Complejidad Computacional

Santiago Figueira

Departamento de Computación - FCEN - UBA

clase 6

Clase 6

La jerarquía de tiempos determinísticos

La jerarquía de tiempos no-determinísticos

Teorema de Ladner

La jerarquía de tiempos determinísticos

Clase 6

Notación de 'o chica'

Definición

Sean $f, g : \mathbb{N} \to \mathbb{N}$. Decimos que $\boldsymbol{f = o(g)}$ si para todo $\epsilon > 0$

$$f(n) \leq \epsilon \cdot g(n)$$

para todo n suficientemente grande o, equivalentemente, si

$$\lim_n f(n)/g(n) = 0.$$

Notar que si $f = o(g)$ entonces $f = O(g)$.

Notación de 'o chica'

Definición

Sean $f, g : \mathbb{N} \to \mathbb{N}$. Decimos que $\boldsymbol{f = o(g)}$ si para todo $\epsilon > 0$

$$f(n) \leq \epsilon \cdot g(n)$$

para todo n suficientemente grande o, equivalentemente, si

$$\lim_n f(n)/g(n) = 0.$$

Notar que si $f = o(g)$ entonces $f = O(g)$.

Ejemplos

- $4(n+2) \neq o(n)$ pero observar que $4(n+2) = O(n)$
- $4(n+2) = o(n^2)$
- $n^c = o(2^n)$ para todo $c \in \mathbb{N}$
- $k^n = o(2^{n^2})$ para todo $k \in \mathbb{N}$

Re-codificación de máquinas determinísticas

En clases pasadas

- codificamos cualquier máquina M con $i = \langle M \rangle \in \{0,1\}^*$.
  - M_i representaba la única máquina M con índice i, es decir $\langle M \rangle = i$.
- definimos la máquina universal U que recibía como parámetros $\langle i, x \rangle$ y simulaba $M_i(x)$.

Re-codificación de máquinas determinísticas

En clases pasadas

- codificamos cualquier máquina M con $i = \langle M \rangle \in \{0,1\}^*$.
  - M_i representaba la única máquina M con índice i, es decir $\langle M \rangle = i$.
- definimos la máquina universal U que recibía como parámetros $\langle i, x \rangle$ y simulaba $M_i(x)$.

En esta clase vamos a

- codificar cada máquina M con *infinitas* cadenas $\langle \langle M \rangle, z \rangle$ para cualquier $z \in \{0,1\}^*$
  - Cualquier $\langle \langle M \rangle, z \rangle$ representa M
- usar una máquina universal U que usa la *nueva* codificación de máquinas:
  - $U(\langle w, x \rangle)$ simula $M_w(x)$ como antes, solo que ahora w es un índice de la *nueva* codificación.

Re-codificación de máquinas determinísticas

- Lo importante es que para la nueva codificación

$$(M_i)_{i \in \{0,1\}^*}$$

vale que dada una máquina M existen infinitos i tal que $M = M_i$.

- Seguimos teniendo que

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.

La jerarquía de tiempos determinísticos

Teorema

Si f, g son construibles en tiempo y cumplen que

$$f(n) \cdot \log f(n) = o(g(n))$$

entonces $\mathbf{DTime}(f(n)) \subsetneq \mathbf{DTime}(g(n))$.

Ejemplo

$\mathbf{DTime}(n^c) \subsetneq \mathbf{DTime}(n^d)$ si $c < d$.

Demostración.

Definimos una máquina determinística D que con entrada x hace esto:

calcular $n = |x|$ *y* $t = g(n)$
simular $U(\langle x, x\rangle)$ *por* t *pasos*
si $U(\langle x, x\rangle)$ *no terminó en* $\leq t$ *pasos, devolver* 0
si no, devolver $1 - U(\langle x, x\rangle)$

Demostración.

Definimos una máquina determinística D que con entrada x hace esto:

calcular $n = |x|$ *y* $t = g(n)$
simular $U(\langle x, x\rangle)$ *por* t *pasos*
si $U(\langle x, x\rangle)$ *no terminó en* $\leq t$ *pasos, devolver* 0
si no, devolver $1 - U(\langle x, x\rangle)$

D corre en tiempo $O(g(n))$, de modo que $\mathcal{L}(D) \in \mathbf{DTime}(g(n))$.

Demostración.

Definimos una máquina determinística D que con entrada x hace esto:

> *calcular* $n = |x|$ *y* $t = g(n)$
> *simular* $U(\langle x, x\rangle)$ *por* t *pasos*
> *si* $U(\langle x, x\rangle)$ *no terminó en* $\leq t$ *pasos, devolver* 0
> *si no, devolver* $1 - U(\langle x, x\rangle)$

D corre en tiempo $O(g(n))$, de modo que $\mathcal{L}(D) \in \mathbf{DTime}(g(n))$.
Supongamos $\mathcal{L}(D) \in \mathbf{DTime}(f(n))$.

Demostración.

Definimos una máquina determinística D que con entrada x hace esto:

calcular $n = |x|$ *y* $t = g(n)$
simular $U(\langle x, x\rangle)$ *por* t *pasos*
si $U(\langle x, x\rangle)$ *no terminó en* $\leq t$ *pasos, devolver* 0
si no, devolver $1 - U(\langle x, x\rangle)$

D corre en tiempo $O(g(n))$, de modo que $\mathcal{L}(D) \in \mathbf{DTime}(g(n))$.
Supongamos $\mathcal{L}(D) \in \mathbf{DTime}(f(n))$.
Sea M una máquina determinística que decide $\mathcal{L}(D)$ en tiempo $c \cdot f(n)$. Existe n_0 tal que si $|x| \geq n_0$, entonces M con entrada x termina en $\leq c \cdot f(|x|)$ pasos.

Demostración.

