Complejidad Computacional

Santiago Figueira

Departamento de Computación - FCEN - UBA

clase 10

Clase 10

Máquinas con oráculo

Teorema de Baker, Gill, Solovay

La jerarquía polinomial y **NP** con oráculos

Máquinas con oráculo

Clase 10

- Máquinas con oráculo
- Teorema de Baker, Gill, Solovay
- La jerarquía polinomial y **NP** con oráculos

Máquinas con oráculo

Son como las máquinas de Turing que vimos, pero con estas diferencias:

- su comportamiento depende de un lenguaje $\mathcal{X} \subseteq \{0,1\}^*$
- tiene una cinta adicional de *consulta*:
- tiene 3 estados distinguidos más:
  - q_{consulta}, $q_{\text{resp:sí}}$, $q_{\text{resp:no}}$
- una nueva

INSTRUCCIÓN

si *estado* $== q_{consulta}$
entonces
supongamos que en la cinta de consulta
está escrito "$\triangleright\, x\, \Box$", *con* $x \in \{0,1\}^*$
si $x \in \mathcal{X}$, *pasar a* $q_{resp:sí}$
si no, pasar a $q_{resp:no}$

Máquinas con oráculo

- Mismas definiciones de cómputo, aceptación, rechazo, tiempo de cómputo, uso de espacio, etc.
- El comportamiento de M depende de la información del oráculo.
- Si M es una máquina (determinística o no-determinística) con oráculo, notamos $M^{\mathcal{X}}(x)$ a la salida de M con oráculo $\mathcal{X}$ y entrada x (si es que terminó).
- Si $M(x)$ termina, solo puede hacer una cantidad finita de consultas al oráculo. Si cambiamos el oráculo en elementos que nunca son consultados, el cómputo no cambia.
  - Ejemplo: $M^{\mathcal{X}}(x)$ termina y a lo largo del cómputo consulta $y_1, \ldots, y_m$ al oráculo. Para cualquier $\mathcal{Y}$ tal que $y_j \in \mathcal{X}$ sii $y_j \in \mathcal{Y}$ para $j = 1, \ldots, m$, tenemos $M^{\mathcal{X}}(x) = M^{\mathcal{Y}}(x)$.
  - Ejemplo: $M^{\mathcal{X}}(x)$ al paso t solo puede hacer finitas consultas al oráculo, independientemente de la respuesta que reciba. Al paso t no puede hacer más que t consultas.
- Las máquinas con oráculo se pueden listar, de la misma forma que hicimos con las máquinas determinísticas o las máquinas no-determinísticas.

Clases de complejidad relativizadas a oráculos

Clase de complejidad: $\mathbf{P}^{\mathcal{X}}, \mathbf{NP}^{\mathcal{X}}$

- $\mathbf{P}^{\mathcal{X}}$ es la clase de lenguajes decidibles por una máquina determinística que corre en tiempo polinomial y tiene acceso al oráculo $\mathcal{X}$.
- $\mathbf{NP}^{\mathcal{X}}$ es la clase de lenguajes decidibles por una máquina no-determinística que corre en tiempo polinomial y tiene acceso al oráculo $\mathcal{X}$.

Ejemplo: $\overline{\mathsf{SAT}} \in \mathbf{P}^{\mathsf{SAT}}$

Considerar la siguiente máquina determinística M con acceso a SAT y entrada x:

> *preguntar al oráculo si $x \in \mathsf{SAT}$ (escribir x en la cinta de consulta)*
> *si responde 'sí' (entra a $q_{resp:sí}$), devolver 0*
> *si no (entra a $q_{resp:no}$), devolver 1*

M corre en tiempo lineal independientemente del oráculo al que tenga acceso y $\mathcal{L}(M^{\mathsf{SAT}}) = \overline{\mathsf{SAT}}$.

Proposición

Si $\mathcal{X} \in \mathbf{P}$, entonces $\mathbf{P} = \mathbf{P}^{\mathcal{X}}$.

Demostración.

Probemos $\mathbf{P}^{\mathcal{X}} \subseteq \mathbf{P}$ (la otra inclusión es trivial).
Sea M una máquina determinística que corre en tiempo polinomial y decide $\mathcal{X}$.

Proposición

Si $\mathcal{X} \in \mathbf{P}$, entonces $\mathbf{P} = \mathbf{P}^{\mathcal{X}}$.

Demostración.

Probemos $\mathbf{P}^{\mathcal{X}} \subseteq \mathbf{P}$ (la otra inclusión es trivial).
Sea M una máquina determinística que corre en tiempo polinomial y decide $\mathcal{X}$.
Sea $\mathcal{L} \in \mathbf{P}^{\mathcal{X}}$ y supongamos que M' es una máquina determinística que corre en tiempo polinomial y decide $\mathcal{L}$ con acceso al oráculo $\mathcal{X}$.

Proposición

Si $\mathcal{X} \in \mathbf{P}$, entonces $\mathbf{P} = \mathbf{P}^{\mathcal{X}}$.

Demostración.

Probemos $\mathbf{P}^{\mathcal{X}} \subseteq \mathbf{P}$ (la otra inclusión es trivial).
Sea M una máquina determinística que corre en tiempo polinomial y decide $\mathcal{X}$.
Sea $\mathcal{L} \in \mathbf{P}^{\mathcal{X}}$ y supongamos que M' es una máquina determinística que corre en tiempo polinomial y decide $\mathcal{L}$ con acceso al oráculo $\mathcal{X}$.
La máquina M'' hace lo mismo que M' pero reemplaza cada pregunta x al oráculo $\mathcal{X}$ por la llamada a $M(x)$.
M'' corre en tiempo polinomial y decide $\mathcal{L}$.
Entonces $\mathcal{L} \in \mathbf{P}$. $\square$

Problema: EXPCOM (Cómputos exponenciales)

$$\mathsf{EXPCOM} = \{\langle M, x, 1^n \rangle : \text{la máquina determinística } M \text{ con entrada } x \text{ devuelve 1 en} \leq 2^n \text{ pasos}\}$$

Proposición

$\textbf{ExpTime} \subseteq \mathbf{P}^{\mathsf{EXPCOM}}$.

