Complejidad Computacional

Santiago Figueira

Departamento de Computación - FCEN - UBA

clase 7

Clase 7

Espacio usado por un cómputo

Espacio polinomial: **PSpace** y **NPSpace**

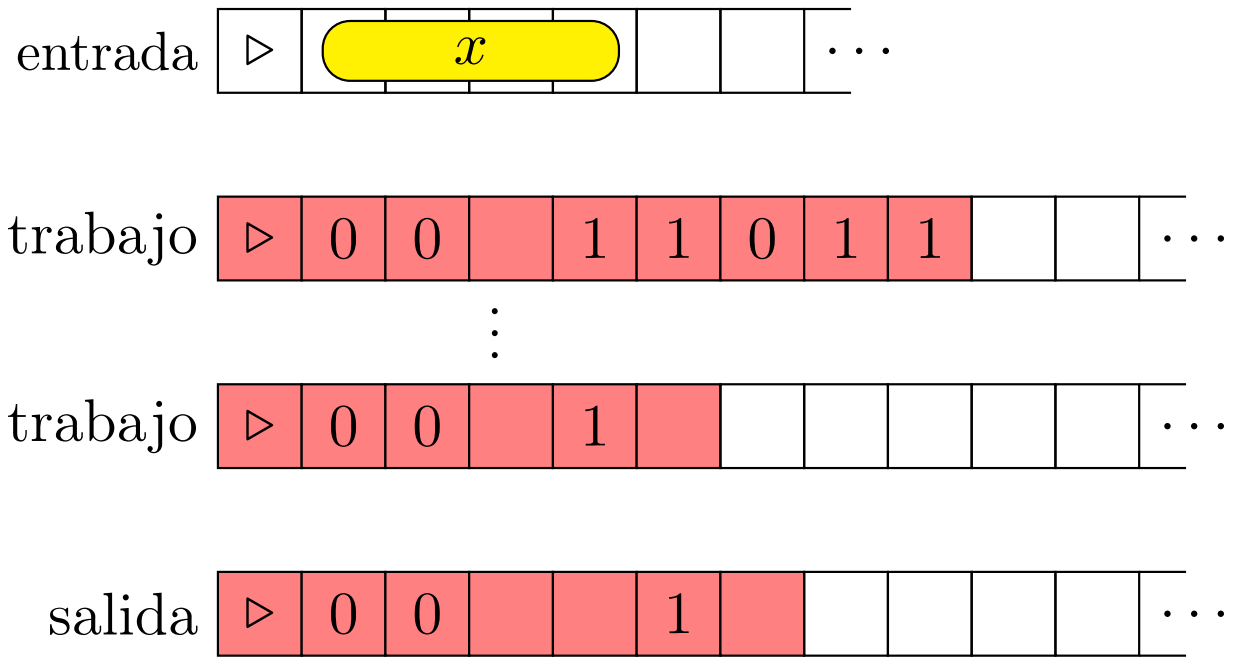
Teorema de Savitch

Espacio usado por un cómputo

Clase 7

- Espacio usado por un cómputo
- Espacio polinomial: **PSpace** y **NPSpace**
- Teorema de Savitch

Espacio en una máquina determinística



Definición

Dada una máquina determinística M y $x \in \{0,1\}^*$ el **espacio** que usa M con entrada x es la cantidad de celdas por las que alguna vez pasó la cabeza en las cintas de trabajo y de salida a lo largo del cómputo de M a partir de x.

Espacio en una máquina no-determinística

Definición

Dada una máquina no-determinística N y $x \in \{0,1\}^*$ el **espacio** que usa N con entrada x es la cantidad de celdas por las que alguna vez pasó la cabeza en las cintas de trabajo y de salida a lo largo de *todos* los cómputos de N a partir de x.

Espacio usado en una máquina

Definición

La máquina M (determinística o no-determinística)

- **usa espacio $S(n)$** si para toda entrada x, el espacio que usa M con entrada x es a lo sumo $S(|x|)$
- **usa espacio $O(S(n))$** si existe una constante c tal que para todo x, salvo finitos, M con entrada x usa espacio $c \cdot S(|x|)$

Definición

Una función f es **computable en espacio** $S(n)$, $O(S(n))$ si existe una máquina determinística que computa f y usa espacio $S(n)$, $O(S(n))$.

$\textsc{Space}(S(n))$ y $\textsc{NSpace}(S(n))$

Clase de complejidad: $\textsc{Space}(S(n))$, $\textsc{NSpace}(S(n))$

$\textsc{Space}(S(n))$ es la clase de lenguajes $\mathcal{L}$ tal que existe una máquina determinística M tal que

- M decide $\mathcal{L}$
- M usa espacio $O(S(n))$

$\textsc{NSpace}(S(n))$ es la clase de lenguajes $\mathcal{L}$ tal que existe una máquina no-determinística N tal que

- N decide $\mathcal{L}$
- N usa espacio $O(S(n))$

Funciones construibles en espacio

Definición

Una función $S : \mathbb{N} \to \mathbb{N}$ es **construible en espacio** si la función $1^n \mapsto [S(n)]$ es computable en espacio $O(S(n))$

Funciones construibles en espacio

Definición

Una función $S : \mathbb{N} \to \mathbb{N}$ es **construible en espacio** si la función $1^n \mapsto [S(n)]$ es computable en espacio $O(S(n))$

Ejemplos de funciones construibles en espacio

$\log n$, n, n^2, 2^n

- No eran interesantes las funciones $T(n) < n$ cuando medíamos el tiempo, porque se espera que las máquinas al menos lean la entrada y eso lleva tiempo n.
- Cuando medimos espacio, la cosa cambia. Como no medimos el espacio de la entrada, tiene sentido estudiar funciones de espacio $S(n) < n$.
- Pero a veces vamos a requerir que $S(n) \geq \log n$, dado que queremos poder 'recordar' cualquier *índice* de la cinta de entrada.
    - La entrada tiene largo n, entonces representamos cualquier $i \leq n$ con $\leq \log n$ bits.

$\mathbf{DTime}(S(n)) \subseteq \mathbf{Space}(S(n)) \subseteq \mathbf{NSpace}(S(n))$

Proposición

$\mathbf{DTime}(S(n)) \subseteq \mathbf{Space}(S(n))$.

Demostración.

En cada paso, una máquina determinística solo puede usar una celda más de cada cinta. Si M corre en tiempo $O(S(n))$, usa espacio $O(S(n))$. □

$\mathbf{DTime}(S(n)) \subseteq \mathbf{Space}(S(n)) \subseteq \mathbf{NSpace}(S(n))$

Proposición

$\mathbf{DTime}(S(n)) \subseteq \mathbf{Space}(S(n))$.

Demostración.

En cada paso, una máquina determinística solo puede usar una celda más de cada cinta. Si M corre en tiempo $O(S(n))$, usa espacio $O(S(n))$. □

Proposición

$\mathbf{Space}(S(n)) \subseteq \mathbf{NSpace}(S(n))$.