Definimos una máquina determinística D que con entrada x hace esto:

calcular $n = |x|$ *y* $t = g(n)$
simular $U(\langle x, x\rangle)$ *por* t *pasos*
si $U(\langle x, x\rangle)$ *no terminó en* $\leq t$ *pasos, devolver* 0
si no, devolver $1 - U(\langle x, x\rangle)$

D corre en tiempo $O(g(n))$, de modo que $\mathcal{L}(D) \in \mathbf{DTime}(g(n))$.
Supongamos $\mathcal{L}(D) \in \mathbf{DTime}(f(n))$.
Sea M una máquina determinística que decide $\mathcal{L}(D)$ en tiempo $c \cdot f(n)$. Existe n_0 tal que si $|x| \geq n_0$, entonces M con entrada x termina en $\leq c \cdot f(|x|)$ pasos.
$U(\langle x, x\rangle)$ simula $M_x(x)$. Si $M_x(x)$ termina en $\leq c \cdot f(|x|)$ pasos, entonces $U(\langle x, x\rangle)$ necesita $\leq c' \cdot c \cdot f(|x|) \cdot \log(c \cdot f(|x|))$ pasos para terminar.

Demostración.

Definimos una máquina determinística D que con entrada x hace esto:

calcular $n = |x|$ *y* $t = g(n)$
simular $U(\langle x, x\rangle)$ *por* t *pasos*
si $U(\langle x, x\rangle)$ *no terminó en* $\leq t$ *pasos, devolver* 0
si no, devolver $1 - U(\langle x, x\rangle)$

D corre en tiempo $O(g(n))$, de modo que $\mathcal{L}(D) \in \mathbf{DTime}(g(n))$.
Supongamos $\mathcal{L}(D) \in \mathbf{DTime}(f(n))$.
Sea M una máquina determinística que decide $\mathcal{L}(D)$ en tiempo $c \cdot f(n)$. Existe n_0 tal que si $|x| \geq n_0$, entonces M con entrada x termina en $\leq c \cdot f(|x|)$ pasos.
$U(\langle x, x\rangle)$ simula $M_x(x)$. Si $M_x(x)$ termina en $\leq c \cdot f(|x|)$ pasos, entonces $U(\langle x, x\rangle)$ necesita $\leq c' \cdot c \cdot f(|x|) \cdot \log(c \cdot f(|x|))$ pasos para terminar. Existen $d, n_1 \geq n_0$ tal que para todo $n \geq n_1$

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

Tomemos x tal que $|x| \geq n_1$ y $M = M_x$.

Demostración.

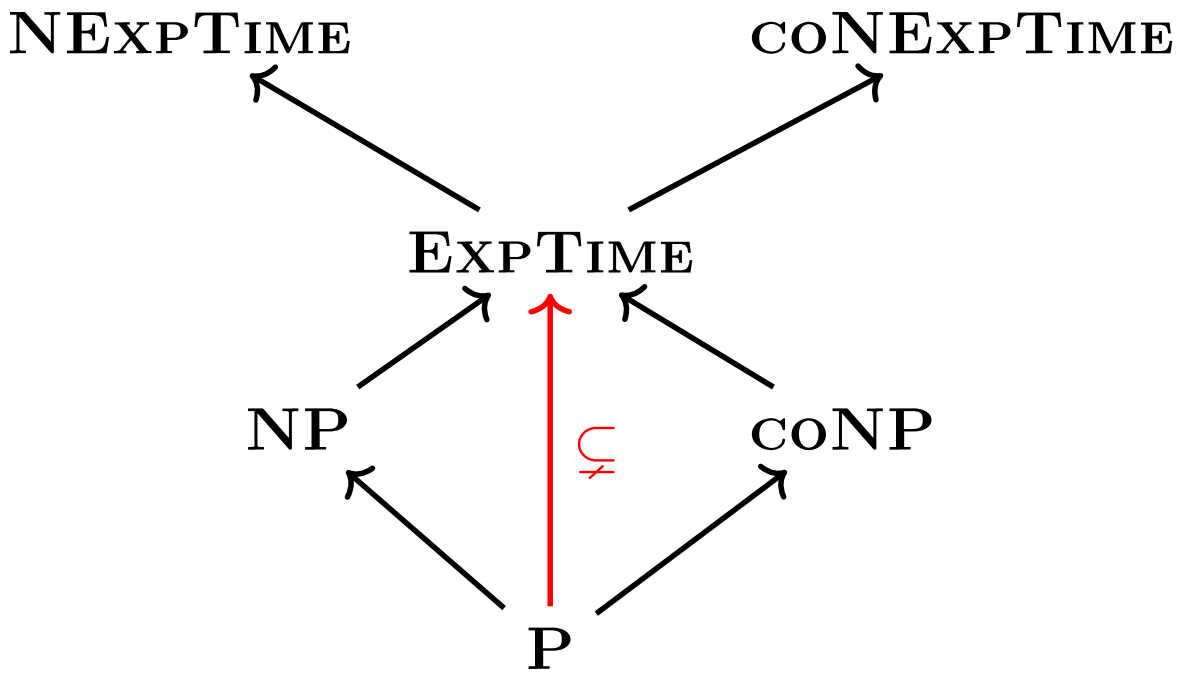
Definimos una máquina determinística D que con entrada x hace esto:

calcular $n = |x|$ *y* $t = g(n)$
simular $U(\langle x, x\rangle)$ *por* t *pasos*
si $U(\langle x, x\rangle)$ *no terminó en* $\leq t$ *pasos, devolver* 0
si no, devolver $1 - U(\langle x, x\rangle)$

D corre en tiempo $O(g(n))$, de modo que $\mathcal{L}(D) \in \mathbf{DTime}(g(n))$.
Supongamos $\mathcal{L}(D) \in \mathbf{DTime}(f(n))$.
Sea M una máquina determinística que decide $\mathcal{L}(D)$ en tiempo $c \cdot f(n)$. Existe n_0 tal que si $|x| \geq n_0$, entonces M con entrada x termina en $\leq c \cdot f(|x|)$ pasos.
$U(\langle x, x\rangle)$ simula $M_x(x)$. Si $M_x(x)$ termina en $\leq c \cdot f(|x|)$ pasos, entonces $U(\langle x, x\rangle)$ necesita $\leq c' \cdot c \cdot f(|x|) \cdot \log(c \cdot f(|x|))$ pasos para terminar. Existen $d, n_1 \geq n_0$ tal que para todo $n \geq n_1$

