

Teoremas y proposiciones evaluadas en el primer parcial (pero con mis formas de ver la demo)

Proposición 1. Sea $f : \{0,1\}^* \rightarrow \{0,1\}^*$, sea T una función construible en tiempo y sea Γ un alfabeto. Si f es computable en tiempo $T(n)$ por una máquina $M = (\Gamma, Q, \delta)$, entonces f es computable en tiempo $O(\log |\Gamma| \cdot T(n))$ por una máquina $M' = (\Sigma, Q', \delta')$ donde $\Sigma = \{0, 1, \triangleright, \square\}$ es el alfabeto estándar.

Demo:

$|\Gamma|$ es la cantidad de estados que hay originalmente, para codificar cada estado de Q en Q' se usa el alfabeto Σ (binario), por lo cual llevará $\log_2 |\Gamma|$ bits por estado ($\log_2 |\Gamma| = \log |\Gamma|$ bits por convención, aunque en big O no importa).

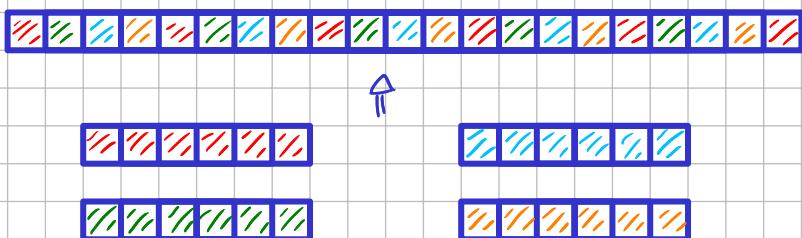
Esto se traslada a δ' también, por lo que lo que antes llevaba un símbolo de Γ ahora lleva $\log |\Gamma|$ de Σ .

Proposición 2. Sea $f : \{0,1\}^* \rightarrow \{0,1\}^*$ y sea T una función construible en tiempo. Si f es computable en tiempo $T(n)$ por una máquina estándar de $k \geq 3$ cintas (entrada, salida y $k-2$ cintas de trabajo), entonces f es computable en tiempo $O(T(n)^2)$ por una máquina de cinta única.

Demo:

Puedo alternar los símbolos de las k cintas en la cinta única y usar símbolos característicos para indicar dónde está la cabeza de cada cinta.

En la posición i está el carácter $\Gamma_i/3\rfloor$ de la cinta $i \bmod k$.



Quedo en $O(T(n)^2)$ debido a que por cada paso de δ debe primero ubicar cada cabeza para ver qué accionar y después modificar cada cabeza como lo indique δ .

Proposición 3. Sea $f : \{0,1\}^* \rightarrow \{0,1\}^*$ y sea T una función construible en tiempo. Si f es computable en tiempo $T(n)$ por una máquina estándar entonces hay una máquina oblivious que computa f en tiempo $O(T(n)^2)$.

Demo:

Esto es similar a Prop. 2.

La máquina oblivious se mueve un patrón fijo en cada paso (Dame todo la cinta) y cambia la cinta con los cambios que se piden.

Proposición 4. Sea $f : \{0,1\}^* \rightarrow \{0,1\}^*$ y sea T una función construible en tiempo. Si f es computable por una máquina con cintas bi-infinitas en tiempo $T(n)$, entonces f es computable por una máquina estándar en tiempo $O(T(n))$.

Demo:

Se puede doblar la cinta bi-infinita de manera que rebota en el símbolo \triangleright para ver ambas infinitas.



Teorema 1. [Turing 1936] halt no es computable.

$$\text{halt}(x) : \begin{cases} 1 & \text{si la } x\text{-ésima máquina con entrada } x \text{ termina } (M_x(x)) \\ 0 & \text{si no} \end{cases}$$

Demo:

Sale por diagonalización. Tengo que:

	M_1	M_2	\dots
1	$M_1(1)$		
2		$M_2(2)$	
3			\ddots
\vdots			

Defino entonces M tal q' $M(x)$ termina sii $\text{halt}(x)=0$.

$M(\langle M \rangle)$ termina sii $\text{halt}(\langle M \rangle)=0$ sii $M(\langle M \rangle)$ no termina abs!

Teorema 2. Existe una máquina U que computa la función $u(\langle i, x \rangle) = M_i(x)$. Más aún, si M_i con entrada x termina en t pasos, entonces U con entrada $\langle i, x \rangle$ termina en $c \cdot t \cdot \log t$ pasos, donde c depende solo de i .

Demo:

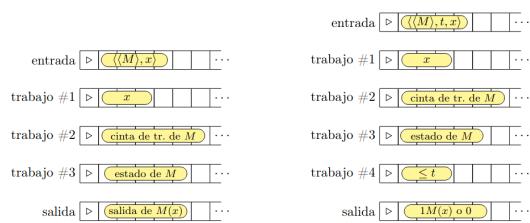


Figura 12: Izquierda: la simulación que hace U (con 3 cintas de trabajo) de M (con una única cinta de trabajo) y entrada x . Derecha: la simulación que hace \tilde{U} (con 4 cintas de trabajo) de M (con una única cinta de trabajo), entrada x hasta el tiempo t .

Pone en cada cinta de trabajo la simulación de la máquina estacionaria de M . O sea, en #1: La entrada x , #2: Cinta de trabajo de M , #3: Estado de M . Si #3 = $[q_f]$ termina la ejecución M .
Por cada paso de M busco & en la entrada de la máquina U .