Grafo de configuraciones

Definición

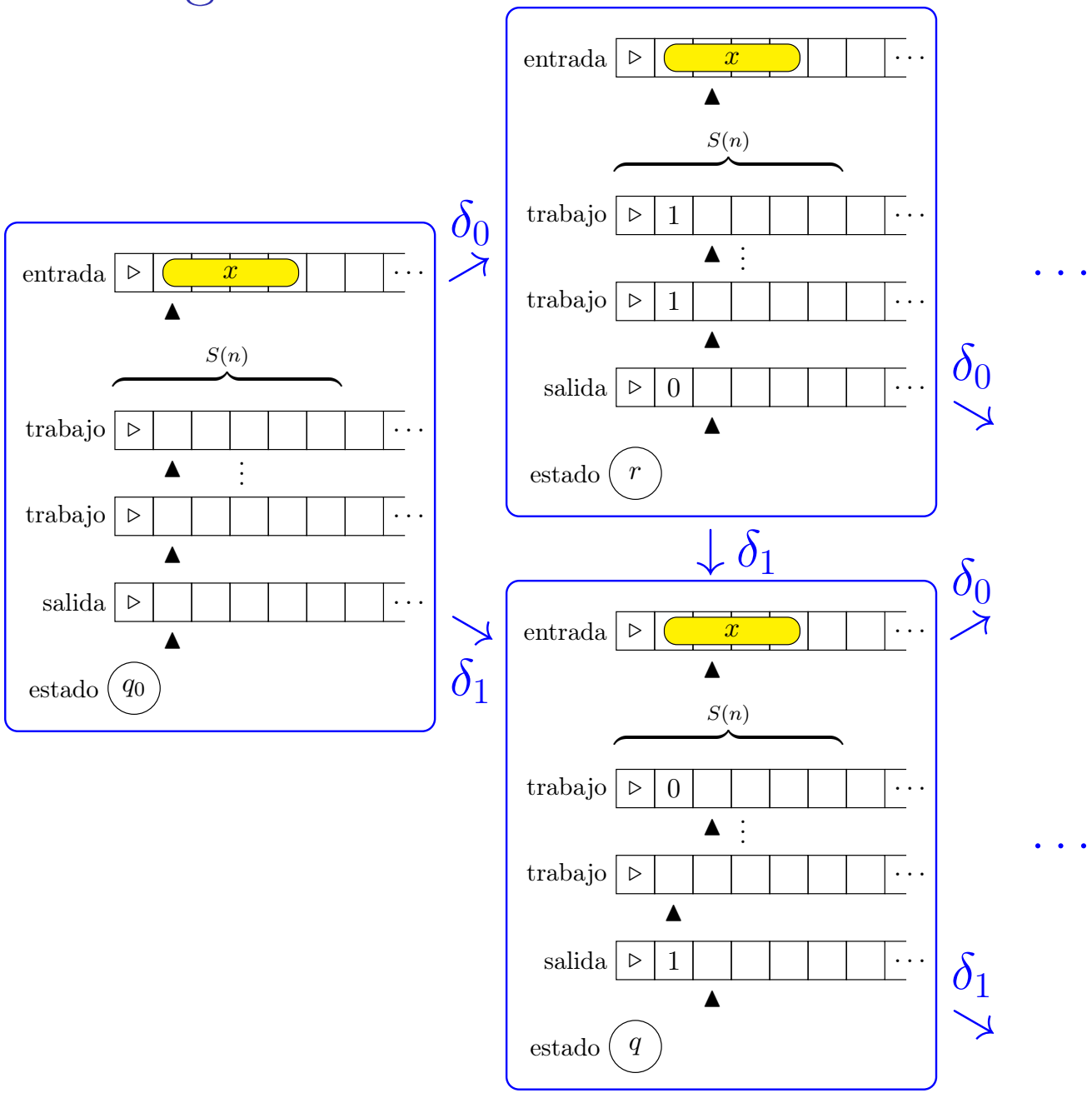
Sea $S : \mathbb{N} \to \mathbb{N}$ y sea M una máquina (determinística o no-determinística) que usa espacio $S(n)$. El **grafo de configuraciones** de M con entrada x, notado $G_{M,x}$ es un grafo dirigido tal que

- el conjunto de vértices son las configuraciones de hasta $S(|x|)$ celdas en cada cinta de trabajo y en la de salida
- hay una arista de C a C' si C' es una evolución en un paso de C dado por M

Observación

Si M es una máquina determinística o no-determinística, M acepta x si y solo si existe un camino en $G_{M,x}$ desde la configuración inicial de M para x hasta alguna configuración final.

Grafo de configuraciones



Grafo de configuraciones

Como siempre, estamos interesados en máquinas que aceptan lenguajes.

- si M es determinística
  - $G_{M,x}$ tiene *out-degree* 1
  - podemos suponer que M borra todas las cintas de trabajo y deja todas las cabezas en la primera celda antes de entrar a q_f. Así hay un *único* estado final $C_{\text{sí}}$ aceptador: es el que tiene el estado en q_f y en la cinta de salida hay un 1 seguido de blancos
- si N es no-determinística
  - $G_{N,x}$ tiene *out-degree* ≤ 2
  - podemos suponer que N borra todas las cintas de trabajo y salida y deja todas las cabezas en la primera celda antes de entrar a $q_{\text{sí}}$. Así hay un único estado final $C_{\text{sí}}$ aceptador.

Codificación de configuraciones

Proposición

Sea M una máquina (determinística o no-determinística) que usa espacio $S(n)$ tal que $S(n) \geq \log n$. Existe una constante c tal que para todo x, cada vértice de $G_{M,x}$ se puede describir usando $c \cdot S(|x|)$ bits (c depende de M). Luego, $G_{M,x}$ tiene $2^{c \cdot S(|x|)}$ nodos.

Demostración.

Sea $M = (\Sigma, Q, \delta)$ con k cintas de trabajo.

- numeramos los estados de Q y usamos la codificación usual (codificamos q con $\langle q \rangle$

Demostración.

Sea $M = (\Sigma, Q, \delta)$ con k cintas de trabajo.

- numeramos los estados de Q y usamos la codificación usual (codificamos q con $\langle q \rangle$
- contenido de i-ésima cinta de trabajo T_i:
  - contenido: $\triangleright\ b_1^i\ b_2^i\ \ldots b_{S(|x|)}^i$ (codificado en binario)
  - posición cabeza: $0 \leq j_i \leq S(|x|)$
  - codificación: $\langle T_i \rangle = 0b_1^i \ldots 0b_{j_i-1}^i\ 10\ 0b_{j_i}^i \ldots 0b_{S(|x|)}^i$

Demostración.

Sea $M = (\Sigma, Q, \delta)$ con k cintas de trabajo.

- numeramos los estados de Q y usamos la codificación usual (codificamos q con $\langle q \rangle$
- contenido de i-ésima cinta de trabajo T_i:
  - contenido: $\triangleright\ b_1^i\ b_2^i\ \ldots b_{S(|x|)}^i$ (codificado en binario)
  - posición cabeza: $0 \leq j_i \leq S(|x|)$
  - codificación: $\langle T_i \rangle = 0b_1^i \ldots 0b_{j_i-1}^i\ 10\ 0b_{j_i}^i \ldots 0b_{S(|x|)}^i$
- configuración de la cinta de salida Z: ídem, $\langle Z \rangle$

Demostración.