Problema: EXPCOM (Cómputos exponenciales)

$$\mathsf{EXPCOM} = \{\langle M, x, 1^n \rangle : \begin{array}{l}\text{la máquina determinística } M \text{ con entra-}\\ \text{da } x \text{ devuelve 1 en } \leq 2^n \text{ pasos}\end{array}\}$$

Proposición

$\textbf{ExpTime} \subseteq \mathbf{P}^{\mathsf{EXPCOM}}$.

Demostración

Sea $\mathcal{L} \in \textbf{DTime}(2^{n^c})$. Supongamos que $\mathcal{L} = \mathcal{L}(M)$ para una máquina determinística M que corre en tiempo $O(2^{n^c})$. Sea k tal que para todo $x \in \{0,1\}^*$ con $|x| > k$ tenemos que M con entrada x termina en $\leq 2^{|x|^{c+1}}$ pasos.

Problema: EXPCOM (Cómputos exponenciales)

$\mathsf{EXPCOM} = \{\langle M, x, 1^n \rangle : \begin{array}{l}\text{la máquina determinística } M \text{ con entra-}\\ \text{da } x \text{ devuelve 1 en } \leq 2^n \text{ pasos}\end{array}\}$

Proposición

$\mathbf{ExpTime} \subseteq \mathbf{P}^{\mathsf{EXPCOM}}$.

Demostración

Sea $\mathcal{L} \in \mathbf{DTime}(2^{n^c})$. Supongamos que $\mathcal{L} = \mathcal{L}(M)$ para una máquina determinística M que corre en tiempo $O(2^{n^c})$. Sea k tal que para todo $x \in \{0,1\}^*$ con $|x| > k$ tenemos que M con entrada x termina en $\leq 2^{|x|^{c+1}}$ pasos.

Considerar la siguiente máquina determinística M_1 con entrada x y acceso al oráculo EXPCOM:

si $|x| \leq k$, devolver 1 si $x \in \mathcal{L}$ y 0 en caso contrario
si no, preguntar al oráculo si $\langle M, x, 1^{|x|^{c+1}} \rangle$ y devolver su respuesta

Problema: EXPCOM (Cómputos exponenciales)

$$\mathsf{EXPCOM} = \{\langle M, x, 1^n\rangle : \text{la máquina determinística } M \text{ con entrada } x \text{ devuelve } 1 \text{ en } \leq 2^n \text{ pasos}\}$$

Proposición

$\mathbf{ExpTime} \subseteq \mathbf{P}^{\mathsf{EXPCOM}}$.

Demostración

Sea $\mathcal{L} \in \mathbf{DTime}(2^{n^c})$. Supongamos que $\mathcal{L} = \mathcal{L}(M)$ para una máquina determinística M que corre en tiempo $O(2^{n^c})$. Sea k tal que para todo $x \in \{0,1\}^*$ con $|x| > k$ tenemos que M con entrada x termina en $\leq 2^{|x|^{c+1}}$ pasos.

Considerar la siguiente máquina determinística M_1 con entrada x y acceso al oráculo EXPCOM:

si $|x| \leq k$, devolver 1 si $x \in \mathcal{L}$ y 0 en caso contrario
si no, preguntar al oráculo si $\langle M, x, 1^{|x|^{c+1}}\rangle$ y devolver su respuesta

M_1 corre en tiempo polinomial, entonces $\mathcal{L}(M_1^{\mathsf{EXPCOM}}) \in \mathbf{P}^{\mathsf{EXPCOM}}$ y

$$x \in \mathcal{L}(M_1^{\mathsf{EXPCOM}}) \quad \text{sii} \quad x \in \mathcal{L} = \mathcal{L}(M)$$

□

Proposición

$\mathbf{NP}^{\mathsf{EXPCOM}} \subseteq \textbf{E}\textsc{xp}\textbf{T}\textsc{ime}$.

Proposición

$\mathbf{NP}^{\mathsf{EXPCOM}} \subseteq \textbf{ExpTime}$.

Demostración.

Sea $\mathcal{L} \in \mathbf{NP}^{\mathsf{EXPCOM}}$ y sea N una máquina no-determinística que corre en tiempo polinomial y decide $\mathcal{L}$ con acceso al oráculo EXPCOM.

Proposición

$\mathbf{NP}^{\mathsf{EXPCOM}} \subseteq \textsc{\textbf{ExpTime}}$.

Demostración.

Sea $\mathcal{L} \in \mathbf{NP}^{\mathsf{EXPCOM}}$ y sea N una máquina no-determinística que corre en tiempo polinomial y decide $\mathcal{L}$ con acceso al oráculo EXPCOM.

En tiempo exponencial en $|x|$ podemos simular determinísticamente a N con entrada x y también cada consulta que hace N al oráculo EXPCOM.

Luego $\mathcal{L} \in \textsc{\textbf{ExpTime}}$. $\square$

Proposición

$\mathbf{NP}^{\mathsf{EXPCOM}} \subseteq \textbf{ExpTime}$.

Demostración.

Sea $\mathcal{L} \in \mathbf{NP}^{\mathsf{EXPCOM}}$ y sea N una máquina no-determinística que corre en tiempo polinomial y decide $\mathcal{L}$ con acceso al oráculo EXPCOM.

En tiempo exponencial en $|x|$ podemos simular determinísticamente a N con entrada x y también cada consulta que hace N al oráculo EXPCOM.

