Nociones de Reducción y Completitud para una Clase de Complejidad

IIC3242

Problemas en una clase de complejidad

Una clase de complejidad contiene un conjunto de problemas de decisión.

- ¿Hay alguno de estos problemas que *represente* a la clase?

Un caso muy conocido: SAT representa a la clase NP.

- ¿En qué sentido la representa?
- ¿Qué sucede si encontramos un algoritmo eficiente para SAT?

Vamos a definir las nociones necesarias para estudiar cuando un problema representa a una clase de complejidad.

- Vamos a empezar por la noción de reducción

La noción de reducción: Máquinas que calculan

Definición

Una MT calculadora (*MTC*) *con* k *cintas de trabajo:*
$(Q, \Sigma, \Gamma, q_0, \delta)$

- Q *es un conjunto finito de estados*
- Σ *es un alfabeto finito tal que* $\vdash, \mathtt{B} \notin \Sigma$
- Γ *es un alfabeto finito tal que* $\Sigma \cup \{\vdash, \mathtt{B}\} \subseteq \Gamma$
- $q_0 \in Q$ *es el estado inicial*
- δ *es una función parcial:*

$$\delta \ : \ Q \times \Gamma^{k+1} \to Q \times \{\leftarrow, \square, \rightarrow\} \times \Gamma^k \times \{\leftarrow, \square, \rightarrow\}^k \times (\Sigma \cup \{\mathtt{B}\})$$

δ *es llamada función de transición*

MTC con k cintas de trabajo: Funcionamiento

La máquina tiene $k + 2$ cintas infinitas hacia la derecha.

- La primera cinta es de lectura, la última de escritura y las restantes para todos los cálculos intermedios
- El símbolo $\vdash$ es usado para demarcar la posición 0 de cada cinta

MTC con k cintas de trabajo: Funcionamiento

Σ es el alfabeto de entrada y salida, y Γ es el alfabeto de las cintas.

- Una palabra $w \in \Sigma^*$ de entrada de largo n es colocada en las posiciones $1, \ldots, n$ de la primera cinta
- Las siguientes posiciones $(n+1, n+2, \ldots)$ de la primera cinta contienen el símbolo `B`
- Las restantes cintas contienen el símbolo `B` en las posiciones $1, 2, 3, \ldots$

La máquina tiene una cabeza lectora por cinta.

- Al comenzar, la máquina se encuentra en el estado q_0, y cada cabeza lectora está en la posición 1 de su cinta

MTC con k cintas de trabajo: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora i se encuentra en la posición p_i.

- Si el símbolo en la posición p_i es a_i y $\delta(q, a_1, \ldots, a_k, a_{k+1}) = (q', X_1, b_2, \ldots, b_{k+1}, X_2, \ldots, X_{k+1}, b_{k+2})$, entonces:
  - Para cada $i \in \{2, \ldots, k+1\}$, la máquina escribe el símbolo b_i en la posición p_i de la i-ésima cinta
  - Cambia de estado desde q a q'
  - Para cada $i \in \{1, \ldots, k+1\}$, la máquina mueve la cabeza lectora de la i-ésima cinta a la posición $p_i - 1$ si X_i es $\leftarrow$, y a la posición $p_i + 1$ si X_i es $\rightarrow$. Si X_i es $\square$, entonces la máquina no mueve la cabeza lectora de la i-ésima cinta
  - Si $b_{k+2} \in \Sigma$, entonces la máquina escribe el símbolo b_{k+2} en la posición p_{k+2} de la cinta $k+2$, y mueve la cabeza lectora de esta cinta a la posición $p_{k+2} + 1$. Si $b_{k+2} = \mathtt{B}$, entonces la máquina no cambia la configuración de la cinta $k+2$

MTC con k cintas de trabajo: Función calculada

Para medir el espacio ocupado por una MTC sólo se considera las celdas visitadas en las k cintas de trabajo.

- No depende ni de la entrada ni de la salida

Definición

Una función $h : \Sigma^* \to \Sigma^*$ puede ser calculada en espacio $O(f(n))$ si existe una MTC M tal que:

- *M para en todas las entradas*
- *s_M es $O(f(n))$*
- *Con entrada $w \in \Sigma^*$: M se detiene en una configuración que tiene en la cinta $k+2$ a $h(w)$, precedido por el símbolo $\vdash$ y seguido por una cadena de símbolos* B

La noción de reducción logarítmica

Dados lenguajes L_1 y L_2 con alfabeto Σ

Definición

L_1 es reducible en espacio logarítmico a L_2, denotado como $L_1 \leq_{\mathrm{m}}^{\log} L_2$, si existe una función f computable en espacio $O(\log n)$ tal que para todo $w \in \Sigma^*$:

$$w \in L_1 \text{ si y sólo si } f(w) \in L_2$$

Reducción logarítmica: Una propiedad fundamental

Proposición

Si $L_1 \leq_{\mathrm{m}}^{\log} L_2$ *y* $L_2 \leq_{\mathrm{m}}^{\log} L_3$, *entonces* $L_1 \leq_{\mathrm{m}}^{\log} L_3$.

¿Por qué es fundamental esta propiedad?

Ejercicio

Demuestre la proposición.

- La demostración no es trivial ...

La noción de completitud

Definición (Hardness)

Dada una clase de complejidad $\mathcal{C}$ que contiene a LOGSPACE, un lenguaje L es hard para $\mathcal{C}$ si para todo $L' \in \mathcal{C}$ existe una reducción logarítmica de L' a L.

Definición (Completitud)

Dada una clase de complejidad $\mathcal{C}$ que contiene a LOGSPACE, un lenguaje L es completo para $\mathcal{C}$ si $L \in \mathcal{C}$ y L es hard para $\mathcal{C}$.

Notación

Si un lenguaje L es completo para una clase de complejidad $\mathcal{C}$, decimos que L es $\mathcal{C}$-completo.

La noción de completitud: Un primer ejemplo

Sea CAMINO el siguiente lenguaje:

$$\text{CAMINO} = \{(G, a, b) \mid G \text{ es un grafo dirigido y } b \text{ es alcanzable desde } a \text{ en } G\}.$$

Teorema

CAMINO es NLOGSPACE-completo.

Ejercicio

Demuestre el teorema.

¿Por qué es importante la noción de completitud?

Corolario

Si CAMINO $\in$ LOGSPACE, entonces LOGSPACE $=$ NLOGSPACE.

Ejercicio

Demuestre el corolario.

- El ingrediente principal es la transitividad de $\leq_{\mathrm{m}}^{\log}$

¡Para demostrar que LOGSPACE $=$ NLOGSPACE, sólo tenemos que demostrar que CAMINO $\in$ LOGSPACE!

Segundo ejemplo: Completitud en NP

Ahora vamos a estudiar la clase NP.

- SAT $= \{\varphi \mid \varphi$ es una fórmula en lógica proposicional y φ es satisfacible$\}$
- Vamos a demostrar que SAT es NP-completo

Teorema (Cook-Levin)

SAT es NP-completo.

Demostración: Primero tenemos que demostrar que SAT $\in$ NP.

- ¿Cómo se demuestra esto?

Teorema de Cook-Levin: Demostración

En segundo lugar, tenemos que demostrar que SAT es NP-hard.

- Sea $L \in \text{NP}$. Tenemos que demostrar que L es reducible en espacio logarítmico a SAT

Por demostrar: Existe una función $f : \Sigma^* \to \Sigma^*$ tal que

- f es computable en espacio logarítmico
- para cada $w \in \Sigma^*$: $w \in L$ si y sólo si $f(w) \in \text{SAT}$

Notación

$f(w) = \varphi_w$

Teorema de Cook-Levin: Demostración

Como $L \in \text{NP}$, existe una MT $M = (Q, \Sigma, \Gamma, q_0, \delta, F)$ no determinista con $k \geq 1$ cintas de trabajo tal que:

- $L = L(M)$
- $t_M(n)$ es $O(n^c)$, donde $c > 0$ es un número natural

Vamos a representar el funcionamiento de M con entrada w usando la fórmula φ_w: M acepta w si y sólo si φ_w es satisfacible.

Teorema de Cook-Levin: Demostración

Sin perdida de generalidad suponemos que:

- $w = a_1 \cdots a_n$, donde cada $a_i \in \Sigma$
- $F = \{q_m\}$
- Para cada $q \in Q$ y $\bar{a} \in \Gamma^{k+1}$, se tiene que $\delta(q, \bar{a})$ está definido si y sólo si $q \neq q_m$

¿Por qué podemos suponer esto?