Teorema 3. Existe una máquina \tilde{U} que computa la función $\tilde{u}(\langle i, t, x \rangle)$ en tiempo $c \cdot t \cdot \log t$, donde c depende solo de i .

Demo:

Muy similar a Teo 2 pero con una cinta de trabajo más para llevar registro de i (cant. de pasos en la simulación).

Teorema 4. $P \subseteq NP$.

Demo:

Sea $L \in P$. $L(M)$, M máq. det. poly.

Tomás $p(|x|) = 0$. Defino M' det. poly y el certificado $c = \emptyset$, entonces tengo que M' con entrada $\langle x, c \rangle$ copia el comportamiento de M con entrada x .

$x \in L$ sii $M(x) = 1$

sii $M'(\langle x, c \rangle) = 1$

sii $\exists c \in \{0, 1\}^0$ tal q' $M'(\langle x, c \rangle) = 1$

Lo D. se $\in NP$.

Teorema 6. Existe una máquina no-determinística NU que tal que NU acepta $((i, x))$ sii N_i acepta x y si N_i corre en tiempo $T(n)$ entonces $NU((i, x))$ decide si N_i acepta o rechaza x en $c \cdot T(|x|)$ pasos, donde c depende solo de i .

Demo: Similar Teo. 2. No tiene el logaritmo porque ...

Teorema 7. La relación \leq_p es transitiva.

Demo:

Sea $L \leq_p L'$ vía f y $L' \leq_p L''$ vía g qvq $L \leq_p L''$

$x \in L$ sii $f(x) \in L'$ sii $g(f(x)) \in L''$

Ahora qvq $g \circ f$ es computable en tiempo poly a $|x|$.

Sea M_f, M_g tal q' M_f computa f en poly y M_g que computa g en poly.

$M_{g \circ f} : \langle x \rangle$

$y_i = M_f(x) \rightarrow O(n^c) \rightarrow$ A la suma su salido es polinomial respecto $n = |x|$

$\text{ret } M_g(y) \rightarrow O((n^c)^d) = O(n^{cd}) \rightarrow$ A la suma es poly respecto $|y|$.

Teorema 8. Si $NP\text{-hard} \cap P \neq \emptyset$, entonces $P = NP$.

Demo:

Si $L \in NP\text{-hard} \cap P$, entonces puedo tomar cualquier $L' \in NP$ y reducirlo a L , por lo cual tenga una f computable poly. tal que:

$x \in L' \text{ sii } f(x) \in L$

Entonces a la $L'(M_f)$ la declaro como:

$M_L(x)$

$$y_i = M_f(x) \rightarrow O(n^c)$$

$$M_L(y) \rightarrow O(n^{cd})$$

}

Es una máq. det. poly por lo q' $L \in P$.

O sea $P = NP$.

Teorema 9. Si $L \in NP$ -completo, entonces $L \in P$ si $P = NP$.

Demo:

$\Rightarrow) L \in P \Rightarrow P = NP$

Si L es NP -C y a su vez $L \in P$, lo único que agrega que $L \in NP$ -C (como L ya era NP (porque $P \subseteq NP$)) es que $L \in NP$ -hard.

Por esto, sale por el Teo. 8.

$\Leftarrow) P = NP \Rightarrow L \in P$

Si $P = NP$, entonces todos los problemas de NP se pueden resolver en tiempo poly, en particular también todos los NP -C.

Problema: TMSAT

$$\text{TMSAT} = \{\langle y, x, 1^n, 1^t \rangle \mid \exists u \in \{0, 1\}^n \ M_y(xu) = 1 \text{ en a lo sumo } t \text{ pasos}\}$$

Proposición 6. $\text{TMSAT} \in NP$ -completo.

Demo: $L \in NP$

$$x \in L \iff \exists u \in \{0, 1\}^{P(|x|)} \ M(x, u) = 1 \quad (\text{def. de } L \text{ a } NP)$$

$$x \in L \iff f(x) \in \text{TMSAT} \quad (\text{def. } L \leq_p \text{TMSAT})$$

La f de la reducción sería:

$$f(x) = \langle \langle M \rangle, x, 1^{P(|x|)}, 1^{q(|x|+p(|x|))} \rangle \rightarrow \text{Computable en tiempo poly.}$$

$$f(x) \in \text{TMSAT} \iff \langle \langle M \rangle, x, 1^{P(|x|)}, 1^{q(|x|+p(|x|))} \rangle \in \text{TMSAT}$$

$$\iff \exists u \in \{0, 1\}^{P(|x|)} \ M(x, u) = 1 \text{ o } (q(|x|) = q(|x|+p(|x|))) \text{ pasa}$$

$$\iff x \in L$$

Proposición 7. Sea $\varphi(p_1, \dots, p_n)$ una fórmula booleana. Si v y v' son dos evaluaciones tales que $v(p_i) = v'(p_i)$ para $i \in \{1, \dots, n\}$, entonces $v \models \varphi(p_1, \dots, p_n)$ si $v' \models \varphi(p_1, \dots, p_n)$.

Demo:

$v(p_i) = v'(p_i)$ significa que la evaluación evalúa a las variables proposicionales con el mismo valor. La prop. sale inmediata.

