



Pontificia Universidad Católica de Chile  
Escuela de Ingeniería  
Departamento de Ciencias de la Computación  
Matías Fernández - matias.fernandez@uc.cl

## IIC2213 - Lógica para ciencia de la computación

### Ayudantía 6 - Viernes 28 de Abril del 2023

**Problema 1.** Demuestre o dé un contraejemplo para las siguientes afirmaciones

- a) Si  $L_1, L_2 \in \text{NP}$  entonces  $L_1 \cap L_2 \in \text{NP}$  y  $L_1 \cup L_2 \in \text{NP}$
- b) Si  $L_1$  y  $L_2$  son NP-completos entonces  $L_1 \cap L_2 \in \text{NP-completo}$
- c) Si  $L_1$  y  $L_2$  son NP-completos entonces  $L_1 \cup L_2 \in \text{NP-completo}$

**Solución:**

- a) Es verdadero. Si  $L_1$  y  $L_2$  están en NP entonces por definición (alternativa) tenemos que existen  $M_1$  y  $M_2$  máquinas de turing en P tales que

$$w \in L_1 \Leftrightarrow \exists w_1 : M_1 \text{ acepta con input } (w, w_1)$$

$$w \in L_2 \Leftrightarrow \exists w_2 : M_2 \text{ acepta con input } (w, w_2)$$

Construiremos una máquina que decide a  $w \in L_1 \cap L_2$  dado un testigo  $w'$ .

$M' :=$  recibe de input  $w$  y  $w'$ . Parsea que  $w'$  sea el par  $(w_1, w_2)$  donde  $w_1$  es el testigo para  $w \in L_1$  y  $w_2$  es el testigo para  $w \in L_2$ . Luego, simulamos  $M_1$  con input  $(w, w_1)$  y  $M_2$  con input  $(w, w_2)$ . Sólo si ambas máquinas aceptan aceptamos, si no rechazamos.

Si  $w \in L_1 \cap L_2$  entonces la máquina va a aceptar con testigo  $w' = (w_1, w_2)$  ya que  $M_1$  acepta a  $(w, w_1)$  y  $M_2$  acepta a  $(w, w_2)$ .

Si  $w \notin L_1 \cap L_2$ , entonces  $w \notin L_1$  o  $w \notin L_2$ . Sin pérdida de generalidad  $w \notin L_1$ . Entonces para toda palabra  $w_1$ , se tiene que  $M_1$  no acepta a  $(w, w_1)$ . Luego  $M'$  rechaza.

Podemos hacer lo análogo para probar que  $L_1 \cup L_2 \in \text{NP}$ .

- b) Contraejemplo: tomar dos lenguajes en NP-completo tales que  $L_1 \cap L_2 = \emptyset$ . Por ejemplo:

$$L_1 = \{1\#w : w \in \text{SAT}\}$$

$$L_2 = \{0\#w : w \in \text{SAT}\}$$

Donde  $\#$  es concatenar ej:  $0\#10111 = 010111$ .

Claramente  $L_1$  y  $L_2$  están en NP-completo pero su intersección es vacía, lo que no está en NP-completo.

- c) Contraejemplo: tomar dos lenguajes en NP-completo tales que  $L_1 \cup L_2 = A^*$ . Por ejemplo:

$$L_1 = \{1\#w : w \in \text{SAT}\} \cup \{0\#w : w \in \{0, 1\}^*\}$$

$$L_2 = \{0\#w : w \in \text{SAT}\} \cup \{1\#w : w \in \{0, 1\}^*\}$$

Los lenguajes  $L_1$  y  $L_2$  están en NP-completo. Pero la unión es todo  $\{0,1\}^*$  lo cual no está en NP-completo.

**Problema 2.** Sean  $L_1 \subseteq \{0,1\}^*$  y  $L_2 \subseteq \{0,1\}^*$  dos lenguajes tales que  $L_1$  es NP-completo y  $L_2 \in P$ .

