Unidad III: Complejidad Computacional

NP-completitud

Clase 17 - Lógica para Ciencia de la Computación/Teoría de la Computación

Prof. Miguel Romero

P vs NP

- Para demostrar P $\neq$ NP necesitamos encontrar un lenguaje $L \in \text{NP} \setminus \text{P}$.
  - ¿Qué L sería un buen candidato?
- ¿Existen lenguajes más difíciles que otros dentro de NP?
- ¿Cómo podemos comparar los lenguajes de NP?
- En los años 70, surge la teoría de NP-completitud.
  - Aborda las preguntas anteriores.
  - Nos ayuda a entender mejor la pregunta P vs NP.

Definición (recordatorio):

Sean alfabetos finitos Σ_1 y Σ_2. Sea una función $f : \Sigma_1^* \to \Sigma_2^*$ y una MTD $M = (Q, \Sigma_1, \Gamma, q_0, q_f, \delta)$. Decimos que M **computa** f si:

1. Para toda entrada $w \in \Sigma_1^*$, la ejecución $\rho = C_0, \ldots, C_m$ de M sobre w es finita. (M se detiene con w.)
2. $C_m = \vdash q_f\, f(w)$.

En otras palabras:

Para toda entrada $w \in \Sigma_1^*$, la MT M acepta, deja la cabeza en la posición 1, y deja escrito en la cinta la palabra $f(w) \in \Sigma_2^*$.

Definición:

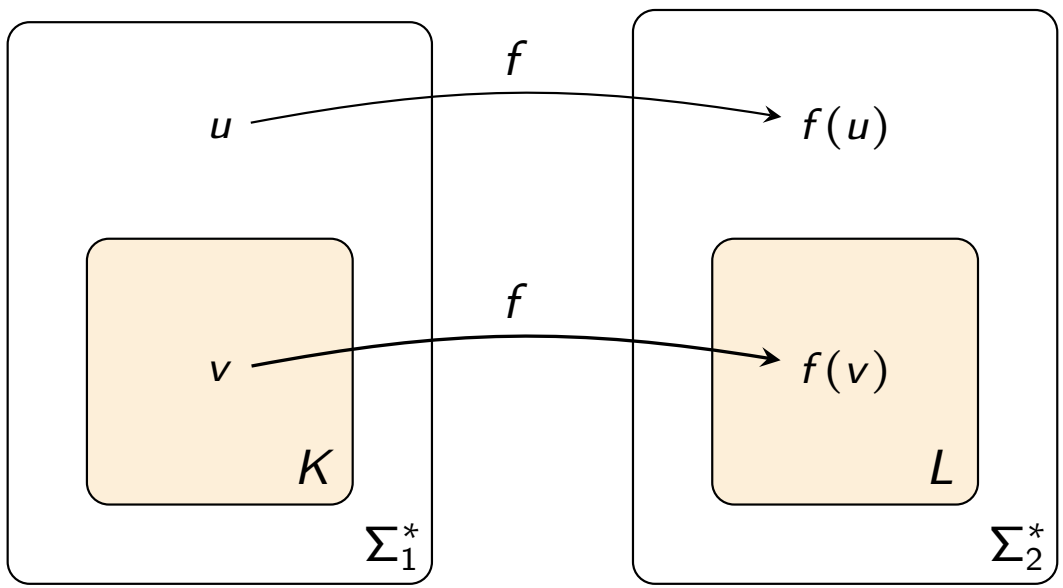
Sean alfabetos finitos Σ_1 y Σ_2, y $f : \Sigma_1^* \to \Sigma_2^*$ una función. Decimos que f es **computable en tiempo polinomial** si existe una MTD M que toma tiempo polinomial y computa f.

Definición:

Sean lenguajes $K \subseteq \Sigma_1^*$ y $L \subseteq \Sigma_2^*$. Una **reducción polinomial** de K a L es una función computable en tiempo polinomial $f : \Sigma_1^* \to \Sigma_2^*$ tal que para toda palabra $w \in \Sigma_1^*$:

$$w \in K \iff f(w) \in L$$

Idea:



Definición:

Sean lenguajes $K \subseteq \Sigma_1^*$ y $L \subseteq \Sigma_2^*$. Una **reducción en tiempo polinomial** de K a L es una función computable en tiempo polinomial $f : \Sigma_1^* \to \Sigma_2^*$ tal que para toda palabra $w \in \Sigma_1^*$:

$$w \in K \iff f(w) \in L$$

Comentarios:

- La función $f : \Sigma_1^* \to \Sigma_2^*$ **no** tiene que ser biyectiva (ni inyectiva, ni sobre).
- Si existe una reducción en tiempo polinomial de K a L, decimos que K se reduce en tiempo polinomial a L y escribimos:

$$K \leq_p L.$$

- La relación $\leq_p$ entre lenguajes es **transitiva**:

$$\text{Si } K \leq_p L \text{ y } L \leq_p J, \text{ entonces } K \leq_p J.$$

(¿por qué?)

