Teoría de la NP-completitud

Luz Marina Moreno de Antonio y Jorge Riera Ledesma

Departamento de Ingeniería Informática y de Sistemas. Universidad de La Laguna

22 de noviembre de 2016

Capítulo 1

Teorema de Cook

SATISFACTIBILIDAD (SAT)

ENTRADA: Un conjunto de cláusulas $C = \{c_1, c_2, \ldots, c_m\}$ sobre un conjunto finito U de variables.
PREGUNTA: ¿Existe una asignación booleana para U, tal que satisfaga todas las cláusulas de C?

Teorema 1. *SAT es $\mathcal{NP}$-completo.*

Demostración. Es fácil comprobar que SAT $\in \mathcal{NP}$, ya que se puede encontrar una algoritmo para una Máquina de Turing No Determinista (NDTM) que reconozca el lenguaje $L(3\text{SAT}, e)$, para un esquema de codificación e, en un número de pasos acotado por una función polinomial. De esta manera, el primer requerimiento para la prueba de $\mathcal{NP}$-completitud se cumple.

Para el segundo requerimiento debemos bajar a nivel de lenguaje, donde SAT se representa por un lenguaje $L_{SAT} = L[SAT, e]$ para algún esquema de codificación razonable e. Debemos demostrar que para todo lenguaje $L \in \mathcal{NP}$, $L \preceq L_{SAT}$. Los lenguajes de la clase $\mathcal{NP}$ son bastante diversos, se trata de una clase infinitamente grande, por lo que no esperemos presentar una demostración para cada uno de los lenguajes por separado.

No obstante, todo lenguaje de la clase $\mathcal{NP}$ puede describirse de una manera estándar: mediante un programa para una NDTM que lo reconozca en tiempo polinomial. Esto nos permitirá trabajar con un programa genérico para una NDTM y derivar una transformación polinomial genérica desde el lenguaje que reconozca esta máquina genérica al lenguaje L_{SAT}. Esta transformación genérica, cuando se particularice para un programa de una NDTM M que reconozca el lenguaje L_M, dará la deseada transformación desde L_M a L_{SAT}. Así, en esencia, presentaremos una demostración simultánea para todos los lenguajes $L \in \mathcal{NP}$ de que $L \preceq L_{SAT}$.

Para empezar, denotemos por M un programa arbitrario en tiempo polinomial para una NDTM, especificado por Γ, Σ, b, Q, q_0, q_Y, q_N, y δ, que reconoce el lenguaje $L = L_M$. Además, sea $p(n)$ un polinomio que acota superiormente la función $T_M(n)$. Sin pérdida de generalidad se asume que $p(n) \geq n$ para todo $n \in \mathbb{Z}^+$. La transformación genérica f_L será derivada en términos de M, Γ, Σ, b, Q, q_0, q_Y, q_N, δ y p.

Por razones de conveniencia describiremos f_L como si fuera una función entre el conjunto de las cadenas de Σ hasta las entradas del problema SAT, mas que hasta las cadenas del alfabeto inducido por el esquema de codificación asociado a SAT, ya que los detalles de la demostración asociados al esquema de codificación pueden ser deducidos fácilmente.

De esta manera, f_L tendrá la propiedad de que para todo $x \in \Sigma^*$, $x \in L$ si, y sólo si, $f_L(x)$ es satisfecho por una asignación booleana. La clave de las construcción de f_L está en

demostrar cómo un conjunto de cláusulas puede ser usado para comprobar si una entrada x es aceptada por el programa para una NDTM M, es decir, si $x \in L$.