Luego $\mathcal{L} \in \textbf{ExpTime}$. □

Corolario

$\mathbf{P}^{\mathsf{EXPCOM}} = \mathbf{NP}^{\mathsf{EXPCOM}}$.

Demostración.

$\textbf{ExpTime} \subseteq \mathbf{P}^{\mathsf{EXPCOM}} \subseteq \mathbf{NP}^{\mathsf{EXPCOM}} \subseteq \textbf{ExpTime}$ □

Teorema de Baker, Gill, Solovay

Clase 10

Máquinas con oráculo

Teorema de Baker, Gill, Solovay

La jerarquía polinomial y **NP** con oráculos

Teorema de Baker, Gill, Solovay

Teorema (Baker, Gill, Solovay)

Existen oráculos $\mathcal{A}$ y $\mathcal{B}$ tal que $\mathbf{P}^{\mathcal{A}} = \mathbf{NP}^{\mathcal{A}}$ y $\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$.

Teorema de Baker, Gill, Solovay

Teorema (Baker, Gill, Solovay)

Existen oráculos $\mathcal{A}$ y $\mathcal{B}$ tal que $\mathbf{P}^{\mathcal{A}} = \mathbf{NP}^{\mathcal{A}}$ y $\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$.

$\mathbf{P}^{\mathcal{A}} = \mathbf{NP}^{\mathcal{A}}$.

Tomar $\mathcal{A} = \mathsf{EXPCOM}$. $\square$

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: definición de $\mathcal{U}_{\mathcal{B}}$, dado $\mathcal{B}$

Para cualquier $\mathcal{B} \subseteq \{0,1\}^*$ definimos

$$\mathcal{U}_{\mathcal{B}} = \{1^n \colon \exists x \in \mathcal{B}, |x| = n\}.$$

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: definición de $\mathcal{U}_{\mathcal{B}}$, dado $\mathcal{B}$

Para cualquier $\mathcal{B} \subseteq \{0,1\}^*$ definimos

$$\mathcal{U}_{\mathcal{B}} = \{1^n \colon \exists x \in \mathcal{B}, |x| = n\}.$$

Veamos que para cualquier $\mathcal{B}$, $\mathcal{U}_{\mathcal{B}} \in \mathbf{NP}^{\mathcal{B}}$. Definimos la máquina no-determinística N que con oráculo $\mathcal{B}$ y entrada y hace esto:

si y *no es de la forma* 1^n *para algún* n, *rechazar*
si no (supongamos que $y = 1^n$*),*
  inventar x *tal que* $|x| = n$
  consultar si $x \in \mathcal{B}$ *y devolver la respuesta*

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: definición de $\mathcal{U}_{\mathcal{B}}$, dado $\mathcal{B}$

Para cualquier $\mathcal{B} \subseteq \{0,1\}^*$ definimos

$$\mathcal{U}_{\mathcal{B}} = \{1^n \colon \exists x \in \mathcal{B}, |x| = n\}.$$

Veamos que para cualquier $\mathcal{B}$, $\mathcal{U}_{\mathcal{B}} \in \mathbf{NP}^{\mathcal{B}}$. Definimos la máquina no-determinística N que con oráculo $\mathcal{B}$ y entrada y hace esto:

si y no es de la forma 1^n para algún n, rechazar
si no (supongamos que $y = 1^n$),
  inventar x tal que $|x| = n$
  consultar si $x \in \mathcal{B}$ y devolver la respuesta

Para todo n tenemos

$$1^n \in \mathcal{L}(N^{\mathcal{B}}) \qquad \text{sii} \qquad \exists x \in \mathcal{B}, |x| = n \qquad \text{sii} \qquad 1^n \in \mathcal{U}_{\mathcal{B}}.$$

(y para y no de la forma 1^n tenemos $y \notin \mathcal{L}(N^{\mathcal{B}})$, $y \notin \mathcal{U}_{\mathcal{B}}$)

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: definición de $\mathcal{U}_{\mathcal{B}}$, dado $\mathcal{B}$

Para cualquier $\mathcal{B} \subseteq \{0,1\}^*$ definimos

$$\mathcal{U}_{\mathcal{B}} = \{1^n \colon \exists x \in \mathcal{B}, |x| = n\}.$$

Veamos que para cualquier $\mathcal{B}$, $\mathcal{U}_{\mathcal{B}} \in \mathbf{NP}^{\mathcal{B}}$. Definimos la máquina no-determinística N que con oráculo $\mathcal{B}$ y entrada y hace esto:

si y no es de la forma 1^n para algún n, rechazar
si no (supongamos que $y = 1^n$),
inventar x tal que $|x| = n$
consultar si $x \in \mathcal{B}$ y devolver la respuesta

Para todo n tenemos

$$1^n \in \mathcal{L}(N^{\mathcal{B}}) \qquad \text{sii} \qquad \exists x \in \mathcal{B}, |x| = n \qquad \text{sii} \qquad 1^n \in \mathcal{U}_{\mathcal{B}}.$$

(y para y no de la forma 1^n tenemos $y \notin \mathcal{L}(N^{\mathcal{B}})$, $y \notin \mathcal{U}_{\mathcal{B}}$)

A continuación definimos un $\mathcal{B}$ para el cual $\mathcal{U}_{\mathcal{B}} \notin \mathbf{P}^{\mathcal{B}}$.

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: propiedades de $\mathcal{B}$

Sea M_i la máquina determinística con oráculo representada por la expansión binaria de $i \in \mathbb{N}$ tal que para toda máquina M con oráculo existen infinitos i tal que $M = M_i$.

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: propiedades de $\mathcal{B}$

Sea M_i la máquina determinística con oráculo representada por la expansión binaria de $i \in \mathbb{N}$ tal que para toda máquina M con oráculo existen infinitos i tal que $M = M_i$.

