# Complejidad Computacional

Santiago Figueira

Departamento de Computación - FCEN - UBA

clase 12

Clase 12

Las clases $\mathbf{NC}_{\text{nu}}$ y $\mathbf{AC}_{\text{nu}}$

Uniformidad: las clases $\mathbf{NC}$ y $\mathbf{AC}$

$\mathbf{P}$-completitud

# Las clases $\mathbf{NC}_{\text{nu}}$ y $\mathbf{AC}_{\text{nu}}$

# Las clases $\mathbf{NC}_{\text{nu}}$ y $\mathbf{AC}_{\text{nu}}$

## Clase de complejidad: $\mathbf{NC}^d_{\text{nu}}$ y $\mathbf{NC}_{\text{nu}}$

- $\mathbf{NC}^d_{\text{nu}}$ es la clase de lenguajes decidibles por una familia de circuitos $(C_n)_{n\in\mathbb{N}}$ tal que
  - $|C_n|$ es polinomial en $n$ y
  - $\text{prof}(C_n) = O(\log^d n)$
- $\mathbf{NC}_{\text{nu}} = \bigcup_{i\geq 0} \mathbf{NC}^i_{\text{nu}}$

# Las clases $\mathbf{NC}_{\text{nu}}$ y $\mathbf{AC}_{\text{nu}}$

## Clase de complejidad: $\mathbf{NC}^d_{\text{nu}}$ y $\mathbf{NC}_{\text{nu}}$

- $\mathbf{NC}^d_{\text{nu}}$ es la clase de lenguajes decidibles por una familia de circuitos $(C_n)_{n\in\mathbb{N}}$ tal que
  - $|C_n|$ es polinomial en $n$ y
  - $\text{prof}(C_n) = O(\log^d n)$
- $\mathbf{NC}_{\text{nu}} = \bigcup_{i\geq 0} \mathbf{NC}^i_{\text{nu}}$

## Clase de complejidad: $\mathbf{AC}^d_{\text{nu}}$ y $\mathbf{AC}_{\text{nu}}$

- $\mathbf{AC}^d_{\text{nu}}$ se define igual que $\mathbf{NC}^d_{\text{nu}}$ pero permitimos nodos $\wedge$ y $\vee$ con *fan-in* arbitrario.
- $\mathbf{AC}_{\text{nu}} = \bigcup_{i\geq 0} \mathbf{AC}^i_{\text{nu}}$

(en el nombre, 'nu' es por 'no uniforme')

# $\mathbf{NC}^0_{\text{nu}}, \mathbf{AC}^0_{\text{nu}}, \mathbf{NC}^1_{\text{nu}}$

- $\mathbf{NC}^0_{\text{nu}}$ es muy limitado porque solo depende de una cantidad finita de bits de entrada
- $\mathbf{AC}^0_{\text{nu}}$ no tiene esa misma limitación, por ejemplo

$$\{1^n : n \geq 1\} \in \mathbf{AC}^0_{\text{nu}}.$$

# $\mathbf{NC}^0_{\text{nu}}, \mathbf{AC}^0_{\text{nu}}, \mathbf{NC}^1_{\text{nu}}$

- $\mathbf{NC}^0_{\text{nu}}$ es muy limitado porque solo depende de una cantidad finita de bits de entrada
- $\mathbf{AC}^0_{\text{nu}}$ no tiene esa misma limitación, por ejemplo

$$\{1^n : n \geq 1\} \in \mathbf{AC}^0_{\text{nu}}.$$

**Problema: Cantidad impar de 1s**

$$\mathsf{PARITY} = \{x : x \text{ tiene una cantidad impar de 1s}\}$$

# $\mathbf{NC}^0_{\text{nu}}, \mathbf{AC}^0_{\text{nu}}, \mathbf{NC}^1_{\text{nu}}$

- $\mathbf{NC}^0_{\text{nu}}$ es muy limitado porque solo depende de una cantidad finita de bits de entrada
- $\mathbf{AC}^0_{\text{nu}}$ no tiene esa misma limitación, por ejemplo

$$\{1^n : n \geq 1\} \in \mathbf{AC}^0_{\text{nu}}.$$

**Problema: Cantidad impar de 1s**

$$\mathsf{PARITY} = \{x : x \text{ tiene una cantidad impar de 1s}\}$$

**Proposición**

$\mathsf{PARITY} \in \mathbf{NC}^1_{\text{nu}}$.