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

Tomemos x tal que $|x| \geq n_1$ y $M = M_x$.
$U(\langle x, x\rangle)$ termina en $\leq g(|x|)$ pasos y $D(x) = 1 - M_x(x) = 1 - M(x)$.
Pero $\mathcal{L}(D) = \mathcal{L}(M)$. Absurdo. □


NExpTime
coNExpTime
ExpTime
NP
coNP
P
⊊

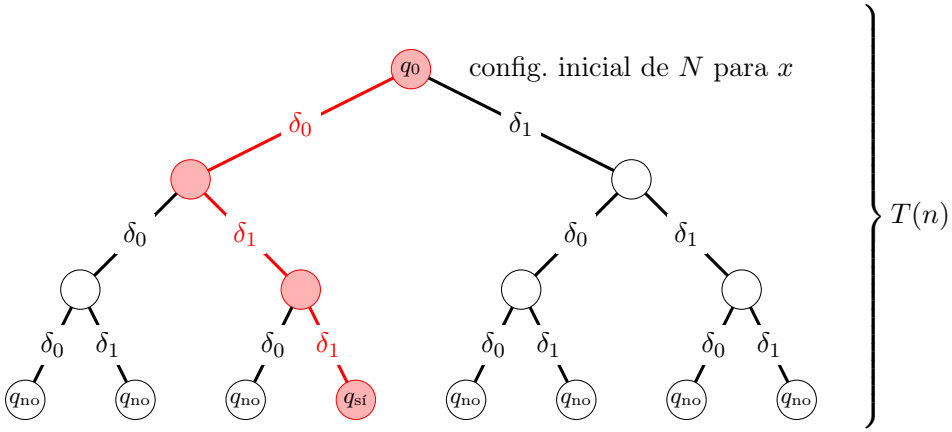
Diagonalización

Parecido a lo que hicimos para demostrar que *halt* no es computable.
Definimos una máquina D tal que

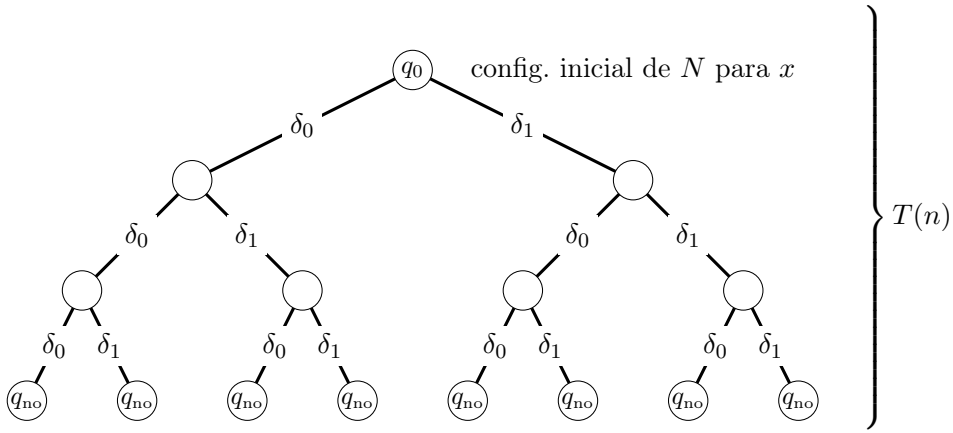
- D corre en tiempo $O(g(n))$
- D se porta distinto a todas las máquinas que corren en tiempo $O(f(n) \cdot \log f(n))$.
  $D(x)$ 'niega' la salida de $M_x(x)$:
  - cuando $M_x(x) = 1$, $D(x) = 0$
  - cuando $M_x(x) = 0$, $D(x) = 1$
- D puede *diagonalizar* en tiempo $O(n^3)$ y definir una función que no es computable por ninguna máquina en tiempo $O(n^2)$

Diagonalización

¿Cómo diagonalizamos una máquina no-determinística N?



N acepta x



N rechaza x

- ¿Podemos diagonalizar en tiempo $O(n^3)$ las máquinas no-deterministas que corren en tiempo $O(n^2)$?
- ¿Cómo 'negamos' la salida de una máquina no-determinística? Se conjetura que $\mathbf{NP} \neq \mathbf{coNP}$.
- Pero sí sabemos que si $\mathcal{L} \in \mathbf{NDTime}(T(n))$, entonces $\overline{\mathcal{L}} \in \mathbf{DTime}(2^{T(n)^c})$ para alguna constante c

La jerarquía de tiempos no-determinísticos

Jerga de cómputos en máquinas no-determinísticas

Recordar que un **cómputo** de una máquina no-determinística $N = (\Sigma, Q, \delta)$ a partir de $x \in \{0,1\}^*$ es una secuencia $C_0, \ldots, C_\ell$ de configuraciones tal que

1. C_0 es inicial a partir de x
2. C_{i+1} es la evolución de C_i en un paso dado por alguna de las 2 tuplas de δ.
3. C_ℓ está en estado $q_{\text{sí}}$ o q_{no}

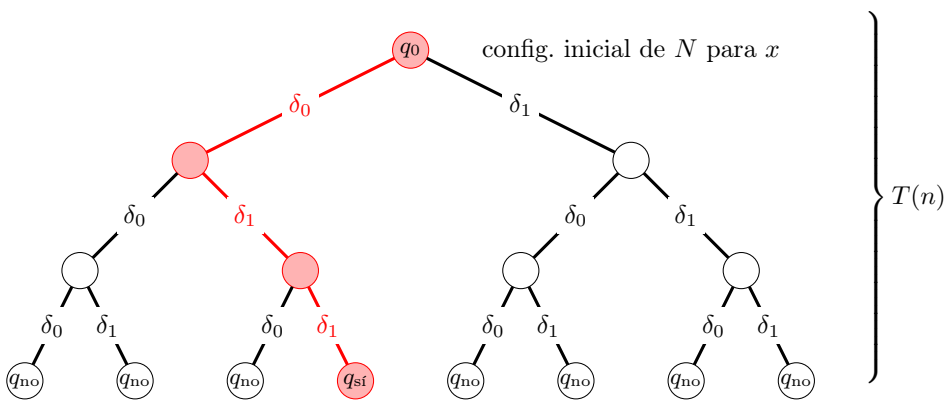
En particular, si existe tal cómputo, decimos que N con entrada x **terminó**.