Propiedades de las reducciones en tiempo polinomial

Proposición:

Si $K \leq_p L$ y $L \in \mathrm{P}$, entonces $K \in \mathrm{P}$.

Demostración:

Sea f la reducción polinomial de K a L y M_L la MTD polinomial que decide L.
(f es computable en tiempo $O(n^k)$ y M_L toma tiempo $O(n^d)$)

La siguiente MTD M_K decide K.

Sobre entrada w:

1. Computar $f(w)$.
2. Ejecutar M_L sobre $f(w)$.
3. Si M_L acepta, **aceptar**.
4. Si M_L rechaza, **rechazar**.

Se cumple que:

$$w \in K \iff f(w) \in L \iff M_L \text{ acepta } f(w) \iff M_K \text{ acepta } w$$

El tiempo de ejecución de M_K es: $O(|w|^k) + O(|w|^{kd})$

Obtenemos que $K \in \mathrm{P}$.

Propiedades de las reducciones en tiempo polinomial

Proposición:

Si $K \leq_p L$ y $L \in \mathrm{P}$, entonces $K \in \mathrm{P}$.

Corolario:

Si $K \leq_p L$ y $K \notin \mathrm{P}$, entonces $L \notin \mathrm{P}$.

Ejemplo

Veamos que 3-COL $\leq_p$ SAT.

La reducción f mapea un grafo G a una fórmula φ_G tal que:

$$\langle G \rangle \in \text{3-COL} \iff \langle \varphi_G \rangle \in \text{SAT}.$$

Sea $G = (V, E)$ un grafo.

Variables proposicionales de φ_G:

- por cada nodo $u \in V$ y cada color $k \in \{1, 2, 3\}$, una variable x_{uk}.
- x_{uk}: "el nodo u recibe el color k".

Ejemplo

Definimos φ_G como:

$$\varphi_G^{\text{color}} = \bigwedge_{u \in V} (x_{u1} \vee x_{u2} \vee x_{u3})$$

$$\varphi_G^{\text{unico}} = \bigwedge_{u \in V} (x_{u1} \to (\neg x_{u2} \wedge \neg x_{u3})) \wedge (x_{u2} \to (\neg x_{u1} \wedge \neg x_{u3})) \wedge (x_{u3} \to (\neg x_{u1} \wedge \neg x_{u2}))$$

$$\varphi_G^{\text{arista}} = \bigwedge_{\{u,v\} \in E} \neg(x_{u1} \wedge x_{v1}) \wedge \neg(x_{u2} \wedge x_{v2}) \wedge \neg(x_{u3} \wedge x_{v3})$$

$$\varphi_G = \varphi_G^{\text{color}} \wedge \varphi_G^{\text{unico}} \wedge \varphi_G^{\text{arista}}$$

La reducción f se puede computar en tiempo polinomial.

Ejemplo

$$\langle G \rangle \in \text{3-COL} \iff \langle \varphi_G \rangle \in \text{SAT}.$$

$(\Rightarrow)$:

Si $\langle G \rangle \in$ 3-COL, existe un 3-coloreo $c : V \to \{1, 2, 3\}$ para G.

Definamos la siguiente valuación σ para φ_G:

$$\sigma(x_{uk}) = \begin{cases} 1 & \text{si } c(u) = k \\ 0 & \text{caso contrario} \end{cases}$$

Por construcción de φ_G, tenemos que $\sigma(\varphi_G) = 1$. (verificarlo!)

Concluimos que $\langle \varphi_G \rangle \in$ SAT.

Ejemplo

$$\langle G \rangle \in \text{3-COL} \iff \langle \varphi_G \rangle \in \text{SAT}.$$

$(\Leftarrow)$:

Si $\langle \varphi_G \rangle \in \text{SAT}$, existe una valuación σ tal que $\sigma(\varphi_G) = 1$.