# $\mathbf{NC}^0_{\text{nu}}, \mathbf{AC}^0_{\text{nu}}, \mathbf{NC}^1_{\text{nu}}$

- $\mathbf{NC}^0_{\text{nu}}$ es muy limitado porque solo depende de una cantidad finita de bits de entrada
- $\mathbf{AC}^0_{\text{nu}}$ no tiene esa misma limitación, por ejemplo

$$\{1^n : n \geq 1\} \in \mathbf{AC}^0_{\text{nu}}.$$

**Problema: Cantidad impar de 1s**

$$\mathsf{PARITY} = \{x : x \text{ tiene una cantidad impar de 1s}\}$$

**Proposición**

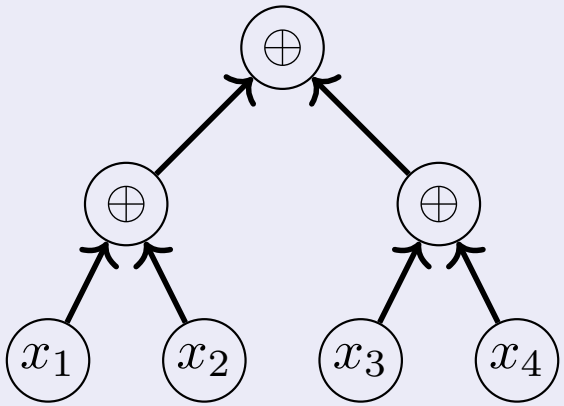
$\mathsf{PARITY} \in \mathbf{NC}^1_{\text{nu}}$.

**Proposición (no lo vamos a probar)**

$\mathsf{PARITY} \notin \mathbf{AC}^0_{\text{nu}}$.

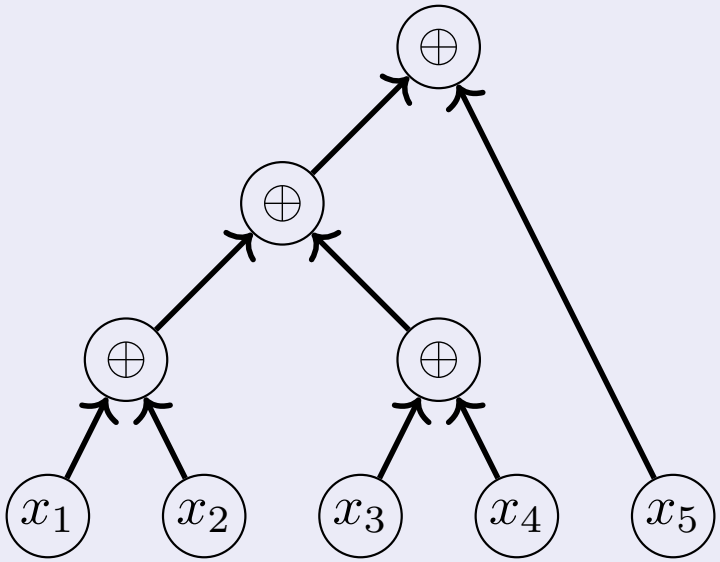
# Demostración de PARITY $\in \mathbf{NC}^1_{\text{nu}}$.

Sumamos (módulo 2) todos los bits de $x$.




$\text{prof}(C_4) = 2$

$C_4$




$\text{prof}(C_5) = 3$

$C_5$

En general,
$\text{prof}(C_n) = O(\log n)$

# La suma en binario

**Problema: Suma en binario**

$$\mathsf{SUMA} = \{xyz : |x| = |y| = i, |z| = i + 1, x + y = z, i \geq 1\}$$

**Proposición**

$\mathsf{SUMA} \in \mathbf{AC}^0_{\mathrm{nu}}$.

## Demostración.

Entrada: $x_n, \ldots, x_1, y_n, \ldots, y_1, z_{n+1}, z_n, \ldots, z_1$.

$$
\begin{array}{rccccccccc}
carry \rightarrow & \ldots & 1 & 1 & 1 & & & 1 & & \\
x = & \ldots & & 1 & 0 & 1 & 1 & 0 & 1 \\
y = & \ldots & & 0 & 1 & 1 & 0 & 0 & 1 \\
\hline
z = & \ldots & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
 & & \downarrow & & & \downarrow & & & \\
 & & i+1 & & & j & & &
\end{array}
$$

## Demostración.

Entrada: $x_n, \ldots, x_1, y_n, \ldots, y_1, z_{n+1}, z_n, \ldots, z_1$.

$$
\begin{array}{rccccccccc}
carry \rightarrow & \ldots & 1 & 1 & 1 & & & 1 & & \\
x = & \ldots & & 1 & 0 & 1 & 1 & 0 & 1 \\
y = & \ldots & & 0 & 1 & 1 & 0 & 0 & 1 \\
\hline
z = & \ldots & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
 & & \downarrow & & & \downarrow & & & \\
 & & i+1 & & & j & & &
\end{array}
$$

Calculamos el *carry* $c_i$ de esta forma:

$$
c_1 = 0 \quad \text{y} \quad c_{i+1} = \bigvee_{i \geq j \geq 1} \left( (x_j \wedge y_j) \wedge \bigwedge_{i \geq k > j} (x_k \vee y_k) \right)
$$

## Demostración.

Entrada: $x_n, \ldots, x_1, y_n, \ldots, y_1, z_{n+1}, z_n, \ldots, z_1$.

