

La Jerarquía Polinomial

IIC3810

Motivación: coloración de grafos

Una grafo $G = (N, A)$ se dice k -coloreable si existe una función $f : N \rightarrow \{1, \dots, k\}$ tal que:

- ▶ Si $(a, b) \in A$, entonces $f(a) \neq f(b)$

El número cromático de un grafo G , denotado como $\chi(G)$, se define como el menor número k tal que G es k -coloreable.

- ▶ Vale decir, G es k -coloreable y no $(k - 1)$ -coloreable

El problema de coloración de un grafo

Considere los siguientes problemas relacionados con el cálculo del número cromático de un grafo:

$$\text{CROM}^{\leq} = \{(G, k) \mid G \text{ es un grafo,} \\ k \text{ es un número natural y } \chi(G) \leq k\}$$

$$\text{CROM}^{\geq} = \{(G, k) \mid G \text{ es un grafo,} \\ k \text{ es un número natural y } \chi(G) \geq k\}$$

Ejercicio

Demuestre que $\text{CROM}^{\leq}$ es NP-completo y $\text{CROM}^{\geq}$ es co-NP-completo.

Cálculo del número cromático

Considere ahora el problema de decisión asociado al cálculo exacto del número cromático de un grafo:

$$\text{CROM} = \{(G, k) \mid G \text{ es un grafo,} \\ k \text{ es un número natural y } \chi(G) = k\}$$

¿Cuál es la complejidad de CROM?

- ¿Está este problema en NP? ¿Está en co-NP? ¿Es completo para alguna de estas clases?

Podemos caracterizar CROM en términos de dos lenguajes en NP.

Ejercicio

Encuentre lenguajes L_1 y L_2 en NP tales que $\text{CROM} = L_1 \setminus L_2$

CROM no está “lejos” de NP

- ▶ De hecho, si $\text{PTIME} = \text{NP}$, entonces CROM está en PTIME
- ▶ CROM puede ser solucionado resolviendo dos problemas en NP. Vemos a estos problemas como sub-rutinas u oráculos

Motivación: evaluación de expresiones aritméticas

Una expresión aritmética es definida recursivamente por las siguientes reglas:

- ▶ Si $n \in \mathbb{N}$, entonces n es una expresión aritmética
- ▶ Si e_1 y e_2 son expresiones aritméticas, entonces $(e_1 + e_2)$ y $(e_1 \cup e_2)$ son expresiones aritméticas

Ejemplo

5, $(5 + 7)$ y $((5 + 7) \cup (4 \cup (6 + 11)))$ son expresiones aritméticas.

El lenguaje $L(e)$ definido por una expresión aritmética e es definido recursivamente por las siguiente reglas:

- ▶ Si $e = n$ con $n \in \mathbb{N}$, entonces $L(e) = \{n\}$
- ▶ Si $e = (e_1 + e_2)$, entonces $L(e) = \{n + m \mid n \in L(e_1) \text{ y } m \in L(e_2)\}$
- ▶ Si $e = (e_1 \cup e_2)$, entonces $L(e) = L(e_1) \cup L(e_2)$

Ejemplo

El lenguaje definido por algunas expresiones aritméticas:

- ▶ $L(5) = \{5\}$ y $L(7) = \{7\}$
- ▶ Si $e = (5 + 7)$, entonces $L(e) = \{n + m \mid n \in L(5) \text{ y } m \in L(7)\} = \{12\}$
- ▶ Si $e = (6 + 11)$, entonces $L(e) = \{17\}$
- ▶ Si $e = (4 \cup (6 + 11))$, entonces $L(e) = L(4) \cup L((6 + 11)) = \{4\} \cup \{17\} = \{4, 17\}$
- ▶ Si $e = (1 \cup 3) + (7 \cup 13)$, entonces $L(e) = \{8, 10, 14, 16\}$

Evaluación de expresiones aritméticas

El problema de evaluación de expresiones aritméticas es definido de la siguiente forma:

$$\text{EVAL} = \{(e, k) \mid e \text{ es una expresión aritmética,} \\ k \text{ es un número natural y } k \in L(e)\}$$

Ejercicio

Demuestre que EVAL es un problema NP-completo.

Equivalencia de expresiones aritméticas

Considere el siguiente problema sobre expresiones aritméticas:

$$\text{EQUIV} = \{(e_1, e_2) \mid e_1 \text{ y } e_2 \text{ son expresiones aritméticas tales que } L(e_1) = L(e_2)\}$$

¿Cuál es la complejidad de EQUIV?