Teorema 10. SAT, 3SAT $\in \text{NP}$.

Demo:

$M(x, u)$

- x representa una fórmula booleana
- Si: $x = \{q\}$ y Ψ tiene m var. proposicionales:
 - ret $\underbrace{u \cap m}_{\text{tanto las primeras } m \text{ elementos de } u} \models \Psi$

Proposición 8. Para toda $F : \{0,1\}^\ell \rightarrow \{0,1\}$ existe una fórmula booleana $\varphi_F(p_1, \dots, p_\ell)$ en CNF tal que $v \models \varphi_F$ si y sólo si $F(v) = 1$ para todo $v \in \{0,1\}^\ell$. Más aún, φ_F se computa en tiempo polinomial a partir de $\langle F \rangle$ y tiene tamaño $O(\ell 2^\ell)$.

Demo:

$\begin{matrix} \text{1 bits por valoración} \\ \text{10 bits por valoración} \end{matrix}$

Concatena todas las posibles combinaciones de ℓ bits ($\ell 2^\ell$) de manera que si $F(v) = 0$, toma un $\varphi_F(p_1, \dots, p_\ell)$ y (p_1, \dots, p_ℓ) erró en φ_F .

Corolario 1. Para toda $F : \{0,1\}^\ell \rightarrow \{0,1\}^k$ existe una fórmula booleana $\varphi_F(p_1, \dots, p_{\ell+k})$ en CNF tal que $uv \models \varphi_F$ si y sólo si $F(u) = v$ para todo $u \in \{0,1\}^\ell$, $v \in \{0,1\}^k$. Más aún, φ_F se computa en tiempo polinomial a partir de $\langle F \rangle$ y tiene tamaño $O((\ell+k)2^{\ell+k})$.

Demo:

Defino $G : \{0,1\}^{\ell+k} \rightarrow \{0,1\}$ tal que $G(u, v) = 1$ si $F(u) = v$ y aplicar Prop 8.

Teorema 11. SAT $\in \text{NP-hard}$. \rightarrow Teorema de Cook Levin

Idea de la demo (no explico todo pq' es muy técnica):

$x \in L$ si: $\exists u \in \{0,1\}^{P(x)}$ $M(x, u) = 1$

sii Existe $u \in \{0,1\}^{P(x)}$ y existe un cálculo $(z_0, \dots, z_{t(x)})$ a partir de la entrada xu

Con el uso de mini-configurationes $z_0, \dots, z_m \in \{0,1\}^K$ con K dependiente de la máquina.

Con la entrada (xu) quiero ver con fórmulas de tamaño polinomial a la entrada puedo simular el cálculo de la M det. que decide a L .

Las fórmulas son:

- ① La que representa la cinta de entrada. (xu)
- ② Configuración inicial de M a partir de (xu) $(q_0, \text{símbolo})$
- ③ Evolución en un paso de z_i a z_{i+1} (δ)
- ④ Configuración final de M aceptadora. $(q_f, \text{símbolo en la cinta de trabajo} = 1)$

Corolario 2. SAT, 3SAT \in NP-completos.

→ Salte del Teo. 10 y 11

Proposición 9. INDSET \in NP-completo.

Demo:

Ver que INDSET \in NP es medianamente fácil, ahora ver q' INDSET es NP-hard
Tenes que hacerlo por def (nunca) o por reducción poly de un NP-hard q' separado.

Vemos que 3SAT \leq_p INDSET:

$$\varphi = (l_1 \vee l_2 \vee l_3) \wedge \dots \wedge (l_m \vee l_{m+1} \vee l_{m+2})$$

A la sumo tengo $3m$ vértices (si todos los literales son \neq), uno para cada variable

Defino una arista entre z y z' si están en la misma cláusula de la fórmula o la variable ligada al nodo z es la negación de la variable ligada al nodo z'

Esto sería la f de q:

$$x \in 3SAT \text{ si } f(x) \in \text{INDSET}$$

(la demo de que funciona quedó en tu imaginación)

Proposición 10. CAMHAM \in NP-completo.

Teorema 12. TSP \in NP-completo.

Proposición 11. KNAPSACK \in NP-completo.

Teorema 13. Si $P = NP$ entonces EXPTIME = NEXPTIME.

Demo:

1) $\text{Exptime} \subseteq \text{NExptime}$

Esto ya se cumple sin que $P = NP$.

2) $\text{NExptime} \subseteq \text{Exptime}$

Toma $L \in \text{NExptime}$ y $L(N) \rightarrow q' N \text{ es no-det y corre en } \mathcal{O}(2^{nc})$

Considéro $L_{\text{pad}}: \langle x, 1^{2^{l|x|}} \rangle : x \in L \rangle$. (L con padding)

Vemos q' $L_{\text{pad}} \in \text{NP}$:

Como la entrada de L_{pad} es una cadena exp. con respecto a $|x|$, puedo correr a N con x por $2^{l|x|}$ pasos pq' es poly con respecto $\langle x, 1^{2^{l|x|}} \rangle$

Jerarquía de Tiempos Determinísticos

Teorema 14. Si f, g son construibles en tiempo y cumplen que $f(n) \cdot \log f(n) = o(g(n))$ entonces $\text{DTIME}(f(n)) \subseteq \text{DTIME}(g(n))$.

Demo:

A ver, esto va a ser lento y detallado pq' me cuesta entenderla.