Por construcción de φ_G (en particular, por φ_G^{color} y φ_G^{unico}):

> para cada $u \in V$, existe un único número $k_u \in \{1, 2, 3\}$ tal que $\sigma(x_{uk_u}) = 1$.

Podemos definir la siguiente función $c : V \to \{1, 2, 3\}$:

$$c(u) = k_u \qquad \text{para cada } u \in V$$

Debido a $\varphi_G^{\text{arista}}$, la función c debe ser un 3-coloreo. (verificarlo!)

Concluimos que $\langle G \rangle \in \text{3-COL}$.

Definición:

Un lenguaje K es **NP-hard** si **para todo** $L \in \text{NP}$, se cumple que $L \leq_p K$.

Idea: K es "al menos tan difícil" como toda la clase NP.

Definición:

Un lenguaje K es **NP-completo** si $K \in \text{NP}$ y K es NP-hard.

Idea: Los problemas NP-completos son los "más difíciles" dentro de NP.

Proposición:

Si K es NP-completo y $K \in \mathrm{P}$, entonces $\mathrm{P} = \mathrm{NP}$.

Demostración:

Sea $L \in \mathrm{NP}$. Como K es NP-hard, sabemos que $L \leq_p K$.

Como $K \in \mathrm{P}$, tenemos que $L \in \mathrm{P}$. Concluimos que $\mathrm{NP} \subseteq \mathrm{P}$, es decir, $\mathrm{P} = \mathrm{NP}$.

Corolario:

Si K es NP-completo, entonces: Tenemos que

$$K \in \mathrm{P} \text{ si y sólo si } \mathrm{P} = \mathrm{NP}.$$

La pregunta P vs NP se reduce a analizar problemas NP-completos.

Proposición:

Si K es NP-completo y $K \in \mathrm{P}$, entonces $\mathrm{P} = \mathrm{NP}$.

Demostración:

Sea $L \in \mathrm{NP}$. Como K es NP-hard, sabemos que $L \leq_p K$.

Como $K \in \mathrm{P}$, tenemos que $L \in \mathrm{P}$. Concluimos que $\mathrm{NP} \subseteq \mathrm{P}$, es decir, $\mathrm{P} = \mathrm{NP}$.

Corolario:

Asumamos que $\mathrm{P} \neq \mathrm{NP}$. Si K es NP-completo, entonces $K \notin \mathrm{P}$.

Para probar que un problema no está en P, basta probar que es NP-completo.
(bajo el supuesto que $\mathrm{P} \neq \mathrm{NP}$.)

Proposición:

Si K es NP-hard y $K \leq_p L$, entonces L es NP-hard.

Demostración: Transitividad de $\leq_p$.

Corolario:

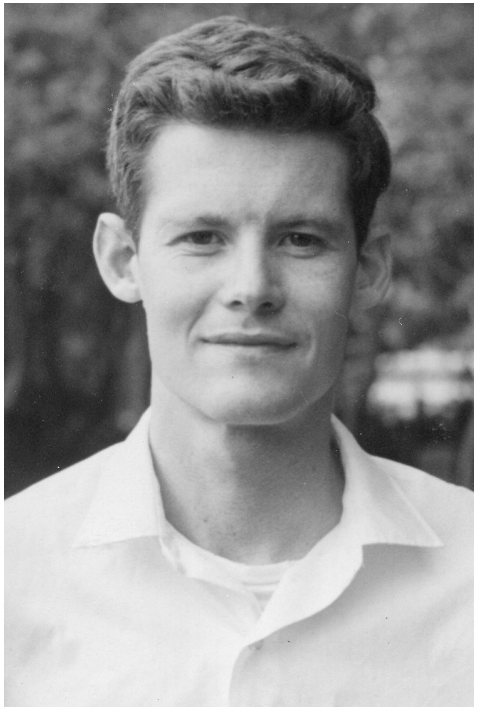
Si K es NP-completo y $K \leq_p L$ para $L \in \text{NP}$, entonces L es NP-completo.

¿Cómo demostramos que un problema L es NP-completo?

- Tomar un problema K que ya sabemos que es NP-completo.
- Demostrar que $L \in \text{NP}$ y $K \leq_p L$.

Necesitamos un punto de partida. ¿Existe algún problema NP-completo?

Teorema de Cook-Levin



Stephen Cook (1939 –)



Leonid Levin (1948 –)

Teorema de Cook-Levin (1971):

SAT es NP-completo.

