MEMORIA

EJERCICIO 24

CHAPTER 7. SIPSER

Algoritmia y Complejidad

Rocío Sánchez Cerván

3ºA Ingeniería Informática, curso 2023/24

Ejercicio 7.24. A 2cnf-formula is an AND of clauses, where each clause is an OR of at most two literals. Let $2SAT = \{ < \varphi > \mid \varphi \ is\ a\ satisfiable\ 2cnf - formula \}$. Show that $2SAT \in P$.

Por definición de P, se dice que un lenguaje L pertenece a P si existe una máquina de Turing determinista M que decide L en tiempo polinómico.

- **Construcción de M:**

Sean m el número de variables y n el número de cláusulas de φ
Sea $V = \{ x_1, x_2, \dots, x_m \}$ el conjunto de variables.

Por ser φ una fórmula booleana en forma normal conjuntiva con al menos dos literales por cláusula, φ es de la forma:

$$\varphi = c_1 \wedge c_2 \wedge \dots c_n = (a_1 \vee b_1) \wedge (a_2 \vee b_2) \wedge \dots \wedge (a_n \vee b_n)$$

Siendo c_i la cláusula i de φ y a_i, b_i variables, con $i \in \{1..n\}$ y $a_i, b_i \in \{ x_1, x_2, \dots, x_m, \overline{x}_1, \overline{x}_2, \dots, \overline{x}_m \}$. En el caso de que c_i tenga solo un literal, consideramos $a_i = b_i$

Vamos a construir un grafo dirigido, G, a partir de φ que tendrá 2m nodos:
$x_1, x_2, \dots, x_m, \overline{x}_1, \overline{x}_2, \dots, \overline{x}_m$

Sabemos que $a_i \vee b_i \equiv \overline{a}_i \rightarrow b_i \equiv \overline{b}_i \rightarrow a_i$, entonces:
$\forall\, c_i = (a_i \vee b_i) \in V$ añadimos a G un arco $\overline{a}_i \rightarrow b_i$ y otro arco $\overline{b}_i \rightarrow a_i$
Por definición del AND lógico, φ es satisfacible si existe una asignación de variables tal que cada cláusula de φ es satisfacible.

Llegamos a una contradicción y, por tanto, llegamos a que φ es insatisfacible si obtenemos $\overline{x}_i \rightarrow x_i$ y $x_i \rightarrow \overline{x}_i$ para algún $x_i \in V$. Por la construcción de G y la transitividad de la implicación, esto ocurre si encontramos un x_i tal que G tiene un ciclo que contenga x_i y $\overline{x}_i$

De esta forma, si $\exists\, x_i \in V \mid G\ tiene\ un\ ciclo\ que\ contiene\ x_i\ y\ \overline{x}_i \Rightarrow \varphi \notin 2SAT$
Esto es equivalente a decir:
$\varphi \in 2SAT \Rightarrow \neg \exists\, x_i \in V \mid G\ tiene\ un\ ciclo\ que\ contiene\ x_i\ y\ \overline{x}_i$

Como G es un grafo dirigido, el problema de encontrar un ciclo que contenga los literales correspondientes se reduce a determinar si estos literales pertenecen a la misma componente fuertemente conexa (SCC, por sus siglas en inglés). Una SCC es un subconjunto maximal de vértices de un grafo tales que para cada par de vértices A, B del subconjunto existe un camino dirigido entre ambos (un camino de A a B y otro desde B hasta A).
Para determinar las componentes fuertemente conexas de un grafo dirigido existen diversos algoritmos, como el algoritmo de Tarjan o el algoritmo de Kosaraju.

Usaremos el algoritmo de Kosaraju, que tiene complejidad temporal O(|Ve| + |E|) , siendo Ve el conjunto de vértices y E el conjunto de arcos del grafo G.

Con este algoritmo encontramos todas las SCC maximales de G.