Teorema de Cook-Levin: Demostración

Usamos las siguientes variables proposicionales:

$$
\begin{array}{lcl}
s_{i,t,p,a} & : & i \in [1, k+1],\ t \in [0, t_M(n)],\ p \in [0, t_M(n)] \text{ y } a \in \Gamma \\
c_{i,t,p} & : & i \in [1, k+1],\ t \in [0, t_M(n)] \text{ y } p \in [0, t_M(n)] \\
e_{t,q} & : & t \in [0, t_M(n)] \text{ y } q \in Q
\end{array}
$$

φ_w es definida como $\varphi_I \wedge \varphi_C \wedge \varphi_A \wedge \varphi_\delta$.

Teorema de Cook-Levin: Demostración

φ_I: Estado inicial

$$\left(\bigwedge_{i=1}^{k+1} c_{i,0,1}\right) \wedge e_{0,q_0} \wedge \left(\bigwedge_{i=1}^{k+1} s_{i,0,0,\vdash}\right) \wedge \left(\bigwedge_{p=1}^{n} s_{1,0,p,a_p}\right) \wedge \left(\bigwedge_{p=n+1}^{t_M(n)} s_{1,0,p,\mathrm{B}}\right) \wedge \left(\bigwedge_{i=2}^{k+1} \bigwedge_{p=1}^{t_M(n)} s_{i,0,p,\mathrm{B}}\right)$$

Teorema de Cook-Levin: Demostración

φ_C: La máquina funciona correctamente

φ_C se define como la conjunción de cinco fórmulas. Primero, cada celda siempre contiene un único símbolo:

$$\bigwedge_{i=1}^{k+1} \bigwedge_{t=0}^{t_M(n)} \bigwedge_{p=0}^{t_M(n)} \left(\bigvee_{a \in \Gamma} \left(S_{i,t,p,a} \wedge \bigwedge_{b \in (\Gamma \setminus \{a\})} \neg S_{i,t,p,b} \right) \right)$$

Teorema de Cook-Levin: Demostración

Segundo, la máquina siempre está en un único estado:

$$\bigwedge_{t=0}^{t_M(n)} \left(\bigvee_{q \in Q} \left(e_{t,q} \wedge \bigwedge_{q' \in (Q \setminus \{q\})} \neg e_{t,q'} \right) \right)$$

Tercero, la cabeza lectora de cada celda siempre está en una única posición:

$$\bigwedge_{i=1}^{k+1} \bigwedge_{t=0}^{t_M(n)} \left(\bigvee_{p=0}^{t_M(n)} \left(c_{i,t,p} \wedge \bigwedge_{p' \in ([0,t_M(n)] \setminus \{p\})} \neg c_{i,t,p'} \right) \right)$$

Teorema de Cook-Levin: Demostración

Cuarto, el contenido de la cinta de lectura no cambia durante la ejecución de la máquina:

$$\bigwedge_{t=0}^{t_M(n)-1} \bigwedge_{p=0}^{t_M(n)} \bigwedge_{a \in \Gamma} \left(S_{1,t+1,p,a} \leftrightarrow S_{1,t,p,a} \right)$$

Teorema de Cook-Levin: Demostración

Quinto, el valor de una celda de una cinta de trabajo no cambia si no es apuntada por la cabeza lectora:

$$\bigwedge_{i=2}^{k+1} \bigwedge_{t=0}^{t_M(n)-1} \bigwedge_{p=0}^{t_M(n)} \left((\neg c_{i,t,p}) \rightarrow \bigwedge_{a \in \Gamma} (s_{i,t+1,p,a} \leftrightarrow s_{i,t,p,a}) \right)$$

Teorema de Cook-Levin: Demostración

φ_A: La máquina acepta w

$$\bigvee_{t=0}^{t_M(n)} e_{t,q_m}$$

Teorema de Cook-Levin: Demostración

φ_δ: relación δ define como funciona la máquina (representamos $\leftarrow$ como -1, $\square$ como 0 y $\rightarrow$ como 1):

$$\bigwedge_{t=0}^{t_M(n)-1} \bigwedge_{p_1=0}^{t_M(n)-1} \cdots \bigwedge_{p_{k+1}=0}^{t_M(n)-1} \Bigg[\bigwedge_{(q,a_1,\ldots,a_{k+1}) \in Q \times \Gamma^{k+1}} \Bigg(\Big(e_{t,q} \wedge \bigwedge_{i=1}^{k+1} \big(c_{i,t,p_i} \wedge s_{i,t,p_i,a_i} \big) \Big) \rightarrow \bigvee_{(q,a_1,\ldots,a_{k+1},q',m_1,b_2,\ldots,b_{k+1},m_2,\ldots,m_{k+1}) \in \delta} \Bigg(e_{t+1,q'} \wedge \Big(\bigwedge_{i=1}^{k+1} c_{i,t+1,p_i+m_i} \Big) \wedge \Big(\bigwedge_{i=2}^{k+1} s_{i,t+1,p_i,b_i} \Big) \Bigg) \Bigg) \Bigg]$$

Teorema de Cook-Levin: Demostración

Tenemos que demostrar que $w \in L$ si y sólo si φ_w es satisfacible.

- ¿Cómo se hace esto?

Para terminar la demostración, sólo nos queda probar que φ_w puede ser construida en espacio $O(\log n)$.

- ¿Cómo se demuestra esto?
  - Nótese que $\log t_M(n)$ es $O(\log n)$

Tercer ejemplo: Completitud en PTIME

Vamos a demostrar que una restricción de SAT es PTIME-completo.

Notación

- *Un literal es una letra proposicional o la negación de una letra proposicional:* p *y* $\neg q$
- *Una clausula es una disyunción de literales:* $(p \vee q)$ *y* $(s \vee \neg q \vee \neg r)$
- *Una cláusula es de Horn si contiene a lo más un literal positivo (de la forma* p*).*
  - $(s \vee \neg q \vee \neg r)$ *es una clausula de Horn y* $(p \vee q)$ *no lo es*
- *HORN-SAT* $= \{\varphi \mid \varphi$ *es una conjunción de cláusulas de Horn y* φ *es satisfacible*$\}$

Completitud en PTIME: HORN-SAT

Teorema

HORN-SAT es PTIME-completo.

Demostración: Primero tenemos que demostrar que HORN-SAT $\in$ PTIME.

- ¿Cómo se demuestra esto?

En segundo lugar, tenemos que demostrar que HORN-SAT es PTIME-hard.

- Vamos a demostrar que $\overline{\text{HORN-SAT}}$ es PTIME-hard
- ¿Por qué nos basta demostrar esto?

HORN-SAT es PTIME-hard: Demostración

Sea $L \in$ PTIME. Tenemos que demostrar que L es reducible en espacio logarítmico a $\overline{\text{HORN-SAT}}$.

Por demostrar: Existe una función $f : \Sigma^* \to \Sigma^*$ tal que

- f es computable en espacio logarítmico
- para cada $w \in \Sigma^*$: $w \in L$ si y sólo si $f(w) \in \overline{\text{HORN-SAT}}$

Notación

$f(w) = \varphi_w$

HORN-SAT es PTIME-hard: Demostración

Como $L \in \text{PTIME}$, existe una MT $M = (Q, \Sigma, \Gamma, q_0, \delta, F)$ determinista con una cinta de trabajo tal que:

- M para con todas las entradas
- $L = L(M)$
- $t_M(n)$ es $O(n^c)$, donde $c > 0$ es un número natural

¿Por qué podemos suponer que M tiene una cinta de trabajo?

- ¿Podíamos suponer esto en la demostración del Teorema de Cook-Levin?

Vamos a representar el funcionamiento de M con entrada w usando la fórmula φ_w: M acepta w si y sólo si φ_w **no** es satisfacible.

HORN-SAT es PTIME-hard: Demostración

Sin perdida de generalidad suponemos que:

- $w = a_1 \cdots a_n$, donde cada $a_i \in \Sigma$
- $F = \{q_m\}$
- Para cada $q \in Q$ y $(a, b) \in \Gamma \times \Gamma$, si $\delta(q, a, b)$ está definido, entonces $q \neq q_m$

¿Por qué podemos suponer esto?

HORN-SAT es PTIME-hard: Demostración

Usamos las siguientes variables proposicionales:

$$
\begin{array}{lcl}
s_{i,t,p,a} & : & i \in [1,2],\ t \in [0, t_M(n)],\ p \in [0, t_M(n)] \text{ y } a \in \Gamma \\
c_{i,t,p} & : & i \in [1,2],\ t \in [0, t_M(n)] \text{ y } p \in [0, t_M(n)] \\
e_{t,q} & : & t \in [0, t_M(n)] \text{ y } q \in Q
\end{array}
$$

φ_w es definida como $\varphi_I \wedge \varphi_N \wedge \varphi_\delta$

HORN-SAT es PTIME-hard: Demostración

φ_I: Estado inicial.

$$c_{1,0,1} \wedge c_{2,0,1} \wedge e_{0,q_0} \wedge s_{1,0,0,\vdash} \wedge s_{2,0,0,\vdash} \wedge \left(\bigwedge_{p=1}^{n} s_{1,0,p,a_p} \right) \wedge \left(\bigwedge_{p=n+1}^{t_M(n)} s_{1,0,p,\mathtt{B}} \right) \wedge \left(\bigwedge_{p=1}^{t_M(n)} s_{2,0,p,\mathtt{B}} \right)$$

HORN-SAT es PTIME-hard: Demostración

φ_N: La máquina no acepta w

$$\bigwedge_{t=0}^{t_M(n)} \neg e_{t,q_m}$$

HORN-SAT es PTIME-hard: Demostración

φ_δ define como funciona la máquina (representamos $\leftarrow$ como -1, $\square$ como 0 y $\rightarrow$ como 1):

$$\bigwedge_{t=0}^{t_M(n)-1} \bigwedge_{p_1=0}^{t_M(n)-1} \bigwedge_{p_2=0}^{t_M(n)-1} \bigwedge_{(q,a,b,q',\ell,c,m)\in\delta} \Big[\Big(\neg e_{t,q} \vee \neg c_{1,t,p_1} \vee \neg s_{1,t,p_1,a} \vee \neg c_{2,t,p_2} \vee \neg s_{2,t,p_2,b} \vee e_{t+1,q'} \Big) \wedge$$
$$\Big(\neg e_{t,q} \vee \neg c_{1,t,p_1} \vee \neg s_{1,t,p_1,a} \vee \neg c_{2,t,p_2} \vee \neg s_{2,t,p_2,b} \vee c_{1,t+1,p_1+\ell} \Big) \wedge$$
$$\Big(\neg e_{t,q} \vee \neg c_{1,t,p_1} \vee \neg s_{1,t,p_1,a} \vee \neg c_{2,t,p_2} \vee \neg s_{2,t,p_2,b} \vee s_{1,t+1,p_1,a} \Big) \wedge$$

HORN-SAT es PTIME-hard: Demostración

$$\left(\neg e_{t,q} \vee \neg c_{1,t,p_1} \vee \neg s_{1,t,p_1,a} \vee \neg c_{2,t,p_2} \vee \neg s_{2,t,p_2,b} \vee c_{2,t+1,p_2+m}\right) \wedge$$

$$\left(\neg e_{t,q} \vee \neg c_{1,t,p_1} \vee \neg s_{1,t,p_1,a} \vee \neg c_{2,t,p_2} \vee \neg s_{2,t,p_2,b} \vee s_{2,t+1,p_2,c}\right) \wedge$$

$$\bigwedge_{p_1' \in ([0,t_M(n)] \setminus \{p_1\})} \bigwedge_{d \in \Gamma} \Big(\neg e_{t,q} \vee \neg c_{1,t,p_1} \vee \neg s_{1,t,p_1,a} \vee$$

$$\neg c_{2,t,p_2} \vee \neg s_{2,t,p_2,b} \vee \neg s_{1,t,p_1',d} \vee s_{1,t+1,p_1',d} \Big) \wedge$$

$$\bigwedge_{p_2' \in ([0,t_M(n)] \setminus \{p_2\})} \bigwedge_{e \in \Gamma} \Big[\Big(\neg e_{t,q} \vee \neg c_{1,t,p_1} \vee \neg s_{1,t,p_1,a} \vee$$

$$\neg c_{2,t,p_2} \vee \neg s_{2,t,p_2,b} \vee \neg s_{2,t,p_2',e} \vee s_{2,t+1,p_2',e} \Big) \Big]$$

HORN-SAT es PTIME-hard: Demostración

Tenemos que demostrar que $w \in L$ si y sólo si φ_w **no** es satisfacible.

- ¿Cómo se hace esto?

Esta propiedad junto con las siguientes condiciones nos dicen que la reducción es correcta:

- φ_w es una conjunción de clausulas de Horn
- φ_w puede ser construida en espacio logarítmico (ya que $\log t_M(n)$ es $O(\log n)$)

Otros problemas completos para PTIME y NP

Una vez encontrado un problema L completo para una clase $\mathcal{C}$, es más simple encontrar otros problemas $\mathcal{C}$-completos.

- L' es $\mathcal{C}$-completo si $L' \in \mathcal{C}$ y $L \leq_{\mathrm{m}}^{\log} L'$

Vamos a usar esta idea para mostrar otros problemas completos para PTIME y NP.

Un problema NP-completo: CNF-SAT

Un problema muy útil al hacer reducciones:

$$\text{CNF-SAT} = \{\varphi \mid \varphi \text{ es una conjunción de cláusulas y } \varphi \text{ es satisfacible}\}$$

Teorema

CNF-SAT es NP-completo.

Ejercicio

Demuestre que SAT $\leq_{\mathrm{m}}^{\log}$ CNF-SAT.

Otro problema NP-completo: 3-CNF-SAT

Notación

Una k-clausula es una clausula con a lo más k literales.

Un problema aun más útil al hacer reducciones:

$$\text{3-CNF-SAT} = \{\varphi \mid \varphi \text{ es una conjunción de 3-cláusulas y } \varphi \text{ es satisfacible}\}$$

Teorema

3-CNF-SAT es NP-completo.

Ejercicio

Demuestre que SAT $\leq_{\mathrm{m}}^{\log}$ 3-CNF-SAT.

Un tercer problema NP-completo: Programación entera

Un sistema de ecuaciones lineales enteras es de la forma:

$$A\vec{x} \leq \vec{b}$$

donde:

- A es una matriz de números enteros de orden $m \times n$
- $\vec{x}$ es un vector de variables de orden $n \times 1$
- $\vec{b}$ es un vector de números enteros de orden $m \times 1$

Un vector $\vec{c}$ de números enteros de orden $n \times 1$ es una solución para el sistema si $A\vec{c} \leq \vec{b}$.

Un tercer problema NP-completo: Programación entera

Un problema muy estudiado por su utilidad práctica:

$$\text{PROG-ENT} = \{(A, \vec{b}) \mid A\vec{x} \leq \vec{b} \text{ es un sistema de ecuaciones lineales enteras que tiene solución}\}$$

Teorema

PROG-ENT es NP-completo.

Ejercicio

Demuestre que 3-CNF-SAT $\leq_{\mathrm{m}}^{\log}$ PROG-ENT.

- ¿Cómo se puede demostrar que PROG-ENT $\in$ NP?

Y ahora PTIME-completitud: Programación lineal

Un sistema de ecuaciones lineales enteras $A\vec{x} \leq \vec{b}$ tiene solución en $\mathbb{R}$ si existe $\vec{c}$ en $\mathbb{R}$ tal que $A\vec{c} \leq \vec{b}$.

Un problema fundamental en ingeniería:

$$\text{PROG-LIN} = \{(A, \vec{b}) \mid A\vec{x} \leq \vec{b} \text{ es un sistema de ecuaciones lineales enteras que tiene solución en } \mathbb{R}\}$$

Teorema

PROG-LIN es PTIME-completo.

Programación lineal es PTIME-completo: Demostración

Dirección complicada: PROG-LIN $\in$ PTIME.

- Primer algoritmo polinomial: Khachiyan (1979)

  No es usado en la práctica por el alto grado del polinomio

- Segundo algoritmo polinomial: Karmarkar (1984)

Dirección simple: PROG-LIN es PTIME-hard.

- Vamos a demostrar que HORN-SAT $\leq_{\mathrm{m}}^{\log}$ PROG-LIN

Programación lineal es PTIME-completo: Demostración

Sea φ una conjunción de clausulas de Horn.

Vamos a construir un sistema de ecuaciones lineales enteras $A\vec{x} \leq \vec{b}$ tal que:

φ es satisfacible si y sólo si $A\vec{x} \leq \vec{b}$ tiene solución en $\mathbb{R}$.

Cada letra proposicional en φ es una variable en el sistema de ecuaciones.

- Por cada letra p se incluye las ecuaciones:

$$\begin{aligned} -p &\leq 0 \\ p &\leq 1 \end{aligned}$$

Programación lineal es PTIME-completo: Demostración

Además, el sistema incluye las siguientes ecuaciones:

- Si p es una conjunción en φ:

$$-p \leq -1$$

- Si $(\neg p_1 \vee \cdots \vee \neg p_n \vee q)$ es una conjunción en φ:

$$\left(\sum_{i=1}^{n} p_i\right) - q \leq n - 1$$

- Si $(\neg p_1 \vee \cdots \vee \neg p_n)$ es una conjunción en φ:

$$\left(\sum_{i=1}^{n} p_i\right) \leq n - 1$$

Completitud en otras clases de complejidad

Encontrar un problema completo para una clase de complejidad nos ayuda a caracterizarla.

- Hemos hecho esto para NLOGSPACE, PTIME y NP

Otra clase muy importante: PSPACE

- Clase de problemas que pueden ser resueltos *eficientemente* en términos de espacio

¿Qué problemas son completos para la clase PSPACE?

Un primer problema completo para PSPACE

Notación

$L(r)$ es el lenguaje representado por una expresión regular r.

Decimos que una expresión regular r_1 está contenida en una expresión regular r_2 si $L(r_1) \subseteq L(r_2)$.

- Por ejemplo: $(01)^*$ está contenida en $(0 + 1)^*$

Notación

$r_1 \subseteq r_2$: r_1 está contenida en r_2.

Un primer problema completo para PSPACE

Sea CONT-REG el siguiente problema:

$$\text{CONT-REG} = \{(r_1, r_2) \mid r_1 \text{ y } r_2 \text{ son expresiones regulares tales que } r_1 \subseteq r_2\}$$

Este es un problema que aparece en muchas aplicaciones prácticas.

- Optimización de consultas sobre bases de datos XML

Teorema (Meyer & Stockmeyer)

CONT-REG es PSPACE-completo.

CONT-REG es PSPACE-completo: Demostración

Primero tenemos que demostrar que CONT-REG $\in$ PSPACE.

- ¿Cómo se demuestra esto?

En segundo lugar, tenemos que demostrar que CONT-REG es PSPACE-hard.

- Vamos a demostrar que para todo $L \in$ PSPACE, se tiene que $L \leq_{\mathrm{m}}^{\log} \overline{\text{CONT-REG}}$
- ¿Por qué basta con demostrar esto?

CONT-REG es PSPACE-completo: Demostración

Por demostrar: Existe una función $f : \Sigma^* \to \Sigma^*$ tal que

- f es computable en espacio logarítmico
- para cada $w \in \Sigma^*$: $w \in L$ si y sólo si $f(w) \in \overline{\text{CONT-REG}}$

Notación

$f(w) = (r_w, s_w)$

CONT-REG es PSPACE-completo: Demostración

Como $L \in$ PSPACE, existe una MT $M = (Q, \Sigma, \Gamma, q_0, \delta, F)$ determinista con una cinta de trabajo tal que:

- M para en todas las entradas
- $L = L(M)$
- $s_M(n)$ es $O(n^c)$, donde $c > 0$ es un número natural

¿Por qué consideramos sólo una cinta de trabajo?

Vamos a representar el funcionamiento de M con entrada w usando los lenguajes regulares r_w y s_w:

$$M \text{ acepta } w \text{ si y sólo si } L(r_w) \not\subseteq L(s_w)$$

CONT-REG es PSPACE-completo: Demostración

Sin perdida de generalidad suponemos que:

- $w = a_1 \cdots a_n$, donde cada $a_i \in \Sigma$ y $n \geq 1$
- $F = \{q_m\}$
- Para cada $q \in Q$ y $(a, b) \in \Gamma^2$, si $\delta(q, a, b)$ está definido, entonces $q \neq q_m$

¿Por qué podemos suponer esto?

- ¿Qué hacemos con la entrada $w = \varepsilon$?

CONT-REG es PSPACE-completo: Demostración

Sea $\Delta = \Gamma \cup (\Gamma \times \{\star\}) \cup Q \cup \{\#, \$, \%\}$

- Suponemos que estos conjuntos son disjuntos

Una configuración de M es representada por una palabra en el lenguaje regular:

$$\#Q\$(\Gamma \cup (\Gamma \times \{\star\}))^{n+2} \%(\Gamma \cup (\Gamma \times \{\star\}))^{s_M(n)+1}$$

- El símbolo $\star$ es usado para indicar las posiciones de las cabezas lectoras
- Dado $\Psi = \{b_1, \ldots, b_k\} \subseteq \Delta$, el conjunto Ψ representa a la expresión regular $(b_1 + \cdots + b_k)$

CONT-REG es PSPACE-completo: Demostración

Definimos $r_w = \Delta^*$ y $s_w = s_1 + s_2 + s_3 + s_4$, donde:

- s_1: Palabras que no representan una sucesión de configuraciones de M con entrada w
- s_2: Palabras que no comienzan con la configuración inicial de M con entrada w
- s_3: Palabras que no terminan con una configuración final de M con entrada w
- s_4: Palabras que contienen de manera consecutiva a dos configuraciones α y β de M con entrada w tales que α y β no están de acuerdo con δ

CONT-REG es PSPACE-completo: Demostración

s_1 se define como la unión de las siguientes expresiones:

- El primer símbolo de la palabra no es #:

$$s_{1,1} \quad = \quad \varepsilon + (\Delta \setminus \{\#\})\Delta^*$$

- Después de un símbolo # no aparece inmediatamente un estado:

$$s_{1,2} \quad = \quad \Delta^*\#(\varepsilon + (\Delta \setminus Q)\Delta^*)$$

CONT-REG es PSPACE-completo: Demostración

- Después de un símbolo en Q no aparece inmediatamente el símbolo \$:

$$s_{1,3} = \Delta^* Q(\varepsilon + (\Delta \setminus \{\$\})\Delta^*)$$

- Después de un símbolo \$ no aparece inmediatamente el símbolo $\vdash$:

$$s_{1,4} = \Delta^* \$(\varepsilon + (\Delta \setminus \{\vdash, (\vdash, \star)\})\Delta^*)$$

- Después de \$ $\vdash$ no aparece el símbolo %:

$$s_{1,5} = \Delta^* \$\{\vdash, (\vdash, \star)\}(\Delta \setminus \{\%\})^*$$

CONT-REG es PSPACE-completo: Demostración

- Entre $\$ \vdash$ y $\%$ se tiene menos de $n+1$ símbolos distintos del símbolo $\%$:

$$s_{1,6} = \Delta^*\$\{\vdash, (\vdash, \star)\}(\varepsilon + (\Delta \setminus \{\%\}))^n \%\Delta^*$$

- Entre $\$ \vdash$ y $\%$ se tiene más de $n+1$ símbolos distintos de $\%$:

$$s_{1,7} = \Delta^*\$\{\vdash, (\vdash, \star)\}(\Delta \setminus \{\%\})^{n+2}(\Delta \setminus \{\%\})^* \%\Delta^*$$

- Entre los n símbolos que aparecen después de $\$ \vdash$, hay al menos uno que no corresponden a w:

$$s_{1,8} = \sum_{i=1}^{n} \left(\Delta^*\$\{\vdash, (\vdash, \star)\}\Delta^{i-1}(\Delta \setminus \{a_i, (a_i, \star)\})\Delta^* \right)$$

CONT-REG es PSPACE-completo: Demostración

- El símbolo $n+1$ que aparece después de $\$\vdash$ no es $\mathtt{B}$:

$$s_{1,9} \quad = \quad \Delta^*\$\{\vdash,(\vdash,\star)\}\Delta^n(\Delta \setminus \{\mathtt{B},(\mathtt{B},\star)\})\Delta^*$$

- Entre los $n+2$ símbolos que aparecen después de \$, no hay ninguno marcado con el símbolo $\star$:

$$s_{1,10} \quad = \quad \Delta^*\$(\Delta \setminus (\Gamma \times \{\star\}))^{n+2}\Delta^*$$

- Entre los $n+2$ símbolos que aparecen después de \$, hay al menos dos marcados con el símbolo $\star$:

$$s_{1,11} \quad = \quad \sum_{1 \leq i < j \leq n+2} \left(\Delta^*\$\Delta^{i-1}(\Gamma \times \{\star\})\Delta^{j-(i+1)}(\Gamma \times \{\star\})\Delta^* \right)$$