Sea $M = (\Sigma, Q, \delta)$ con k cintas de trabajo.

- numeramos los estados de Q y usamos la codificación usual (codificamos q con $\langle q \rangle$
- contenido de i-ésima cinta de trabajo T_i:
  - contenido: $\triangleright\ b_1^i\ b_2^i\ \ldots b_{S(|x|)}^i$ (codificado en binario)
  - posición cabeza: $0 \leq j_i \leq S(|x|)$
  - codificación: $\langle T_i \rangle = 0b_1^i \ldots 0b_{j_i-1}^i\ 10\ 0b_{j_i}^i \ldots 0b_{S(|x|)}^i$
- configuración de la cinta de salida Z: ídem, $\langle Z \rangle$
- configuración de la cinta de entrada E:
  - solo guardamos la *posición* p de la cabeza (en binario): $\langle E \rangle = \langle p \rangle$. No usamos la misma configuración que antes porque podría ser $n > S(n)$

Demostración.

Sea $M = (\Sigma, Q, \delta)$ con k cintas de trabajo.

- numeramos los estados de Q y usamos la codificación usual (codificamos q con $\langle q \rangle$
- contenido de i-ésima cinta de trabajo T_i:
  - contenido: $\triangleright\ b_1^i\ b_2^i\ \ldots b_{S(|x|)}^i$ (codificado en binario)
  - posición cabeza: $0 \leq j_i \leq S(|x|)$
  - codificación: $\langle T_i \rangle = 0b_1^i \ldots 0b_{j_i-1}^i\ 10\ 0b_{j_i}^i \ldots 0b_{S(|x|)}^i$
- configuración de la cinta de salida Z: ídem, $\langle Z \rangle$
- configuración de la cinta de entrada E:
  - solo guardamos la *posición* p de la cabeza (en binario): $\langle E \rangle = \langle p \rangle$. No usamos la misma configuración que antes porque podría ser $n > S(n)$

Sea $n = |x|$. Cada C se codifica con $\langle C \rangle = \langle \langle q \rangle, \langle E \rangle, \langle T_1 \rangle, \ldots, \langle T_k \rangle, \langle Z \rangle \rangle$.

$$|C| = d \cdot ((k+1) \cdot S(n) + \log n) = c \cdot S(n).$$

Cada $C \in \{0,1\}^*$, luego hay $2^{c \cdot S(n)}$ posibles configuraciones (para c dependiente de M pero independiente de x). □

$\mathbf{NSpace}(S(n)) \subseteq \mathbf{DTime}(2^{O(S(n))})$

Proposición

Si S es construible en espacio,
$\mathbf{NSpace}(S(n)) \subseteq \mathbf{DTime}(2^{O(S(n))})$.

$\mathbf{NSpace}(S(n)) \subseteq \mathbf{DTime}(2^{O(S(n))})$

Proposición

Si S es construible en espacio,
$\mathbf{NSpace}(S(n)) \subseteq \mathbf{DTime}(2^{O(S(n))})$.

Demostración.

Sea N una máquina no-determinística que decide $\mathcal{L}$ en espacio $O(S(n))$. Definimos la máquina determinística M con entrada x:

> *Construir $G_{N,x}$ (toma tiempo $2^{O(S(n))}$). Usar BFS para decidir si existe un camino entre la configuración inicial de N a partir de x y la (única) configuración final. Si existe tal camino, escribir 1 en la salida; si no escribir 0. Luego pasar a q_f.*

Para cualquier grafo $G = (V, E)$, BFS corre en tiempo $O(|V| + |E|)$. Como el *out-degree* de cada vértice de $G_{N,x}$ es ≤ 2, la cantidad de aristas de $G_{N,x}$ es $O(2^{O(S(|x|))})$. Luego BFS sobre $G_{N,x}$ toma tiempo $O(2^{O(S(|x|))})$.
Entonces $\mathcal{L}(M) = \mathcal{L}$ y M corre en tiempo $O(2^{O(S(n))})$. □

Espacio polinomial: **PSpace** y **NPSpace**

Clase 7

- Espacio usado por un cómputo
- Espacio polinomial: **PSpace** y **NPSpace**
- Teorema de Savitch

PSpace, NPSpace

Clase de complejidad: PSpace y NPSpace

$$\mathbf{PSpace} = \bigcup_{c>0} \mathbf{Space}(n^c)$$

$$\mathbf{NPSpace} = \bigcup_{c>0} \mathbf{NSpace}(n^c)$$

Observación

$\mathbf{PSpace} \subseteq \mathbf{NPSpace} \subseteq \mathbf{ExpTime}$.

NP ⊆ PSpace

Proposición

NP ⊆ PSpace.

NP ⊆ PSpace

Proposición

NP ⊆ PSpace.

Demostración.

Supongamos $\mathcal{L} \in \mathbf{NP}$. Existe una máquina determinística M y un polinomio $p : \mathbb{N} \to \mathbb{N}$ tal que M corre en tiempo polinomial y para todo x:

$$x \in \mathcal{L} \quad \text{sii} \quad \text{existe } u \in \{0,1\}^{p(|x|)} \text{ tal que } M(\langle x, u \rangle) = 1$$

NP ⊆ PSpace

Proposición

NP ⊆ PSpace.

Demostración.