- ▶ ¿Está en NP o co-NP?
- ▶ ¿Puede encontrar dos lenguajes L_1 y L_2 en NP tales que $\text{EQUIV} = L_1 \setminus L_2$?
- ▶ ¿Puede al menos demostrar que si $\text{PTIME} = \text{NP}$, entonces EQUIV está en PTIME?
 - ▶ ¿Tiene sentido la idea de sub-rutina (u oráculo) en este caso?

¿Qué tienen en común los problemas CROM y EQUIV?

- ▶ Si encontramos un algoritmo polinomial para un problema NP-completo, entonces estos problemas pueden ser resueltos en tiempo polinomial
- ▶ Un problema NP-completo puede ser visto como un *oráculo* para estos problemas

Vamos a introducir la noción de MT con oráculo.

- ▶ Vamos a mostrar que sirve para caracterizar a los problemas anteriores, y a muchos otros problemas ...

Definición

MT determinista con oráculo para $A \subseteq \Sigma^$:*

$$M^A = (Q, \Sigma, \Gamma, q_0, \delta, F)$$

- ▶ Q es un conjunto finito de estados tal que $q_?, q_{YES}, q_{NO} \in Q$
- ▶ Σ es un alfabeto finito tal que $\vdash, B \notin \Sigma$
- ▶ Γ es un alfabeto finito tal que $\Sigma \cup \{\vdash, B\} \subseteq \Gamma$
- ▶ $q_0 \in Q$ es el estado inicial
- ▶ $F \subseteq Q$ es un conjunto de estados finales
- ▶ δ es una función parcial:

$$\delta : Q \times \Gamma \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \square, \rightarrow\} \times \Gamma \times \{\leftarrow, \square, \rightarrow\}$$

*La segunda cinta es la **cinta de consulta***

MT con oráculo: Funcionamiento

La definición anterior puede extenderse fácilmente al caso de no determinismo.

Una MT M con oráculo para A funciona como una MT tradicional excepto cuando entra al estado $q_?$:

- ▶ Cinta de consulta: $\vdash wBB\cdots$, para $w \in \Sigma^*$

La cabeza lectora de la cinta de consulta está en la posición 1

- ▶ M invoca al oráculo para A , y su siguiente estado es q_{YES} o q_{NO} dependiendo de su respuesta
 - ▶ $w \in A$ si y sólo si el estado es q_{YES}

MT con oráculo: Tiempo de ejecución

El tiempo de ejecución de una MT con oráculo se define como para el caso de las MTs tradicionales.

- ▶ Una invocación al oráculo se considera como un paso

Ejemplo

CROM es aceptado por una MT determinista $M^{\text{CROM}^\leq}$ que funciona en tiempo $O(n)$

- ▶ También podemos usar SAT como oráculo. ¿En que tiempo funcionaría la MT?

Una primera clase definida en términos de MTs con oráculos:

Definición

$PTIME^A$: *Lenguajes L para los cuales existe una MT determinista M^A tal que $L = L(M^A)$ y M^A funciona en tiempo $O(n^k)$.*

Tenemos que:

- ▶ $CROM$ está en $PTIME^{SAT}$
 - ▶ También está contenido en $PTIME^A$, para cualquier problema A que es NP-completo
- ▶ NP y co-NP están contenidos en $PTIME^{SAT}$

Clases de complejidad y la noción de oráculo

Una definición mas general:

Definición

$$PTIME^{NP} = \bigcup_{A \in NP} PTIME^A$$

En realidad esta definición no es más general:

Proposición

$$PTIME^{NP} = PTIME^{SAT}$$

Ejercicio

Demuestre la proposición.

¿Cuál es la complejidad de EQUIV?

- ▶ ¿Está en PTIME^{NP} ?

Nuevamente la noción de oráculo nos puede ayudar a entender la complejidad de un problema.

Una segunda clase definida en términos de oráculos

Definición

- ▶ NP^A : Lenguajes L para los cuales existe una MT **no** determinista M^A tal que $L = L(M^A)$ y M^A funciona en tiempo $O(n^k)$
- ▶ NP^{NP} : $\bigcup_{A \in NP} NP^A$

Podemos describir mejor la complejidad de EQUIV.

Ejercicio

Demuestre que $EQUIV \in co-NP^{NP}$.

- ▶ Esto es equivalente a demostrar que $\overline{EQUIV} \in NP^{NP}$

La jerarquía polinomial

Podemos generalizar las definiciones anteriores considerando una clase de complejidad $\mathcal{C}$.