CONT-REG es PSPACE-completo: Demostración

- Después de un símbolo % no aparece inmediatamente el símbolo $\vdash$:

$$s_{1,12} = \Delta^* \%(\varepsilon + (\Delta \setminus \{\vdash, (\vdash, \star)\})\Delta^*)$$

- Después de $\% \vdash$ hay menos de $s_M(n)$ símbolos:

$$s_{1,13} = \Delta^* \%\{\vdash, (\vdash, \star)\}(\varepsilon + \Delta)^{s_M(n)-1}$$

- Entre $\% \vdash$ y # aparecen menos de $s_M(n)$ símbolos distintos de #:

$$s_{1,14} = \Delta^* \%\{\vdash, (\vdash, \star)\}(\varepsilon + (\Delta \setminus \{\#\}))^{s_M(n)-1}\#\Delta^*$$

CONT-REG es PSPACE-completo: Demostración

- Después de $\%\vdash$ aparecen más de $s_M(n)$ símbolos distintos de $\#$:

$$s_{1,15} \quad = \quad \Delta^* \%\{\vdash, (\vdash, \star)\}(\Delta \setminus \{\#\})^{s_M(n)+1}\Delta^*$$

- Entre los $s_M(n)$ símbolos que aparecen después de $\%\vdash$, hay uno que no está en Γ:

$$s_{1,16} \quad = \quad \sum_{i=1}^{s_M(n)} \left(\Delta^* \%\{\vdash, (\vdash, \star)\}\Delta^{i-1}(\Delta \setminus (\Gamma \cup (\Gamma \times \{\star\})))\Delta^* \right)$$

CONT-REG es PSPACE-completo: Demostración

- Entre los $s_M(n)+1$ símbolos que aparecen después de %, no hay ninguno marcado con el símbolo $\star$:

$$s_{1,17} \quad = \quad \Delta^* \% (\Delta \setminus (\Gamma \times \{\star\}))^{s_M(n)+1} \Delta^*$$

- Entre los $s_M(n)+1$ símbolos que aparecen después de %, hay al menos dos marcados con el símbolo $\star$:

$$s_{1,18} \quad = \quad \sum_{1 \leq i < j \leq s_M(n)+1} \left(\Delta^* \% \Delta^{i-1} (\Gamma \times \{\star\}) \Delta^{j-(i+1)} (\Gamma \times \{\star\}) \Delta^* \right)$$

CONT-REG es PSPACE-completo: Demostración

s_2 se define como la unión de las siguientes expresiones:

- La primera configuración en la palabra no tiene al estado inicial:

$$s_{2,1} \quad = \quad \#(Q \setminus \{q_0\})\Delta^*$$

- La primera configuración en la palabra no tiene a la cabeza lectora de la cinta de lectura en la posición 1:

$$s_{2,2} \quad = \quad \#q_0\$\Delta a_1 \Delta^*$$

CONT-REG es PSPACE-completo: Demostración

- La primera configuración en la palabra no tiene a la cabeza lectora de la cinta de trabajo en la posición 1:

$$s_{2,3} \quad = \quad \#q_0\$ \vdash (a_1, \star)a_2 \cdots a_n \mathrm{B}\,\%\Delta\Gamma\Delta^*$$

- La primera configuración en la palabra no tiene sólo blancos en la cinta de trabajo:

$$s_{2,4} \quad = \quad \#q_0\$ \vdash (a_1, \star)a_2 \cdots a_n \mathrm{B}\,\% \vdash$$
$$\left[(\Delta \setminus \{(\mathrm{B}, \star)\})\Delta^* + \sum_{i=2}^{s_M(n)} \left((\mathrm{B}, \star)\mathrm{B}^{i-2}(\Delta \setminus \{\mathrm{B}\})\Delta^*\right)\right]$$

CONT-REG es PSPACE-completo: Demostración

s_3 se define como:

$$s_3 \quad = \quad \Delta^*\#(Q \setminus \{q_m\})\$\Delta^{n+2}\,\%\Delta^{s_M(n)+1}$$

CONT-REG es PSPACE-completo: Demostración

s_4 se define como la unión de las siguientes expresiones:

- Una celda de la cinta de trabajo que no es apuntada por la cabeza lectora cambió de contenido:

$$s_{4,1} = \sum_{i=1}^{s_M(n)+1} \sum_{a \in \Gamma} \Big(\Delta^* \# Q \$ \Delta^{n+2} \, \% \Gamma^{i-1} a \Gamma^{s_M(n)+1-i} \; \# Q \$ \Delta^{n+2} \, \% \Gamma^{i-1} ((\Gamma \setminus \{a\}) + ((\Gamma \times \{\star\}) \setminus \{(a, \star)\})) \Gamma^{s_M(n)+1-i} \Delta^* \Big)$$

CONT-REG es PSPACE-completo: Demostración

- Hay un cambio de estado para un triple en $Q \times \Gamma \times \Gamma$ que no está en el dominio de δ:

$$s_{4,2} = \sum_{i=1}^{n+2} \sum_{j=1}^{s_M(n)+1} \sum_{a \in \Gamma} \sum_{b \in \Gamma} \sum_{q \in Q \,:\, \delta(q,a,b) \text{ no está definido}} \left(\Delta^* \# q \$ \Delta^{i-1}(a, \star) \Delta^{n+2-i} \,\%\, \Delta^{j-1}(b, \star) \Delta^{s_M(n)+1-j} \right.$$

$$\left. \#(Q \setminus \{q\}) \$ \Delta^{n+2} \,\%\, \Delta^{s_M(n)+1} \Delta^* \right)$$

CONT-REG es PSPACE-completo: Demostración

- Hay un cambio en la posición de la cabeza lectora de la cinta de lectura para un triple en $Q \times \Gamma \times \Gamma$ que no está en el dominio de δ:

$$s_{4,3} = \sum_{i=1}^{n+2} \sum_{j=1}^{s_M(n)+1} \sum_{a \in \Gamma} \sum_{b \in \Gamma} \sum_{q \in Q \,:\, \delta(q,a,b) \text{ no está definido}} \Big(\Delta^* \# q \$ \Delta^{i-1}(a, \star) \Delta^{n+2-i} \,\%\, \Delta^{j-1}(b, \star) \Delta^{s_M(n)+1-j} \# Q \$ \Delta^{i-1} \Gamma \Delta^{n+2-i} \,\%\, \Delta^{s_M(n)+1} \Delta^* \Big)$$

CONT-REG es PSPACE-completo: Demostración

- Hay un cambio en la posición de la cabeza lectora de la cinta de trabajo para un triple en $Q \times \Gamma \times \Gamma$ que no está en el dominio de δ:

$$s_{4,4} = \sum_{i=1}^{n+2} \sum_{j=1}^{s_M(n)+1} \sum_{a \in \Gamma} \sum_{b \in \Gamma} \sum_{q \in Q \,:\, \delta(q,a,b) \text{ no está definido}} \Big(\Delta^* \# q \$ \Delta^{i-1} (a, \star) \Delta^{n+2-i} \,\%\, \Delta^{j-1} (b, \star) \Delta^{s_M(n)+1-j} \# Q \$ \Delta^{n+2} \,\%\, \Delta^{j-1} \Gamma \Delta^{s_M(n)+1-j} \Delta^* \Big)$$

CONT-REG es PSPACE-completo: Demostración

- Hay un cambio en el contenido de la celda de la cinta de trabajo apuntada por la cabeza lectora para un triple en $Q \times \Gamma \times \Gamma$ que no está en el dominio de δ:

$$s_{4,5} = \sum_{i=1}^{n+2} \sum_{j=1}^{s_M(n)+1} \sum_{a \in \Gamma} \sum_{b \in \Gamma} \sum_{q \in Q \,:\, \delta(q,a,b) \text{ no está definido}} \Big(\Delta^* \# q \$ \Delta^{i-1} (a, \star) \Delta^{n+2-i} \,\%\, \Delta^{j-1} (b, \star) \Delta^{s_M(n)+1-j} \# Q \$ \Delta^{n+2} \,\%\, \Delta^{j-1} ((\Gamma \times \{\star\}) \setminus \{(b, \star)\}) \Delta^{s_M(n)+1-j} \Delta^* \Big)$$