Teorema de Cook-Levin: demostración

Sabemos que SAT $\in$ NP. Basta probar que SAT es NP-hard.

Sea $L \in$ NP. Existe MTN $M = (Q, \Sigma, \Gamma, q_0, q_f, \Delta)$ y polinomio $p : \mathbb{N} \to \mathbb{N}$ tq:

M decide L y el tiempo de M está acotado por p.

Debemos demostrar que $L \leq_p$ SAT.

La reducción debe mapear $w \in \Sigma^*$ a una fórmula proposicional φ_w tal que:

$$w \in L \iff \langle \varphi_w \rangle \in \text{SAT}$$

Idea:

Valuaciones que satisfacen a φ_w **codifican** ejecuciones de aceptación para M y w.

$$\begin{aligned} w \in L &\iff M \text{ acepta } w \\ &\iff \text{existe ejecución de aceptación para } M \text{ y } w \\ &\iff \text{existe valuación que satisface a } \varphi_w \\ &\iff \langle \varphi_w \rangle \in \text{SAT} \end{aligned}$$

Teorema de Cook-Levin: demostración

Dado: MTN $M = (Q, \Sigma, \Gamma, q_0, q_f, \Delta)$ que decide L en tiempo a lo más p.

Asumiremos sin pérdida de generalidad lo siguiente:

Para todo $w \in \Sigma^*$ con $|w| = n$,
toda ejecución de M con w hace exactamente una cantidad de pasos $p(n)$.

(¿por qué podemos asumir esto?)

Teorema de Cook-Levin: demostración

Dado: MTN $M = (Q, \Sigma, \Gamma, q_0, q_f, \Delta)$ que decide L en tiempo a lo más p.

Sea $w \in \Sigma^*$ y $n = |w|$.

Variables proposicionales de φ_w:

- $s_{t,i,a}$, para cada $0 \leq t \leq p(n)$, $0 \leq i \leq p(n)$ y $a \in \Gamma$.
  - $s_{t,i,a}$ = "En el tiempo t, la posición i de la cinta tiene el símbolo a".
- $c_{t,i}$, para cada $0 \leq t \leq p(n)$ y $0 \leq i \leq p(n)$.
  - $c_{t,i}$ = "En el tiempo t, la cabeza está en la posición i de la cinta".
- $e_{t,q}$, para cada $0 \leq t \leq p(n)$ y $q \in Q$.
  - $e_{t,q}$ = "En el tiempo t, el estado es q".

La fórmula φ_w se define como

$$\varphi_w = \varphi_{\text{unico}} \wedge \varphi_{\text{inicio}} \wedge \varphi_{\text{sig}} \wedge \varphi_{\text{aceptar}}$$

donde:

- φ_{unico}: En cada tiempo t, hay un único símbolo en cada posición i, la cabeza está en una única posición y estamos en un único estado.
- φ_{inicio}: La primera configuración es la configuración inicial de M con w.
- φ_{sig}: La configuración en tiempo $t+1$ sigue de la configuración en tiempo t.
- φ_{aceptar}: Terminamos en una configuración de aceptación.

φ_{unico}: En cada tiempo t, hay un único símbolo en cada posición i, la cabeza está en una única posición y estamos en un único estado.

φ_{unico} es la conjunción de las siguientes 3 fórmulas:

- En cada tiempo t y posición i, hay un único símbolo en esa posición:

$$\bigwedge_{t=0}^{p(n)} \bigwedge_{i=0}^{p(n)} \bigvee_{a \in \Gamma} \left(s_{t,i,a} \wedge \bigwedge_{\substack{b \in \Gamma \\ b \neq a}} \neg s_{t,i,b} \right).$$

- En cada tiempo t, la cabeza está en una única posición:

$$\bigwedge_{t=0}^{p(n)} \bigvee_{i=0}^{p(n)} \left(c_{t,i} \wedge \bigwedge_{\substack{j=0 \\ j \neq i}}^{p(n)} \neg c_{t,j} \right).$$

- En cada tiempo t, estamos en un único estado:

$$\bigwedge_{t=0}^{p(n)} \bigvee_{q \in Q} \left(e_{t,q} \wedge \bigwedge_{\substack{r \in Q \\ r \neq q}} \neg e_{t,r} \right).$$

φ_{inicio}: La primera configuración es la configuración inicial de M con w.