Definición

$$\triangleright PTIME^{\mathcal{C}}: \bigcup_{A \in \mathcal{C}} PTIME^A$$

$$\triangleright NP^{\mathcal{C}}: \bigcup_{A \in \mathcal{C}} NP^A$$

La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

Definición (Jerarquía Polinomial)

$$\Sigma_0^P = PTIME$$

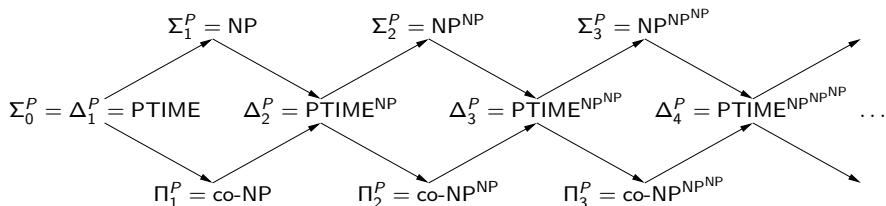
$$\Sigma_{n+1}^P = NP^{\Sigma_n^P} \quad n \geq 0$$

$$\Delta_{n+1}^P = PTIME^{\Sigma_n^P} \quad n \geq 0$$

$$\Pi_{n+1}^P = co-\Sigma_{n+1}^P \quad n \geq 0$$

La jerarquía polinomial

¿Cómo se ve la jerarquía polinomial en una figura?



Definición

$$PH = \bigcup_{k \geq 0} \Sigma_k^P$$

¿Cuál es la relación entre PH y EXPTIME?

- ▶ $PH \subseteq EXPTIME$

La jerarquía polinomial y EXPTIME

¿Puede ser cierto que $\text{EXPTIME} \subseteq \text{PH}$?

- ▶ Esto implicaría que PH tiene problemas completos

¿Puede tener problemas completos PH?

- ▶ Cada clase Σ_k^P es cerrada bajo $\leq_m^P$
 - ▶ Si $L_1 \leq_m^P L_2$ y $L_2 \in \Sigma_k^P$, entonces $L_1 \in \Sigma_k^P$
- ▶ Si PH tiene un problema completo, entonces la jerarquía polinomial colapsa a algún nivel finito

Proposición

Si la jerarquía polinomial no colapsa a algún nivel finito, entonces $\text{PH} \subsetneq \text{EXPTIME}$.

El colapso de la jerarquía polinomial: un resultado fundamental

Teorema

Para $k \geq 1$:

- (a) Si $\Sigma_k^P = \Pi_k^P$, entonces $PH = \Sigma_k^P$
- (b) Si $\Sigma_k^P = \Delta_k^P$, entonces $PH = \Delta_k^P$

Problemas completos en la jerarquía polinomial

El lenguaje QBF_i ($i \geq 1$) está formado por todas las fórmulas proposicionales cuantificadas que son válidas y de la forma:

$$\begin{aligned} &\exists x_{1,1} \cdots \exists x_{1,m_1} \\ &\forall x_{2,1} \cdots \forall x_{2,m_2} \\ &\exists x_{3,1} \cdots \exists x_{3,m_3} \\ &\quad \dots \\ &Q_i x_{i,1} \cdots Q_i x_{i,m_i} \varphi \end{aligned}$$

donde:

- ▶ $Q_i = \exists$ si i es impar y $Q_i = \forall$ si i es par
- ▶ φ es una fórmula proposicional sobre las variables $x_{1,1}, \dots, x_{1,m_1}, \dots, x_{i,1}, \dots, x_{i,m_i}$

Un problema completo para Σ_k^P

La clase de problemas $\{QBF_i\}_{i \geq 1}$ es adecuada para representar la jerarquía polinomial.

Teorema

Para cada $k \geq 1$, QBF_k es Σ_k^P -completo.

Otro problema completo para Σ_2^P

Teorema

$\overline{EQUIV}$ es Σ_2^P -completo.

- Se deduce que $EQUIV$ es Π_2^P -completo

Hay problemas naturales que son completos para los distintos niveles de la jerarquía polinomial.

Comentario final: Una noción de reducción más general

Los problemas en las transparencias anteriores son completos bajo la noción de reducción $\leq_m^P$

- ▶ Una propiedad fundamental de esta noción de reducción: si $L_1 \leq_m^P L_2$ y $L_2 \in \text{PTIME}$, entonces $L_1 \in \text{PTIME}$

Podemos generalizar $\leq_m^P$ manteniendo esta propiedad fundamental:

$$L_1 \leq_T^P L_2 \text{ si y sólo si } L_1 \in \text{PTIME}^{L_2}$$

Comentario final: Una noción de reducción más general

$\leq_T^p$ es llamada **reducción de Turing de tiempo polinomial**

Teorema

- ▶ Si $L_1 \leq_m^p L_2$, entonces $L_1 \leq_T^p L_2$
- ▶ Existen lenguajes L_1 y L_2 tales que $L_1 \leq_T^p L_2$ y $L_1 \not\leq_m^p L_2$

Demostración del teorema

La primera parte del teorema es fácil de demostrar, sólo vamos a demostrar la segunda parte.