Algoritmo de Kosaraju:

Paso 1. Búsqueda en Profundidad (DFS):
Realizamos un recorrido primero en profundidad (DFS) en G. Los vértices visitados los vamos añadiendo en orden de visita a una pila (inicialmente vacía), que usaremos en el paso 2.

Paso 2. Búsqueda en Profundidad en el grafo transpuesto:
Realizamos un recorrido DFS en el grafo transpuesto de G empezando, sucesivamente, por cada vértice de la pila anterior. Cada vértice visitado lo añadimos a una lista de nodos visitados y a una lista de posibles vértices pertenecientes a la misma SCC (en adelante, listaSCC). Nos encontramos con dos posibilidades:

- Si en este recorrido llegamos de nuevo al vértice del tope de la pila, todos los vértices de listaSCC pertenecen a la misma SCC.
- Si no llegamos al vértice del tope, este vértice forma por sí mismo una SCC.

En ambos casos, desapilamos el tope de la pila y seguimos haciendo DFS empezando en el siguiente vértice de la pila.
Para extraer vértices del tope de la pila tenemos en cuenta que los nodos visitados y los que forman parte de una SCC ya encontrada son descartados. Con esta búsqueda identificamos todas las SCC del grafo.

- **Descripción de M:**

M, mTD multicinta (6 cintas).
Entrada: codificación de φ, < φ > en la cinta c1.

Paso 1: recorremos la cinta c1 y generamos en la cinta c2 los arcos de G (separados por ";"). Para cada cláusula $(a \vee b)$ de φ escribimos las implicaciones $\overline{a} \rightarrow b$ y $\overline{b} \rightarrow a$, representadas en la cinta como: $(\neg a,\ b);\ (\neg b,\ a);$

Paso 2: escribimos en c3 el conjunto de variables, V, y sus negaciones.

A continuación, aplicamos el algoritmo de Kosaraju.
Paso 3: empezando por el primer vértice de la cinta c3, aplicamos el algoritmo de Kosaraju tal y como lo hemos descrito anteriormente. Para trabajar con el grafo transpuesto, simplemente leeremos los arcos de G de derecha a izquierda desde la cinta c2. Escribiremos la pila en la cinta c4 y los nodos visitados los almacenaremos en la cinta c5. En la propia cinta c5 marcaremos los nodos de listaSCC (ampliando el alfabeto). Con cada SCC encontrada, desmarcamos estos nodos (dejando de nuevo sin marcar todos los nodos visitados hasta el momento). Cada SCC encontrada se escribirá en la cinta c6. Las distintas SCC se separan entre sí con ";" y dentro de una misma SCC los vértices irán separados por comas (",").

Paso 4: finalizado el algoritmo de Kosaraju, recorremos la cinta c6 comprobando si en una misma SCC se encuentra una variable y su negación. En este punto hay dos opciones:

1. Que sí haya dentro de una misma SCC una variable y su negación, en cuyo caso la fórmula φ inicial no es satisfacible:
   $M(<\varphi>) = q_{reject} \Rightarrow \varphi \notin 2SAT$
2. Que no encontremos de una misma SCC una variable y su negación, en cuyo caso la fórmula φ inicial sí es satisfacible:
   $M(<\varphi>) = q_{accept} \Rightarrow \varphi \in 2SAT$

- **Complejidad temporal de M:**

Vamos a calcular la complejidad temporal de M en función de n, número de cláusulas de φ.

Paso 1: $n\ cláusulas\ *\ 2\ implicaciones/cláusula$: $O(2n)$
Paso 2: $habrá\ a\ lo\ sumo\ 2n\ variables\ *\ 2\ nodos/variable\ = 4n\ nodos$: $O(4n)$
Paso 3: $4n\ (primer\ DFS)\ +\ 4n\ (segundo\ DFS)$: $O(8n)$
Paso 4: $O(n^2)$, para cada variable comprobamos si su negación está en el mismo SCC

Entonces, M decide 2SAT en tiempo $O(n^2)$ (polinómico), por lo que hemos demostrado que $2SAT \in P$.