción f es total y pertenece a FP.

Se cumple $\varphi \in$ 3-SAT si y sólo si $(G, 2C) \in$ VC.

a. Si φ es una fórmula de 3-SAT, y A es una asignación de valores de verdad que la satisface, entonces al menos un literal de toda cláusula es verdadero. Considerando A, el siguiente conjunto de vértices V' es un cubrimiento de tamaño 2C del grafo G construido. Todo vértice asociado a un literal falso se incluye en V', y si luego de esta inclusión hay triángulos de tipo 1 que no tienen dos vértices en V' (caso de cláusulas con dos o tres literales verdaderos), entonces se agregan a V' vértices cualesquiera de dichos triángulos hasta lograrlo. Así se cumple que $|V'| = 2C$. También se cumple que V' cubre G: los enlaces de tipo 1 están cubiertos porque V' tiene dos vértices de cada triángulo de tipo 1, y los enlaces de tipo 2 están cubiertos porque si un literal es verdadero, entonces el literal negado es falso, y así el vértice asociado a este último pertenece a V'.
b. Si (G, 2C) está en VC, y V' es un cubrimiento de 2C vértices del grafo G construido a partir de la fórmula φ, entonces la siguiente asignación de valores de verdad A satisface φ. A los literales asociados a los vértices que no están en V', les asigna el valor verdadero, y al resto les asigna valores consistentes cualesquiera. La asignación A satisface φ porque V' tiene necesariamente dos vértices por triángulo de tipo 1, y así al menos un literal de cada cláusula es verdadero. Por otra parte, A no puede ser inconsistente, no puede suceder que un literal sea verdadero en una cláusula y el literal negado sea verdadero en otra, porque si no, el enlace de tipo 2 asociado a ellos no estaría cubierto por V'.