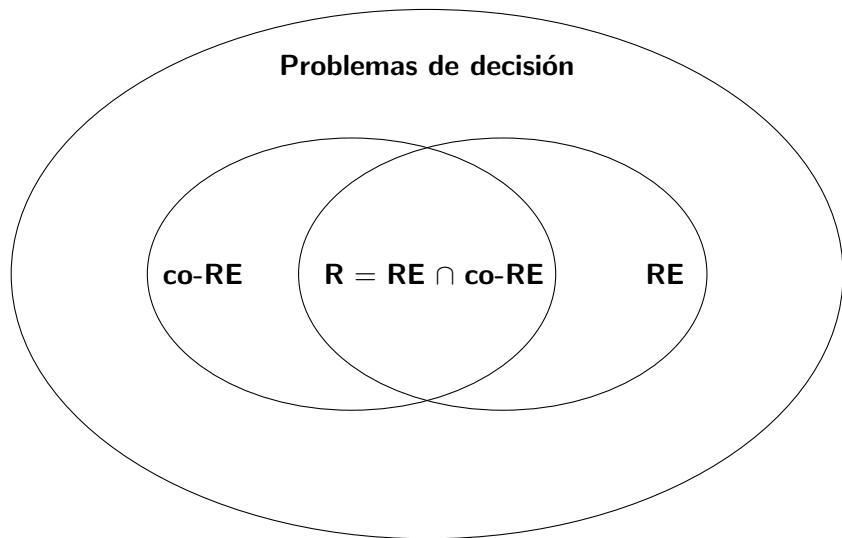
- Vamos a considerar los lenguajes sobre el alfabeto $\{0, 1\}$

Recuerde la definición de las siguientes clases de complejidad:

$R = \{L \subseteq \{0, 1\}^* \mid L \text{ es un lenguaje decidable}\}$

$RE = \{L \subseteq \{0, 1\}^* \mid L \text{ es un lenguaje recursivamente enumerable}\}$

Las clases R, RE y co-RE



Un lenguaje que separa RE de co-RE

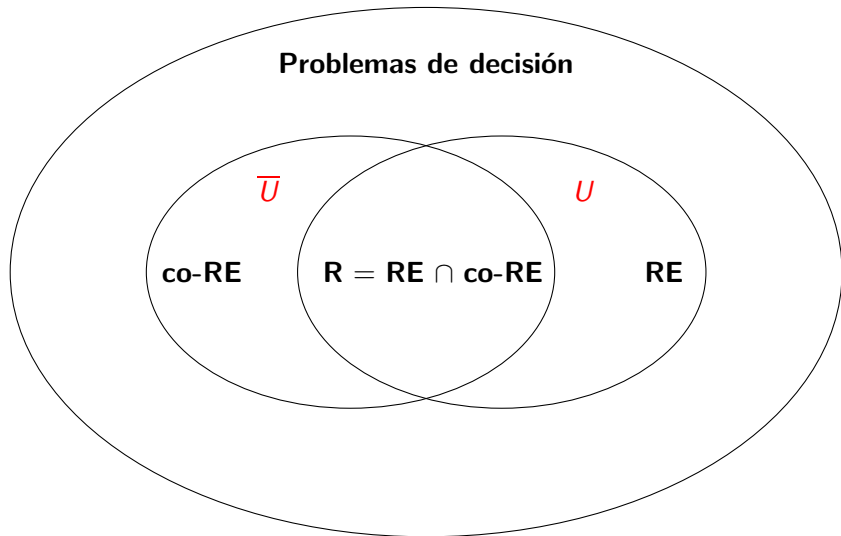
Recuerde la definición del problema de la parada de la máquina de Turing:

$$U = \{(M, w) \mid M \text{ es una MT, } w \in \{0, 1\}^* \text{ y} \\ M \text{ se detiene con entrada } w\}$$

Sabemos que $U \notin R$ y $U \in RE$

- ▶ Concluimos que $U \notin \text{co-RE}$
- ▶ Entonces tenemos que $\overline{U} \in \text{co-RE}$ y $\overline{U} \notin RE$

U y $\overline{U}$ en la figura



Parte final de la demostración

Sabemos que si $L \leq_m^P U$, entonces $L \in \text{RE}$

- ▶ ¿Cómo se demuestra esto?

Tenemos entonces que $\overline{U} \not\leq_m^P U$

Esto concluye la demostración puesto que $\overline{U} \leq_T^P U$

- ▶ De hecho, para cada lenguaje L se tiene que $\overline{L} \leq_T^P L$