Un **cómputo aceptador** es uno en el que C_ℓ está en estado $q_{\text{sí}}$.

Un **cómputo incompleto** es una secuencia $y = C_0, \ldots, C_\ell$ como la anterior donde valen 1 y 2 pero no vale 3. En este caso decimos que N **no terminó** siguiendo dicho cómputo.

Codificación de cómputos en máquinas no-deterministas

- Codificamos un cómputo de N con una secuencia $y \in \{0,1\}^*$ que representa las alternativas de δ.
- Si N es una máquina no-determinística que corre en tiempo $T(n)$, toda cadena $y_x \in \{0,1\}^{T(|x|)}$ representa un cómputo de N a partir de x
- Observar que secuencias más cortas que $T(|x|)$ son posiblemente cómputos incompletos.
- Por simplicidad suponemos que cualquier extensión de y_x codifica el mismo cómputo que y_x.



- N acepta x a través del cómputo 011 y también a través de los cómputos $0110, 0111, 01100, \ldots$
- 01 es un cómputo incompleto de N a partir x
- N todavía no terminó siguiendo el cómputo 01

Re-codificación de máquinas no-determinísticas

- Lo mismo que hicimos para las máquinas determinísticas
- Nueva codificación

  $$(N_i)_{i \in \{0,1\}^*}$$

  tal que para toda máquina no-determinística N existen infinitos i tal que $N = N_i$.
- Seguimos teniendo que

  Existe una máquina no-determinística NU que tal que NU acepta $(\langle i, x \rangle)$ sii N_i acepta x y si N_i corre en tiempo $T(n)$ entonces $NU(\langle i, x \rangle)$ decide si N_i acepta o rechaza x en $c \cdot T(|x|)$ pasos, donde c depende solo de i.
- A veces vamos a usar una enumeración $(N_i)_{i \in \mathbb{N}}$ de las máquinas no-determinísticas indexadas por números naturales:
  - para $i \in \mathbb{N}$, N_i representa N_z, donde z es la representación de i en binario sin el 1 inicial

La jerarquía de tiempos no-determinísticos

Teorema

Si f, g son construibles en tiempo, no decrecientes y cumplen que

$$f(n+1) = o(g(n)),$$

entonces $\mathbf{NDTime}(f(n)) \subsetneq \mathbf{NDTime}(g(n))$.

Ejemplo

$\mathbf{NDTime}(n^c) \subsetneq \mathbf{NDTime}(n^d)$ si $c < d$.

Demostración.

Sea $(N_i)_{i\in\mathbb{N}}$ una enumeración de todas las máquinas no-determinísticas. Definimos la máquina no-determinística N con entrada x así:

si x *no es de la forma* 1^i0y, *rechazar*
si no, supongamos $x = 1^i0y$
  si $|y| < g(i)$,
    simular no determinísticamente $N_i(x0)$ *y*
      $N_i(x1)$ *por* $g(|x|)$ *pasos* (*)
    si alguna no terminó: rechazar
    si las dos terminaron:
      aceptar sii $N_i(x0)$ *acepta y* $N_i(x1)$ *acepta*
  si $|y| = g(i)$,
    simular $N_i(1^i0)$ *siguiendo el cómputo*
      codificado por y (**)
    si no terminó: rechazar
    si terminó: aceptar sii y *es no aceptador*
  si $|y| > g(i)$, *rechazar*

Demostración.

Sea $(N_i)_{i\in\mathbb{N}}$ una enumeración de todas las máquinas no-deterministas. Definimos la máquina no-determinística N con entrada x así:

si x no es de la forma 1^i0y, rechazar
si no, supongamos $x = 1^i0y$
  si $|y| < g(i)$,
    simular no determinísticamente $N_i(x0)$ *y*
      $N_i(x1)$ *por $g(|x|)$ pasos* (*)
    si alguna no terminó: rechazar
    si las dos terminaron:
      aceptar sii $N_i(x0)$ acepta y $N_i(x1)$ acepta
  si $|y| = g(i)$,
    simular $N_i(1^i0)$ siguiendo el cómputo
      codificado por y (**)
    si no terminó: rechazar
    si terminó: aceptar sii y es no aceptador
  si $|y| > g(i)$, rechazar

Como es no-determinística, en (*) simulamos en tiempo $O(g(|x|))$.
En (**) simulamos en tiempo $O(|y|)$.
N corre en tiempo $O(g(n))$, luego $\mathcal{L}(N) \in \mathbf{NDTime}(g(n))$.

Supongamos que $\mathcal{L}(N) \in \mathbf{NDTime}(f(n))$
Sea N' una máquina no-determinística tal que

- $\mathcal{L}(N') = \mathcal{L}(N)$ y
- con entrada x suficientemente larga termina en $\leq c \cdot f(|x|)$ pasos.

Como $f(n+1) = o(g(n))$, para todo n suficientemente grande,

$$c \cdot f(n+1) \leq g(n).$$

Supongamos que $\mathcal{L}(N) \in \mathbf{NDTime}(f(n))$
Sea N' una máquina no-determinística tal que

- $\mathcal{L}(N') = \mathcal{L}(N)$ y
- con entrada x suficientemente larga termina en $\leq c \cdot f(|x|)$ pasos.

Como $f(n+1) = o(g(n))$, para todo n suficientemente grande,

$$c \cdot f(n+1) \leq g(n).$$

Tomar i suficientemente grande tal que

- $N' = N_i$ y
- para todo $x \in \{0,1\}^*$, si $|x| > i$ entonces $N_i(x0)$ y $N_i(x1)$ terminan en tiempo no-determinístico $g(|x|)$
  - $|x0| = |x1| = |x| + 1$ y N_i corre en tiempo $c \cdot f(n)$

Recordemos que N con entrada x hacía esto:

si x no es de la forma 1^i0y, rechazar
si no, supongamos $x = 1^i0y$
  si $|y| < g(i)$,
    simular no determinísticamente *$N_i(x0)$ y $N_i(x1)$ por $g(|x|)$ pasos*
    si alguna no terminó: rechazar
    si las dos terminaron:
      aceptar sii $N_i(x0)$ acepta y $N_i(x1)$ acepta
  si $|y| = g(i)$,
    simular $N_i(1^i0)$ siguiendo el cómputo codificado por y
    si no terminó: rechazar
    si terminó: aceptar sii y es no aceptador
  si $|y| > g(i)$, rechazar

N_i es tal que $\mathcal{L}(N_i) = \mathcal{L}(N)$ y corre en tiempo $c \cdot f(n)$
Para $x = 1^i0y$, $N_i(x0)$ y $N_i(x0)$ terminan en $c \cdot f(|x| + 1) \leq g(|x|)$ pasos. $N_i(1^i0)$ termina en $c \cdot f(i + 1) \leq g(i)$ pasos.

Recordemos que N con entrada x hacía esto:

si x no es de la forma 1^i0y, rechazar
si no, supongamos $x = 1^i0y$
  si $|y| < g(i)$,
    simular no determinísticamente $N_i(x0)$ y $N_i(x1)$ por $g(|x|)$ pasos
    si alguna no terminó: rechazar
    si las dos terminaron:
      aceptar sii $N_i(x0)$ acepta y $N_i(x1)$ acepta
  si $|y| = g(i)$,
    simular $N_i(1^i0)$ siguiendo el cómputo codificado por y
    si no terminó: rechazar
    si terminó: aceptar sii y es no aceptador
  si $|y| > g(i)$, rechazar

N_i es tal que $\mathcal{L}(N_i) = \mathcal{L}(N)$ y corre en tiempo $c \cdot f(n)$
Para $x = 1^i0y$, $N_i(x0)$ y $N_i(x0)$ terminan en $c \cdot f(|x| + 1) \leq g(|x|)$ pasos. $N_i(1^i0)$ termina en $c \cdot f(i + 1) \leq g(i)$ pasos.

Recordemos que N con entrada x hacía esto:

si x no es de la forma 1^i0y, rechazar
si no, supongamos $x = 1^i0y$
  si $|y| < g(i)$,
    simular no determinísticamente $N_i(x0)$ y $N_i(x1)$ por $g(|x|)$ pasos
    si alguna no terminó: rechazar
    si las dos terminaron:
      aceptar sii $N_i(x0)$ acepta y $N_i(x1)$ acepta
  si $|y| = g(i)$,
    simular $N_i(1^i0)$ siguiendo el cómputo codificado por y
    si no terminó: rechazar
    si terminó: aceptar sii y es no aceptador
  si $|y| > g(i)$, rechazar

N_i es tal que $\mathcal{L}(N_i) = \mathcal{L}(N)$ y corre en tiempo $c \cdot f(n)$
Para $x = 1^i0y$, $N_i(x0)$ y $N_i(x0)$ terminan en $c \cdot f(|x| + 1) \leq g(|x|)$ pasos. $N_i(1^i0)$ termina en $c \cdot f(i + 1) \leq g(i)$ pasos.

Recordemos que N con entrada x hacía esto:

si x no es de la forma 1^i0y, rechazar
si no, supongamos $x = 1^i0y$
  si $|y| < g(i)$,
    simular no determinísticamente $N_i(x0)$ y $N_i(x1)$ por $g(|x|)$ pasos
    si alguna no terminó: rechazar
    si las dos terminaron:
    aceptar sii $N_i(x0)$ acepta y $N_i(x1)$ acepta
  si $|y| = g(i)$,
    simular $N_i(1^i0)$ siguiendo el cómputo codificado por y
    si no terminó: rechazar
    si terminó: aceptar sii y es no aceptador
si $|y| > g(i)$, rechazar

N_i es tal que $\mathcal{L}(N_i) = \mathcal{L}(N)$ y corre en tiempo $c \cdot f(n)$
Para $x = 1^i0y$, $N_i(x0)$ y $N_i(x0)$ terminan en $c \cdot f(|x|+1) \leq g(|x|)$ pasos. $N_i(1^i0)$ termina en $c \cdot f(i+1) \leq g(i)$ pasos.

$1^i0 \in \mathcal{L}(N)$ sii $\forall y \in \{0,1\}^1\ 1^i0y \in \mathcal{L}(N_i)$

