Modelos Avanzados de Computación
NP-completitud

Daniel Pozo Escalona
Antonio Coín Castro

13 de junio de 2019

Problema 12 e). Un circuito lógico es un grafo con entradas binarias, puertas AND, OR y NOT y una salida binaria. El problema `CSAT` es dado un circuito lógico, determinar si existe una entrada que hace que la salida sea verdadera. Probar que `CSAT` es NP-completo.

Solución. En primer lugar probamos que `CSAT` está en NP. Para ello, consideramos el algoritmo que consiste en generar de forma no determinista una entrada (una sucesión de valores de verdad), y comprobar si la salida que proporciona el circuito para esa entrada es verdadera. Claramente este algoritmo no determinista es polinómico en función del tamaño de la entrada y resuelve el problema tanto en los casos positivos como en los negativos. Para ver que de hecho `CSAT` es NP-Completo, reducimos `3-SAT` a él.

3-SAT $\propto$ CSAT

Dada una instancia de `3-SAT` formada por un conjunto $U = \{x_0, \ldots, x_{n-1}\}$ de variables junto con un conjunto C de m cláusulas, cada una consistente en una disyunción de exactamente tres literales en las variables de U, construimos por cada cláusula una puerta OR cuyas entradas son las variables involucradas, eventualmente negadas mediante una puerta NOT en caso de que aparecieran con el símbolo $\neg$ delante en la fórmula. Las salidas de todas las puertas OR van a parar a una puerta AND final que las combina todas, y cuya salida es la salida del circuito. Notemos que si queremos obligar a que cada puerta lógica tenga únicamente dos entradas podemos emplear dos puertas OR para cada cláusula y $m-1$ puertas AND al final, encadenándolas correctamente.

Es decir, si denotamos indistintamente x_i^* a x_i ó $\neg x_i$, y $C = \{c_1, \ldots, c_m\}$ con $c_i = x_{i1}^* \vee x_{i2}^* \vee x_{i3}^*$, la satisfabilidad de C equivale a la de

$$\bigwedge_{i=1}^{m} (x_{i1}^* \vee x_{i2}^* \vee x_{i3}^*),$$

y de esta forma el circuito lógico resultante de la reducción se puede escribir (en notación prefija) como:

$$\text{AND}(\text{OR}(X_{11}, X_{12}, X_{13}), \ldots, \text{OR}(X_{m1}, X_{m2}, X_{m3})),$$

donde $X_{ih} = x_i$ ó $X_{ih} = \text{NOT}(x_i)$ dependiendo de si era $x_{ih}^* = x_{ih}$ ó $x_{ih}^* = \neg x_{ih}$.

La equivalencia entre ambos problemas es clara: si hay una asignación de valores de verdad que hace verdaderas todas las cláusulas de la instancia de `3-SAT`, esa misma asignación hará que la salida del circuito construido sea verdadera, y viceversa.

Además, esta reducción se realiza en espacio logarítmico, pues consiste únicamente en ir leyendo la entrada e ir escribiendo en la salida, sustituyendo los símbolos tal y como se ha explicado. Si suponemos, por ejemplo, que la entrada al problema `3-SAT` viene dada en la forma:

$$n_0 c n_1 c n_2 c \dots c n_{3m-3} c n_{3m-2} c n_{3m-1}$$

con n_i un número entre 0 y $n-1$ en binario, más un bit de negación delante, y entendiendo que el conjunto de cláusulas es $C = \{x^*_{n_{3i}} \vee x^*_{n_{3i+1}} \vee x^*_{n_{3i+2}} : i = 0, \dots, m-1\}$, y que la entrada a `CSAT` que calcula la reducción es:

$$\underbrace{n_0 c \dots c n_{3m-1}}_{\text{nodos}} \ X \ \underbrace{n_0 e \texttt{or}_1 c n_1 e \texttt{or}_1 c n_2 e \texttt{or}_1 c \ \dots \ c n_{3m-1} e \texttt{or}_m}_{\text{disyunciones entre los nodos}} \ X \ \underbrace{\texttt{or}_1 e \texttt{and} c \dots \texttt{or}_m e \texttt{and}}_{\text{conjunciones}}$$

Notamos que los símbolos c, e y X son separadores. Para calcular esta entrada a partir de la primera solo son necesarios un número constante de contadores para almacenar, a lo más, m, y que por tanto solo requieren de espacio logarítmico en la entrada.

Ejemplo

Sea $U = \{x_0, x_1, x_2, x_3\}$ y $C = \{x_0 \vee \neg x_1 \vee x_2, \neg x_0 \vee x_1 \vee x_3\}$. La entrada codificada para la reducción sería

$$000c101c010c100c001c011$$

y por tanto la salida calculada que representa el circuito lógico asociado es

$$000c101c010c100c001c011 \ X$$

$$000e\texttt{or}_1 c101e\texttt{or}_1 c010e\texttt{or}_1 c100e\texttt{or}_2 c001e\texttt{or}_2 c011e\texttt{or}_2 \ X$$

$$\texttt{or}_1 e\texttt{and} c\texttt{or}_2 e\texttt{and}$$

El circuito lógico que pintaríamos sería el siguiente: