

# La Jerarquía Polinomial

IIC3810

# Motivación: Soluciones exactas

Al estudiar la clase NP consideramos preguntas “existenciales”:

- ▶ ¿Existe una asignación de verdad que satisface a una fórmula proposicional?
- ▶ ¿Existe una solución entera para un sistema de ecuaciones lineales enteras?
- ▶ ¿Existe un camino Hamiltoniano en un grafo?

¿Cambia la complejidad de estos problemas si hacemos preguntas “exactas”?

# Motivación: Soluciones exactas

Al estudiar la clase NP consideramos preguntas “existenciales”:

- ▶ ¿Existe una asignación de verdad que satisface a una fórmula proposicional?
- ▶ ¿Existe una solución entera para un sistema de ecuaciones lineales enteras?
- ▶ ¿Existe un camino Hamiltoniano en un grafo?

¿Cambia la complejidad de estos problemas si hacemos preguntas “exactas”?

- ▶ ¿Existe **exactamente** una asignación de verdad que satisface a una fórmula proposicional?

# Motivación: Soluciones exactas

Al estudiar la clase NP consideramos preguntas “existenciales”:

- ▶ ¿Existe una asignación de verdad que satisface a una fórmula proposicional?
- ▶ ¿Existe una solución entera para un sistema de ecuaciones lineales enteras?
- ▶ ¿Existe un camino Hamiltoniano en un grafo?

¿Cambia la complejidad de estos problemas si hacemos preguntas “exactas”?

- ▶ ¿Existe **exactamente** una asignación de verdad que satisface a una fórmula proposicional?
- ▶ ¿Cuál es la complejidad de este problema? ¿Está en NP?

# Soluciones exactas: La clase DP

Para responder a las preguntas anteriores necesitamos definir una nueva clase de complejidad.

Idea detrás de esta clase: Noción de solución exacta se reduce a hacer dos consultas.

- ▶ ¿Es cierto que existe una asignación de verdad que satisface la fórmula proposicional  $\varphi$ ?
- ▶ ¿Es cierto que no existen al menos dos asignaciones de verdad que satisfacen  $\varphi$ ?

# Soluciones exactas: La clase DP

Para responder a las preguntas anteriores necesitamos definir una nueva clase de complejidad.

Idea detrás de esta clase: Noción de solución exacta se reduce a hacer dos consultas.

- ▶ ¿Es cierto que existe una asignación de verdad que satisface la fórmula proposicional  $\varphi$ ?
- ▶ ¿Es cierto que no existen al menos dos asignaciones de verdad que satisfacen  $\varphi$ ?

## Definición

$L \in DP$  si y sólo si existen lenguajes  $L_1 \in NP$  y  $L_2 \in co-NP$  tales que  $L = L_1 \cap L_2$ .

# unique-SAT y la clase DP

Una definición formal del primer problema que queremos estudiar:

$\text{unique-SAT} = \{\varphi \mid \varphi \text{ es una fórmula proposicional tal que}$   
 $\text{existe exactamente una asignación de verdad que satisface } \varphi\}$

Ejercicio

Demuestre que  $\text{unique-SAT} \in \text{DP}$ .

# unique-SAT y la clase DP

Una definición formal del primer problema que queremos estudiar:

$\text{unique-SAT} = \{\varphi \mid \varphi \text{ es una fórmula proposicional tal que}$   
 $\text{existe exactamente una asignación de verdad que satisface } \varphi\}$

Ejercicio

Demuestre que  $\text{unique-SAT} \in \text{DP}$ .

¿Es unique-SAT completo para DP?



# unique-SAT y la clase DP

Una definición formal del primer problema que queremos estudiar:

$\text{unique-SAT} = \{\varphi \mid \varphi \text{ es una fórmula proposicional tal que}$   
 $\text{existe exactamente una asignación de verdad que satisface } \varphi\}$

Ejercicio

Demuestre que  $\text{unique-SAT} \in \text{DP}$ .

¿Es unique-SAT completo para DP?

- ▶ Este es un problema abierto

# unique-SAT y la clase DP

Una definición formal del primer problema que queremos estudiar:

unique-SAT =  $\{\varphi \mid \varphi \text{ es una fórmula proposicional tal que}$   
existe exactamente una asignación de verdad que satisface  $\varphi\}$

Ejercicio

Demuestre que unique-SAT  $\in$  DP.

¿Es unique-SAT completo para DP?

- ▶ Este es un problema abierto
- ▶ ¿Existen problemas completos para esta clase?

# Un primer problema completo para DP

Sea 3-CNF-SAT-UNSAT el siguiente lenguaje:

3-CNF-SAT-UNSAT =  $\{(\varphi, \psi) \mid \varphi \text{ y } \psi \text{ son conjunciones}$   
de 3-clausulas,  $\varphi$  es satisfacible y  $\psi$  no es satisfacible}

# Un primer problema completo para DP

Sea 3-CNF-SAT-UNSAT el siguiente lenguaje:

3-CNF-SAT-UNSAT =  $\{(\varphi, \psi) \mid \varphi \text{ y } \psi \text{ son conjunciones de 3-clausulas, } \varphi \text{ es satisfacible y } \psi \text{ no es satisfacible}\}$

Teorema

*3-CNF-SAT-UNSAT es DP-completo.*

# Un primer problema completo para DP

Sea 3-CNF-SAT-UNSAT el siguiente lenguaje:

3-CNF-SAT-UNSAT =  $\{(\varphi, \psi) \mid \varphi \text{ y } \psi \text{ son conjunciones de 3-clausulas, } \varphi \text{ es satisfacible y } \psi \text{ no es satisfacible}\}$

## Teorema

*3-CNF-SAT-UNSAT es DP-completo.*

## Ejercicio

Demuestre el teorema.

# Un problema natural que representan a DP

## Notación

$$\text{clique-number}(G) = \text{máx}\{k \mid G \text{ tiene un clique de tamaño } k\}$$

Usamos este número para definir dos problemas en grafos:

$$\begin{aligned}\text{CLIQUE} &= \{(G, k) \mid \text{clique-number}(G) \geq k\} \\ \text{exact-CLIQUE} &= \{(G, k) \mid \text{clique-number}(G) = k\}\end{aligned}$$

# Un problema natural que representan a DP

## Notación

$$\text{clique-number}(G) = \max\{k \mid G \text{ tiene un clique de tamaño } k\}$$

Usamos este número para definir dos problemas en grafos:

$$\begin{aligned}\text{CLIQUE} &= \{(G, k) \mid \text{clique-number}(G) \geq k\} \\ \text{exact-CLIQUE} &= \{(G, k) \mid \text{clique-number}(G) = k\}\end{aligned}$$

¿Cuál es la complejidad de CLIQUE?

- ▶ ¿Son distintas las complejidades de CLIQUE y exact-CLIQUE?