sii $\forall y \in \{0,1\}^2\ 1^i0y \in \mathcal{L}(N_i)$

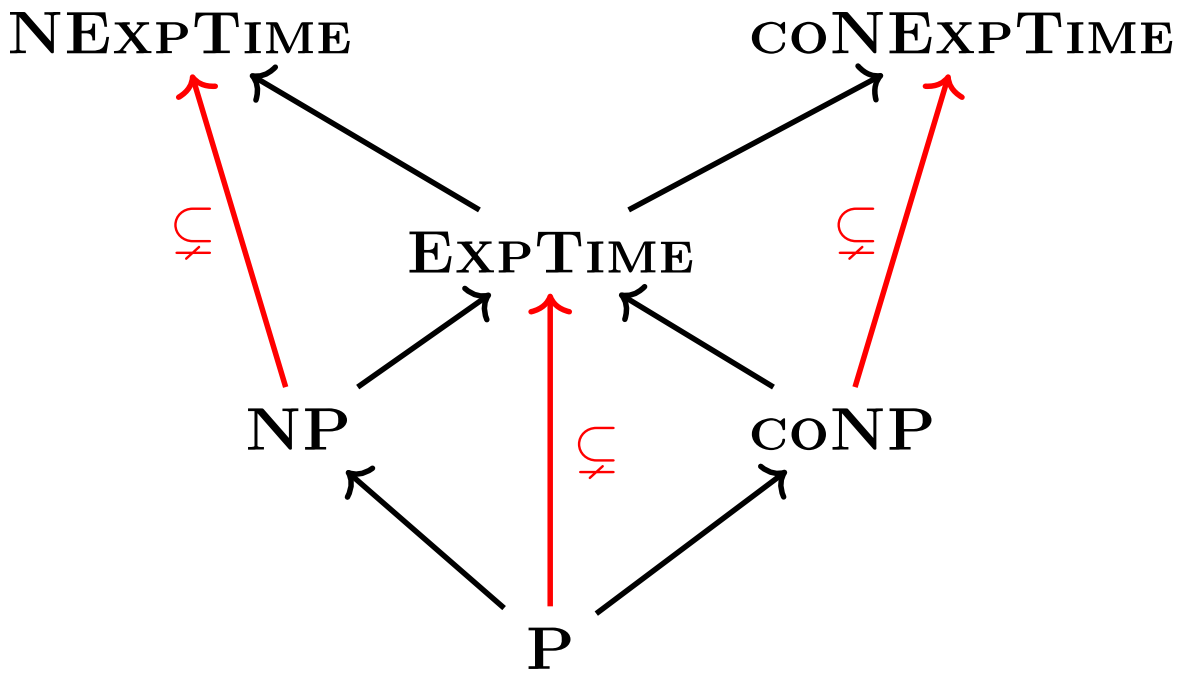
$\ldots$

sii $\forall y \in \{0,1\}^{g(i)}\ 1^i0y \in \mathcal{L}(N_i)$

sii $N_i(1^i0)$ rechaza siguiendo todos los cómputos y tal que $|y| = g(i)$

sii $1^i0 \notin \mathcal{L}(N_i) = \mathcal{L}(N)$

Absurdo. □


NExpTime
coNExpTime
ExpTime
NP
coNP
P
⊊
⊊
⊊

Teorema de Ladner

Problemas **NP**-intermedios

Muchos problemas **NP** terminan siendo **P** o **NP-completos**.
¿Les pasa esto a todos los problemas **NP**?

Teorema (Ladner)

Si $\mathbf{P} \neq \mathbf{NP}$ entonces existe $\mathcal{L}$ tal que $\mathcal{L} \in \mathbf{NP}$, $\mathcal{L} \notin \mathbf{P}$, y $\mathcal{L} \notin \mathbf{NP\text{-}completo}$.

Demostración.

Como siempre, sea M_i la i-ésima máquina determinística.
Para simplificar, escribimos ψ en lugar de $\langle\psi\rangle$.

Demostración.

Como siempre, sea M_i la i-ésima máquina determinística.
Para simplificar, escribimos ψ en lugar de $\langle\psi\rangle$.

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

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

Demostración.

Como siempre, sea M_i la i-ésima máquina determinística.
Para simplificar, escribimos ψ en lugar de $\langle\psi\rangle$.

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

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

- cuando H crece relativamente rápido, por ejemplo $H(n) = n$, entonces $\mathsf{SAT}_H \in \mathbf{P}$: si $x_\psi = \psi 01^{|\psi|^{|\psi|}}$ entonces $|x_\psi| \geq 2^{|\psi|}$ para toda ψ salvo finitos casos. Dada x_ψ decidimos si ψ es satisfacible en tiempo $O(2^{|\psi|}) = O(|x_\psi|)$, o sea tiempo lineal.

Demostración.

Como siempre, sea M_i la i-ésima máquina determinística.
Para simplificar, escribimos ψ en lugar de $\langle\psi\rangle$.

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

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

- cuando H crece relativamente rápido, por ejemplo $H(n) = n$, entonces $\mathsf{SAT}_H \in \mathbf{P}$: si $x_\psi = \psi 01^{|\psi|^{|\psi|}}$ entonces $|x_\psi| \geq 2^{|\psi|}$ para toda ψ salvo finitos casos. Dada x_ψ decidimos si ψ es satisfacible en tiempo $O(2^{|\psi|}) = O(|x_\psi|)$, o sea tiempo lineal.
- cuando H es constante, $\mathsf{SAT}_H \in$ **NP-completo**.

En la demostración que sigue, vamos a elegir una H adecuada: H tiende a infinito pero crece lentamente

Definimos H así:

$$H(n) = \begin{array}{l} \text{mínimo } i < \log\log n \text{ tq } \forall x \in \{0,1\}^{\leq \log n}, \\ M_i(x) = \chi_{\mathsf{SAT}_H}(x) \text{ en } \leq i \cdot |x|^i \text{ pasos; o} \\ \log\log n \text{ si no existe tal } i \end{array}$$

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

Definimos H así:

$$H(n) = \begin{array}{l} \text{mínimo } i < \log\log n \text{ tq } \forall x \in \{0,1\}^{\leq \log n}, \\ M_i(x) = \chi_{\mathsf{SAT}_H}(x) \text{ en } \leq i \cdot |x|^i \text{ pasos; o} \\ \log\log n \text{ si no existe tal } i \end{array}$$

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

SAT_H y H están definidas por recursión mutua, pero están bien definidas:

- $H(n)$ usa $\mathsf{SAT}_H(x)$ para palabras x de longitud $\leq \log n$
- $x \in \mathsf{SAT}_H$ depende de $H(n)$ para $n < |x|$
  (aquí $x = \psi 01^{n^{H(n)}}$, con $n = |\psi|$)

Definimos H así:

$$H(n) = \begin{array}{l} \text{mínimo } i < \log\log n \text{ tq } \forall x \in \{0,1\}^{\leq \log n}, \\ M_i(x) = \chi_{\mathsf{SAT}_H}(x) \text{ en } \leq i \cdot |x|^i \text{ pasos; o} \\ \log\log n \text{ si no existe tal } i \end{array}$$

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

SAT_H y H están definidas por recursión mutua, pero están bien definidas:

- $H(n)$ usa $\mathsf{SAT}_H(x)$ para palabras x de longitud $\leq \log n$
- $x \in \mathsf{SAT}_H$ depende de $H(n)$ para $n < |x|$
  (aquí $x = \psi 01^{n^{H(n)}}$, con $n = |\psi|$)

Ejercicio

H es computable en tiempo $O(n^3)$.

Definimos H así:

$$H(n) = \begin{array}{l} \text{mínimo } i < \log\log n \text{ tq } \forall x \in \{0,1\}^{\leq \log n}, \\ M_i(x) = \chi_{\mathsf{SAT}_H}(x) \text{ en } \leq i \cdot |x|^i \text{ pasos; o} \\ \log\log n \text{ si no existe tal } i \end{array}$$

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

SAT_H y H están definidas por recursión mutua, pero están bien definidas:

- $H(n)$ usa $\mathsf{SAT}_H(x)$ para palabras x de longitud $\leq \log n$
- $x \in \mathsf{SAT}_H$ depende de $H(n)$ para $n < |x|$
  (aquí $x = \psi 01^{n^{H(n)}}$, con $n = |\psi|$)

Ejercicio

H es computable en tiempo $O(n^3)$.