Si la cadena $x \in \Sigma^*$ es aceptada por M, entonces, habrá una secuencia de estados de M que acepte x acotada superiormente por un polinomio $p(n)$, donde $n = |x|$. Dicha secuencia no puede involucrar ninguna celda de la cinta excepto aquellas comprendidas entre $-p(n)$ y $p(n) + 1$ ya que la la cabeza de lectura/escritura comienza en la celda 1, y se mueve una única posición en cada paso. El estado de la computación en cualquier instante puede determinado completamente dando el contenido de las celdas, el estado actual, y la posición de la cabeza de lectura/escritura. Es más, puesto que no se llevarán a cabo más de $p(n)$ pasos durante la ejecución, habrá a los sumo $p(n) + 1$ instantes que deben ser considerados. Esto nos permitirá describir la computación completamente usando sólo un número limitado de variables booleanas y una asignación booleana para las mismas.

A continuación se muestra cómo f_L construye el conjunto de variables U para SAT a partir de M. Se etiquetarán los estados de Q como $q_0, q_1 = q_Y, q_2 = q_N, q_3, \ldots, q_r$, donde $r = |Q| - 1$, y los elementos de Γ como $s_0 = b, s_1, s_2, \ldots, s_v$, donde $v = |\Gamma| - 1$. Se crearán tres tipos de variables, cuyo sentido se especifica en la tabla 1.1. Por "En el instante i" se entiende "después de la terminación del paso i-ésimo".

Tabla 1.1: Variables creadas por f_l

Variable	*Rango*	*Significado*
$Q[i,k]$	$0 \le i \le p(n)$ $0 \le k \le t$	En el instante i, M está en el estado q_k
$H[i,i]$	$0 \le i \le p(n)$ $-p(n) \le j \le p(n)+1$	En el instante i, la cabeza de lectura/escritura está examinando la celda j
$S[i,j,k]$	$0 \le i \le p(n)$ $-p(n) \le j \le p(n)+1$ $0 \le k \le v$	En el instante i, el contenido de la celda j es el símbolo s_k

Una computación de M induce una asignación booleana de estas variables de una manera obvia, bajo la convención de que, si el programa para antes de $p(n)$, la configuración permanece estática durante los instantes posteriores, manteniendo el mismo estado de parada, posición de la cabeza de lectura/escritura, y contenido de la cinta. El contenido de la cinta en el instance 0 consiste en la entrada x, escrita entre las celdas 1 y n, con todas las demás celdas en blanco.

Por otro lado, una asignación booleana arbitraria para esas variables no necesita corresponderse en absoluto con una computación, y mucho menos con una que lleve a un estado de aceptación.

De acuerdo con una asignación booleana arbitraria, una celda de la cinta podría contener distintos símbolos a la vez, y la máquina podría estar simultáneamente en diferentes estados, y la cabeza de lectura escritura podría estar en cualquier subconjunto de las posiciones $-p(n)$ a $p(n) + 1$.

La transformación f_L construye un conjunto de cláusulas que involucran las variables descritas en la tabla 1.1, tal que una asignación booleana de las mismas es satisfactible, si y sólo si, es una asignación booleana inducida por una computación que conduce a un

estado de aceptación para x, a lo sumo en $p(n)$ pasos, y que la cadena producida durante esta computación tenga una longitud de a los sumo $p(n)$ celdas. De esta manera, tenemos los siguiente

$$
\begin{array}{lll}
x \in L & \Longleftrightarrow & \text{existe una computación que lleva a } M \text{ a un estado de aceptación con la entrada } x \\
& \Longleftrightarrow & \text{existe una computación que lleva a } M \text{ a un estado de aceptación con la entrada } x \text{ en a los sumo } p(n) \text{ pasos, produciendo una cadena de salida de longitud } p(n) \\
& \Longleftrightarrow & \text{existe una asignación booleana para el conjunto de cláusulas de } f_L(x).
\end{array}
$$

Esto significa que f_L satisfará una de las dos condiciones requeridas para una transformación polinomial. La otra condición, que f_L sea computable en tiempo polinomial, se verificará una vez hayamos completado la descripción de f_L.

Dividiremos las cláusulas producidas por f_L en seis grupos, cada una de ellas imponiendo un tipo independiente de restricción sobre cualquiera de las asignaciones booleanas satifactibles con se describe en la tabla 1.2.