# Un problema natural que representan a DP

## Teorema

*exact-CLIQUE es DP-completo.*



# Un problema natural que representan a DP

## Teorema

*exact-CLIQUE es DP-completo.*

**Demostración:** Primero tenemos que demostrar que  $\text{exact-CLIQUE} \in \text{DP}$

▶ ¿Cómo se demuestra esto?

Después tenemos que demostrar que  $\text{exact-CLIQUE}$  es DP-hard.

# Un problema natural que representan a DP

## Teorema

*exact-CLIQUE es DP-completo.*

**Demostración:** Primero tenemos que demostrar que  $\text{exact-CLIQUE} \in \text{DP}$

- ▶ ¿Cómo se demuestra esto?

Después tenemos que demostrar que  $\text{exact-CLIQUE}$  es DP-hard.

- ▶ Vamos a reducir 3-CNF-SAT-UNSAT a  $\text{exact-CLIQUE}$

## exact-CLIQUE es DP-hard: Demostración

Vamos a utilizar una reducción usual de 3-CNF-SAT a CLIQUE.

## exact-CLIQUE es DP-hard: Demostración

Vamos a utilizar una reducción usual de 3-CNF-SAT a CLIQUE.

- ▶ Recordemos esta reducción a través del ejemplo:

$$\beta = (p \vee q \vee r) \wedge (\neg p \vee \neg q \vee s) \wedge (p \vee \neg q \vee \neg s)$$

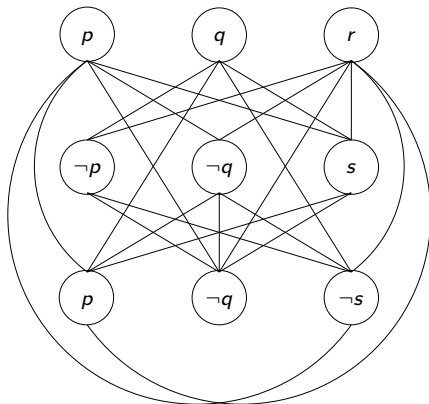
# exact-CLIQUE es DP-hard: Demostración

Vamos a utilizar una reducción usual de 3-CNF-SAT a CLIQUE.

- Recordemos esta reducción a través del ejemplo:

$$\beta = (p \vee q \vee r) \wedge (\neg p \vee \neg q \vee s) \wedge (p \vee \neg q \vee \neg s)$$

Grafo  $G_\beta$ :



## exact-CLIQUE es DP-hard: Demostración

Tenemos que:  $\beta \in 3\text{-CNF-SAT}$  si y sólo si  $(G_\beta, 3) \in \text{CLIQUE}$

# exact-CLIQUE es DP-hard: Demostración

Tenemos que:  $\beta \in 3\text{-CNF-SAT}$  si y sólo si  $(G_\beta, 3) \in \text{CLIQUE}$

En general: Si  $\varphi = C_1 \wedge \dots \wedge C_n$ , donde cada  $C_i$  es una clausula, entonces:

$\varphi \in 3\text{-CNF-SAT}$  si y sólo si  $(G_\varphi, n) \in \text{CLIQUE}$

# exact-CLIQUE es DP-hard: Demostración

Tenemos que:  $\beta \in 3\text{-CNF-SAT}$  si y sólo si  $(G_\beta, 3) \in \text{CLIQUE}$

En general: Si  $\varphi = C_1 \wedge \dots \wedge C_n$ , donde cada  $C_i$  es una clausula, entonces:

$\varphi \in 3\text{-CNF-SAT}$  si y sólo si  $(G_\varphi, n) \in \text{CLIQUE}$

Vamos a utilizar esta reducción en nuestra demostración.

- ▶ Pero antes: Necesitamos algunas operaciones sobre grafos



## exact-CLIQUE es DP-hard: Demostración

Dados: Grafos  $G_1 = (N_1, A_1)$  y  $G_2 = (N_2, A_2)$

# exact-CLIQUE es DP-hard: Demostración

Dados: Grafos  $G_1 = (N_1, A_1)$  y  $G_2 = (N_2, A_2)$

## Notación

$G_1 \uplus G_2$ : Unión disjunta de  $G_1$  y  $G_2$

► Suponiendo que  $N_1 \cap N_2 = \emptyset$ :  $G_1 \uplus G_2 = (N_1 \cup N_2, A_1 \cup A_2)$

# exact-CLIQUE es DP-hard: Demostración

Dados: Grafos  $G_1 = (N_1, A_1)$  y  $G_2 = (N_2, A_2)$

## Notación

$G_1 \uplus G_2$ : Unión disjunta de  $G_1$  y  $G_2$

► Suponiendo que  $N_1 \cap N_2 = \emptyset$ :  $G_1 \uplus G_2 = (N_1 \cup N_2, A_1 \cup A_2)$

Si  $N_1 \cap N_2 \neq \emptyset$ : Los nodos de  $G_2$  pueden ser renombrados para poder ejecutar la operación  $\uplus$

## exact-CLIQUE es DP-hard: Demostración

### Notación

$G_1 \times G_2 = (N_1 \times N_2, A)$ , donde:

$$A = \{((a_1, a_2), (b_1, b_2)) \mid (a_1, b_1) \in A_1 \text{ y } (a_2, b_2) \in A_2, \text{ o } \\ a_1 = b_1 \text{ y } (a_2, b_2) \in A_2, \text{ o } \\ (a_1, b_1) \in A_1 \text{ y } a_2 = b_2\}$$

## exact-CLIQUE es DP-hard: Demostración

### Notación

$G_1 \times G_2 = (N_1 \times N_2, A)$ , donde:

$$A = \{((a_1, a_2), (b_1, b_2)) \mid (a_1, b_1) \in A_1 \text{ y } (a_2, b_2) \in A_2, \text{ o } \\ a_1 = b_1 \text{ y } (a_2, b_2) \in A_2, \text{ o } \\ (a_1, b_1) \in A_1 \text{ y } a_2 = b_2\}$$

### Ejercicio

Si  $G_1$  y  $G_2$  tienen cliques con  $n_1$  y  $n_2$  elementos, respectivamente, ¿qué puede decir sobre los cliques en  $G_1 \times G_2$ ?

# exact-CLIQUE es DP-hard: Demostración

Tenemos los ingredientes para reducir de 3-CNF-SAT-UNSAT a exact-CLIQUE.

Dado  $(\varphi, \psi)$ , vamos a construir  $(G_{(\varphi, \psi)}, k_{(\varphi, \psi)})$  tal que:

$$\begin{aligned} &(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT} \\ &\quad \text{si y sólo si} \\ & (G_{(\varphi, \psi)}, k_{(\varphi, \psi)}) \in \text{exact-CLIQUE} \end{aligned}$$

# exact-CLIQUE es DP-hard: Demostración

Tenemos los ingredientes para reducir de 3-CNF-SAT-UNSAT a exact-CLIQUE.