Entonces $\mathsf{SAT}_H \in \mathbf{NP}$.

$$H(n) = \begin{array}{l} \text{mínimo } i < \log\log n \text{ tq } \forall x \in \{0,1\}^{\leq \log n}, \\ M_i(x) = \chi_{\mathsf{SAT}_H}(x) \text{ en } \leq i \cdot |x|^i \text{ pasos; o} \\ \log\log n \text{ si no existe tal } i \end{array}$$

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

$$H(n) = \begin{array}{l} \text{mínimo } i < \log\log n \text{ tq } \forall x \in \{0,1\}^{\leq \log n}, \\ M_i(x) = \chi_{\mathsf{SAT}_H}(x) \text{ en } \leq i \cdot |x|^i \text{ pasos; o} \\ \log\log n \text{ si no existe tal } i \end{array}$$

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

Proposición 1

$\mathsf{SAT}_H \in \mathbf{P} \Longrightarrow \exists c\, \forall n\ H(n) \leq c$

$$H(n) = \begin{array}{l} \text{mínimo } i < \log\log n \text{ tq } \forall x \in \{0,1\}^{\leq \log n}, \\ M_i(x) = \chi_{\mathsf{SAT}_H}(x) \text{ en } \leq i \cdot |x|^i \text{ pasos; o} \\ \log\log n \text{ si no existe tal } i \end{array}$$

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

Proposición 1

$\mathsf{SAT}_H \in \mathbf{P} \Longrightarrow \exists c\, \forall n\ H(n) \leq c$

Demostración.

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

$$H(n) \leq \text{máx}\left(\{H(m) : m < 2^{2^i}\} \cup \{i\}\right).$$

□

$$H(n) = \begin{array}{l} \text{mínimo } i < \log\log n \text{ tq } \forall x \in \{0,1\}^{\leq \log n}, \\ M_i(x) = \chi_{\mathsf{SAT}_H}(x) \text{ en } \leq i \cdot |x|^i \text{ pasos; o} \\ \log\log n \text{ si no existe tal } i \end{array}$$

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

$$H(n) = \begin{array}{l} \text{mínimo } i < \log\log n \text{ tq } \forall x \in \{0,1\}^{\leq \log n}, \\ M_i(x) = \chi_{\mathsf{SAT}_H}(x) \text{ en } \leq i \cdot |x|^i \text{ pasos; o} \\ \log\log n \text{ si no existe tal } i \end{array}$$

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

Proposición 2

$\exists c\, \exists^\infty n\ H(n) \leq c \Longrightarrow \mathsf{SAT}_H \in \mathbf{P}$.

$$H(n) = \begin{array}{l} \text{mínimo } i < \log\log n \text{ tq } \forall x \in \{0,1\}^{\leq \log n}, \\ M_i(x) = \chi_{\mathsf{SAT}_H}(x) \text{ en } \leq i \cdot |x|^i \text{ pasos; o} \\ \log\log n \text{ si no existe tal } i \end{array}$$

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

Proposición 2

$\exists c\ \exists^\infty n\ H(n) \leq c \Longrightarrow \mathsf{SAT}_H \in \mathbf{P}$.

Demostración.

Supongamos $\exists c\ \exists^\infty n\ H(n) \leq c$. Entonces $\exists i\ \exists^\infty n\ H(n) = i$.
Veamos que M_i acepta SAT_H en tiempo $i \cdot n^i$: Sup. que M_i no acepta SAT_H. Existe x tal que $M_i(x) \neq \chi_{\mathsf{SAT}_H}(x)$.
Entonces $\forall n > 2^{|x|}\ H(n) \neq i$. Absurdo.
Similar caso en que M_i no corre en tiempo $i \cdot n^i$. □

$$H(n) = \begin{array}{l} \text{mínimo } i < \log\log n \text{ tq } \forall x \in \{0,1\}^{\leq \log n}, \\ M_i(x) = \chi_{\mathsf{SAT}_H}(x) \text{ en } \leq i \cdot |x|^i \text{ pasos; o} \\ \log\log n \text{ si no existe tal } i \end{array}$$

$$\mathsf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\}$$

Proposición 2

$\exists c\ \exists^\infty n\ H(n) \leq c \Longrightarrow \mathsf{SAT}_H \in \mathbf{P}$.

Demostración.

Supongamos $\exists c\ \exists^\infty n\ H(n) \leq c$. Entonces $\exists i\ \exists^\infty n\ H(n) = i$.
Veamos que M_i acepta SAT_H en tiempo $i \cdot n^i$: Sup. que M_i no acepta SAT_H. Existe x tal que $M_i(x) \neq \chi_{\mathsf{SAT}_H}(x)$.
Entonces $\forall n > 2^{|x|}\ H(n) \neq i$. Absurdo.
Similar caso en que M_i no corre en tiempo $i \cdot n^i$. □

Contra-recíproco: $\mathsf{SAT}_H \notin \mathbf{P} \Longrightarrow \lim_{n\to\infty} H(n) = \infty$

Veamos que $\mathsf{SAT}_H \notin \mathbf{P}$:

Supongamos $\mathsf{SAT}_H \in \mathbf{P}$. Por la Proposición 1, $\exists c\ \forall n\ H(n) \leq c$.

$$\begin{aligned}\mathsf{SAT}_H &= \{\psi 01^{n^{H(n)}} : \psi \in \mathsf{SAT}, n = |\psi|\} \\ &= \{\psi 0 \underbrace{1 \ldots\ldots\ldots 1}_{|\psi|^{H(|\psi|)} \leq |\psi|^c} : \psi \in \mathsf{SAT}\}\end{aligned}$$

Pero entonces un algoritmo de tiempo polinomial para SAT_H se puede usar para SAT, por lo que $\mathsf{SAT} \in \mathbf{P}$. Como SAT es **NP-completo** concluimos que $\mathbf{P} = \mathbf{NP}$. Absurdo.