La idea principal es demostrarlo por diagonalización que hay lenguajes que están en $\text{DTIME}(g(n))$ pero no en $\text{DTIME}(f(n))$.

Declaro D det. tal que:

$D: \{x\}$

$$\text{Calcula } n = |x| \text{ y } t = g(n) \rightarrow O(g(n) + n) = O(g(n))$$

Simula $U(\langle x, x \rangle)$ por t pasos $\rightarrow O(g(n))$

Si: $U(\langle x, x \rangle)$ no terminó en $\leq t$ pasos : ret 0 $\rightarrow O(1)$

Si: no : ret $\neg(U(\langle x, x \rangle)) \rightarrow O(1)$

$L(D) \in \text{DTIME}(g(n))$ es inmediato.

Ahora vamos a ver por absurdo la prueba:

Supongo q' $L(D) \in \text{DTIME}(f(n))$. Sea $L(M) = L(D)$, donde M termina en $\leq t$ pasos $O(f(|x|))$ pasos ($c \cdot f(|x|)$) (así $|x| \geq n$ q' es el punto donde siempre se vuelve mayor)

Por el Teo 2 sé que $U(\langle x, x \rangle)$ simula $M_x(x)$ en $c' \cdot c \cdot f(|x|) \cdot \log(c \cdot f(|x|))$ pasos (teo 2 dice q' $U(\langle x, x \rangle)$ corre en $K \cdot Z \cdot \log Z$ donde Z es el tiempo de corriente de x)

Luego, existen $d, n_1 \geq n_0$ tal q' $\forall n \geq n_1$:

$$c' \cdot c \cdot f(n) \cdot \log(c \cdot f(n)) \leq d \cdot f(n) \cdot \log(f(n)) \leq g(n)$$

Tomo x tal q' $|x| \geq n_1$, y $M_x = M$. La existencia de este x es garantizada pq' se pueden hacer ∞ mág. q' hagan lo mismo.

Ahora bien, $U(\langle x, x \rangle)$ termina en $g(|x|)$ pasos. Luego, $D(x) = \neg(M_x(x)) = \neg(M(x)) = 0$

OK, tiene sentido, te armás una mág. q' corra en $g(|x|)$ pasos, forzas a una mág. a identificar el mismo lenguaje pero en $\log(f(|x|)) \cdot f(|x|)$ pasos y como siempre termina, siempre pasa q' da una contradicción con el lenguaje q' suponía q' decidió.

¿Que pasa si le resto D_x ? No me da una contradicción también?

$D(\langle D_x \rangle) = U(\langle \langle D_x \rangle, \langle D_x \rangle \rangle)$ por t pasos = $g(|x|)$ pasos.. Ojo! Esto no termina, requiere pasos $\log(g(|x|)) \cdot g(|x|)$ pasos para terminar

Jerarquía de Tiempos no-determinísticos

Teorema 15. Si f, g son construibles en tiempo, no decrecientes y cumplen que $f(n+1) = o(g(n))$, entonces $\text{NDTIME}(f(n)) \subseteq \text{NDTIME}(g(n))$.

Demo:

Dejo espacio para verlo después.

Tercero de Ladner

Teorema 16 (Ladner). Si $P \neq NP$ entonces existe L tal que $L \in NP$, $L \notin P$, y $L \notin NP$ -completo.

Demo:

Q.VQ si $P \neq NP \Rightarrow \exists L. L \in NP \wedge L \notin P \wedge L \notin NP$ -Completo

Cómo hago esto? Me crea un L y voy viendo q' asumiendo el antecedente $P \neq NP$, llego a obtener q' se cumple el consecuente si lo hice bien al L .

Definimos el problema SAT_H relativo a la función $H: \mathbb{N} \rightarrow \mathbb{N}$ tal que:

$$SAT_H = \{ \psi 0^L : \psi \in SAT, L = | \psi | \}$$

↳ Es como un padding de SAT, a sea, no agrega nada a ψ .

Ahora, si $H(n) = n$, $SAT_H \in P$: Esto pasa pq' la entrada sería 0^L , por lo cual resolver que ψ es SAT en una M det. tardaría 2^L , lo cual es polinomial respecto a la entrada (pq' $2^L \leq 1^n = n$ para todo ψ , salvo finitas casos)

Si $H(n) = c + \epsilon n$, $SAT_H \in EXP$: El tener 0^L me dejó la entrada con 1^n y esto no es gran cambio para el SAT original, ya q' sigue siendo NP-C para una entrada poly.

Necesita que $H(n)$ tienda a infinito pero q' crezca lentamente para llegar a un NP-intermedio

Se define H tal que:

$$H(n) = \begin{cases} \min \{ i \in \log \log n \text{ tal q' } \forall x. x \in \{0,1\}^i \} & \text{if } \log n \\ \log \log n & \text{if } \log n \end{cases}$$

↳ Función característica de SAT_H

X' No entiend q' hace así, veámoslo más tranqui:

Buscamos si alguna máq. M_i decide SAT_H sobre todas las entradas chicas ($\log n$) en a lo sumo $i|x|$ pasos.