Dado  $(\varphi, \psi)$ , vamos a construir  $(G_{(\varphi, \psi)}, k_{(\varphi, \psi)})$  tal que:

$$\begin{aligned} &(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT} \\ &\quad \text{si y sólo si} \\ & (G_{(\varphi, \psi)}, k_{(\varphi, \psi)}) \in \text{exact-CLIQUE} \end{aligned}$$

Suponemos que  $\varphi$  y  $\psi$  están formadas por  $m$  y  $n$  cláusulas, respectivamente, donde  $m \geq 2$ ,  $n \geq 2$  y  $m \neq n$ .

# exact-CLIQUE es DP-hard: Demostración

Tenemos los ingredientes para reducir de 3-CNF-SAT-UNSAT a exact-CLIQUE.

Dado  $(\varphi, \psi)$ , vamos a construir  $(G_{(\varphi, \psi)}, k_{(\varphi, \psi)})$  tal que:

$$\begin{aligned} (\varphi, \psi) &\in \text{3-CNF-SAT-UNSAT} \\ &\text{si y sólo si} \\ (G_{(\varphi, \psi)}, k_{(\varphi, \psi)}) &\in \text{exact-CLIQUE} \end{aligned}$$

Suponemos que  $\varphi$  y  $\psi$  están formadas por  $m$  y  $n$  cláusulas, respectivamente, donde  $m \geq 2$ ,  $n \geq 2$  y  $m \neq n$ .

- ¿Por qué podemos suponer que  $m \geq 2$ ,  $n \geq 2$  y  $m \neq n$ ?



# exact-CLIQUE es DP-hard: Demostración

Sea:

$$G_{(\varphi, \psi)} = (G_{\varphi} \uplus K_{m-1}) \times (G_{\psi} \uplus K_{n-1})$$

$$k_{(\varphi, \psi)} = m \cdot (n - 1)$$

# exact-CLIQUE es DP-hard: Demostración

Sea:

$$\begin{aligned}G_{(\varphi, \psi)} &= (G_{\varphi} \uplus K_{m-1}) \times (G_{\psi} \uplus K_{n-1}) \\k_{(\varphi, \psi)} &= m \cdot (n - 1)\end{aligned}$$

Tenemos que:

$\varphi$	$\psi$	Tamaño del mayor clique en $G_{(\varphi, \psi)}$
CNF-SAT	CNF-SAT	$m \cdot n$
CNF-SAT	$\overline{\text{CNF-SAT}}$	$m \cdot (n - 1)$
$\overline{\text{CNF-SAT}}$	CNF-SAT	$(m - 1) \cdot n$
$\overline{\text{CNF-SAT}}$	$\overline{\text{CNF-SAT}}$	$(m - 1) \cdot (n - 1)$

## exact-CLIQUE es DP-hard: Demostración

Dado que  $m \neq n$ :  $m \cdot (n - 1) \neq (m - 1) \cdot n$

Por lo tanto, dado que  $(m - 1) \cdot (n - 1) < m \cdot (n - 1) < m \cdot n$ , concluimos que:

$$\begin{aligned} (\varphi, \psi) &\in \text{3-CNF-SAT-UNSAT} \\ &\text{si y sólo si} \\ (G_{(\varphi, \psi)}, k_{(\varphi, \psi)}) &\in \text{exact-CLIQUE} \end{aligned}$$



# La relación entre NP y DP

Es fácil ver que  $\text{NP} \subseteq \text{DP}$  y  $\text{co-NP} \subseteq \text{DP}$ .

▶ ¿Cómo se demuestra esto?

¿Cuál es la relación entre NP y DP?

▶ ¿Es  $\text{NP} \neq \text{DP}$ ?

# La relación entre NP y DP

Es fácil ver que  $\text{NP} \subseteq \text{DP}$  y  $\text{co-NP} \subseteq \text{DP}$ .

- ▶ ¿Cómo se demuestra esto?

¿Cuál es la relación entre NP y DP?

- ▶ ¿Es  $\text{NP} \neq \text{DP}$ ?
- ▶ Este es un problema abierto, y hay una buena explicación de por qué

# La relación entre NP y DP

## Teorema

*$NP = DP$  si y sólo si  $NP = co-NP$ .*

# La relación entre NP y DP

## Teorema

$NP = DP$  si y sólo si  $NP = co-NP$ .

**Demostración:** ( $\Leftarrow$ ) Suponga que  $NP = co-NP$ .

Si  $L \in DP$ :  $L = L_1 \cap L_2$ , con  $L_1 \in NP$  y  $L_2 \in co-NP$ .

► Por hipótesis:  $L_2 \in NP$

Por lo tanto:  $L = L_1 \cap L_2$ , con  $L_1, L_2 \in NP$ .

Pero NP es cerrado bajo intersección.

► Concluimos que  $L \in NP$

# La relación entre NP y DP

( $\Rightarrow$ ) Suponga que  $NP = DP$ .

Como  $co-NP \subseteq DP$ :  $co-NP \subseteq NP$

Por lo que también tenemos que  $NP \subseteq co-NP$ :



# La relación entre NP y DP

( $\Rightarrow$ ) Suponga que  $NP = DP$ .

Como  $co-NP \subseteq DP$ :  $co-NP \subseteq NP$

Por lo que también tenemos que  $NP \subseteq co-NP$ :

$$L \in NP$$

# La relación entre NP y DP

( $\Rightarrow$ ) Suponga que  $NP = DP$ .

Como  $co-NP \subseteq DP$ :  $co-NP \subseteq NP$

Por lo que también tenemos que  $NP \subseteq co-NP$ :

$$L \in NP \Rightarrow \bar{L} \in co-NP$$

# La relación entre NP y DP

( $\Rightarrow$ ) Suponga que  $NP = DP$ .

Como  $co-NP \subseteq DP$ :  $co-NP \subseteq NP$

Por lo que también tenemos que  $NP \subseteq co-NP$ :

$$\begin{aligned} L \in NP &\Rightarrow \overline{L} \in co-NP \\ &\Rightarrow \overline{L} \in NP \end{aligned} \quad (\text{ya que } co-NP \subseteq NP)$$

# La relación entre NP y DP

( $\Rightarrow$ ) Suponga que  $NP = DP$ .

Como  $co-NP \subseteq DP$ :  $co-NP \subseteq NP$

Por lo que también tenemos que  $NP \subseteq co-NP$ :

$$\begin{aligned} L \in NP &\Rightarrow \overline{L} \in co-NP \\ &\Rightarrow \overline{L} \in NP && (\text{ya que } co-NP \subseteq NP) \\ &\Rightarrow L \in co-NP \end{aligned}$$



# Motivación: Problemas de optimización

Un tour  $\pi$  en un grafo  $G$  es una secuencia de arcos  $(a_1, a_2), \dots, (a_{k-1}, a_k), (a_k, a_1)$  en  $G$  tal que:

- ▶  $a_i \neq a_j$  para cada  $i \neq j$ ,
- ▶  $\{a_1, \dots, a_k\}$  es el conjunto de nodos de  $G$ .