Definimos $\mathcal{B} = \bigcup_i \mathcal{B}_i$ en etapas y definimos $(n_i)_{i \in \mathbb{N}}$ con estas propiedades:

- $\mathcal{B}_0 = \emptyset$ y $n_0 = 1$
- $\mathcal{B}_i \subseteq \mathcal{B}_{i+1}$ y $n_i < n_{i+1}$
- $x \in \mathcal{B}_i \Rightarrow |x| \leq n_i$ (en particular, cada $\mathcal{B}_i$ es finito)

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: idea de la prueba

Idea: diagonalizar.

si M_i con oráculo $\mathcal{B}$ corre en tiempo polinomial, entonces toma la decisión equivocada cuando la entrada es 1^{n_i}:

- si M_i acepta 1^{n_i} entonces nos aseguramos de que ninguna cadena de longitud n_i esté en $\mathcal{B}$ (y por lo tanto $1^{n_i} \notin \mathcal{U}_{\mathcal{B}}$)
- si M_i rechaza 1^{n_i}, entonces metemos en $\mathcal{B}$ alguna cadena de longitud n_i (y por lo tanto $1^{n_i} \in \mathcal{U}_{\mathcal{B}}$)

Así, ninguna máquina que corra en tiempo polinomial decide $\mathcal{U}_{\mathcal{B}}$, de modo que $\mathcal{U}_{\mathcal{B}} \notin \mathbf{P}^{\mathcal{B}}$.

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: construcción de $\mathcal{B}_i$ y para $i > 0$

- definimos $n_i > n_{i-1}$ y $n_i >$ máximo de las longitudes consultadas por $M_k^{\mathcal{B}_{i-1}}(1^{n_k})$ al tiempo $2^{n_k - 1}$ para todo $k < i$

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: construcción de $\mathcal{B}_i$ y para $i > 0$

- definimos $n_i > n_{i-1}$ y $n_i >$ máximo de las longitudes consultadas por $M_k^{\mathcal{B}_{i-1}}(1^{n_k})$ al tiempo 2^{n_k-1} para todo $k < i$
- simular M_i con entrada 1^{n_i} por 2^{n_i-1} pasos
  - si M_i consulta al oráculo por un x con $|x| < n_i$, le respondemos lo mismo que '¿$x \in \mathcal{B}_{i-1}$?'
  - si M_i consulta al oráculo por un x con $|x| \geq n_i$, le respondemos 'no'

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: construcción de $\mathcal{B}_i$ y para $i > 0$

- definimos $n_i > n_{i-1}$ y $n_i >$ máximo de las longitudes consultadas por $M_k^{\mathcal{B}_{i-1}}(1^{n_k})$ al tiempo 2^{n_k-1} para todo $k < i$
- simular M_i con entrada 1^{n_i} por 2^{n_i-1} pasos
  - si M_i consulta al oráculo por un x con $|x| < n_i$, le respondemos lo mismo que '¿$x \in \mathcal{B}_{i-1}$?'
  - si M_i consulta al oráculo por un x con $|x| \geq n_i$, le respondemos 'no'
- si M_i acepta 1^{n_i} en $\leq 2^{n_i-1}$ pasos, definimos $\mathcal{B}_i = \mathcal{B}_{i-1}$
  - $\mathcal{B}_i$ no contiene cadenas de tamaño n_i ($\mathcal{B}$ tampoco)

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: construcción de $\mathcal{B}_i$ y para $i > 0$

- definimos $n_i > n_{i-1}$ y $n_i >$ máximo de las longitudes consultadas por $M_k^{\mathcal{B}_{i-1}}(1^{n_k})$ al tiempo 2^{n_k-1} para todo $k < i$
- simular M_i con entrada 1^{n_i} por 2^{n_i-1} pasos
  - si M_i consulta al oráculo por un x con $|x| < n_i$, le respondemos lo mismo que '¿$x \in \mathcal{B}_{i-1}$?'
  - si M_i consulta al oráculo por un x con $|x| \geq n_i$, le respondemos 'no'
- si M_i acepta 1^{n_i} en $\leq 2^{n_i-1}$ pasos, definimos $\mathcal{B}_i = \mathcal{B}_{i-1}$
  - $\mathcal{B}_i$ no contiene cadenas de tamaño n_i ($\mathcal{B}$ tampoco)
- si M_i rechaza 1^{n_i} o no llegó a una decisión todavía en 2^{n_i-1} pasos, elegimos un x, $|x| = n_i$ que no haya sido consultado y definimos $\mathcal{B}_i = \mathcal{B}_{i-1} \cup \{x\}$.
  - tal x existe porque simulamos M_i por 2^{n_i-1} pasos y por lo tanto hay $\leq 2^{n_i-1}$ consultas, pero hay 2^{n_i} cadenas de tamaño n_i
  - agregar x no 'rompe' ninguna de las simulaciones de M_k a tiempo 2^{n_k-1}, para $k < i$. A la simulación de M_k por 2^{n_k-1} pasos le daba lo mismo el oráculo $\mathcal{B}_k$ o $\mathcal{B}_i$ (o $\mathcal{B}$)

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: verificación de que $\mathcal{U}_{\mathcal{B}} \notin \mathbf{P}^{\mathcal{B}}$

Supongamos que M es una máquina determinística y p es un polinomio tal que

- $M^{\mathcal{B}}$ corre en tiempo $p(n)$
- $M^{\mathcal{B}}$ acepta $\mathcal{U}_{\mathcal{B}}$

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: verificación de que $\mathcal{U}_{\mathcal{B}} \notin \mathbf{P}^{\mathcal{B}}$

Supongamos que M es una máquina determinística y p es un polinomio tal que

- $M^{\mathcal{B}}$ corre en tiempo $p(n)$
- $M^{\mathcal{B}}$ acepta $\mathcal{U}_{\mathcal{B}}$

Sea i suficientemente grande tal que $M_i = M$ y $2^{n_i - 1} > p(n_i)$. Simular $M_i = M$ por $2^{n_i - 1}$ pasos es suficiente para saber si M_i acepta o rechaza 1^{n_i}.

$\mathbf{P}^{\mathcal{B}} \neq \mathbf{NP}^{\mathcal{B}}$: verificación de que $\mathcal{U}_{\mathcal{B}} \notin \mathbf{P}^{\mathcal{B}}$

Supongamos que M es una máquina determinística y p es un polinomio tal que

- $M^{\mathcal{B}}$ corre en tiempo $p(n)$
- $M^{\mathcal{B}}$ acepta $\mathcal{U}_{\mathcal{B}}$

Sea i suficientemente grande tal que $M_i = M$ y $2^{n_i - 1} > p(n_i)$. Simular $M_i = M$ por $2^{n_i - 1}$ pasos es suficiente para saber si M_i acepta o rechaza 1^{n_i}.

- si $M_i^{\mathcal{B}}$ acepta 1^{n_i}, ninguna cadena de longitud n_i está en $\mathcal{B}$. Entonces $1^{n_i} \notin \mathcal{U}_{\mathcal{B}}$.
- si $M_i^{\mathcal{B}}$ rechaza 1^{n_i}, $\mathcal{B}_i$ contiene una cadena de tamaño n_i. Entonces $1^{n_i} \in \mathcal{U}_{\mathcal{B}}$.

Luego $M^{\mathcal{B}} = M_i^{\mathcal{B}}$ no puede decidir $\mathcal{U}_{\mathcal{B}}$ porque falla para la entrada 1^{n_i}. $\square$

Clases de complejidad relativizadas a clases de complejidad

Clase de complejidad: $\mathbf{P}^{\mathbf{C}}, \mathbf{NP}^{\mathbf{C}}$

$$\mathbf{P}^{\mathbf{C}} = \bigcup_{\mathcal{X} \in \mathbf{C}} \mathbf{P}^{\mathcal{X}}$$

$$\mathbf{NP}^{\mathbf{C}} = \bigcup_{\mathcal{X} \in \mathbf{C}} \mathbf{NP}^{\mathcal{X}}$$

Observación

Si $\mathcal{X} \in$ **C-completo**, entonces $\mathbf{P}^{\mathbf{C}} = \mathbf{P}^{\mathcal{X}}$ y $\mathbf{NP}^{\mathbf{C}} = \mathbf{NP}^{\mathcal{X}}$.

Ejemplo

$\mathbf{P}^{\mathbf{NP}} = \mathbf{P}^{\mathsf{SAT}} = \mathbf{P}^{\overline{\mathsf{SAT}}} = \mathbf{P}^{\mathbf{coNP}}$

La jerarquía polinomial y **NP** con oráculos

Clase 10

Máquinas con oráculo

Teorema de Baker, Gill, Solovay

La jerarquía polinomial y **NP** con oráculos

$\boldsymbol{\Sigma}^{\mathbf{p}}_{\boldsymbol{i+1}} = \mathbf{NP}^{\Sigma_i \mathsf{SAT}}$

Teorema

Para $i \geq 1$, $\boldsymbol{\Sigma}^{\mathbf{p}}_{\boldsymbol{i+1}} = \mathbf{NP}^{\Sigma_i \mathsf{SAT}}$.

Ejemplo

- $\boldsymbol{\Sigma}^{\mathbf{p}}_{\mathbf{1}} = \mathbf{NP}$ (ya lo vimos; no cae dentro del teorema, salvo que definamos $\Sigma_0\mathsf{SAT} = \emptyset$)
- $\boldsymbol{\Sigma}^{\mathbf{p}}_{\mathbf{2}} = \mathbf{NP}^{\Sigma_1\mathsf{SAT}} = \mathbf{NP}^{\mathsf{SAT}} = \mathbf{NP}^{\mathbf{NP}}$ (recordar que $\Sigma_1\mathsf{SAT} = \mathsf{SAT}$)
- $\boldsymbol{\Sigma}^{\mathbf{p}}_{\mathbf{3}} = \mathbf{NP}^{\Sigma_2\mathsf{SAT}} = \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}$

$\mathbf{\Sigma^p_{i+1}} \subseteq \mathbf{NP}^{\Sigma_i\mathsf{SAT}}$

Sea $\mathcal{L} \in \mathbf{\Sigma^p_{i+1}}$ y una máquina determinística M que corre en tiempo polinomial tal que

$$x \in \mathcal{L} \qquad \text{sii} \qquad \exists u_1 \forall u_2 \ldots Q u_{i+1}\ M(\langle x, u_1, \ldots, u_i \rangle) = 1$$

donde $|u_i| = q(|x|)$ para algún polinomio fijo q.

$\mathbf{\Sigma^p_{i+1}} \subseteq \mathbf{NP}^{\Sigma_i\mathsf{SAT}}$

Sea $\mathcal{L} \in \mathbf{\Sigma^p_{i+1}}$ y una máquina determinística M que corre en tiempo polinomial tal que

$$x \in \mathcal{L} \qquad \text{sii} \qquad \exists u_1 \forall u_2 \ldots Q u_{i+1}\ M(\langle x, u_1, \ldots, u_i \rangle) = 1$$

donde $|u_i| = q(|x|)$ para algún polinomio fijo q. Definamos

$$\mathcal{L}' = \{\langle x, u_1 \rangle \colon \forall u_2 \ldots Q_{i+1} u_{i+1}\ M(\langle x, u_1, \ldots, u_i \rangle) = 1\} \in \mathbf{\Pi^p_i}$$
$$\leq_{\mathrm{p}} \overline{\Sigma_i\mathsf{SAT}}$$