El i debe ser el mínimo menor a $\log \log n$

Si no existe, ponemos $H(n) = \log \log n$ para que crezca un poco

SAT_H y H quedan definidas por recursión mutua (H depende de SAT_H y viceversa)

• $H(n)$ usa $SAT_H(x)$ para palabras x tal que $|x| \leq \log(n)$

• $SAT_H(x)$ usa $H(n)$ para palabras x tal que $|x| > n$

donde $x = \psi 0^L$ y $n = |\psi|$.

Proposición 12. $SAT_H \in P \Rightarrow \exists c \forall n H(n) \leq c$.

Demostración. Supongamos que la máquina determinística M decide SAT_H en tiempo $c \cdot n^c$. Existe $i > c$ tal que $L(M) = L(M_i)$. Entonces para todo $n \geq 2^{2^i}$, tenemos que $H(n) \leq i$. Luego

$$H(n) \leq \max \left(\{H(m) : m < 2^{2^i}\} \cup \{i\} \right),$$

y esto termina la demostración de la Proposición. \square

Proposición 13. $\exists c \exists^{\infty} n H(n) \leq c \Rightarrow \text{SAT}_H \in \mathbf{P}$.

Demuestra. Supongamos $\exists c \exists^{\infty} n H(n) \leq c$. Entonces $\exists i \exists^{\infty} n H(n) = i$. Veamos que M_i decide SAT_H en tiempo $i \cdot n^i$. Supongamos que M_i no decide SAT_H . Existe x tal que $M_i(x) \neq \chi_{\text{SAT}_H}(x)$. Entonces, por definición de H , tenemos que $\forall n > 2^{|x|} H(n) \neq i$ y esto es un absurdo. Si suponemos que M_i no corre en tiempo $i \cdot n^i$, razonamos análogamente y llegamos también a un absurdo. Esto termina la demostración de la Proposición. \square

Proposición 14. Existe k tal que para toda fórmula booleana ψ , $|\psi| > k \Rightarrow |F_\psi| < |\psi|$.

Demuestra. Supongamos que vale lo contrario. Para todo k , existe una fórmula booleana ψ tal que $|\psi| > k$ y $|F_\psi| \geq |\psi| > k$. Como $\text{SAT}_H \notin \mathbf{P}$, por el contrarecíproco de la Proposición 13, tenemos que $\lim_{n \rightarrow \infty} H(n) = \infty$. Entonces

$$\exists k' \forall n > k' n^{H(n)} > c \cdot n^c. \quad (18)$$

Sea ψ tal que $|F_\psi| \geq |\psi| > k'$. Luego

$$\begin{aligned} c \cdot |\psi|^c &\leq c \cdot |F_\psi|^c && \text{(pues } |F_\psi| \geq |\psi|) \\ &< |F_\psi|^{H(|F_\psi|)} && \text{(por (18) y el hecho de que } |F_\psi| > k') \\ &\leq c \cdot |\psi|^c, && \text{(por (17))} \end{aligned}$$

y esto es un absurdo, que termina la demostración de la Proposición. \square

Proposición 15. $\text{DTIME}(S(n)) \subseteq \text{SPACE}(S(n))$.

Demo:

En cada paso una máq. det. solo puede usar 1 celda más de cada cinta.

Proposición 16. $\text{SPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.

Demo:

$\text{Space}(S(n))$ es la clase de lenguajes L tal q' $\exists M$. M det., $L(M)$ y M usa espacio $S(n)$

$\text{NSpace}(S(n))$ es lo mismo pero con M no-det.

Luego, fácilmente puedo decir q' si $L \in \text{Space}(S(n)) \Rightarrow L \in \text{NSpace}(S(n))$ pq' es ignorar el certificado e imitar a la M det. lo que hace la máq. no-det.

Proposición 18. Si S es construible en espacio, $\text{NSPACE}(S(n)) \subseteq \text{DTIME}(2^{O(S(n))})$.

Demo:

Hago una M det. que construya el grafo de configuraciones (Toma $\mathcal{Q}^{O(S(n))}$) de la N no-det. y luego hago BFS para ver si existe un camino entre la configuración inicial a la final de N con entrada x .

Proposición 19. $\text{NP} \subseteq \text{PSPACE}$.

Demo:

Tengo $L \in \text{NP}$,

$$x \in L \quad \text{si: } \exists u \in \{0,1\}^{p(|x|)} \cdot M(x, u) = 1$$

M máq. det. poly y $p(|x|)$ es un polinomio de $|x|$

Luego, Creo M' det tal que genere todos los Certificados posibles en los mismos $p(x)$ bits siempre y corra $M(x, u)$ con todos los certificados hasta recorrer todos o que $M(x, u)$ sea 1.

Jerarquía de Espacio

Teorema 17. Si f, g son construibles en espacio y cumplen que $f(n) = o(g(n))$ entonces

$$\text{SPACE}(f(n)) \subsetneq \text{SPACE}(g(n)).$$

Demo:

Sea D una máq. det. tal que $\mathcal{L}(D)$:

$D: \langle x \rangle \rightarrow O(2^{g(|x|)})$ con espacio $g(n)$ usado
 $t := g(n)$ Recorde q' $\text{Space}(S(n)) \subseteq \text{Space}(s(n)) \subseteq \text{BTire}(2^{O(s(n))})$

Ejecuto $U(\langle x, x \rangle)$ por 2^t pasos

Si no terminó: ret 1

Si terminó: ret $\neg U(\langle x, x \rangle)$

Luego supongo q' tengo M máq. det. tal que $\mathcal{L}(M)$ y ocupa $f(n)$ espacio en ejecutarse.