## Notación

- ▶ Una función de costo para un grafo  $G = (N, A)$  es una función  $\text{costo} : A \rightarrow \mathbb{N}$ .
- ▶ Costo de un tour  $\pi$  en  $G$ :

$$\text{costo}(\pi) = \left( \sum_{i=1}^{k-1} \text{costo}((a_i, a_{i+1})) \right) + \text{costo}((a_k, a_1))$$

# Motivación: Problemas de optimización

El problema del agente viajero se define como:

$$\text{TSP} = \{(G, \text{costo}, k) \mid \text{existe un tour } \pi \text{ en } G \text{ tal que } \text{costo}(\pi) \leq k\}$$

¿Cuál es la complejidad de TSP?

# Motivación: Problemas de optimización

El problema del agente viajero se define como:

$$\text{TSP} = \{(G, \text{costo}, k) \mid \text{existe un tour } \pi \text{ en } G \text{ tal que } \text{costo}(\pi) \leq k\}$$

¿Cuál es la complejidad de TSP?

Teorema

*TSP es NP-completo.*

# Motivación: Problemas de optimización

TSP es un problema de decisión.



# Motivación: Problemas de optimización

TSP es un problema de decisión.

- ▶ Existe una versión de optimización de TSP: Encuentre el menor  $k$  tal que  $(G, \text{costo}, k) \in \text{TSP}$

# Motivación: Problemas de optimización

TSP es un problema de decisión.

- ▶ Existe una versión de optimización de TSP: Encuentre el menor  $k$  tal que  $(G, \text{costo}, k) \in \text{TSP}$
- ▶ El problema de optimización es al menos tan difícil como el problema de decisión

# Motivación: Problemas de optimización

TSP es un problema de decisión.

- ▶ Existe una versión de optimización de TSP: Encuentre el menor  $k$  tal que  $(G, \text{costo}, k) \in \text{TSP}$
- ▶ El problema de optimización es al menos tan difícil como el problema de decisión
- ▶ Si se puede resolver TSP en tiempo polinomial, ¿se puede resolver la versión de optimización en tiempo polinomial?

# Motivación: Problemas de optimización

TSP es un problema de decisión.

- ▶ Existe una versión de optimización de TSP: Encuentre el menor  $k$  tal que  $(G, \text{costo}, k) \in \text{TSP}$
- ▶ El problema de optimización es al menos tan difícil como el problema de decisión
- ▶ Si se puede resolver TSP en tiempo polinomial, ¿se puede resolver la versión de optimización en tiempo polinomial?
  - ▶ TSP tiene que ser utilizado un número polinomial de veces

# Motivación: Problemas de optimización

Un tour es óptimo si no existe otro tour con costo menor.

- ▶ Puede haber más de un tour óptimo

Otra versión de TSP:

$$\text{unique-TSP} = \{(G, \text{costo}) \mid \text{existe un único tour óptimo para } G\}$$

Este es un problema de decisión: Parece ser más difícil que TSP.

# Motivación: Problemas de optimización

Un tour es óptimo si no existe otro tour con costo menor.

- ▶ Puede haber más de un tour óptimo

Otra versión de TSP:

$$\text{unique-TSP} = \{(G, \text{costo}) \mid \text{existe un único tour óptimo para } G\}$$

Este es un problema de decisión: Parece ser más difícil que TSP.

- ▶ Si se puede resolver TSP en tiempo polinomial, ¿se puede resolver unique-TSP en tiempo polinomial?

# Motivación: Problemas de optimización

Un tour es óptimo si no existe otro tour con costo menor.

- Puede haber más de un tour óptimo

Otra versión de TSP:

$$\text{unique-TSP} = \{(G, \text{costo}) \mid \text{existe un único tour óptimo para } G\}$$

Este es un problema de decisión: Parece ser más difícil que TSP.

- Si se puede resolver TSP en tiempo polinomial, ¿se puede resolver unique-TSP en tiempo polinomial?
  - Nuevamente TSP tiene que ser utilizado un número polinomial de veces

# La noción de oráculo

¿Qué tienen en común los problemas anteriores?



# La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- ▶ Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces

# La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- ▶ Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces
  - ▶ Para el caso de DP: Dos veces

# La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- ▶ Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces
  - ▶ Para el caso de DP: Dos veces
- ▶ Si encontramos un algoritmo polinomial para un problema NP-completo, entonces estos problemas pueden ser resueltos en tiempo polinomial

# La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- ▶ Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces
  - ▶ Para el caso de DP: Dos veces
- ▶ 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

# La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- ▶ Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces
  - ▶ Para el caso de DP: Dos veces
- ▶ 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 definir la noción de MT con oráculo.

# La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- ▶ Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces
  - ▶ Para el caso de DP: Dos veces
- ▶ 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 definir la noción de MT con oráculo.

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

# MT con oráculo

## 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  $q_?, q_{YES}, q_{NO} \in Q$
- ▶  $\Sigma$  es un alfabeto finito tal que  $\vdash, B \notin \Sigma$
- ▶  $\Gamma$  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
- ▶  $\delta$  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.



# 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 esto  $q_?$ :

# 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 esto  $q_?$ :

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

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

# 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 esto  $q_?$ :

- ▶ 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

Para los ejemplos iniciales:

# 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

Para los ejemplos iniciales:

3-CNF-SAT-UNSAT : aceptado por  $M^{\text{SAT}}$  que funciona en tiempo  $O(n)$

# 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

Para los ejemplos iniciales:

3-CNF-SAT-UNSAT : aceptado por  $M^{\text{SAT}}$  que funciona en tiempo  $O(n)$

exact-CLIQUE : aceptado por  $M^{\text{CLIQUE}}$  que funciona en tiempo  $O(n)$

# 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

Para los ejemplos iniciales:

3-CNF-SAT-UNSAT : aceptado por  $M^{\text{SAT}}$  que funciona en tiempo  $O(n)$

exact-CLIQUE : aceptado por  $M^{\text{CLIQUE}}$  que funciona en tiempo  $O(n)$

unique-TSP : aceptado por  $M^{\text{TSP}}$  que funciona en tiempo  $O(n^k)$

# Clases de complejidad y la noción de oráculo

Una primera clase definida en términos de MTs con oráculos:

## Definición

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



# Clases de complejidad y la noción de oráculo

Una primera clase definida en términos de MTs con oráculos:

## Definición

$P^A$ : 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:

# Clases de complejidad y la noción de oráculo

Una primera clase definida en términos de MTs con oráculos:

## Definición

$P^A$ : 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:

- ▶ 3-CNF-SAT-UNSAT, exact-CLIQUE y unique-TSP están en  $P^{\text{SAT}}$

# Clases de complejidad y la noción de oráculo

Una primera clase definida en términos de MTs con oráculos:

## Definición

$P^A$ : 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:

- ▶ 3-CNF-SAT-UNSAT, exact-CLIQUE y unique-TSP están en  $P^{\text{SAT}}$
- ▶ También están contenidos en  $P^A$ , para  $A$  problema NP-completo

# Clases de complejidad y la noción de oráculo

Una primera clase definida en términos de MTs con oráculos:

## Definición

$P^A$ : 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:

- ▶ 3-CNF-SAT-UNSAT, exact-CLIQUE y unique-TSP están en  $P^{\text{SAT}}$ 
  - ▶ También están contenidos en  $P^A$ , para  $A$  problema NP-completo
- ▶ NP, co-NP y DP están contenidas en  $P^{\text{SAT}}$

# Clases de complejidad y la noción de oráculo

Una definición mas general:

Definición

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

# Clases de complejidad y la noción de oráculo

Una definición mas general:

Definición

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

En realidad esta definición no es más general:

Proposición

$$P^{NP} = P^{SAT}$$

# Clases de complejidad y la noción de oráculo

Una definición mas general:

Definición

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

En realidad esta definición no es más general:

Proposición

$$P^{NP} = P^{SAT}$$

Ejercicio

Demuestre la proposición.