Asumamos que $w = a_1 \dots a_n$. La fórmula φ_{inicio} es:

$$S_{0,0,\vdash} \wedge \bigwedge_{i=1}^{n} S_{0,i,a_i} \wedge \bigwedge_{i=n+1}^{p(n)} S_{0,i,\mathtt{B}} \wedge C_{0,1} \wedge e_{0,q_0}.$$

Teorema de Cook-Levin: demostración

φ_{sig}: La configuración en tiempo $t+1$ sigue de la configuración en tiempo t.

φ_{sig} es la conjunción de 2 fórmulas:

- Si en tiempo t, la cabeza **no** está en la posición i, el símbolo en esa posición no cambia:

$$\bigwedge_{t=0}^{p(n)-1} \bigwedge_{i=0}^{p(n)} \Big(\neg c_{t,i} \rightarrow \bigwedge_{a \in \Gamma} \big(s_{t,i,a} \rightarrow s_{t+1,i,a} \big) \Big).$$

- Si en tiempo t, el estado es q y la cabeza está leyendo el símbolo a, entonces el símbolo, la cabeza y el estado en $t+1$ se actualizan según **alguna** transición $(q', b, D) \in \Delta(q, a)$:

$$\bigwedge_{t=0}^{p(n)-1} \bigwedge_{i=0}^{p(n)} \bigwedge_{(q,a) \in Q \times \Gamma} \Big((e_{t,q} \wedge c_{t,i} \wedge s_{t,i,a}) \rightarrow \bigvee_{(q',b,D) \in \Delta(q,a)} (e_{t+1,q'} \wedge X \wedge s_{t+1,i,b}) \Big)$$

donde:

$$X = \begin{cases} c_{t+1,i+1} & \text{si } D = \triangleright \text{ y } 0 \leq i \leq p(n) - 1 \\ c_{t+1,i-1} & \text{si } D = \triangleleft \text{ y } 1 \leq i \leq p(n) \\ 0 & \text{en caso contrario} \end{cases}$$

Teorema de Cook-Levin: demostración

φ_{aceptar}: Terminamos en una configuración de aceptación.

$$e_{p(n),q_f}.$$

De la construcción (y discusión) anterior, se sigue que:

$$w \in L \iff \langle \varphi_w \rangle \in \mathrm{SAT}.$$

Concluimos que SAT es NP-completo.

Definamos el siguiente lenguaje:

$$\text{CNF-SAT} = \{\langle \varphi \rangle \mid \varphi \text{ fórmula proposicional en CNF satisfacible}\}.$$

Recordatorio:

- Un literal es una variable proposicional x o su negación $\neg x$.
- Una cláusula es una disyunción de literales.
- Una fórmula está en CNF si es una conjunción de cláusulas.
- Ejemplo: $(\neg x \vee y) \wedge (y \vee z \vee \neg w) \wedge (w \vee \neg y) \wedge z$.

Teorema:

CNF-SAT es NP-completo.

Demostración (idea):

Usando distributividad entre $\vee$ y $\wedge$, podemos pasar todas las fórmulas anteriores a CNF en tiempo polinomial (verifíquelo!), salvo por:

$$\bigwedge_{t=0}^{p(n)} \bigvee_{i=0}^{p(n)} \left(c_{t,i} \wedge \bigwedge_{\substack{j=0 \\ j \neq i}}^{p(n)} \neg c_{t,j} \right).$$

Pero está fórmula es equivalente a:

$$\left(\bigwedge_{t=0}^{p(n)} \bigvee_{i=0}^{p(n)} c_{t,i} \right) \wedge \left(\bigwedge_{t=0}^{p(n)} \bigwedge_{i=0}^{p(n)} \bigwedge_{\substack{j=0 \\ j \neq i}}^{p(n)} \left(\neg c_{t,i} \vee \neg c_{t,j} \right) \right).$$

Definamos el siguiente lenguaje:

$$\text{DNF-SAT} = \{\langle \varphi \rangle \mid \varphi \text{ fórmula proposicional en DNF satisfacible}\}.$$

Recordatorio:

- Un término es una conjunción de literales.
- Una fórmula está en DNF si es una disyunción de términos.
- Ejemplo: $(x \wedge \neg y \wedge w) \vee (z \wedge \neg w) \vee (w \wedge \neg y \wedge \neg x) \vee z$

Proposición:

DNF-SAT $\in$ P.

Demostración: Ejercicio.