Luego, si tengo M_x (Máq. de configuración x) y hago lo siguiente:

$$D(x) = \neg U(\langle x, x \rangle) = \neg M_x(x) = \neg M(x)$$

Sé q' termina pq' $2^{f(x)} < 2^{g(x)}$

Abs! Si: $\mathcal{L}(D) = \mathcal{L}(M)$ no puede ser q' para un mismo x dan resultados opuestos!

Pregunto q' se surgió y por ahí al q' le esto también (lo vas a sacar yo?) Ufa, otro loop lapo!

Si yo le pongo D_y . No me da contradicción también?

C

$$D(y) = \begin{cases} \text{termina?} & \text{Si:} \\ & \text{No:} \\ & \text{Mas o menos?} \end{cases}$$

$D(y)$ llama a $U(\langle y, y \rangle)$ y corre esta máquina, que a su vez es $D(y)$... y de repente **basta!** Se cumplieron los $2^{g(x)}$ pasos y retorna 1.

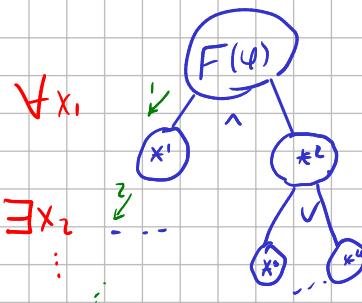
Entonces, $D(y)$ si termina, es mas, devuelve 1! !!

Teorema 18. TQBF $\in \text{PSPACE}$.

Demo:

Se puede definir una función recursiva F de manera que se reutilice espacio y siempre use el mismo espacio para x

• Pasos de ejecución.



$$\psi = \forall x_1 \exists x_2 \forall x_3 \exists x_4 \dots$$

$$\begin{aligned} x^1 &= F(\psi \wedge x_1=0) \\ x^2 &= F(\psi \wedge x_1=1) \\ x^3 &= F(\psi \wedge x_2=0) \\ x^4 &= F(\psi \wedge x_2=1) \end{aligned}$$

Cuando se termina de evaluar pone el resultado en el anterior y sigue con el otro hijo, tipo DFS.

Proposición 20. En tiempo polinomial se puede transformar cualquier QBF generalizada $\psi(y_1, \dots, y_m)$ en una QBF $\psi'(y_1, \dots, y_m)$ posiblemente con variables libres que es equivalente, es decir, tal que para todo $v \in \{0,1\}^m$

$$v \models \psi(y_1, \dots, y_m) \quad \text{sii} \quad v \models \psi'(y_1, \dots, y_m).$$

Demo:

Ejercicio 9. Demostrar la Proposición 20. Idea: renombrar variables cuantificadas si hace falta y aplicar reglas:

$$\begin{aligned} \neg \forall x \psi &= \exists x \neg \psi \\ \neg \exists x \psi &= \forall x \neg \psi \\ \psi * (Qx \varphi(x)) &= Qx (\varphi * \psi) \quad (Q \in \{\forall, \exists\}, * \in \{\wedge, \vee\}, \psi \text{ no contiene a } x \text{ libre}) \end{aligned}$$

Proposición 21. $\text{CHECKQBF} \leq_p \text{TQBF}$.

Demo:

Demostración. Sea ψ una QBF generalizada con variables libres y_1, \dots, y_m y sea $v \in \{0,1\}^m$. Primero llevamos ψ a una ψ' equivalente pero en forma prenexa (ψ' también tiene variables libres y_0, \dots, y_{m-1}). Por la Proposición 20 esto toma tiempo polinomial. Luego definimos ψ'' como el resultado de reemplazar cada variable y_j en ψ' por la constante $v(j)$; nos queda una QBF tal que¹³

$$\begin{array}{ll} \langle \psi, v \rangle \in \text{CHECKQBF} & \text{sii} \quad v \models \psi(y_1, \dots, y_m) \\ & \text{sii} \quad \models \psi' \\ & \text{sii} \quad \psi'' \in \text{TQBF} \end{array}$$

La transformación $\langle \psi, v \rangle \mapsto \psi''$ es computable en tiempo polinomial y esto prueba que $\text{CHECKQBF} \leq_p \text{TQBF}$. \square

Proposición 22. Sea M una máquina (determinística o no-determinística) que usa espacio $S(n) \geq \log n$ y sea c una constante tal que para todo $x \in \{0,1\}^*$ cualquier configuración C de un cálculo de M con entrada x se representa con $|C| = c \cdot S(|x|)$ bits. Dado $x \in \{0,1\}^*$, podemos computar en tiempo polinomial en $S(n)$ una fórmula $\varphi_{M,x}(\bar{s}, \bar{t})$ en CNF con variables libres $\bar{s} = s_1, \dots, s_{c \cdot S(|x|)}$, $\bar{t} = t_1, \dots, t_{c \cdot S(|x|)}$ tal que $\langle C \rangle \langle C' \rangle \models \varphi_{M,x}(\bar{s}, \bar{t})$ si y sólo si hay una flecha de C a C' en $G_{M,x}$.