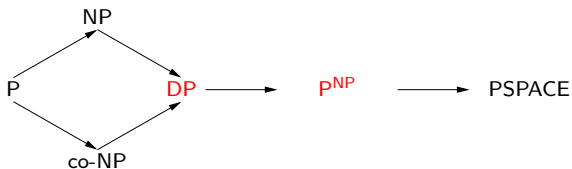
# ¿Dónde estamos?

¿Cuál es la relación de la clase  $P^{NP}$  con las clases que habíamos definido antes?



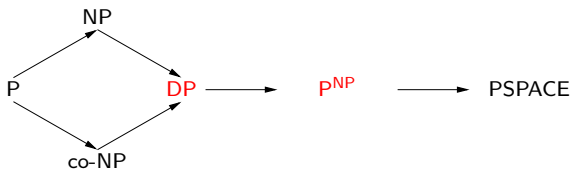
# ¿Dónde estamos?

¿Cuál es la relación de la clase  $P^{NP}$  con las clases que habíamos definido antes?



# ¿Dónde estamos?

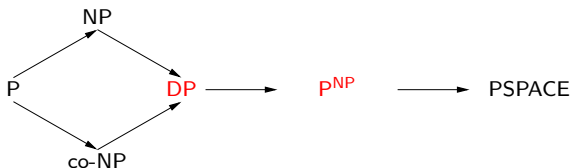
¿Cuál es la relación de la clase  $P^{NP}$  con las clases que habíamos definido antes?



¿Por qué se tiene que  $P^{NP} \subseteq PSPACE$ ?

# ¿Dónde estamos?

¿Cuál es la relación de la clase  $P^{NP}$  con las clases que habíamos definido antes?



¿Por qué se tiene que  $P^{NP} \subseteq PSPACE$ ?

¿Basta con esta clase?

► No, hay otros problemas naturales entre  $P^{NP}$  y PSPACE

# Un problema en inteligencia artificial: Revisión de conocimiento

Un problema fundamental: Actualización de conocimiento.

- ▶ Dada una base de conocimiento  $\Sigma$  y una nueva premisa  $\varphi$ , queremos actualizar el conocimiento en  $\Sigma$  de acuerdo con  $\varphi$

# Un problema en inteligencia artificial: Revisión de conocimiento

Un problema fundamental: Actualización de conocimiento.

- ▶ Dada una base de conocimiento  $\Sigma$  y una nueva premisa  $\varphi$ , queremos actualizar el conocimiento en  $\Sigma$  de acuerdo con  $\varphi$
- ▶ Queremos actualizar lo que sea *necesario*

# Un problema en inteligencia artificial: Revisión de conocimiento

Un problema fundamental: Actualización de conocimiento.

- ▶ Dada una base de conocimiento  $\Sigma$  y una nueva premisa  $\varphi$ , queremos actualizar el conocimiento en  $\Sigma$  de acuerdo con  $\varphi$
- ▶ Queremos actualizar lo que sea *necesario*
- ▶ No queremos perder información que no es *necesario* eliminar

# Un problema en inteligencia artificial: Revisión de conocimiento

Un problema fundamental: Actualización de conocimiento.

- ▶ Dada una base de conocimiento  $\Sigma$  y una nueva premisa  $\varphi$ , queremos actualizar el conocimiento en  $\Sigma$  de acuerdo con  $\varphi$
- ▶ Queremos actualizar lo que sea *necesario*
- ▶ No queremos perder información que no es *necesario* eliminar

Supuestos:

- ▶  $\Sigma$  es un conjunto de fórmulas proposicionales
- ▶  $\varphi$  es una fórmula proposicional

# Revisión de conocimiento

Dado  $\Sigma$  y  $\varphi$ : queremos generar una fórmula que refleje la actualización de  $\Sigma$  dado  $\varphi$ .

Notación

$\Sigma \circ \varphi$



# Revisión de conocimiento

Dado  $\Sigma$  y  $\varphi$ : queremos generar una fórmula que refleje la actualización de  $\Sigma$  dado  $\varphi$ .

## Notación

$$\Sigma \circ \varphi$$

¿Cómo podemos hacer esto?

- ▶ ¿Qué debería ser  $\{p, p \rightarrow q\} \circ \neg q$ ?

# Revisión de conocimiento

Una posible solución: **Belief Revision**

# Revisión de conocimiento

Una posible solución: **Belief Revision**

## Notación

- ▶  $modelos(\Sigma)$ : Conjunto de las valuaciones  $\sigma$  que satisfacen  $\Sigma$
- ▶  $\Delta(\sigma_1, \sigma_2)$ : Conjunto de las variables proposicionales  $p$  tales que  $\sigma_1(p) \neq \sigma_2(p)$ 
  - ▶ Consideramos valuaciones con el mismo dominio

# Revisión de conocimiento

Una posible solución: **Belief Revision**

## Notación

- ▶  $\text{modelos}(\Sigma)$ : Conjunto de las valuaciones  $\sigma$  que satisfacen  $\Sigma$
- ▶  $\Delta(\sigma_1, \sigma_2)$ : Conjunto de las variables proposicionales  $p$  tales que  $\sigma_1(p) \neq \sigma_2(p)$ 
  - ▶ Consideramos valuaciones con el mismo dominio

## Ejemplo

Si  $\sigma_1(p) = 1$ ,  $\sigma_1(q) = 1$ ,  $\sigma_2(p) = 1$  y  $\sigma_2(q) = 0$ , entonces  $\Delta(\sigma_1, \sigma_2) = \{q\}$

- ▶  $\Delta(\sigma_1, \sigma_2)$  mide la *distancia* entre  $\sigma_1$  y  $\sigma_2$

# Revisión de conocimiento

Para actualizar  $\Sigma$  dado  $\varphi$ : Actualizamos los modelos de  $\Sigma$  con respecto a  $\varphi$ .

Dado  $\sigma$  tal que  $\sigma(\Sigma) = 1$ , queremos seleccionar los modelos  $\sigma_1$  de  $\varphi$  que están a distancia mínima de  $\sigma$ .