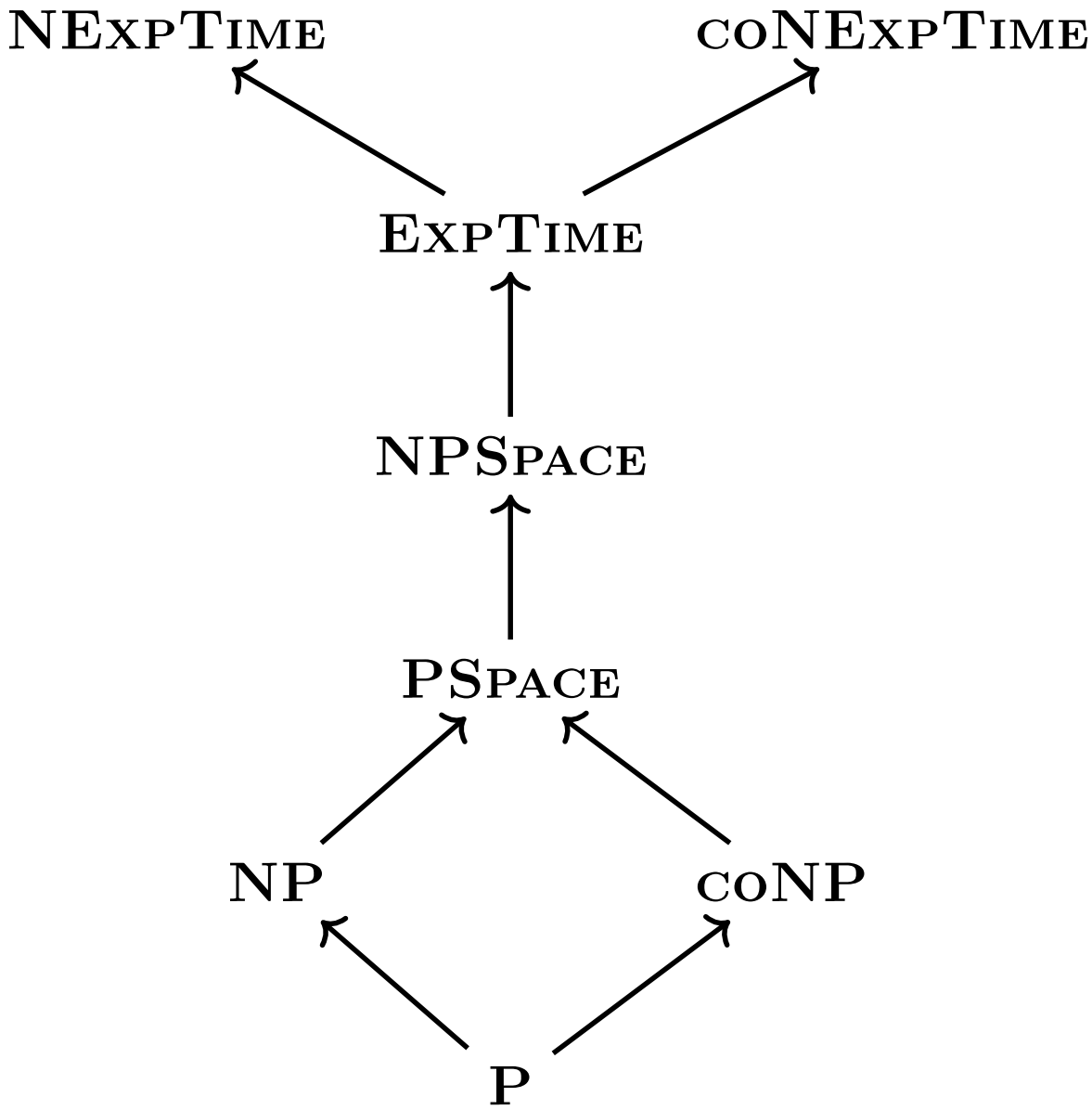
Supongamos $\mathcal{L} \in \mathbf{NP}$. Existe una máquina determinística M y un polinomio $p : \mathbb{N} \to \mathbb{N}$ tal que M corre en tiempo polinomial y para todo x:

$$x \in \mathcal{L} \quad \text{sii} \quad \text{existe } u \in \{0,1\}^{p(|x|)} \text{ tal que } M(\langle x, u\rangle) = 1$$

Definimos una máquina determinística M' que con entrada x hace esto:

> *Simula $M(\langle x, u\rangle)$ para cada $u \in \{0,1\}^{p(|x|)}$. Cada u lo escribe en las* mismas *$p(|x|)$ celdas. Si encuentra u tal que $M(\langle x, u\rangle) = 1$, escribe 1 en la salida. Si no, escribe 0. Luego pasa a q_f.*

M' corre en tiempo exponencial pero usa espacio $p(|x|)$ más el que use M con entrada $\langle x, u\rangle$. Luego $\mathcal{L} \in \mathbf{PSpace}$. □


NExpTime
coNExpTime
ExpTime
NPSpace
PSpace
NP
coNP
P

La jerarquía de espacios

Teorema

Si f, g son construibles en espacio y cumplen que

$$f(n) = o(g(n))$$

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

Ejercicio

Demostrarlo. Misma idea que para la jerarquía de tiempos determinísticos, solo que podemos simular cualquier máquina con un un factor constante de costo extra en lugar de logarítmico.

Fórmulas booleanas cuantificadas (QBF)

Definición

Una **fórmula booleana cuantificada** (o **QBF**) es una expresión de la forma

$$Q_1 x_1 \; Q_2 x_2 \ldots Q_n x_n \; \varphi(x_1, x_2, \ldots, x_n)$$

donde

- Q_i es $\forall$ o $\exists$
- $\varphi(x_1, x_2, \ldots, x_n)$ es una fórmula booleana con variables entre $x_1, \ldots, x_n$ y posiblemente con constantes $0, 1$

Los cuantificadores $\forall$ o $\exists$ tienen la semántica usual de 'para todo' y 'existe', pero las variables x_i son *booleanas*, es decir solo pueden tomar valor 0 (falso) o 1 (verdadero).

Las QBF están en forma *prenexa* (cuantificadores adelante)

Fórmulas booleanas cuantificadas (QBF)

Ejemplo

$$\psi = \forall x_1 \exists x_2 \forall x_3 \ (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3)$$

es falsa: si x_1 es verdadera, entonces x_3 tiene que ser verdadera, pero x_3 está cuantificada universalmente.

Fórmulas booleanas cuantificadas (QBF)

El problema TQBF

Notar que todas las variables de una QBF están cuantificadas (se llaman *sentencias*), entonces son verdaderas o falsas independiente de cualquier asignación de variables.

Notación: $\models$

Sea ψ una QBF. Notamos

- $\models \psi$ cuando ψ es verdadera
- $\not\models \psi$ cuando ψ es falsa

Problema: TQBF (*True Quantified Boolean Formula*)

$$\mathsf{TQBF} = \{\langle \psi \rangle : \psi \text{ es una QBF tal que } \models \psi\}$$

Fórmulas booleanas cuantificadas (QBF)

Eliminación de cuantificadores

Supongamos la QBF

$$\psi = Q_1 x_1 \overbrace{Q_2 x_2 \dots Q_n x_n \ \varphi(x_1, x_2, \dots, x_n)}^{\rho}$$

Para $b \in \{0, 1\}$, definimos $\psi \restriction_{x_1=b}$ como el resultado de reemplazar en ρ todas las ocurrencias de x_1 por b.

Ejemplo

$(\forall y \exists x \ y \wedge x) \restriction_{y=1} = \exists x \ 1 \wedge x.$

- si $Q_1 = \forall$ entonces $\models \psi$ sii $\models \psi \restriction_{x_1=0}$ y $\models \psi \restriction_{x_1=1}$
- si $Q_1 = \exists$ entonces $\models \psi$ sii $\models \psi \restriction_{x_1=0}$ o $\models \psi \restriction_{x_1=1}$

TQBF $\in$ PSPACE

Teorema

TQBF $\in$ **PSPACE**.

Demostración.

Definimos una función recursiva F con

- entrada: (codificación de) una QBF

$$\psi = Q_1 x_1 \; Q_2 x_2 \ldots Q_n x_n \; \varphi(x_1, x_2, \ldots, x_n)$$

- salida: 1 si $\models \psi$ y 0 en caso contrario.

Demostración.