Demo:

Similar al Teo. de Cook Levin

Idea de la demostración. Es parecido a lo que hicimos con la demostración del Teorema 11. En (14), construimos la fórmula booleana ψ_3 que expresaba que la mini-configuración z_i evolucionaba en un paso en z_{i+1} , donde $z_0 \dots z_m$ representaba un cálculo. Esa fórmula booleana predicaba sobre mini-configuraciones. Ahora tenemos configuraciones completas, es decir, con todo el contenido de todas las cintas, de tamaño $|C| = c \cdot S(|x|)$ y tenemos que expresar cuándo una configuración C , codificada con las variables \bar{t} , es la evolución en un paso de otra configuración C' , codificada con las variables \bar{s} .

Recordemos la codificación de las configuraciones de la demostración de la Proposición 17. Cada cinta de trabajo codifica simultáneamente su contenido y la posición de su cabeza. Supongamos que M tiene una sola cinta de trabajo. Alcanza con mirar qué pasa con la codificación de 3 bloques de 2 bits en la codificación de la cinta. Por ejemplo, supongamos que la cabeza de la cinta de entrada está leyendo b_e y que M está en el estado q . Si en la parte que codifica la cinta de trabajo de $\langle C \rangle$ (variables en \bar{s}) vemos una ventana $0b_1 \ 0 \ b_2$ (recordar que cada bit se codifica con una variable booleana en $\varphi_{M,x}$), con $b_i \in \{0,1\}$, sabemos que está leyendo el bit b_2 y entonces podemos determinar el contenido de la misma ventana pero en la codificación de $\langle C' \rangle$ (variables en \bar{t}) porque contamos con la información de δ y del símbolo leído en la entrada. Más en detalle, si $\delta(q, b_e, b_2) = (*, b_3, R, *, *)$ sabemos que la cabeza de la cinta en $\langle C' \rangle$ escribe b_3 y se mueve a la derecha; por lo tanto la ventana correspondiente en $\langle C' \rangle$ es $0b_1 \ 0b_3 \ 10$. Lo importante es que el comportamiento solo depende de la ventana, es decir, de variables proposicionales que participan en la descripción de esa ventana. De la misma forma, se puede expresar cómo cambia el estado, la posición de cabeza de la cinta de entrada (recordemos que no codificamos el contenido de la cinta de entrada) y el contenido y posición de la cabeza en la cinta de salida. La fórmula final tiene que ‘deslizar’ esas ventanas por todas las posiciones de las cintas en $\langle C \rangle$ y $\langle C' \rangle$. Así, la fórmula que buscamos se computa en tiempo polinomial en $S(n)$, y por lo tanto tiene tamaño también polinomial en $S(n)$. \square

Teorema 19. TQBF $\in \text{PSPACE-hard}$.

Demuestra. Sea $\mathcal{L} \in \text{PSPACE}$ y sea M una máquina determinística que decide \mathcal{L} en espacio $S(n)$, con S un polinomio. Consideramos el grafo de configuraciones $G_{M,x}$ de la Definición 21. Cada vértice $\langle C \rangle$ de $G_{M,x}$ es la codificación con $c \cdot S(n)$ bits de una configuración, donde $n = |x|$. Sea C_0 la configuración inicial y C_f la configuración final del cálculo de M a partir de x . Definimos fórmulas $\psi_i(\bar{s}, \bar{t})$ con variables libres \bar{s}, \bar{t} , donde \bar{s} y \bar{t} son tuplas de dimensión $c \cdot S(n)$ con la siguiente propiedad: $\langle C \rangle \langle C' \rangle \models \psi_i(\bar{s}, \bar{t})$ si existe un camino dirigido de longitud a lo sumo 2^i en $G_{M,x}$ entre C y C' . Si hay un camino de C a C' en $G_{M,x}$, entonces hay uno de longitud a lo sumo $2^{c \cdot S(n)}$, porque $G_{M,x}$ tiene a lo sumo $2^{c \cdot S(n)}$ vértices.

Luego, tenemos que

$$\begin{aligned} x \in \mathcal{L} &\text{ si } \langle C_0 \rangle \langle C_f \rangle \models \psi_{c \cdot S(n)}(\bar{s}, \bar{t}) \\ &\text{ si } \langle \psi_{c \cdot S(n)}(\bar{s}, \bar{t}), \langle C_0 \rangle \langle C_f \rangle \rangle \in \text{CHECKQBF}. \end{aligned} \quad (22)$$

Entonces $\mathcal{L} \leq_p \text{CHECKQBF} \leq_p \text{TQBF}$, siempre y cuando podamos definir $\psi_{c \cdot S(n)}$ en tiempo polinomial en n .

Definimos $\psi_i(\bar{s}, \bar{t})$ por recursión en i . Para $i = 0$, definimos $\psi_0(\bar{s}, \bar{t})$ como la fórmula $\bar{s} = \bar{t} \vee \varphi_{M,x}(\bar{s}, \bar{t})$, donde $\varphi_{M,x}(\bar{s}, \bar{t})$ es la fórmula de la Proposición 22 y $\bar{s} = \bar{t}$ abrevia $\bigwedge_{i=1}^{2^{c \cdot S(n)}} (s_i \wedge t_i) \vee (\neg s_i \wedge \neg t_i)$ si tomamos que $\bar{s} = s_1, \dots, s_{2^{c \cdot S(n)}}$ y $\bar{t} = t_1, \dots, t_{2^{c \cdot S(n)}}$. Entonces, computamos ψ_0 en tiempo polinomial en n . Para $i > 0$ podríamos definir