CONT-REG es PSPACE-completo: Demostración

- El cambio de estado para un triple en $Q \times \Gamma \times \Gamma$ no está de acuerdo con δ:

$$s_{4,6} = \sum_{i=1}^{n+2} \sum_{j=1}^{s_M(n)+1} \sum_{a\in\Gamma} \sum_{b\in\Gamma} \sum_{q\in Q} \sum_{q'\in Q \,:\, \delta(q,a,b)=(q'',X,c,Y) \text{ y } q'\neq q''} \Big(\Delta^* \# q \$ \Delta^{i-1}(a,\star)\Delta^{n+2-i} \,\%\, \Delta^{j-1}(b,\star)\Delta^{s_M(n)+1-j} \# q' \$ \Delta^{n+2} \,\%\, \Delta^{s_M(n)+1}\Delta^* \Big)$$

CONT-REG es PSPACE-completo: Demostración

- El cambio de la posición de la cabeza lectora de la cinta de lectura para un triple en $Q \times \Gamma \times \Gamma$ no está de acuerdo con δ:

$$s_{4,7} = \sum_{i=1}^{n+2} \sum_{j=1}^{s_M(n)+1} \sum_{a\in\Gamma} \sum_{b\in\Gamma} \sum_{q\in Q} \sum_{k\in[1,n+2]\,:\,\delta(q,a,b)=(q'',X,c,Y) \text{ y } i+X\neq k}$$
$$\Big(\Delta^* \# q \$ \Delta^{i-1}(a,\star)\Delta^{n+2-i} \,\%\, \Delta^{j-1}(b,\star)\Delta^{s_M(n)+1-j}$$
$$\#Q\$\Delta^{k-1}(\Gamma\times\{\star\})\Delta^{n+2-k} \,\%\, \Delta^{s_M(n)+1}\Delta^* \Big)$$

CONT-REG es PSPACE-completo: Demostración

- El cambio de la posición de la cabeza lectora de la cinta de trabajo para un triple en $Q \times \Gamma \times \Gamma$ no está de acuerdo con δ:

$$s_{4,8} = \sum_{i=1}^{n+2} \sum_{j=1}^{s_M(n)+1} \sum_{a \in \Gamma} \sum_{b \in \Gamma} \sum_{q \in Q} \sum_{k \in [1, s_M(n)+1] \,:\, \delta(q,a,b) = (q'', X, c, Y) \text{ y } j+Y \neq k}$$

$$\Big(\Delta^* \# q \$ \Delta^{i-1}(a, \star) \Delta^{n+2-i} \,\%\, \Delta^{j-1}(b, \star) \Delta^{s_M(n)+1-j}$$

$$\# Q \$ \Delta^{n+2} \,\%\, \Delta^{k-1}(\Gamma \times \{\star\}) \Delta^{s_M(n)+1-k} \Delta^* \Big)$$

CONT-REG es PSPACE-completo: Demostración

- El cambio en el valor de la celda apuntada por la cabeza lectora de la cinta de trabajo para un triple en $Q \times \Gamma \times \Gamma$ no está de acuerdo con δ:

$$s_{4,9} = \sum_{i=1}^{n+2} \sum_{j=1}^{s_M(n)+1} \sum_{a\in\Gamma} \sum_{b\in\Gamma} \sum_{q\in Q} \sum_{d\in\Gamma \,:\, \delta(q,a,b)=(q'',X,c,Y) \text{ y } c\neq d}$$
$$\Big(\Delta^* \# q \$ \Delta^{i-1}(a,\star)\Delta^{n+2-i} \,\%\, \Delta^{j-1}(b,\star)\Delta^{s_M(n)+1-j}$$
$$\# Q \$ \Delta^{n+2} \,\%\, \Delta^{j-1}(d+(d,\star))\Delta^{s_M(n)+1-j}\Delta^* \Big)$$

□

Un segundo problema completo para PSPACE

Sea TOTAL-REG el siguiente problema:

$$\text{TOTAL-REG} = \{r \mid r \text{ es una expresión regular sobre un alfabeto } \Sigma \text{ y } L(r) = \Sigma^*\}$$

Teorema (Meyer & Stockmeyer)

TOTAL-REG es PSPACE-completo.

Ejercicio

Demuestre el teorema.

Un tercer problema completo para PSPACE

Decimos que dos expresiones regulares r_1 y r_2 son equivalentes si $L(r_1) = L(r_2)$.

- Usamos la notación $r_1 \equiv r_2$

Sea EQUIV-REG el siguiente problema:

$$\text{EQUIV-REG} = \{(r_1, r_2) \mid r_1 \text{ y } r_2 \text{ son expresiones regulares tales que } r_1 \equiv r_2\}$$

Un tercer problema completo para PSPACE

Teorema (Meyer & Stockmeyer)

EQUIV-REG es PSPACE-completo.

Ejercicio

Demuestre que CONT-REG $\leq_{\mathrm{m}}^{\log}$ EQUIV-REG.

Otro problema completo para PSPACE

Sabemos que SAT es completo para la clase NP.

- Dada una fórmula proposicional φ, el problema a resolver es verificar si existe una asignación de verdad que satisface φ

Podemos entonces escribir $\exists x_1 \cdots \exists x_k \, \varphi$ en lugar de φ.

- Por ejemplo: $\exists p \exists q \exists r \, (p \vee \neg q) \wedge (\neg p \vee q \vee r) \wedge (\neg p \vee \neg r)$

¿Qué pasaría si usáramos el cuantificador universal?

- ¿Qué significa $\forall p \forall q \, (p \vee q) \wedge (\neg p \vee \neg q)$?

Otro problema completo para PSPACE

Una *fórmula proposicional cuantificada* es de la forma:

$$Q_1 x_1 \, Q_2 x_2 \cdots Q_k x_k \, \varphi$$

donde:

- φ es una fórmula proposicional sobre las variables $x_1, \ldots, x_k$
- cada $Q_i \in \{\exists, \forall\}$

La semántica de estas fórmulas se define como para el caso de la lógica de primer-orden.

- ¿Es *válida* la fórmula $\exists q \forall p \, (p \vee q) \wedge (\neg p \vee \neg q)$?
- ¿Y qué pasa en el caso de $\forall p \exists q \, (p \vee q) \wedge (\neg p \vee \neg q)$?

Otro problema completo para PSPACE

El lenguaje QBF se define como:

$$\text{QBF} = \{\varphi \mid \varphi \text{ es una fórmula proposicional cuantificada y } \varphi \text{ es válida}\}$$

Teorema

QBF es PSPACE-completo.

Ejercicio

Demuestre que QBF está en PSPACE.

Completitud sobre PSPACE

También es importante conocer problemas completos para clases de complejidad sobre PSPACE.

- Nuestro estudio va a concluir con las clases EXPTIME y NEXPTIME

Vamos a ver que estas clases están relacionas con PTIME y NP.

- Esto nos va a dar algunos problemas completos para NEXPTIME

¿Es EXPTIME $\neq$ NEXPTIME?

Teorema

Si EXPTIME $\neq$ NEXPTIME, entonces PTIME $\neq$ NP.

Demostración: Suponga que PTIME = NP.

Sea $L \in$ NEXPTIME: Existe una MT no determinista M que acepta L y funciona en tiempo $O(2^{n^k})$.

Por demostrar: $L \in$ EXPTIME.

- Vamos a utilizar la técnica de "padding"
- Agregando símbolos a la entrada se puede cambiar la complejidad de un problema

¿Es EXPTIME $\neq$ NEXPTIME?

Sea $L' = \{w\#^{2^{|w|^k}} \mid w \in L\}$.

- Suponemos que $\#$ es un símbolo nuevo

Como $L \in$ NEXPTIME, se tiene que $L' \in$ NP.

- ¿Por qué?

Pero por hipótesis tenemos que PTIME $=$ NP, de lo que concluimos que $L' \in$ PTIME.

- Existe una MT determinista M' que acepta L' y funciona en tiempo $O(n^\ell)$

¿Es EXPTIME $\neq$ NEXPTIME?

Como $L' \in$ PTIME, podemos construir una MT determinista $M^\star$ que acepta L y funciona en tiempo $O(2^{n^c})$.

- Con entrada w: $M^\star$ genera la palabra $w\#^{2^{|w|^k}}$ y después utiliza M' para verificar si $w\#^{2^{|w|^k}} \in L'$

Concluimos que $L \in$ EXPTIME.

La relación entre NP y NEXPTIME

Vimos una primera relación entre las clases NP y NEXPTIME.