Definimos una función recursiva F con

- entrada: (codificación de) una QBF

$$\psi = Q_1 x_1 \; Q_2 x_2 \ldots Q_n x_n \; \varphi(x_1, x_2, \ldots, x_n)$$

- salida: 1 si $\models \psi$ y 0 en caso contrario.

si $n = 0$, ψ no tiene variables (solo tiene constantes)
decidir si es verdadera o falsa *(tiempo $O(|\psi|)$)*
si no, ψ es de la forma $\psi = Qx \ldots$
computar $i = F(\psi \restriction_{x=0})$ y $j = F(\psi \restriction_{x=1})$
(se reutiliza el espacio en los dos llamados)
solo nos quedamos con los bits i, j
si $Q = \forall$, devolver 1 si $i = j = 1$; si no, 0
si $Q = \exists$, devolver 1 si $i = 1$ o $j = 1$; si no, 0

Demostración.

Definimos una función recursiva F con

- entrada: (codificación de) una QBF

$$\psi = Q_1 x_1 \; Q_2 x_2 \ldots Q_n x_n \; \varphi(x_1, x_2, \ldots, x_n)$$

- salida: 1 si $\models \psi$ y 0 en caso contrario.

si $n = 0$, ψ no tiene variables (solo tiene constantes)
decidir si es verdadera o falsa *(tiempo $O(|\psi|)$)*
si no, ψ es de la forma $\psi = Qx \ldots$
computar $i = F(\psi \restriction_{x=0})$ y $j = F(\psi \restriction_{x=1})$
(se reutiliza el espacio en los dos llamados)
solo nos quedamos con los bits i, j
si $Q = \forall$, devolver 1 si $i = j = 1$; si no, 0
si $Q = \exists$, devolver 1 si $i = 1$ o $j = 1$; si no, 0

Necesita espacio $O(|\psi|)$ para escribir $\psi \restriction_{x=b}$ y en cada llamado decrementamos el tamaño de la entrada. En total usa espacio $O(|\psi|^2)$. □

Fórmulas booleanas cuantificadas (QBF)

QBFs con variables libres

Podemos extender las QBFs con variables libres:

$$\psi(y_1, \ldots, y_m) = Q_1 x_1 \; Q_2 x_2 \ldots Q_n x_n \; \varphi(x_1, x_2, \ldots, x_n, y_1, \ldots, y_m)$$

donde

- Q_i es $\forall$ o $\exists$
- $\psi(x_1, x_2, \ldots, x_n, y_1, \ldots, y_m)$ es una fórmula booleana con variables entre $x_1, \ldots, x_n, y_1, \ldots, y_m$
- $y_1, \ldots, y_m$ son variables libres de ψ
- $x_1, \ldots, x_n$ son variables ligadas de ψ

Fórmulas booleanas cuantificadas (QBF)

QBFs con variables libres

Podemos extender las QBFs con variables libres:

$$\psi(y_1, \ldots, y_m) = Q_1 x_1 \; Q_2 x_2 \ldots Q_n x_n \; \varphi(x_1, x_2, \ldots, x_n, y_1, \ldots, y_m)$$

donde

- Q_i es $\forall$ o $\exists$
- $\psi(x_1, x_2, \ldots, x_n, y_1, \ldots, y_m)$ es una fórmula booleana con variables entre $x_1, \ldots, x_n, y_1, \ldots, y_m$
- $y_1, \ldots, y_m$ son variables libres de ψ
- $x_1, \ldots, x_n$ son variables ligadas de ψ

Cuando hay variables libres, la verdad o falsedad depende de una valuación $\{y_1, \ldots, y_m\} \to \{0, 1\}$. Como con las CNF:

- representamos valuaciones $\{y_1, \ldots, y_m\} \to \{0, 1\}$ con palabras v en $\{0, 1\}^m$.
- notamos $v \models \psi(y_1, \ldots, y_m)$ cuando ψ es verdadera para v
- notamos $v \not\models \psi(y_1, \ldots, y_m)$ cuando ψ es falsa para v

Fórmulas booleanas cuantificadas (QBF)

Sin la restricción de forma prenexa

Ejemplo

$$\psi(y_1, y_2) = \exists x\ (x \leftrightarrow y_1) \wedge (x \leftrightarrow y_2)$$

(acá $u \leftrightarrow v$ es una abreviatura para $(u \wedge v) \vee (\neg u \wedge \neg v)$)
$00 \models \psi(y_1, y_2)$, $11 \models \psi(y_1, y_2)$, $01 \not\models \psi(y_1, y_2)$

Podemos considerar también una gramática más general:

$$\varphi ::= x \mid 0 \mid 1 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \forall x\ \varphi \mid \exists x\ \varphi$$

con la semántica esperada. Llamémoslas **QBF generalizadas**

Fórmulas booleanas cuantificadas (QBF)

Sin la restricción de forma prenexa

Ejemplo

$$\psi(y_1, y_2) = \exists x \ (x \leftrightarrow y_1) \wedge (x \leftrightarrow y_2)$$

(acá $u \leftrightarrow v$ es una abreviatura para $(u \wedge v) \vee (\neg u \wedge \neg v)$)
$00 \models \psi(y_1, y_2)$, $11 \models \psi(y_1, y_2)$, $01 \not\models \psi(y_1, y_2)$

Podemos considerar también una gramática más general:

$$\varphi ::= x \mid 0 \mid 1 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \forall x \ \varphi \mid \exists x \ \varphi$$

con la semántica esperada. Llamémoslas **QBF generalizadas**

No son QBFs porque no están en forma prenexa.

Ejemplo

$$\psi(y) = \exists x_1 \ (\forall x_2 \ (y \wedge \neg x_1 \wedge x_2) \vee \forall x_3 \ (\neg y \vee x_1 \vee x_3))$$

no es una QBF pero es una QBF generalizada.

Fórmulas booleanas cuantificadas (QBF)

Sin la restricción de forma prenexa

Proposición

En tiempo polinomial se puede transformar cualquier QBF generalizada $\psi(y_1, \ldots, y_m)$ en una QBF $\psi'(y_1, \ldots, y_m)$ equivalente, es decir, tal que para todo $v \in \{0,1\}^m$

$$v \models \psi(y_1, \ldots, y_m) \qquad \text{sii} \qquad v \models \psi'(y_1, \ldots, y_m)$$

Ejercicio

Demostrarlo. Idea: renombrar variables ligadas para no repetir y aplicar reglas:

$$\neg \forall x\ \psi = \exists x\ \neg\psi$$

$$\neg \exists x\ \psi = \forall x\ \neg\psi$$

$$\psi * (Qx\ \varphi(x)) = Qx\ (\varphi * \psi)$$

$(Q \in \{\forall, \exists\}, * \in \{\wedge, \vee\}, \psi$ no contiene a x libre$)$

Chequeo de QBF generalizada

Problema: CHECKQBF (*Chequeo de QBF generalizada*)

$$\mathsf{CHECKQBF} = \{\langle \psi, v \rangle : \begin{array}{l} \psi \text{ es una QBF generalizada con } m \text{ variables} \\ \text{libres, } v \in \{0,1\}^m \text{ y } v \models \psi \end{array} \}$$

Chequeo de QBF generalizada

Problema: CHECKQBF (*Chequeo de QBF generalizada*)

$$\mathsf{CHECKQBF} = \{\langle \psi, v \rangle : \begin{array}{l} \psi \text{ es una QBF generalizada con } m \text{ variables} \\ \text{libres, } v \in \{0,1\}^m \text{ y } v \models \psi \end{array} \}$$

Proposición

$\mathsf{CHECKQBF} \leq_{\mathrm{p}} \mathsf{TQBF}$.