Tabla 1.2: Cláusulas creadas por f_l, y las restricciones que imponen al satisfacerse una asignación booleana

Grupo	*Restricción*
G_1	En cada instante i, M está en un único estado
G_2	En cada instante i, la cabeza de lectura/escritura lee una única celda de la cinta
G_3	En cada instante i, cada celda de la cinta contiene un único símbolo de Γ
G_4	En el instante 0, la máquina se encuentra en el estado inicial para la entrada x
G_5	En el instante $p(n)$, M ha entrado en el estado q_Y, y x has sido aceptada
G_6	Para cada uno de los instantes i, $0 \leq i \leq p(n)$, la configuración de M en el instante $i+1$ sigue una única transición de la función de transición δ desde la configuración en el estado i.

Es fácil darse cuenta de que si los seis grupos de cláusulas cumplen la misión que tienen asignada, entonces, una asignación booleana de las mismas corresponderá con una computación de aceptación para la cadena x. De esa manera, todo lo que necesitamos demostrar es cómo construir cada uno de los grupos de cláusulas que llevan a cabo esta misión.

El grupo G_1 consta de las siguientes cláusula:

$$\{Q[i,0], Q[i,1], \ldots, Q[i,r]\}\,,\ 0 \leq i \leq p(n)$$

$$\left\{\overline{Q[i,j]}, \overline{Q[i,j']}\right\},\ 0 \leq i \leq p(n), 0 \leq j < j' \leq r$$

Las primera $p(n)+1$ cláusulas pueden ser satisfechas simultáneamente si y sólo sim para cada instante i, M está en al menos un estado. Las restantes $(p(n)+1)(r+1)\frac{r}{2}$ cláusulas pueden se satisfechas simultáneamente si y sólo si, en un instante distinto a i, M está en más de un estado simultáneamente. De esta manera, G_1 lleva a cabo su misión.

Los grupos G_2 y G_3 se construyen de manera similar, y los grupos G_4 y G_5 son bastante simples, cada uno consiste en una única cláusula de un único literal. La tabla 1.3 proporciona una especificación completa sobre los cinco grupos de cláusulas.

Tabla 1.3: Cinco primers grupos de cláusulas creados por f_l

Grupo	*Cáusulas del grupo*
G_1	$\{Q[i,0], Q[i,1], \ldots, Q[i,r]\},\ 0 \leq i \leq p(n)$
	$\left\{\overline{Q[i,j]}, \overline{Q[i,j']}\right\},\ 0 \leq i \leq p(n), 0 \leq j < j' \leq r$
G_2	$\{H[i,-p(n)], H[i,-p(n)+1], \ldots, H[i,p(n)+1]\},\ 0 \leq i \leq p(n)$
	$\left\{\overline{H[i,j]}, \overline{H[i,j']}\right\},\ 0 \leq i \leq p(n), -p(n) \leq j < j' \leq p(n)+1$
G_3	$\{S[i,j,0], S[i,j,1], \ldots, S[i,j,v]\},\ 0 \leq i \leq p(n), -p(n) \leq j \leq p(n)+1$
	$\left\{\overline{S[i,j,k]}, \overline{S[i,j,k']}\right\},\ 0 \leq i \leq -p(n), -p(n) \leq j \leq p(n)+1, 0 \leq k < k' \leq v$
G_4	$\{Q[0,0]\}, \{H[0,1]\}, \{S[0,0,0]\},$
	$\{S[0,1,k_1]\}, \{S[0,1,k_2]\}, \ldots, \{S[0,1,k_n]\},$
	$\{S[0,n+1,0]\}, \{S[0,n+2,0]\}, \ldots, \{S[0,p(n)+1,0]\},$
	donde $x = s_{k_1}, s_{k_2}, \ldots, s_{k_n}$
G_5	$\{Q[p(n),1]\}$

Nótese que el número de cláusulas en estos grupos, y el máximo número de literales que aparece en cada cláusula están acotados por una función polinómica en n (ya que r y v son constantes determinadas por M, y por tanto por L).