$$\psi_i(\bar{s}, \bar{t}) = \exists \bar{r} \psi_{i-1}(\bar{s}, \bar{r}) \wedge \psi_{i-1}(\bar{r}, \bar{t})$$

pero el tamaño de ψ_i sería exponencial en i (porque duplica su longitud en cada llamado recursivo) y por lo tanto no podríamos computarla en tiempo polinomial. Esta idea no nos sirve para probar la reducción polinomial (22) que buscamos. El truco es observar que existe un camino de longitud a lo sumo 2^j entre C y C' en el grafo de configuraciones $G_{M,x}$ si existe una configuración intermedia C'' tal que hay un camino de longitud a lo sumo 2^{i-1} entre C y C'' y uno de longitud a lo sumo 2^{j-1} entre C'' y C' . Así, definimos la fórmula

$$\psi_i(\bar{s}, \bar{t}) = \exists \bar{r} \forall \bar{u}, \bar{v} ((\bar{u} = \bar{s} \wedge \bar{v} = \bar{r}) \vee (\bar{u} = \bar{r} \wedge \bar{v} = \bar{t})) \rightarrow \psi_{i-1}(\bar{u}, \bar{v}) \quad (23)$$

En esta fórmula estamos abreviando $\phi \rightarrow \rho$ por $\neg \phi \vee \rho$ y $\bar{a} = \bar{b}$ por $\bigwedge_{i \leq k} (a_i \wedge b_i) \vee (\neg a_i \wedge \neg b_i)$, para $\bar{a} = a_1, \dots, a_k$ y $\bar{b} = b_1, \dots, b_k$. La QBF generalizada (23) expresa que hay una configuración intermedia \bar{r} entre \bar{s} y \bar{t} tal que vale ψ_{i-1} tanto si $\bar{u} = \bar{s}$ y $\bar{v} = \bar{r}$ como si $\bar{u} = \bar{r}$ y $\bar{v} = \bar{t}$, como se muestra a continuación:

$$\begin{array}{c} (\bar{u} = \bar{s} \wedge \bar{v} = \bar{r}) \rightarrow \psi_{i-1}(\bar{u}, \bar{v}) \\ \bar{s} \rightarrow \dots \rightarrow \bar{r} \rightarrow \dots \rightarrow \bar{t} \\ (\bar{u} = \bar{r} \wedge \bar{v} = \bar{t}) \rightarrow \psi_{i-1}(\bar{u}, \bar{v}) \end{array}$$

Llamemos $|\phi|$ al tamaño de la representación de la fórmula ϕ . Notemos que para $i \geq 1$, tenemos que $|\psi_i| \leq |\psi_{i-1}| + O(S(n))$. Para computar ψ_i necesitamos i llamados recursivos hasta llegar al caso base. Luego computamos $\psi_{c \cdot S(n)}$ en tiempo polinomial en $S(n)$, que implica polinomial en n . Notemos que $\psi_{c \cdot S(n)}$ es una QBF generalizada. Usando la Proposición 20, podemos llevarla a forma prenexa en tiempo polinomial. Entonces la equivalencia (22) efectivamente demuestra la reducción polinomial $\mathcal{L} \leq_p \text{CHECKQBF}$ que buscábamos. \square

Como consecuencia del Teorema 18 y del Teorema 19, obtenemos

Corolario 3. $\text{TQBF} \in \text{PSPACE-completo}$.

Teorero de Savitch

Teorema 20 (Savitch). $\text{NSPACE}(S(n)) \subseteq \text{SPACE}(S(n)^2)$, de modo que $\text{PSPACE} = \text{NPSPACE}$.

Demo:

Sea $\mathcal{L} \in \text{NSPACE}(S(n))$ q.vq $\mathcal{L} \in \text{Space}(S(n)^2)$.

Sea N no-det, de manera que:

$x \in \mathcal{L}$ si: $N(x) = 1$ si: hay un camino de longitud a lo sumo 2^i en $G_{N,x}$ desde C_0 a C_f .

Definimos Reach , una función recursiva tal q' $\text{Reach}(C, C', i) \in \{0, 1\}$:

$\text{Reach}(C, C', i) = 1$ si: hay un camino de a lo sumo 2^i en $G_{N,x}$ de C a C'

$x \in \mathcal{L}$ si: $\text{Reach}(C_0, C_f, c \cdot S(n))$

$\text{Reach}(C, C', i)$

Si: $i = 0$: ret 1 si $C \rightarrow C'$ están en $G_{N,x}$, 0 si no

Para cada posible configuración C'' de $G_{N,x}$: \rightarrow Todas las C'' ocupan $O(S(n))$

$j := \text{Reach}(C, C'', i-1)$ \rightarrow M fijo desde el inicio a un promedio
 $K := \text{Reach}(C'', C', i-1)$ \rightarrow M fijo desde el final a un promedio

Si: $j = K = 1$: ret 1

ret 0.

Sea $n = |x|$. $\text{Reach}(C, C', i)$ enumera las configuraciones C'' en espacio $O(S(n))$. Reusa el espacio para calcular j y K .

Entonces, $\text{Reach}(C, C', i)$ usa espacio $O(i \cdot S(n))$.

Luego, computamos $\text{Reach}(C_0, C_f, c \cdot S(n))$ en espacio $O(S(n)^2)$