Chequeo de QBF generalizada

Problema: CHECKQBF (*Chequeo de QBF generalizada*)

$$\mathsf{CHECKQBF} = \{\langle \psi, v \rangle : \begin{array}{l} \psi \text{ es una QBF generalizada con } m \text{ variables} \\ \text{libres, } v \in \{0,1\}^m \text{ y } v \models \psi \end{array} \}$$

Proposición

$\mathsf{CHECKQBF} \leq_{\mathrm{p}} \mathsf{TQBF}$.

Demostración.

Sea ψ una QBF generalizada con variables libres $y_1, \ldots, y_m$ y sea $v \in \{0,1\}^m$.

Chequeo de QBF generalizada

Problema: CHECKQBF (*Chequeo de QBF generalizada*)

$$\mathsf{CHECKQBF} = \{\langle \psi, v \rangle : \begin{array}{l} \psi \text{ es una QBF generalizada con } m \text{ variables} \\ \text{libres, } v \in \{0,1\}^m \text{ y } v \models \psi \end{array} \}$$

Proposición

$\mathsf{CHECKQBF} \leq_{\mathrm{p}} \mathsf{TQBF}$.

Demostración.

Sea ψ una QBF generalizada con variables libres $y_1, \ldots, y_m$ y sea $v \in \{0,1\}^m$. Primero llevamos ψ a una ψ' equivalente pero en forma prenexa (tiempo polinomial).

Chequeo de QBF generalizada

Problema: CHECKQBF (*Chequeo de QBF generalizada*)

$$\mathsf{CHECKQBF} = \{\langle \psi, v \rangle : \begin{array}{l} \psi \text{ es una QBF generalizada con } m \text{ variables} \\ \text{libres, } v \in \{0,1\}^m \text{ y } v \models \psi \end{array} \}$$

Proposición

$\mathsf{CHECKQBF} \leq_{\mathrm{p}} \mathsf{TQBF}$.

Demostración.

Sea ψ una QBF generalizada con variables libres $y_1, \ldots, y_m$ y sea $v \in \{0,1\}^m$. Primero llevamos ψ a una ψ' equivalente pero en forma prenexa (tiempo polinomial). Luego definimos ψ'' como el resultado de reemplazar cada variable y_j en ψ' por la constante $v(j)$;

Chequeo de QBF generalizada

Problema: CHECKQBF (*Chequeo de QBF generalizada*)

$$\mathsf{CHECKQBF} = \{\langle \psi, v \rangle : \begin{array}{l} \psi \text{ es una QBF generalizada con } m \text{ variables} \\ \text{libres, } v \in \{0,1\}^m \text{ y } v \models \psi \end{array} \}$$

Proposición

$\mathsf{CHECKQBF} \leq_{\mathrm{p}} \mathsf{TQBF}$.

Demostración.

Sea ψ una QBF generalizada con variables libres $y_1, \ldots, y_m$ y sea $v \in \{0,1\}^m$. Primero llevamos ψ a una ψ' equivalente pero en forma prenexa (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 que $v \models \psi(y_1, \ldots, y_m)$ sii $\models \psi''$. □

'Evolución en un paso' en CNF

Proposición

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 cómputo de M con entrada x se representa con $|\langle C \rangle| = c \cdot S(|x|)$ bits.
Dado x, podemos computar en tiempo polinomial en $S(n)$ una fórmula $\varphi_{M,x}(\bar{s}, \bar{t})$ en CNF con variables libres $\bar{s} = s_1, \ldots, s_{c \cdot S(|x|)}$, $\bar{t} = t_1, \ldots, t_{c \cdot S(|x|)}$ tal que

$$\langle C \rangle \langle C' \rangle \models \varphi_{M,x}(\bar{s}, \bar{t}) \quad \text{sii} \quad \text{hay una flecha de } C \text{ a } C' \text{ en } G_{M,x}$$

Ejercicio

Demostrarlo (parecido a lo que hicimos con la demostración del teorema de Cook-Levin).

TQBF $\in$ **PSPACE-hard**

Teorema

TQBF $\in$ **PSPACE-hard**

Demostración.

Sea $\mathcal{L} \in \mathbf{PSpace}$ y sea M una máquina determinística que decide $\mathcal{L}$ usando espacio $S(n)$, con S un polinomio.

Demostración.

Sea $\mathcal{L} \in \mathbf{PSpace}$ y sea M una máquina determinística que decide $\mathcal{L}$ usando espacio $S(n)$, con S un polinomio.

Consideramos $G_{M,x}$. Cada vértice $\langle C \rangle$ (C es una configuración) de $G_{M,x}$ se representa con $c \cdot S(n)$ bits, donde $n = |x|$. Sea C_0 la configuración inicial y C_f la final del cómputo de M a partir de x.

Demostración.

Sea $\mathcal{L} \in \mathbf{PSpace}$ y sea M una máquina determinística que decide $\mathcal{L}$ usando espacio $S(n)$, con S un polinomio.

Consideramos $G_{M,x}$. Cada vértice $\langle C \rangle$ (C es una configuración) de $G_{M,x}$ se representa con $c \cdot S(n)$ bits, donde $n = |x|$. Sea C_0 la configuración inicial y C_f la final del cómputo de M a partir de x.
Definimos fórmulas $\psi_i(\bar{s}, \bar{t})$ con variables libres $\bar{s}$, $\bar{t}$, que son tuplas de dimensión $c \cdot S(|x|)$.

$$\langle C \rangle \langle C' \rangle \models \psi_i(\bar{s}, \bar{t}) \quad \text{sii} \quad \text{existe un camino dirigido de longitud} \leq 2^i \text{ en } G_{M,x} \text{ entre } C \text{ y } C'$$

Demostración.

Sea $\mathcal{L} \in \mathbf{PSpace}$ y sea M una máquina determinística que decide $\mathcal{L}$ usando espacio $S(n)$, con S un polinomio.

Consideramos $G_{M,x}$. Cada vértice $\langle C \rangle$ (C es una configuración) de $G_{M,x}$ se representa con $c \cdot S(n)$ bits, donde $n = |x|$. Sea C_0 la configuración inicial y C_f la final del cómputo de M a partir de x.
Definimos fórmulas $\psi_i(\bar{s}, \bar{t})$ con variables libres $\bar{s}$, $\bar{t}$, que son tuplas de dimensión $c \cdot S(|x|)$.

$$\langle C \rangle \langle C' \rangle \models \psi_i(\bar{s}, \bar{t}) \quad \text{sii} \quad \begin{array}{l} \text{existe un camino dirigido de longi-} \\ \text{tud} \leq 2^i \text{ en } G_{M,x} \text{ entre } C \text{ y } C' \end{array}$$

Sea $n = |x|$. Si hay un camino de C a C' en $G_{M,x}$, entonces hay uno de longitud $\leq 2^{c \cdot S(n)}$, porque $G_{M,x}$ tiene $\leq 2^{c \cdot S(n)}$ vértices.