$\Sigma_{i+1}^{\mathbf{p}} \subseteq \mathbf{NP}^{\Sigma_i\mathsf{SAT}}$

Sea $\mathcal{L} \in \mathbf{\Sigma}_{i+1}^{\mathbf{p}}$ y una máquina determinística M que corre en tiempo polinomial tal que

$$x \in \mathcal{L} \qquad \text{sii} \qquad \exists u_1 \forall u_2 \ldots Q u_{i+1}\; M(\langle x, u_1, \ldots, u_i\rangle) = 1$$

donde $|u_i| = q(|x|)$ para algún polinomio fijo q. Definamos

$$\mathcal{L}' = \{\langle x, u_1\rangle \colon \forall u_2 \ldots Q_{i+1} u_{i+1}\; M(\langle x, u_1, \ldots, u_i\rangle) = 1\} \in \mathbf{\Pi}_i^{\mathbf{p}}$$
$$\leq_{\mathrm{p}} \overline{\Sigma_i\mathsf{SAT}}$$

Definimos una máquina no-determinística N que con oráculo $\mathcal{L}'$ y entrada x hace esto:

inventar u_1
consultar si $\langle x, u_1\rangle \in \mathcal{L}'$ *y devolver su respuesta*

$\mathbf{\Sigma}^{\mathbf{p}}_{\boldsymbol{i+1}} \subseteq \mathbf{NP}^{\Sigma_i\mathsf{SAT}}$

Sea $\mathcal{L} \in \mathbf{\Sigma}^{\mathbf{p}}_{\boldsymbol{i+1}}$ y una máquina determinística M que corre en tiempo polinomial tal que

$$x \in \mathcal{L} \qquad \text{sii} \qquad \exists u_1 \forall u_2 \dots Q u_{i+1}\ M(\langle x, u_1, \dots, u_i \rangle) = 1$$

donde $|u_i| = q(|x|)$ para algún polinomio fijo q. Definamos

$$\mathcal{L}' = \{\langle x, u_1\rangle \colon \forall u_2 \dots Q_{i+1} u_{i+1}\ M(\langle x, u_1, \dots, u_i\rangle) = 1\} \in \mathbf{\Pi}^{\mathbf{p}}_{\boldsymbol{i}} \\ \leq_{\mathrm{p}} \overline{\Sigma_i\mathsf{SAT}}$$

Definimos una máquina no-determinística N que con oráculo $\mathcal{L}'$ y entrada x hace esto:

> *inventar* u_1
> *consultar si* $\langle x, u_1\rangle \in \mathcal{L}'$ *y devolver su respuesta*

N corre en tiempo lineal.
Tenemos $\mathcal{L}(N^{\mathcal{L}'}) = \mathcal{L}$ y $\mathcal{L}' \leq_{\mathrm{p}} \overline{\Sigma_i\mathsf{SAT}}$.
Entonces $\mathcal{L} \in \mathbf{NP}^{\overline{\Sigma_i\mathsf{SAT}}} = \mathbf{NP}^{\Sigma_i\mathsf{SAT}}$. □

$\mathbf{NP}^{\Sigma_i\mathsf{SAT}} \subseteq \mathbf{\Sigma}^{\mathbf{p}}_{i+1}$

Sea $\mathcal{L} \in \mathbf{NP}^{\Sigma_i\mathsf{SAT}}$ y sea N una máquina no-determinística que corre en tiempo polinomial $t(n)$ tal que con oráculo $\Sigma_i\mathsf{SAT}$ decide $\mathcal{L}$.

$x \in \mathcal{L}$ sii existe un cómputo $u \in \{0,1\}^{t(|x|)}$ de $N^{\Sigma_i\mathsf{SAT}}$ con entrada x que llega a $q_{\text{sí}}$

$\mathbf{NP}^{\Sigma_i\mathsf{SAT}} \subseteq \mathbf{\Sigma^p_{i+1}}$

Sea $\mathcal{L} \in \mathbf{NP}^{\Sigma_i\mathsf{SAT}}$ y sea N una máquina no-determinística que corre en tiempo polinomial $t(n)$ tal que con oráculo $\Sigma_i\mathsf{SAT}$ decide $\mathcal{L}$.

$x \in \mathcal{L}$ sii existe un cómputo $u \in \{0,1\}^{t(|x|)}$ de $N^{\Sigma_i\mathsf{SAT}}$ con entrada x que llega a $q_{\text{sí}}$

A lo largo de este cómputo u, N hace consultas $\varphi_1, \ldots, \varphi_k$ ($k \leq t(|x|)$), del tipo

$$\varphi_j = \underbrace{\exists \bar{u}_1^j \forall \bar{u}_2^j \ldots Q \bar{u}_i^j}_{i-1 \text{ alternancias}} \quad \psi_j(\bar{u}_1, \ldots, \bar{u}_i)$$

para $j = 1, \ldots, k$ y recibe respuesta $r_j \in \{0,1\}$ ('sí' $= q_{\text{resp:sí}} = 1$ o no $= q_{\text{resp:no}} = 0$)

$\mathbf{NP}^{\Sigma_i\mathsf{SAT}} \subseteq \mathbf{\Sigma}^{\mathbf{p}}_{i+1}$

Sea $\mathcal{L} \in \mathbf{NP}^{\Sigma_i\mathsf{SAT}}$ y sea N una máquina no-determinística que corre en tiempo polinomial $t(n)$ tal que con oráculo $\Sigma_i\mathsf{SAT}$ decide $\mathcal{L}$.