- La técnica de padding nos dice que cada problema en NEXPTIME corresponde con un problema en NP pero con una entrada exponencialmente más sucinta

Esta idea nos permite generar problema completos para la clase NEXPTIME.

Pero antes tenemos que mostrar como representar las entradas de manera sucinta.

- Vamos a utilizar circuitos Booleanos

Circuitos Booleanos

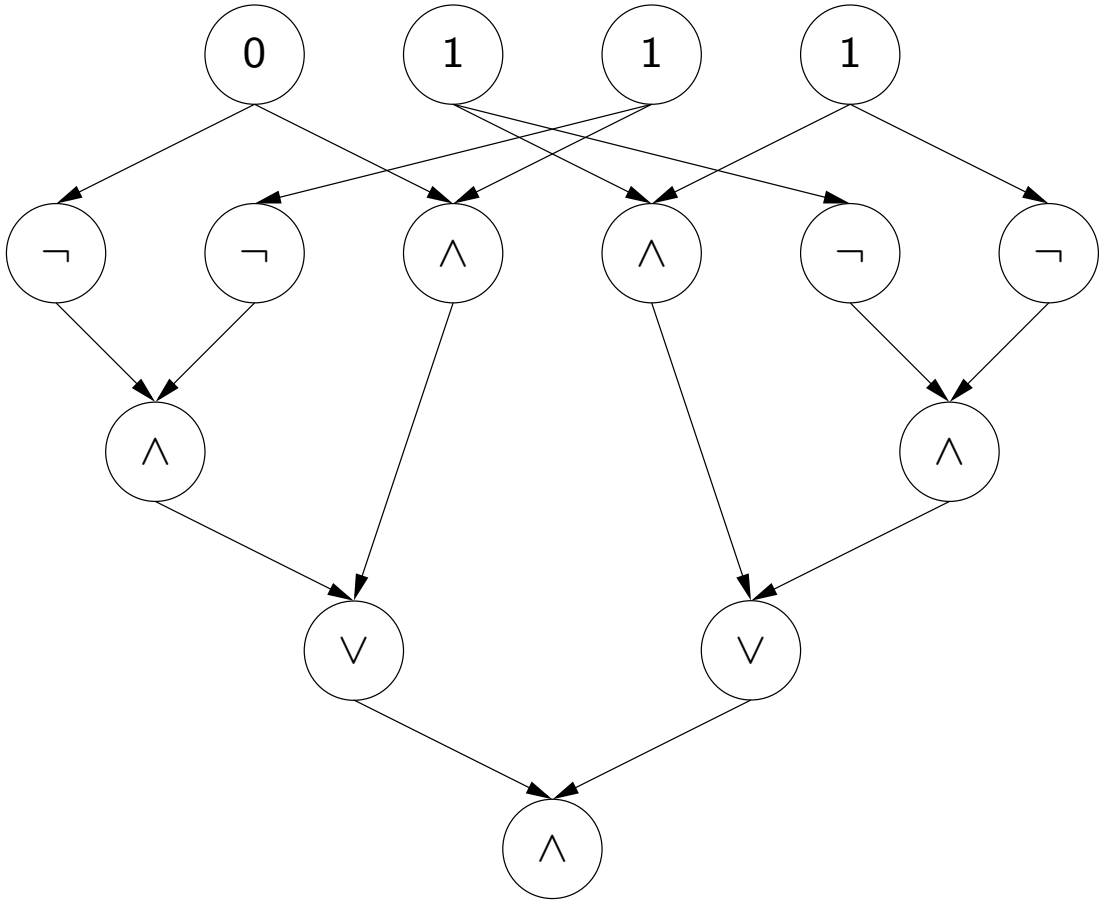
Un circuito Booleano es un grafo dirigido acíclico tal que:

- Cada nodo sin arcos de entrada tiene etiqueta 0 ó 1
- Cada nodo con arcos de entrada tiene etiqueta $\neg$, $\wedge$ o $\vee$
  - Un nodo con etiqueta $\neg$ tienen un arco de entrada
- Existe un único nodo sin arcos de salida

Notación

- *Nodo de entrada: Sin arcos de entrada*
- *Nodo interior: Con arcos de entrada*
- *Nodo de salida: Sin arcos de salida*

Circuito Booleano: Ejemplo

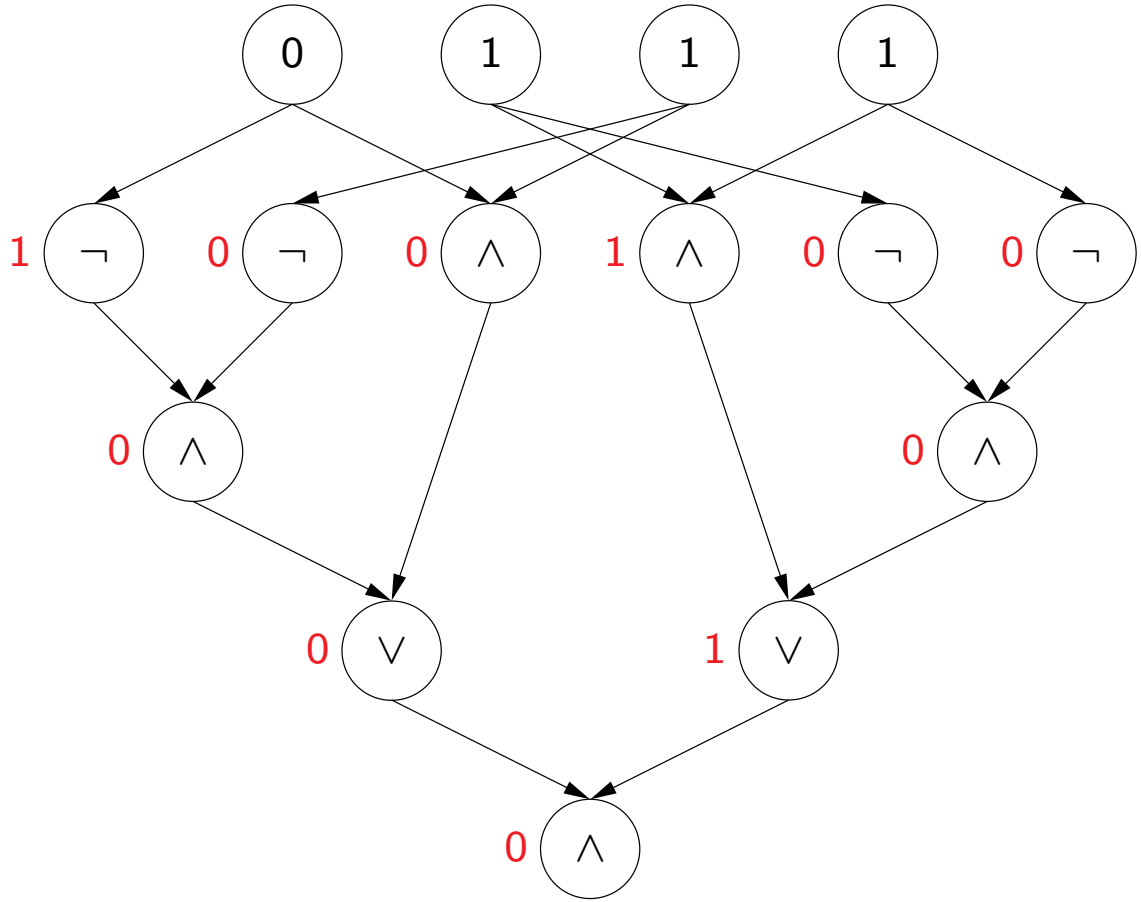


Circuitos Booleanos como funciones

Cada circuito Booleano puede ser interpretado como una función:

- Toma como entrada los valores en los nodos de entrada
- Asocia a cada nodo interior N el resultado de evaluar la etiqueta de N sobre los valores asociados a los nodos que tienen arcos a N
- Tiene como resultado el valor asociado al nodo de salida