Luego,

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

Entonces

$$\mathcal{L} \leq_{\mathrm{p}} \mathsf{CHECKQBF} \leq_{\mathrm{p}} \mathsf{TQBF}$$

siempre y cuando podamos definir $\psi_{c \cdot S(n)}$ en tiempo polinomial en $n = |x|$.

Luego,

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

Entonces

$$\mathcal{L} \leq_{\mathrm{p}} \mathsf{CHECKQBF} \leq_{\mathrm{p}} \mathsf{TQBF}$$

siempre y cuando podamos definir $\psi_{c \cdot S(n)}$ en tiempo polinomial en $n = |x|$. Definimos $\psi_i(\bar{s}, \bar{t})$ por recursión en i.

- $\psi_0(\bar{s}, \bar{t})$ se define como $\bar{s} = \bar{t} \vee \varphi_{M,x}(\bar{s}, \bar{t})$. Computamos ψ_0 en tiempo polinomial en n.
  ($\bar{s} = \bar{t}$ abrevia $\bigwedge_{i=1}^{2^{c \cdot S(n)}} (s_i \wedge t_i) \vee (\neg s_i \wedge \neg t_i)$)

Luego,

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

Entonces

$$\mathcal{L} \leq_{\mathrm{p}} \mathsf{CHECKQBF} \leq_{\mathrm{p}} \mathsf{TQBF}$$

siempre y cuando podamos definir $\psi_{c \cdot S(n)}$ en tiempo polinomial en $n = |x|$. Definimos $\psi_i(\bar{s}, \bar{t})$ por recursión en i.

- $\psi_0(\bar{s}, \bar{t})$ se define como $\bar{s} = \bar{t} \vee \varphi_{M,x}(\bar{s}, \bar{t})$. Computamos ψ_0 en tiempo polinomial en n.
  ($\bar{s} = \bar{t}$ abrevia $\bigwedge_{i=1}^{2^{c \cdot S(n)}} (s_i \wedge t_i) \vee (\neg s_i \wedge \neg t_i)$)
- Para $i > 0$ podemos definir

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

Luego,

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

Entonces

$$\mathcal{L} \leq_{\mathrm{p}} \mathsf{CHECKQBF} \leq_{\mathrm{p}} \mathsf{TQBF}$$

siempre y cuando podamos definir $\psi_{c \cdot S(n)}$ en tiempo polinomial en $n = |x|$. Definimos $\psi_i(\bar{s}, \bar{t})$ por recursión en i.

- $\psi_0(\bar{s}, \bar{t})$ se define como $\bar{s} = \bar{t} \vee \varphi_{M,x}(\bar{s}, \bar{t})$. Computamos ψ_0 en tiempo polinomial en n.
  ($\bar{s} = \bar{t}$ abrevia $\bigwedge_{i=1}^{2^{c \cdot S(n)}} (s_i \wedge t_i) \vee (\neg s_i \wedge \neg t_i)$)
- Para $i > 0$ podemos 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 y por lo tanto no podríamos computarla en tiempo polinomial. No sirve.

- Para $i > 0$, cambiamos por:

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

($\phi \rightarrow \rho$ abrevia $\neg\phi \vee \rho$)

$$\overbrace{\bar{s} \rightarrow \cdots\cdots\cdots \rightarrow \bar{r}}^{(\bar{u}=\bar{s} \wedge \bar{v}=\bar{r}) \rightarrow \psi_{i-1}(\bar{u},\bar{v})} \underbrace{\rightarrow \cdots\cdots\cdots \rightarrow \bar{t}}_{(\bar{u}=\bar{r} \wedge \bar{v}=\bar{t}) \rightarrow \psi_{i-1}(\bar{u},\bar{v})}$$

- Para $i > 0$, cambiamos por:

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

($\phi \rightarrow \rho$ abrevia $\neg\phi \vee \rho$)

$$\overbrace{\bar{s} \rightarrow \cdots\cdots\cdots \rightarrow \bar{r}}^{(\bar{u}=\bar{s}\wedge\bar{v}=\bar{r})\rightarrow\psi_{i-1}(\bar{u},\bar{v})} \underbrace{\rightarrow \cdots\cdots\cdots \rightarrow \bar{t}}_{(\bar{u}=\bar{r}\wedge\bar{v}=\bar{t})\rightarrow\psi_{i-1}(\bar{u},\bar{v})}$$

- Para computar ψ_i necesitamos i llamados recursivos hasta llegar al caso base.

- Para $i > 0$, cambiamos por:

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

($\phi \rightarrow \rho$ abrevia $\neg\phi \vee \rho$)

$$\overbrace{\bar{s} \rightarrow \cdots\cdots\cdots \rightarrow \bar{r}}^{(\bar{u}=\bar{s} \wedge \bar{v}=\bar{r}) \rightarrow \psi_{i-1}(\bar{u},\bar{v})} \underbrace{\rightarrow \cdots\cdots\cdots \rightarrow \bar{t}}_{(\bar{u}=\bar{r} \wedge \bar{v}=\bar{t}) \rightarrow \psi_{i-1}(\bar{u},\bar{v})}$$

- Para computar ψ_i necesitamos i llamados recursivos hasta llegar al caso base.
- Entonces computamos $\psi_{c \cdot S(n)}$ en tiempo polinomial en $S(n)$, que es polinomial en n.
- $\psi_{c \cdot S(n)}$ es una QBF generalizada. En tiempo polinomial la convertimos en QBF.

Corolario

TQBF $\in$ **PSPACE-completo**.

Teorema de Savitch

Teorema de Savitch

Teorema (Savitch)

$\mathbf{NSpace}(S(n)) \subseteq \mathbf{Space}(S(n)^2)$.

Corolario

$\mathbf{PSpace} = \mathbf{NPSpace}$.

Demostración de $\mathbf{NSpace}(S(n)) \subseteq \mathbf{Space}(S(n)^2)$.

Parecido a la demostración que vimos recién.
Sea $\mathcal{L} \in \mathbf{NSpace}(S(n))$, sea N una máquina no-determinística tal que para todo x, N acepta x sii $x \in \mathcal{L}$ sii hay un camino de longitud $\leq 2^{c \cdot S(|x|)}$ en $G_{N,x}$ desde C_0 hasta C_f.

Demostración de $\mathbf{NSpace}(S(n)) \subseteq \mathbf{Space}(S(n)^2)$.

Parecido a la demostración que vimos recién.
Sea $\mathcal{L} \in \mathbf{NSpace}(S(n))$, sea N una máquina no-determinística tal que para todo x, N acepta x sii $x \in \mathcal{L}$ sii hay un camino de longitud $\leq 2^{c \cdot S(|x|)}$ en $G_{N,x}$ desde C_0 hasta C_f.
Definimos una función booleana recursiva (traducible a una máquina determinística) *Reach* tal que $Reach(C, C', i) = 1$ si hay un camino de longitud $\leq 2^i$ en $G_{N,x}$ desde C hasta C'. Luego

$$x \in \mathcal{L} \qquad \text{sii} \qquad Reach(C_0, C_f, c \cdot S(n)) = 1.$$

Demostración de $\mathbf{NSpace}(S(n)) \subseteq \mathbf{Space}(S(n)^2)$.