$x \in \mathcal{L}$ sii existe un cómputo $u \in \{0,1\}^{t(|x|)}$ de $N^{\Sigma_i\mathsf{SAT}}$ con entrada x que llega a $q_{\text{sí}}$

A lo largo de este cómputo u, N hace consultas $\varphi_1, \ldots, \varphi_k$ ($k \leq t(|x|)$), del tipo

$$\varphi_j = \underbrace{\exists \bar{u}_1^j \forall \bar{u}_2^j \ldots Q \bar{u}_i^j}_{i-1 \text{ alternancias}} \quad \psi_j(\bar{u}_1, \ldots, \bar{u}_i)$$

para $j = 1, \ldots, k$ y recibe respuesta $r_j \in \{0,1\}$ ('sí' $= q_{\text{resp:sí}} = 1$ o no $= q_{\text{resp:no}} = 0$)

- si $r_j = 1$ entonces existe $\bar{v}_1^j$ tal que $\bar{v}_1^j \models \forall \bar{v}_2^j \ldots Q \bar{v}_i^j \ \psi_j(\bar{v}_1^j, \bar{v}_2^j, \ldots, \bar{v}_i^j)$ ($i-2$ alternancias)

$\mathbf{NP}^{\Sigma_i\mathsf{SAT}} \subseteq \mathbf{\Sigma}^{\mathbf{p}}_{i+1}$

Sea $\mathcal{L} \in \mathbf{NP}^{\Sigma_i\mathsf{SAT}}$ y sea N una máquina no-determinística que corre en tiempo polinomial $t(n)$ tal que con oráculo $\Sigma_i\mathsf{SAT}$ decide $\mathcal{L}$.

$x \in \mathcal{L}$ sii existe un cómputo $u \in \{0,1\}^{t(|x|)}$ de $N^{\Sigma_i\mathsf{SAT}}$ con entrada x que llega a $q_{\text{sí}}$

A lo largo de este cómputo u, N hace consultas $\varphi_1, \ldots, \varphi_k$ $(k \leq t(|x|))$, del tipo

$$\varphi_j = \underbrace{\exists \bar{u}_1^j \forall \bar{u}_2^j \ldots Q \bar{u}_i^j}_{i-1 \text{ alternancias}} \quad \psi_j(\bar{u}_1, \ldots, \bar{u}_i)$$

para $j = 1, \ldots, k$ y recibe respuesta $r_j \in \{0,1\}$ ('sí' $= q_{\text{resp:sí}} = 1$ o no $= q_{\text{resp:no}} = 0$)

- si $r_j = 1$ entonces existe $\bar{v}_1^j$ tal que
  $\bar{v}_1^j \models \forall \bar{v}_2^j \ldots Q \bar{v}_i^j \;\; \psi_j(\bar{v}_1^j, \bar{v}_2^j, \ldots, \bar{v}_i^j)$ ($i-2$ alternancias)
- si $r_j = 0$ entonces para todo $\bar{w}_1^j$ tenemos
  $\bar{w}_1^j \not\models \forall \bar{w}_2^j \ldots Q \bar{w}_i^j \;\; \psi_j(\bar{w}_1^j, \bar{w}_2^j, \ldots, \bar{w}_i^j)$, o sea
  $\bar{w}_1^j \models \exists \bar{w}_2^j \ldots \overline{Q} \bar{w}_i^j \;\; \neg\psi_j(\bar{w}_1^j, \bar{w}_2^j, \ldots, \bar{w}_i^j)$ ($i-2$ alternancias)

$x \in \mathcal{L}$

sii

$$\exists u \ \exists (r_j)_j \ \exists (\bar{v}_1^j)_j \ \forall (\bar{w}_1^j)_j$$

N acepta x siguiendo el cómputo u, recibe como respuestas $r_1, \ldots, r_k$ (en orden) y para $j = 1, \ldots, k$
$r_j = 1$ y $\bar{v}_1^j \models \forall \bar{v}_2^j \exists \bar{v}_3^j \ldots Q \bar{v}_i^j \ \psi_j(\bar{v}_1^j, \bar{v}_2^j, \ldots, \bar{v}_i^j)$ o bien
$r_j = 0$ y $\bar{w}_1^j \models \exists \bar{w}_2^j \forall \bar{w}_3^j \ldots \overline{Q} \bar{w}_i^j \ \neg\psi_j(\bar{w}_1^j, \bar{w}_2^j, \ldots, \bar{w}_i^j)$

$x \in \mathcal{L}$

sii

$$\exists u\ \exists (r_j)_j\ \exists (\bar{v}_1^j)_j\ \forall (\bar{w}_1^j)_j$$

N acepta x siguiendo el cómputo u, recibe como respuestas $r_1, \ldots, r_k$ (en orden) y para $j = 1, \ldots, k$
$r_j = 1$ y $\bar{v}_1^j \models \forall \bar{v}_2^j \exists \bar{v}_3^j \ldots Q \bar{v}_i^j\ \psi_j(\bar{v}_1^j, \bar{v}_2^j, \ldots, \bar{v}_i^j)$ o bien
$r_j = 0$ y $\bar{w}_1^j \models \exists \bar{w}_2^j \forall \bar{w}_3^j \ldots \overline{Q} \bar{w}_i^j\ \neg\psi_j(\bar{w}_1^j, \bar{w}_2^j, \ldots, \bar{w}_i^j)$

sii

$$\overbrace{\overbrace{\exists u, (r_j)_j, (\bar{v}_1^j)_j}^{\exists}\ \overbrace{\forall (\bar{w}_1^j)_j \forall (\bar{v}_2^j)_j}^{\forall}\ \overbrace{\exists (\bar{w}_2^j)_j \exists (\bar{v}_3^j)_j \forall (\bar{w}_3^j)_j \ldots Q(\bar{v}_i^j)_j \overline{Q}(\bar{w}_i^j)_j}^{i-2 \text{ alternancias}}}^{i \text{ alternancias}}$$

$$\left.\begin{array}{l} N \text{ acepta } x \text{ siguiendo el cómputo } u \text{, recibe como} \\ \text{respuestas } r_1, \ldots, r_k \text{ (en orden) y para } j = 1, \ldots, k\text{:} \\ r_j = 1 \text{ y } \models \psi_j(\bar{v}_1^j, \bar{v}_2^j, \ldots \bar{v}_i^j) \text{ o bien} \\ r_j = 0 \text{ y } \models \neg\psi_j(\bar{w}_1^j, \bar{w}_2^j, \ldots \bar{w}_i^j) \end{array}\right\} \begin{array}{l} \text{computable} \\ \text{en tiempo} \\ \text{polinomial} \end{array}$$

Entonces $\mathcal{L} \in \boldsymbol{\Sigma}^{\mathbf{p}}_{\boldsymbol{i+1}}$.