# Revisión de conocimiento

Para actualizar  $\Sigma$  dado  $\varphi$ : Actualizamos los modelos de  $\Sigma$  con respecto a  $\varphi$ .

Dado  $\sigma$  tal que  $\sigma(\Sigma) = 1$ , queremos seleccionar los modelos  $\sigma_1$  de  $\varphi$  que están a distancia mínima de  $\sigma$ .

## Notación

$$\begin{aligned} \text{mínimo}(\sigma, \varphi) = \{ \sigma_1 \mid \sigma_1(\varphi) = 1 \text{ y no existe } \sigma_2 \text{ tal que} \\ \sigma_2(\varphi) = 1 \text{ y } \Delta(\sigma, \sigma_2) \subsetneq \Delta(\sigma, \sigma_1) \} \end{aligned}$$

# Revisión de conocimiento

Definimos los modelos de  $\Sigma \circ \varphi$  como los modelos de  $\varphi$  que están *más cerca* de los modelos de  $\Sigma$ :

$$\text{modelos}(\Sigma \circ \varphi) = \bigcup_{\sigma : \sigma(\Sigma)=1} \text{mínimo}(\sigma, \varphi)$$

y definimos  $\Sigma \circ \varphi$  como una fórmula  $\psi$  arbitraria tal que  $\text{modelos}(\psi) = \text{modelos}(\Sigma \circ \varphi)$ .

# Revisión de conocimiento

Definimos los modelos de  $\Sigma \circ \varphi$  como los modelos de  $\varphi$  que están *más cerca* de los modelos de  $\Sigma$ :

$$\text{modelos}(\Sigma \circ \varphi) = \bigcup_{\sigma : \sigma(\Sigma)=1} \text{mínimo}(\sigma, \varphi)$$

y definimos  $\Sigma \circ \varphi$  como una fórmula  $\psi$  arbitraria tal que  $\text{modelos}(\psi) = \text{modelos}(\Sigma \circ \varphi)$ .

- ¿Siempre existe esta fórmula? ¿Es única?



# Revisión de conocimiento

## Ejemplo

$$\Sigma = \{p, p \rightarrow q\} \text{ y } \varphi = \neg q$$

$$\begin{aligned} \text{modelos}(\Sigma) &= \{\sigma\}, & \sigma(p) = \sigma(q) &= 1 \\ \text{modelos}(\varphi) &= \{\sigma_1, \sigma_2\}, & \sigma_1(p) &= 1, \sigma_1(q) = 0 \\ & & \sigma_2(p) &= 0, \sigma_2(q) = 0 \end{aligned}$$

Modelos mínimos:

$$\begin{aligned} \Delta(\sigma, \sigma_1) &= \{q\} \\ \Delta(\sigma, \sigma_2) &= \{p, q\} \\ \text{mínimo}(\sigma, \varphi) &= \{\sigma_1\} \\ \text{modelos}(\Sigma \circ \varphi) &= \{\sigma_1\} \end{aligned}$$

$$\text{Resultado: } \{p, p \rightarrow q\} \circ \neg q = p \wedge \neg q$$

# Revisión de conocimiento: Complejidad

Un problema fundamental en el área: Calcular respuestas *certeras*.

- ▶  $\psi$  es una **respuesta certera** en  $\Sigma \circ \varphi$  si para cada  $\sigma \in \text{modelos}(\Sigma \circ \varphi)$ , se tiene que  $\sigma$  satisface  $\psi$

Ejemplo

$p$  es una respuesta certera en  $\{p, p \rightarrow q\} \circ \neg q$ .

# Revisión de conocimiento: Complejidad

Un problema fundamental en el área: Calcular respuestas *certeras*.

- $\psi$  es una **respuesta certera** en  $\Sigma \circ \varphi$  si para cada  $\sigma \in \text{modelos}(\Sigma \circ \varphi)$ , se tiene que  $\sigma$  satisface  $\psi$

Ejemplo

$p$  es una respuesta certera en  $\{p, p \rightarrow q\} \circ \neg q$ .

Problema que queremos estudiar:

**CERTAIN-ANSWERS** =  $\{(\Sigma, \varphi, \psi) \mid \psi \text{ es una respuesta certera en } \Sigma \circ \varphi\}$

# Revisión de conocimiento: Complejidad

¿Cuál es la complejidad de CERTAIN-ANSWERS?

# Revisión de conocimiento: Complejidad

¿Cuál es la complejidad de CERTAIN-ANSWERS?

- ▶ ¿Está en NP? ¿En co-NP?

# Revisión de conocimiento: Complejidad

¿Cuál es la complejidad de CERTAIN-ANSWERS?

- ▶ ¿Está en NP? ¿En co-NP?
- ▶ ¿Está en  $P^{NP}$ ?

# Revisión de conocimiento: Complejidad

¿Cuál es la complejidad de CERTAIN-ANSWERS?

- ▶ ¿Está en NP? ¿En co-NP?
- ▶ ¿Está en  $P^{NP}$ ?
- ▶ ¿Al menos está en PSPACE?

# Revisión de conocimiento: Complejidad

¿Cuál es la complejidad de CERTAIN-ANSWERS?

- ▶ ¿Está en NP? ¿En co-NP?
- ▶ ¿Está en  $P^{NP}$ ?
- ▶ ¿Al menos está en PSPACE?

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  $O(n^k)$
- ▶  $NP^{NP}$ :  $\bigcup_{A \in NP} NP^A$

# 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 CERTAIN-ANSWERS.

# 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 CERTAIN-ANSWERS.

## Ejercicio

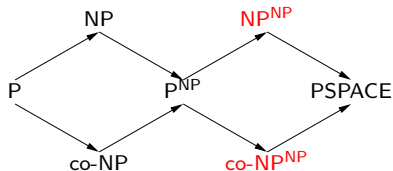
Demuestre que CERTAIN-ANSWERS  $\in$  co- $NP^{NP}$ .

¿Y ahora dónde estamos?

¿Cuál es la relación de la clase  $\text{NP}^{\text{NP}}$  con las clases que habíamos definido antes?

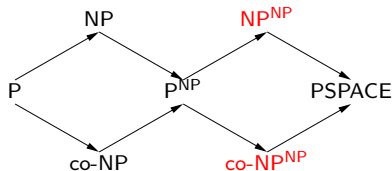
# ¿Y ahora dónde estamos?

¿Cuál es la relación de la clase  $NP^{NP}$  con las clases que habíamos definido antes?



¿Y ahora dónde estamos?

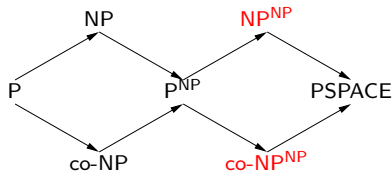
¿Cuál es la relación de la clase  $\text{NP}^{\text{NP}}$  con las clases que habíamos definido antes?