Circuitos Booleanos como funciones: Ejemplo



Circuitos Booleanos como funciones

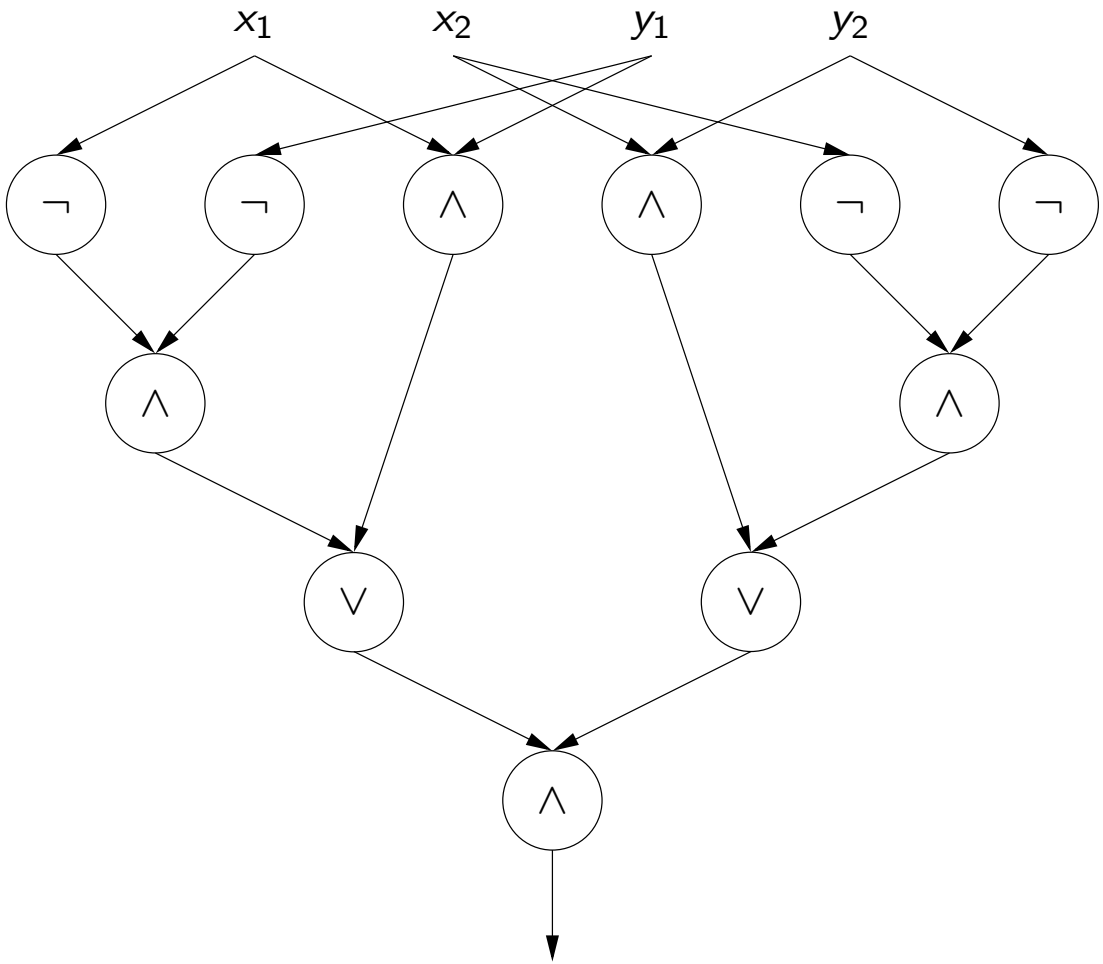
Circuito en el ejemplo anterior representa la función:

$$f(x_1, x_2, y_1, y_2) = \begin{cases} 1 & x_1 = y_1 \text{ y } x_2 = y_2 \\ 0 & \text{en otro caso} \end{cases}$$

Para indicar la función que representa un circuito:

- Reemplazamos las etiquetas 0 y 1 por variables
- Usamos un arco sin nodo de destino para indicar la salida

Circuito Booleano como función: Ejemplo



Representando grafos como circuito Booleanos

Vamos a representar cada grafo G como un circuito Booleano C_G.

Suponga que G tiene 2^n nodos:

- C_G tiene $2n$ entradas
- A cada nodo le corresponde un número binario de largo n

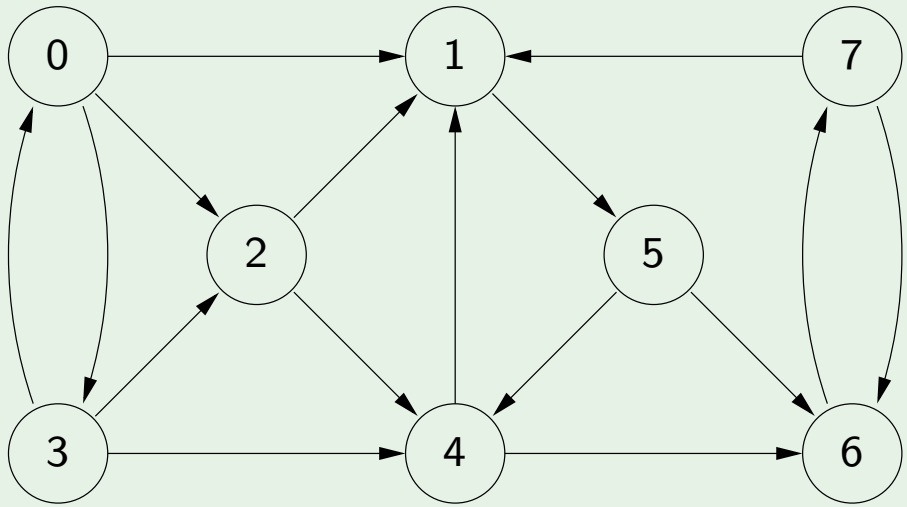
Sean a, b dos nodos en G, y sean $a_1 \cdots a_n$, $b_1 \cdots b_n$ sus representaciones binarias.

- Se tiene que: (a, b) es un arco en G si y sólo si $C_G(a_1, \ldots, a_n, b_1, \ldots, b_n) = 1$

Representando grafos como circuito Booleanos

Ejercicio

1. Represente el siguiente grafo como un circuito Booleano:



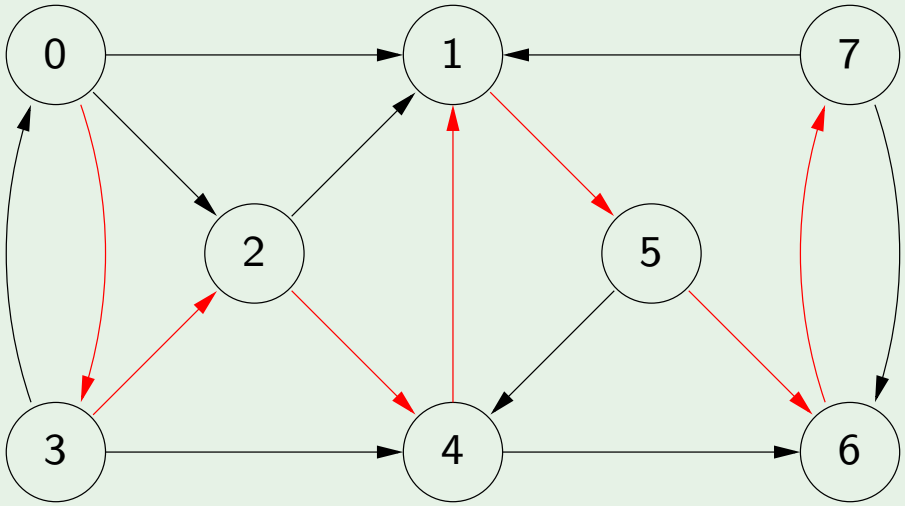
2. ¿Cómo representaría un clique con 2^k nodos? ¿De qué tamaño es su representación?

Un problema NEXPTIME-completo: Succinct-HP

Un camino en un grafo es *Hamiltoniano* si pasa exactamente una vez por cada nodo del grafo.

Ejemplo

El camino en rojo es Hamiltoniano:



Un problema NEXPTIME-completo: Succinct-HP

Un problema muy conocido:

$$\text{HP} \quad = \quad \{G \mid G \text{ tiene un camino Hamiltoniano}\}$$

¿Cuál es la complejidad de HP?

Teorema

HP es NP-completo.

Un problema NEXPTIME-completo: Succinct-HP

Ahora consideramos la versión sucinta de HP:

$$\text{Succinct-HP} \quad = \quad \{C_G \mid G \text{ tiene un camino Hamiltoniano}\}$$

¿Cuál es la complejidad de Succinct-HP?

Teorema

Succinct-HP es NEXPTIME-completo.

Ejercicio

Demuestre que Succinct-HP está en NEXPTIME.

Otro problema completo para NEXPTIME

También existe una versión sucinta de 3-CNF-SAT.

- ¿Cómo representaría una conjunción de clausulas de manera sucinta?
- Este problema también es NEXPTIME-completo