Juan Carlos Llamas Núñez 3º DG Mat-Inf

Ejercicio 18.- El problema SAT-k-CNF es el problema de satisfactibilidad de una fórmula lógica del cálculo de proposiciones expresada en forma normal conjuntiva, tal que cada cláusula tiene a lo sumo k literales. Suponiendo demostrado que el problema SAT-4-CNF es NP-completo, demostrar que el problema SAT-3-CNF también lo es.

Para demostrar que SAT-3-CNF es NP-completo hay que comprobar

1) SAT-3-CNF está en NP

2) para algún problema B que es NP-completo, entonces $B \leq^P$ SAT-3-CNF, es decir, B es reducible polinómicamente a SAT-3-CNF.

1) Comenzamos viendo que SAT-3-CNF es NP, es decir, es un problema de decisión que puede ser resuelto por algoritmos no deterministas en tiempo polinómico. Necesitamos por tanto encontrar un algoritmo de verificación en tiempo polinómico que, dada una instancia del problema SAT-3-CNF, es decir, una fórmula lógica en 3-CNF, y una asignación a las variables, verifique si la expresión es cierta para esa asignación. Damos el siguiente algoritmo es pseudo-código:

```
fun verificar(E, asig) dev res:bool
    for each Clau in E
        if (Clau[1].neg ⊗ asig[Clau[1].var]) v ... v (Clau[3].neg ⊗ asig[Clau[3].var])
            continue;
        else
            res = false, return;
    fin for;
    res = true;
fin fun
```

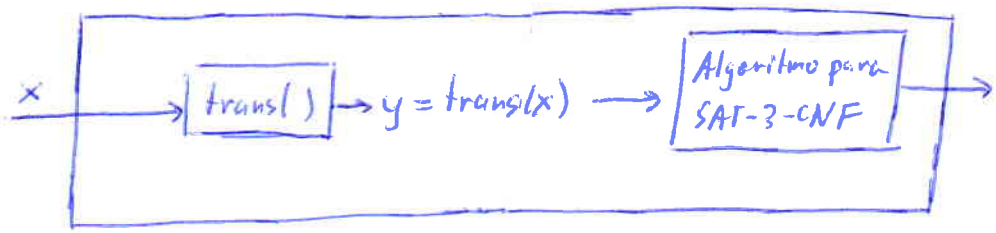
Si alguno de los literales de la cláusula para la asignación da true...

Por tanto, al ser este un algoritmo de tiempo polinómico se tiene que SAT-3-CNF es NP.

2) Según como está planteado el ejercicio parece razonable pensar que B = SAT-4-CNF y tenemos que probar que

$$\text{SAT-4-CNF} \leq^P \text{SAT-3-CNF}$$

Esto quiere decir que tenemos que construir un algoritmo trans(·) que para cada instancia "x" de SAT-4-CNF produzca una instancia "y" del problema SAT-3-CNF en tiempo polinómico y que cumpla que un algoritmo para SAT-3-CNF responde "sí" para "y" sí y solo sí la respuesta al problema SAT-4-CNF para "x" es "sí".



Algoritmo para SAT-4-CNF

La idea intuitiva es que en x cada cláusula tiene un máximo de 4 literales y debemos hacer los cambios pertinentes para que cada cláusula tenga únicamente tres (posiblemente añadiendo más cláusulas).

Sea entonces una cláusula $C_j = (x_1^j \vee x_2^j \vee x_3^j \vee x_4^j)$ con $j \in \{1, \ldots, m\}$ y donde x_i^j son literales para $i = 1,2,3,4$. (Si la cláusula tiene tres literales o menos la dejamos como está).

Realizamos el cambio de variable

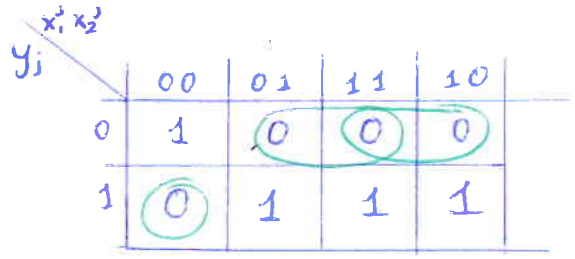
$$y_j = x_1^j \vee x_2^j$$ de tal forma que la nueva cláusula

$$C_j^1 = (y_j \vee x_1^j \vee x_2^j)$$ tiene solamente 3 literales.

Todavía no hemos terminado porque tenemos que dejar reflejado el cambio de variable en la nueva expresión.

Es claro que $C_j \equiv C_j^1 \wedge (y_j = x_1^j \vee x_2^j)$ luego basta expresar la relación $y_j = x_1^j \vee x_2^j$ como una CNF. Lo podemos hacer de manera sencilla con una tabla de verdad y mapas de Karnaugh.

y_j	x_1^j	x_2^j	$y_j = x_1^j \vee x_2^j$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



$$(y_j = x_1^j \vee x_2^j) \equiv (\overline{y_j} \vee x_1^j \vee x_2^j) \wedge (y_j \vee \overline{x_1^j}) \wedge (y_j \vee \overline{x_2^j})$$

Por tanto,

$$(x_1^j \vee x_2^j \vee x_3^j \vee x_4^j) \equiv (y^j \vee x_3^j \vee x_4^j) \wedge (\overline{y^j} \vee x_1^j \vee x_2^j) \wedge (y_j \vee \overline{x_1^j}) \wedge (y_j \vee \overline{x_2^j})$$

$$C_j \qquad C_j^1 \qquad C_j^2 \qquad C_j^3 \qquad C_j^4$$

Esto quiere decir que la cláusula C_j es lógicamente equivalente a la conjunción de las clausula C_j^1, C_j^2, C_j^3 y C_j^4, todas ellas con 3 o menos literales.

Por tanto, dada una expresión en 4-CNF podemos recorrer las distintas cláusulas y transformalas con la relación que aparece arriba para convertirlas en 3-CNF. Nótese que la transformación es lineal en relación al tamaño del problema y que $y^j \neq y^i \; \forall i \neq j$. Finalmente, es conveniente comentar

que se da la equivalencia entre la satisfactibilidad de la fórmula x en 4-CNF y la de trans(x) en 3-CNF.

Esto es que, si tenemos una instancia x del problema SAT-4-CNF para la cual el algoritmo responde "sí", entonces la instancia y = trans(x) del problema SAT-3-CNF cumple que al ejecutar el algoritmo para el SAT-3-CNF este responde "sí", Esto es así porque si existe una asignación para las variables que hagan cierta x basta dar la asignación para y que es igual para todas las variables que se conservan y para $y_j = x_i \vee x_k$. $\forall j$.

En sentido contrario, basta asignarle a las variables de x los valores que toman en y.

Esto prueba que SAT-4-CNF $\leq^P$ SAT-3-CNF y podemos concluir que si SAT-4-CNF es NP-completo entonce también lo es SAT-3-CNF.