¿Por qué se tiene que  $\text{NP}^{\text{NP}} \subseteq \text{PSPACE}$ ?

# ¿Y ahora dónde estamos?

¿Cuál es la relación de la clase  $\text{NP}^{\text{NP}}$  con las clases que habíamos definido antes?



¿Por qué se tiene que  $\text{NP}^{\text{NP}} \subseteq \text{PSPACE}$ ?

Esta construcción se puede generalizar.

► Vamos a definir la jerarquía polinomial

# La jerarquía polinomial

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



# La jerarquía polinomial

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

## Definición

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

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

# La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

# La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

## Definición

# La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

Definición

$$\Sigma_0^P = P$$

# La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

## Definición

$$\begin{aligned}\Sigma_0^P &= P \\ \Sigma_{n+1}^P &= NP^{\Sigma_n^P} \quad n \geq 0\end{aligned}$$

# La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

## Definición

$$\Sigma_0^P = P$$

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

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

# La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

## Definición

$$\begin{aligned}\Sigma_0^P &= P \\ \Sigma_{n+1}^P &= NP^{\Sigma_n^P} & n \geq 0 \\ \Delta_{n+1}^P &= P^{\Sigma_n^P} & n \geq 0 \\ \Pi_{n+1}^P &= co-\Sigma_{n+1}^P & n \geq 0\end{aligned}$$

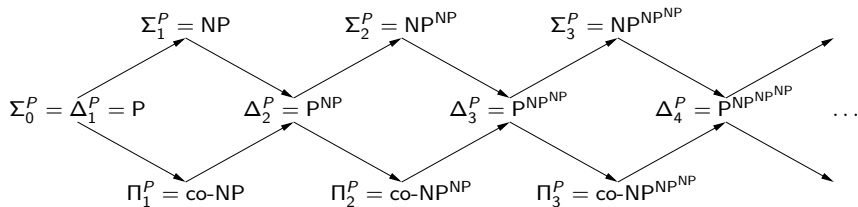
# La jerarquía polinomial

¿Cómo se ve la jerarquía polinomial en una figura?



# La jerarquía polinomial

¿Cómo se ve la jerarquía polinomial en una figura?



# La jerarquía polinomial y PSPACE

## Definición

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

# La jerarquía polinomial y PSPACE

## Definición

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

¿Cuál es la relación entre PH y PSPACE?

# La jerarquía polinomial y PSPACE

## Definición

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

¿Cuál es la relación entre PH y PSPACE?

- ▶  $PH \subseteq PSPACE$

# La jerarquía polinomial y PSPACE

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

# La jerarquía polinomial y PSPACE

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

- ▶ Esto implicaría que PH tiene problemas completos

# La jerarquía polinomial y PSPACE

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

- ▶ Esto implicaría que PH tiene problemas completos

¿Puede tener problemas completos PH?

# La jerarquía polinomial y PSPACE

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

- ▶ Esto implicaría que PH tiene problemas completos

¿Puede tener problemas completos PH?

- ▶ Cada clase  $\Sigma_k^P$  es cerrada bajo  $\leq_m^P$  (reducción many-to-one de tiempo polinomial)



# La jerarquía polinomial y PSPACE

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

- ▶ Esto implicaría que PH tiene problemas completos

¿Puede tener problemas completos PH?

- ▶ Cada clase  $\Sigma_k^P$  es cerrada bajo  $\leq_m^P$  (reducción many-to-one de tiempo polinomial)
- ▶ Si PH tiene un problema completo, entonces la jerarquía polinomial colapsa a algún nivel finito

# La jerarquía polinomial y PSPACE

¿Puede ser cierto que  $PSPACE \subseteq PH$ ?

- ▶ Esto implicaría que PH tiene problemas completos

¿Puede tener problemas completos PH?

- ▶ Cada clase  $\Sigma_k^P$  es cerrada bajo  $\leq_m^P$  (reducción many-to-one de tiempo polinomial)
- ▶ 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  $PH \subsetneq PSPACE$ .*

# La jerarquía polinomial: Una primera propiedad

## Notación

$$\Pi_0^P = P$$

## Proposición

*Para  $k \geq 1$ : Un lenguaje  $L$  sobre un alfabeto  $\Sigma$  está en  $\Sigma_k^P$  si y sólo si existe  $A \in \Pi_{k-1}^P$  y un polinomio  $p(n)$  tal que para todo  $w \in \Sigma^*$  :*

*$w \in L$  si y sólo si existe  $z \in \Sigma^*$  tal que  $|z| \leq p(|w|)$  y  $(w, z) \in A$ .*

# La jerarquía polinomial: Una primera propiedad

## Notación

$$\Pi_0^P = P$$

## Proposición

*Para  $k \geq 1$ : Un lenguaje  $L$  sobre un alfabeto  $\Sigma$  está en  $\Sigma_k^P$  si y sólo si existe  $A \in \Pi_{k-1}^P$  y un polinomio  $p(n)$  tal que para todo  $w \in \Sigma^*$ :*

*$w \in L$  si y sólo si existe  $z \in \Sigma^*$  tal que  $|z| \leq p(|w|)$  y  $(w, z) \in A$ .*

## Ejercicio

Demuestre la proposición para  $k = 1$ .

# La jerarquía polinomial: Caracterizando $\Sigma_k^P$

## Teorema

Para  $k \geq 1$ : Un lenguaje  $L$  sobre un alfabeto  $\Sigma$  está en  $\Sigma_k^P$  si y sólo si existe  $A \in P$  y un polinomio  $p(n)$  tal que para todo  $w \in \Sigma^*$ :

$w \in L$  si y sólo si

$(\exists z_1 \in \Sigma^*, |z_1| \leq p(|w|)) (\forall z_2 \in \Sigma^*, |z_2| \leq p(|w|)) \dots$

$(Q_k z_k \in \Sigma^*, |z_k| \leq p(|w|)) (w, z_1, z_2, \dots, z_k) \in A,$

donde  $Q_k = \exists$  si  $k$  es impar y  $Q_k = \forall$  si  $k$  es par.

# La jerarquía polinomial: Caracterizando $\Sigma_k^P$

## Teorema

Para  $k \geq 1$ : Un lenguaje  $L$  sobre un alfabeto  $\Sigma$  está en  $\Sigma_k^P$  si y sólo si existe  $A \in P$  y un polinomio  $p(n)$  tal que para todo  $w \in \Sigma^*$ :

$w \in L$  si y sólo si

$(\exists z_1 \in \Sigma^*, |z_1| \leq p(|w|)) (\forall z_2 \in \Sigma^*, |z_2| \leq p(|w|)) \dots$

$(Q_k z_k \in \Sigma^*, |z_k| \leq p(|w|)) (w, z_1, z_2, \dots, z_k) \in A,$

donde  $Q_k = \exists$  si  $k$  es impar y  $Q_k = \forall$  si  $k$  es par.

## Ejercicio

Demuestre el teorema usando la propiedad anterior.