- a) ¿Es  $L_1 \cup L_2$  NP-completo?
- b) Si  $L_1 \cap L_2 = \emptyset$ , ¿Es  $L_1 \cup L_2$  NP-completo?
- c) Si  $L_1 \cap L_2 = \emptyset$  y  $NP \neq P$ , ¿Es  $L_1 \cup L_2$  NP-completo?

**Solución:**

- a) No necesariamente, podemos tomar cualquier lenguaje tal que  $L_1 \subseteq L_2$ . Así,  $L_1 \cup L_2 = L_2 \in P$ . Por ejemplo;  $L_1$  el problema de  $K$ -Clique y  $L_2$  el lenguaje de todas las codificaciones de grafos  $G$  y un entero  $k$ .
- b) Vamos a mostrar una reducción que hace que  $L_1 \cup L_2$  esté en NP bajo la condición de que  $L_2 \subsetneq \overline{L_1}$ , o sea, que existe al menos una palabra en el complemento de  $L_1$  pero no está en  $L_2$ . La reducción de  $L_1$  a  $L_1 \cup L_2$  es la siguiente:

$$f(w) = \begin{cases} w_0 & \text{si } w \in L_2 \\ w & \text{si no} \end{cases}$$

donde  $w_0$  es una palabra arbitraria que no está en  $L_1 \cup L_2$ .

Por construcción  $w \in L_1 \Leftrightarrow f(w) \in L_1 \cup L_2$ . Por otro lado es claro que  $L_1 \cup L_2$  está en NP. Así que  $L_1 \cup L_2$  es NP-completo.

- c) Si  $NP \neq P$  entonces se tiene que  $\overline{L_1}$  no está en P, caso contrario si  $\overline{L_1}$  entonces  $L_1$  también está en P (contradicción). Como  $\overline{L_1}$  no está en P pero  $L_2$  si está en P, debe haber al menos un elemento que esté en  $\overline{L_1}$  pero no en  $L_2$ , lo que cumple con la condición que impusimos en b) y por lo cual  $L_1 \cup L_2$  debe ser NP-completo.

**Problema 3.** Sea  $U = \{(M, w, \#^t) \mid M \text{ MT no determinista tal que acepta a } w \text{ en } t \text{ pasos en al menos una ejecución}\}$ . Pruebe que  $U$  es NP-completo.

**Solución:** Dado cualquier lenguaje en NP, tenemos que existe una máquina no determinista  $M_L$  tal que para todo  $w \in L$ ,  $M_L$  acepta a  $w$  en al menos una ejecución en a lo más una cantidad de pasos  $p_L(|w|)$ , donde  $p_L$  es un polinomio que depende de la máquina. También  $M_L$  no acepta a todo  $w \in L$ . Entonces dado  $w$  creamos en tiempo polinomial  $f(w) = (M_L, w, \#^{p_L(|x|)})$ . Por el argumento anterior  $w \in L \Leftrightarrow f(w) \in U$ . Así  $U$  es NP-hard.

Para mostrar que  $U$  está en NP, creamos una máquina  $M_U$  tal que para un input  $l = (M, w, \#^t)$  simula  $M$  con input  $w$  por  $t$  pasos.  $M_U$  va por todas las posibles ejecuciones de  $M$  y acepta a  $l$  si y sólo si  $M$  acepta a  $w$  en  $t$  pasos. Como simulamos hasta a lo más  $t$  pasos y el input de de a lo menos largo  $t$  el tiempo de ejecución es de complejidad polinómica en función de  $t$ . Es claro que  $M_U$  acepta el lenguaje  $U$ , esto prueba que  $U \in NP$ . Finalmente,  $U$  es NP-completo.

**Problema 4.** Pruebe que el lenguaje HAMPATH es NP-completo.

$$\text{HAMPATH} = \{\text{cod}(G) \mid G \text{ posee un camino hamiltoniano}\}$$

**Solución:** La idea es hacer una reducción de 3SAT a HAMPATH... (pendiente)