Parecido a la demostración que vimos recién.
Sea $\mathcal{L} \in \mathbf{NSpace}(S(n))$, sea N una máquina no-determinística tal que para todo x, N acepta x sii $x \in \mathcal{L}$ sii hay un camino de longitud $\leq 2^{c \cdot S(|x|)}$ en $G_{N,x}$ desde C_0 hasta C_f.
Definimos una función booleana recursiva (traducible a una máquina determinística) *Reach* tal que $Reach(C, C', i) = 1$ si hay un camino de longitud $\leq 2^i$ en $G_{N,x}$ desde C hasta C'. Luego

$$x \in \mathcal{L} \qquad \text{sii} \qquad Reach(C_0, C_f, c \cdot S(n)) = 1.$$

$Reach(C, C', i)$ hace esto:

si $i = 0$, devolver 1 si $C = C'$ o $C \to C'$ en $G_{N,x}$; 0 si no
para cada posible configuración C'' de $G_{N,x}$:
  $j = Reach(C, C'', i-1)$; $k = Reach(C'', C', i-1)$
  si $j = k = 1$, devolver 1
devolver 0

Demostración de $\mathbf{NSPACE}(S(n)) \subseteq \mathbf{SPACE}(S(n)^2)$.

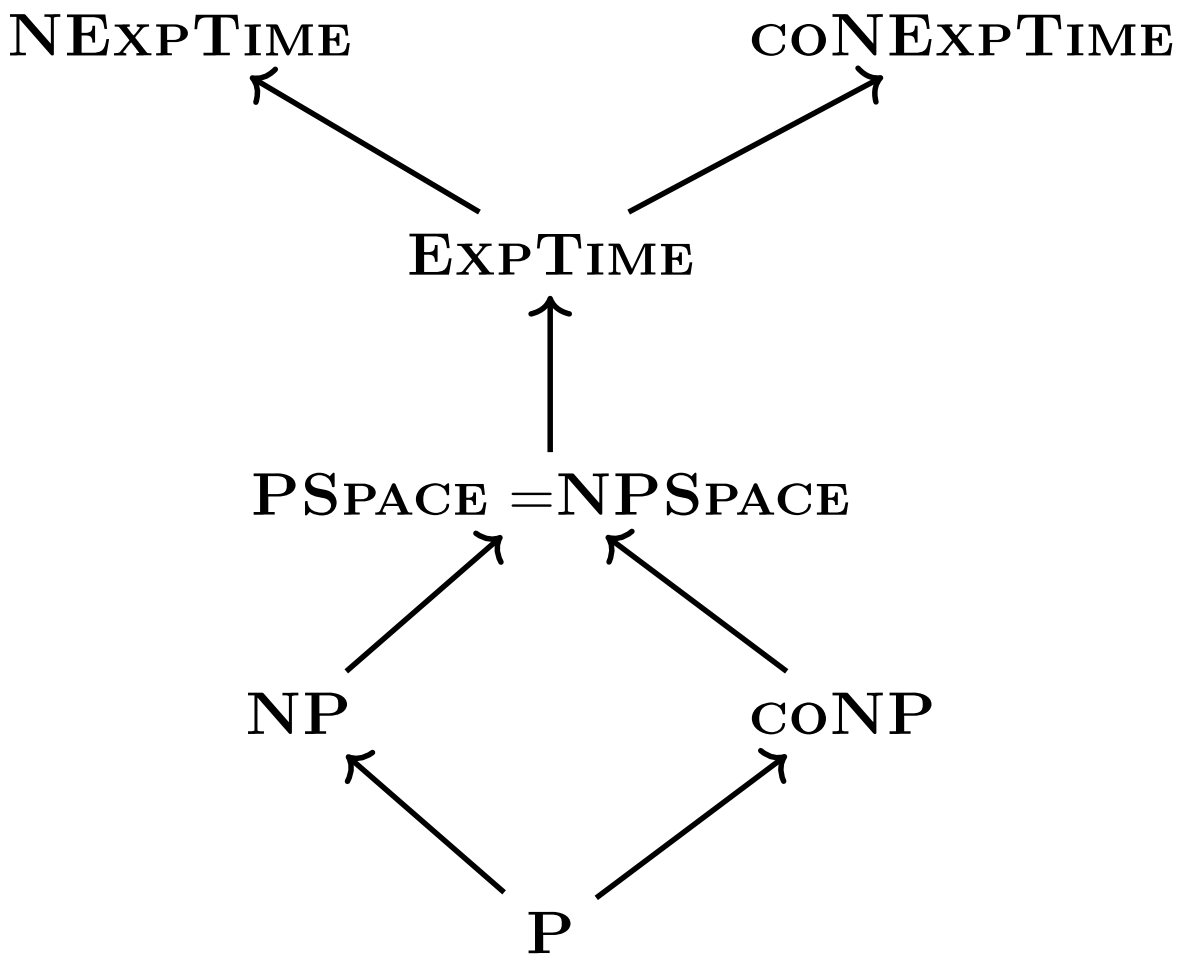
Parecido a la demostración que vimos recién.
Sea $\mathcal{L} \in \mathbf{NSPACE}(S(n))$, sea N una máquina no-determinística tal que para todo x, N acepta x sii $x \in \mathcal{L}$ sii hay un camino de longitud $\leq 2^{c \cdot S(|x|)}$ en $G_{N,x}$ desde C_0 hasta C_f.
Definimos una función booleana recursiva (traducible a una máquina determinística) *Reach* tal que $Reach(C, C', i) = 1$ si hay un camino de longitud $\leq 2^i$ en $G_{N,x}$ desde C hasta C'. Luego

$$x \in \mathcal{L} \qquad \text{sii} \qquad Reach(C_0, C_f, c \cdot S(n)) = 1.$$

$Reach(C, C', i)$ hace esto:

> *si* $i = 0$, *devolver 1 si* $C = C'$ *o* $C \to C'$ *en* $G_{N,x}$; *0 si no*
> *para cada posible configuración* C'' *de* $G_{N,x}$:
> $j = Reach(C, C'', i-1)$; $k = Reach(C'', C', i-1)$
> *si* $j = k = 1$, *devolver 1*
> *devolver 0*

Sea $n = |x|$. $Reach(C, C', i)$ enumera las configuraciones C'' usando espacio $O(S(n))$. Reusa el espacio para calcular j y k. Entonces $Reach(C, C', i)$ usa espacio $O(i \cdot S(n))$. Luego, computamos $Reach(C_0, C_f, c \cdot S(n))$ usando espacio $O(S(n)^2)$. □


NExpTime
coNExpTime
ExpTime
PSpace =NPSpace
NP
coNP
P