Veamos que $\mathsf{SAT}_H \notin$ **NP-completo**:

Supongamos $\mathsf{SAT}_H \in$ **NP-completo**. Hay una reducción polinomial f de SAT a SAT_H:

$$\begin{array}{lll} \psi \in \mathsf{SAT} & \text{sii} & f(\psi) \in \mathsf{SAT}_H \\ & \text{sii} & f(\psi) \text{ es de la forma } F_\psi 01^{|F_\psi|^{H(|F_\psi|)}} \text{ con } F_\psi \in \mathsf{SAT}. \end{array}$$

Supongamos que f es computable en tiempo $c \cdot n^c$ por una cierta máquina determinística. Como esa máquina no puede escribir más que $c \cdot n^c$ símbolos en la salida,

$$|f(x)| \leq c \cdot |x|^c$$

Luego

$$\begin{aligned} |f(\psi)| &= |F_\psi 01^{|F_\psi|^{H(|F_\psi|)}}| \\ &= |F_\psi| + 1 + |F_\psi|^{H(|F_\psi|)} \leq c \cdot |\psi|^c \end{aligned}$$

Sabemos:

- $\psi \in \mathsf{SAT}$ sii $f(\psi) = F_\psi 01^{|F_\psi|^{H(|F_\psi|)}}$ con $F_\psi \in \mathsf{SAT}$
- $|F_\psi| + 1 + |F_\psi|^{H(|F_\psi|)} \leq c \cdot |\psi|^c$

Sabemos:

- $\psi \in \mathsf{SAT}$ sii $f(\psi) = F_\psi 01^{|F_\psi|^{H(|F_\psi|)}}$ con $F_\psi \in \mathsf{SAT}$
- $|F_\psi| + 1 + |F_\psi|^{H(|F_\psi|)} \leq c \cdot |\psi|^c$

Proposición 3

Existe k tal que para todo ψ, $|\psi| > k \Longrightarrow |F_\psi| < |\psi|$.

Sabemos:

- $\psi \in \mathsf{SAT}$ sii $f(\psi) = F_\psi 01^{|F_\psi|^{H(|F_\psi|)}}$ con $F_\psi \in \mathsf{SAT}$
- $|F_\psi| + 1 + |F_\psi|^{H(|F_\psi|)} \leq c \cdot |\psi|^c$

Proposición 3

Existe k tal que para todo ψ, $|\psi| > k \Longrightarrow |F_\psi| < |\psi|$.

Demostración.

Supongamos que no. Para todo k, existe ψ tal que

$$|\psi| > k \quad \text{y} \quad |F_\psi| \geq |\psi| > k.$$

Sabemos:

- $\psi \in \mathsf{SAT}$ sii $f(\psi) = F_\psi 01^{|F_\psi|^{H(|F_\psi|)}}$ con $F_\psi \in \mathsf{SAT}$
- $|F_\psi| + 1 + |F_\psi|^{H(|F_\psi|)} \leq c \cdot |\psi|^c$

Proposición 3

Existe k tal que para todo ψ, $|\psi| > k \Longrightarrow |F_\psi| < |\psi|$.

Demostración.

Supongamos que no. Para todo k, existe ψ tal que

$$|\psi| > k \quad \text{y} \quad |F_\psi| \geq |\psi| > k.$$

Como $\mathsf{SAT}_H \notin \mathbf{P}$, por Prop. 2, $\lim_{n\to\infty} H(n) = \infty$.

Sabemos:

- $\psi \in \mathsf{SAT}$ sii $f(\psi) = F_\psi 01^{|F_\psi|^{H(|F_\psi|)}}$ con $F_\psi \in \mathsf{SAT}$
- $|F_\psi| + 1 + |F_\psi|^{H(|F_\psi|)} \leq c \cdot |\psi|^c$

Proposición 3

Existe k tal que para todo ψ, $|\psi| > k \Longrightarrow |F_\psi| < |\psi|$.

Demostración.

Supongamos que no. Para todo k, existe ψ tal que

$$|\psi| > k \quad \text{y} \quad |F_\psi| \geq |\psi| > k.$$

Como $\mathsf{SAT}_H \notin \mathbf{P}$, por Prop. 2, $\lim_{n\to\infty} H(n) = \infty$.
Entonces $\exists k' \; \forall n > k' \; n^{H(n)} > c \cdot n^c$

Sabemos:

- $\psi \in \mathsf{SAT}$ sii $f(\psi) = F_\psi 01^{|F_\psi|^{H(|F_\psi|)}}$ con $F_\psi \in \mathsf{SAT}$
- $|F_\psi| + 1 + |F_\psi|^{H(|F_\psi|)} \leq c \cdot |\psi|^c$

Proposición 3

Existe k tal que para todo ψ, $|\psi| > k \Longrightarrow |F_\psi| < |\psi|$.

Demostración.

Supongamos que no. Para todo k, existe ψ tal que

$$|\psi| > k \quad \text{y} \quad |F_\psi| \geq |\psi| > k.$$

Como $\mathsf{SAT}_H \notin \mathbf{P}$, por Prop. 2, $\lim_{n\to\infty} H(n) = \infty$.
Entonces $\exists k' \; \forall n > k' \; n^{H(n)} > c \cdot n^c$
Existe ψ tal que $|F_\psi| \geq |\psi| > k'$. Luego

$$c \cdot |\psi|^c \leq c \cdot |F_\psi|^c < |F_\psi|^{H(|F_\psi|)} \leq c \cdot |\psi|^c$$

Absurdo. □

Algoritmo recursivo G para decidir $\psi \in \mathsf{SAT}$:

entrada: ψ
si $|\psi| \leq k$,
  devolver 'sí' si $\psi \in \mathsf{SAT}$ *y 'no' en caso contrario*
    (finitos casos porque k es constante)
si no,
  computar $f(\psi)$
  si $f(\psi)$ *no es de la forma* $F_\psi 01^{|F_\psi|^{H(|F_\psi|)}}$,
    devolver 'no'
  si no, devolver $G(F_\psi)$ (por Prop. 3, $|F_\psi| < |\psi|$)

Notar que G corre en tiempo polinomial y

$$\psi \in \mathsf{SAT} \qquad \text{sii} \qquad G(\psi) \in \mathsf{SAT}$$

Entonces $\mathbf{P} = \mathbf{NP}$. Absurdo. □