El grupo de cláusulas G_6, que garantiza que cada estado sucesivo durante la computación viene de uno previo a partir de un único paso en de M, es un poco más complicado. Consiste en dos subgrupos de cláusulas.

El primer subgrupo garantiza que si la cabeza de lectura escritura no está leyendo la celda j en el instante i, entonces el símbolo dentro de esa celda no cambiará entre el instante i y el instante $i+1$. Las cláusulas de este subgrupo son las siguientes:

$$\left\{\overline{S[i,j,l]}, H[i,j], S[i+1,j,l]\right\},\ 0 \leq i < p(n), -p(n) \leq j \leq p(n)+1, 0 \leq l \leq v.$$

Para todo instante i, celda j, y símbolo s_l, si la cabeza de lectura/escritura no está leyendo la celda j en el instante i, y la celda j contiene s_l en el instante i, pero no en el instante $i+1$, entonces la cláusula anterior, basada en i, j, y l, no será satisfecha, y en caso contrario sí será satisfecha. De esta manera, las $2(p(n)+1)^2(v+1)$ cláusulas de este subgrupo cumplen su misión.

El otro sugrupo que conforma G_6 garantiza que los cambios desde una configuración a la siguiente se hacen de acuerdo con la función de transición δ de M. Por cada cuádrupla (i,j,k,l), $0 \leq i < p(n)$, $-p(n) \leq j \leq p(n)+1$, $0 \leq k \leq r$, y $0 \leq 0 \leq v$, este subgrupo contiene las tres siguientes cláusulas:

$$\left\{\overline{H[i,j]}, \overline{Q[i,k]}, \overline{S[i,j,l]}, H[i+1,j+\Delta]\right\}$$

$$\left\{\overline{H[i,j]}, \overline{Q[i,k]}, \overline{S[i,j,l]}, Q[i+1,k']\right\}$$

$$\left\{\overline{H[i,j]}, \overline{Q[i,k]}, \overline{S[i,j,l]}, S[i+1,j,l'],\right\}$$

donde si $q_k \in Q \setminus \{q_Y, q_N\}$, entonces los valores de Δ, k', y l' serán tales que $\delta(q_k, s_l) = (q_{k'}, s_{l'}, \Delta)$, y si $q_k \in \{q_Y, q_N\}$, entonces $\Delta = 0$, $k' = k$, y $l' = l$.

Ahora debemos demostrar cómo la construcción de las cláusulas G_1–G_6 lleva a cabo la misión especificada. Si $x \in L$, entonces existe una computación conducente a un estado de aceptación en M sobre la cadena x en $p(n)$ pasos o menos, y esta computación, dada la interpretación de las variables, impone una asignación booleana que satisfaga las cláusulas $C = G_1 \cup G_2 \cup G_3 \cup G_4 \cup G_5 \cup G_6$.

En el otro sentido, la construcción de C es tal que cualquier asignación booleana para C debe corresponder a una computación conducente a un estado de aceptación de M para la cadena x. Esto implica que $f_L(x)$ da lugar a una asignación booleana satisfactible, si y sólo si, $x \in L$.

Lo que resta de la demostración es demostrar que para cualquier lenguaje L, se puede construir $f_L(x)$ a partir de x en un número de pasos acotado por una función polinomial de $n = |x|$. Dado un lenguaje L, elijamos una NDTM particular M que reconozca L en tiempo acotado por un polinomio p. Una vez dispongamos de la NDTM específica M y un polinomio específico p, la construcción del conjunto U de variables y la colección C de cláusulas.

□

Siglas

NDTM Máquina de Turing No Determinista. 3, 4, 7

SAT SATISFACTIBILIDAD. 3, 4