Ejemplo (esquema): Supongamos que N acepta x y a lo largo del computo u, N hace 1 sola consulta r al oráculo $\Sigma_4\mathsf{SAT}$:

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

Entonces:

$r = 1 \Rightarrow$ ($q_{\text{resp:sí}}$)

$r = 0 \Rightarrow$ ($q_{\text{resp:sí}}$)

$$\begin{tikzcd}
\exists v_1 \arrow[d] & \forall v_2 \arrow[d] & \exists v_3 \arrow[d] & \forall v_4 \arrow[d] & \psi(v_1, v_2, v_3, v_4) \\
\forall w_1 \arrow[ur] & \exists w_2 \arrow[ur] & \forall w_3 \arrow[ur] & \exists w_4 & \neg\psi(w_1, w_2, w_3, w_4)
\end{tikzcd}$$

$$\underbrace{\exists u \quad \exists r \quad \exists v_1 \quad \forall w_1 \forall v_2 \; \exists w_2 \exists v_3 \; \forall w_3 \forall v_4 \; \exists w_4}_{4 \text{ alternancias} \Rightarrow \mathbf{\Sigma_5^p}} \; [\text{Polinomial}]$$

sii $\quad x \in \mathcal{L}$

$\square$

$\mathbf{P}$ relativizado a oráculos

Clase de complejidad: $\boldsymbol{\Delta}_{\boldsymbol{i}}^{\mathbf{p}}$

$$\boldsymbol{\Delta}_{\mathbf{1}}^{\mathbf{p}} = \mathbf{P}$$

$$\boldsymbol{\Delta}_{\boldsymbol{i}+\mathbf{1}}^{\mathbf{p}} = \mathbf{P}^{\boldsymbol{\Sigma}_{\boldsymbol{i}}^{\mathbf{p}}}$$

Ejemplo

- $\boldsymbol{\Delta}_{\mathbf{2}}^{\mathbf{p}} = \mathbf{P}^{\boldsymbol{\Sigma}_{\mathbf{1}}^{\mathbf{p}}} = \mathbf{P}^{\mathbf{NP}}$
- $\boldsymbol{\Delta}_{\mathbf{3}}^{\mathbf{p}} = \mathbf{P}^{\boldsymbol{\Sigma}_{\mathbf{2}}^{\mathbf{p}}} = \mathbf{P}^{\mathbf{NP}^{\mathbf{NP}}}$

$\mathbf{P}$ relativizado a oráculos

Clase de complejidad: $\boldsymbol{\Delta}_i^{\mathbf{p}}$

$$\boldsymbol{\Delta}_{\mathbf{1}}^{\mathbf{p}} = \mathbf{P}$$

$$\boldsymbol{\Delta}_{\boldsymbol{i}+\mathbf{1}}^{\mathbf{p}} = \mathbf{P}^{\boldsymbol{\Sigma}_i^{\mathbf{p}}}$$

Ejemplo

- $\boldsymbol{\Delta}_{\mathbf{2}}^{\mathbf{p}} = \mathbf{P}^{\boldsymbol{\Sigma}_{\mathbf{1}}^{\mathbf{p}}} = \mathbf{P}^{\mathbf{NP}}$
- $\boldsymbol{\Delta}_{\mathbf{3}}^{\mathbf{p}} = \mathbf{P}^{\boldsymbol{\Sigma}_{\mathbf{2}}^{\mathbf{p}}} = \mathbf{P}^{\mathbf{NP}^{\mathbf{NP}}}$

Problema: LEXSAT (Mínima valuación en orden lex)

$$\mathsf{LEXSAT} = \{\langle v, \varphi\rangle : \begin{array}{l} v \text{ es la valuación más chica en orden} \\ \text{lexicográfico tal que } v \models \varphi \end{array}\}$$

Teorema

LEXSAT es $\boldsymbol{\Delta}_{\mathbf{2}}^{\mathbf{p}}$-completo.

$$
\begin{tikzcd}
 & \vdots & \\
 & \mathbf{\Delta_3^p = P^{NP^{NP}}} \arrow[u] & \\
 & \mathbf{\Sigma_2^p \cup \Pi_2^p = NP^{NP} \cup coNP^{NP}} \arrow[u] & \\
\mathbf{\Sigma_2^p = NP^{NP}} \arrow[ur] & & \mathbf{\Pi_2^p = coNP^{NP}} \arrow[ul] \\
 & \mathbf{\Sigma_2^p \cap \Pi_2^p = NP^{NP} \cap coNP^{NP}} \arrow[ul] \arrow[ur] & \\
 & \mathbf{\Delta_2^p = P^{NP}} \arrow[u] & \\
 & \mathbf{\Sigma_1^p \cup \Pi_1^p = NP \cup coNP} \arrow[u] & \\
\mathbf{\Sigma_1^p = NP} \arrow[ur] & & \mathbf{\Pi_1^p = coNP} \arrow[ul] \\
 & \mathbf{\Sigma_1^p \cap \Pi_1^p = NP \cap coNP} \arrow[ul] \arrow[ur] & \\
 & \mathbf{P = \Sigma_0^p = \Pi_0^p = \Delta_1^p} \arrow[u] &
\end{tikzcd}
$$