# La jerarquía polinomial: Caracterizando $\Sigma_k^P$

La caracterización anterior nos va a servir para encontrar problemas completos para cada clase  $\Sigma_k^P$ .

- ▶ Cada uno de estos problemas es una extensión de SAT

# La jerarquía polinomial: Caracterizando $\Sigma_k^P$

La caracterización anterior nos va a servir para encontrar problemas completos para cada clase  $\Sigma_k^P$ .

- ▶ Cada uno de estos problemas es una extensión de SAT

Pero antes vamos a demostrar un resultado fundamental sobre la jerarquía polinomial.

- ▶ Vamos a razonar sobre un posible colapso de la jerarquía polinomial
- ▶ En la demostración, suponemos la existencia de problemas completos para cada clase  $\Sigma_k^P$



# El colapso de la jerarquía polinomial

## 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  $PH = \Delta_k^P$

# El colapso de la jerarquía polinomial

## 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  $PH = \Delta_k^P$

**Demostración:** Usamos el siguiente lema:

## Lema

Para  $k \geq 1$ :  $NP^{\Sigma_k^P \cap \Pi_k^P} = \Sigma_k^P$

# El colapso de la jerarquía polinomial

## Demostración del lema:

( $\supseteq$ ) Si  $L \in \Sigma_k^P$ , entonces  $L \in \text{NP}^A$ , para  $A \in \Sigma_{k-1}^P$ .

Pero  $\Sigma_{k-1}^P \subseteq \Delta_k^P \subseteq \Sigma_k^P \cap \Pi_k^P$ .

► Concluimos que  $L \in \text{NP}^{\Sigma_k^P \cap \Pi_k^P}$

( $\subseteq$ ) Suponga que  $L \in \text{NP}^{\Sigma_k^P \cap \Pi_k^P}$ .

Se tiene que  $L = \text{NP}^A$ , para  $A \in \Sigma_k^P \cap \Pi_k^P$ .

# El colapso de la jerarquía polinomial

Como  $A \in \Sigma_k^P \cap \Pi_k^P$ :  $A = L(M_1^B)$  y  $\overline{A} = L(M_2^B)$ , donde:

- ▶  $B$  es un problema completo para  $\Sigma_{k-1}^P$
- ▶  $M_1^B$  y  $M_2^B$  son MTs no deterministas que funcionan en tiempo polinomial y tienen oráculo para  $B$

# El colapso de la jerarquía polinomial

Como  $A \in \Sigma_k^P \cap \Pi_k^P$ :  $A = L(M_1^B)$  y  $\overline{A} = L(M_2^B)$ , donde:

- ▶  $B$  es un problema completo para  $\Sigma_{k-1}^P$
- ▶  $M_1^B$  y  $M_2^B$  son MTs no deterministas que funcionan en tiempo polinomial y tienen oráculo para  $B$

Por lo tanto:  $L \in \text{NP}^B$

# El colapso de la jerarquía polinomial

Como  $A \in \Sigma_k^P \cap \Pi_k^P$ :  $A = L(M_1^B)$  y  $\overline{A} = L(M_2^B)$ , donde:

- ▶  $B$  es un problema completo para  $\Sigma_{k-1}^P$
- ▶  $M_1^B$  y  $M_2^B$  son MTs no deterministas que funcionan en tiempo polinomial y tienen oráculo para  $B$

Por lo tanto:  $L \in \text{NP}^B$

- ▶ Idea: Reemplazamos cada llamada en  $M^A$  al oráculo  $A$  por una llamada simultanea a las rutinas  $M_1^B$  y  $M_2^B$

# El colapso de la jerarquía polinomial

Como  $A \in \Sigma_k^P \cap \Pi_k^P$ :  $A = L(M_1^B)$  y  $\overline{A} = L(M_2^B)$ , donde:

- ▶  $B$  es un problema completo para  $\Sigma_{k-1}^P$
- ▶  $M_1^B$  y  $M_2^B$  son MTs no deterministas que funcionan en tiempo polinomial y tienen oráculo para  $B$

Por lo tanto:  $L \in \text{NP}^B$

- ▶ Idea: Reemplazamos cada llamada en  $M^A$  al oráculo  $A$  por una llamada simultanea a las rutinas  $M_1^B$  y  $M_2^B$

Concluimos que  $L \in \Sigma_k^P$ .



# El colapso de la jerarquía polinomial

Ahora volvemos a la demostración inicial.

(a) Suponemos que  $\Sigma_k^P = \Pi_k^P$ .

Vamos a demostrar por inducción en  $j \geq k$  que  $\Sigma_j^P = \Pi_j^P = \Sigma_k^P$ .

Para  $j = k$  la propiedad se tiene por hipótesis.

Suponemos que la propiedad se cumple para  $j \geq k$ , y vamos a demostrar que también es cierta para  $j + 1$ .



# El colapso de la jerarquía polinomial

Tenemos que:

$$\begin{aligned}\Sigma_{j+1}^P &= \text{NP}^{\Sigma_j^P} \\ &= \text{NP}^{\Sigma_j^P \cap \Pi_j^P} && \text{ya que } \Sigma_j^P = \Pi_j^P \\ &= \Sigma_j^P && \text{por lema } (j \geq k \geq 1) \\ &= \Sigma_k^P && \text{por hipótesis de inducción}\end{aligned}$$

Además tenemos que:

$$\begin{aligned}\Pi_{j+1}^P &= \text{co-}\Sigma_{j+1}^P \\ &= \text{co-}\Sigma_k^P && \text{por demostración de arriba} \\ &= \Sigma_k^P && \text{por hipótesis}\end{aligned}$$

# El colapso de la jerarquía polinomial

(b) Suponemos que  $\Delta_k^P = \Sigma_k^P$ .

Como  $\Delta_k^P$  es cerrada bajo complemento, tenemos que  $\Sigma_k^P = \Pi_k^P$ .

► Por (a) tenemos que  $PH = \Sigma_k^P$

Concluimos que  $PH = \Delta_k^P$ .



# Problemas completos en la jerarquía polinomial

El lenguaje  $\text{QBF}_i$  ( $i \geq 1$ ) 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
- ▶  $\varphi$  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 $\Sigma_k^P$

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  $\Sigma_k^P$ -completo.*

# Un problema completo para $\Sigma_k^P$

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  $\Sigma_k^P$ -completo.*

## Ejercicio

Demuestre que  $QBF_k$  está en  $\Sigma_k^P$ .

## Otro problema completo para $\Sigma_2^P$

### Teorema

$\overline{CERTAIN-ANSWERS}$  es  $\Sigma_2^P$ -completo.

- ▶ Se deduce que  $CERTAIN-ANSWERS$  es  $\Pi_2^P$ -completo

## Otro problema completo para $\Sigma_2^P$

### Teorema

$\overline{CERTAIN-ANSWERS}$  es  $\Sigma_2^P$ -completo.

► Se deduce que  $CERTAIN-ANSWERS$  es  $\Pi_2^P$ -completo

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