$$
\begin{array}{rccccccccc}
\textit{carry} \rightarrow & \ldots & 1 & 1 & 1 & & & 1 & & \\
x = & \ldots & & 1 & 0 & 1 & 1 & 0 & 1 \\
y = & \ldots & & 0 & 1 & 1 & 0 & 0 & 1 \\
\hline
z = & \ldots & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
 & & \downarrow & & & \downarrow & & & \\
 & & i+1 & & & j & & &
\end{array}
$$

Calculamos el *carry* $c_i$ de esta forma:

$$c_1 = 0 \quad \text{y} \quad c_{i+1} = \bigvee_{i \geq j \geq 1} \left( (x_j \wedge y_j) \wedge \bigwedge_{i \geq k > j} (x_k \vee y_k) \right)$$

Cada circuito para $c_i$ tiene profundidad 3.

## Demostración.

Entrada: $x_n, \ldots, x_1, y_n, \ldots, y_1, z_{n+1}, z_n, \ldots, z_1$.

$$
\begin{array}{rccccccccc}
\mathit{carry} \rightarrow & \ldots & 1 & 1 & 1 & & & 1 & & \\
x = & \ldots & & 1 & 0 & 1 & 1 & 0 & 1 \\
y = & \ldots & & 0 & 1 & 1 & 0 & 0 & 1 \\
\hline
z = & \ldots & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
 & & \downarrow & & & \downarrow & & & \\
 & & i+1 & & & j & & &
\end{array}
$$

Calculamos el *carry* $c_i$ de esta forma:

$$c_1 = 0 \quad \text{y} \quad c_{i+1} = \bigvee_{i \geq j \geq 1} \left( (x_j \wedge y_j) \wedge \bigwedge_{i \geq k > j} (x_k \vee y_k) \right)$$

Cada circuito para $c_i$ tiene profundidad 3. Calculamos la suma de la forma usual: $s_i = x_i \oplus y_i \oplus c_i$ y $s_{n+1} = c_{n+1}$. Finalmente, la salida es

$$\bigwedge_{n+1 \geq j \geq 1} z_i = s_i$$

y lo calculamos con un circuito de profundidad constante. □

## Proposición

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$. Por lo tanto, $\mathbf{AC}_{\text{nu}} = \mathbf{NC}_{\text{nu}}$.

## Proposición

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$. Por lo tanto, $\mathbf{AC}_{\text{nu}} = \mathbf{NC}_{\text{nu}}$.

## Demostración.

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}}$ es trivial.

## Proposición

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$. Por lo tanto, $\mathbf{AC}_{\text{nu}} = \mathbf{NC}_{\text{nu}}$.

## Demostración.

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}}$ es trivial. Veamos $\mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$.
Supongamos que $\mathcal{L} \in \mathbf{AC}^d_{\text{nu}}$. Existe una familia de circuitos con $\wedge, \vee$ de *fan-in* arbitrario $(C_i)_{i\in\mathbb{N}}$ que decide $\mathcal{L}$,

$$|C_n| = O(n^c) \quad \text{y} \quad \text{prof}(C_n) = O(\log^d n).$$

## Proposición

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$. Por lo tanto, $\mathbf{AC}_{\text{nu}} = \mathbf{NC}_{\text{nu}}$.

## Demostración.

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}}$ es trivial. Veamos $\mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$.
Supongamos que $\mathcal{L} \in \mathbf{AC}^d_{\text{nu}}$. Existe una familia de circuitos con $\wedge, \vee$ de *fan-in* arbitrario $(C_i)_{i\in\mathbb{N}}$ que decide $\mathcal{L}$,

$$|C_n| = O(n^c) \quad \text{y} \quad \text{prof}(C_n) = O(\log^d n).$$

Simulamos el *fan-in* arbitrario con un circuito con *fan-in* binario de profundidad $O(\log n)$.

## Proposición

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$. Por lo tanto, $\mathbf{AC}_{\text{nu}} = \mathbf{NC}_{\text{nu}}$.

## Demostración.

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}}$ es trivial. Veamos $\mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$.
Supongamos que $\mathcal{L} \in \mathbf{AC}^d_{\text{nu}}$. Existe una familia de circuitos con $\wedge, \vee$ de *fan-in* arbitrario $(C_i)_{i\in\mathbb{N}}$ que decide $\mathcal{L}$,

$$|C_n| = O(n^c) \quad \text{y} \quad \text{prof}(C_n) = O(\log^d n).$$

Simulamos el *fan-in* arbitrario con un circuito con *fan-in* binario de profundidad $O(\log n)$.
Definimos $C'_n$ como el reemplazo en $C_n$ de todo nodo de *fan-in* arbitrario con el subcircuito de profundidad $O(\log n^c) = O(\log n)$.

## Proposición

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$. Por lo tanto, $\mathbf{AC}_{\text{nu}} = \mathbf{NC}_{\text{nu}}$.

## Demostración.

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}}$ es trivial. Veamos $\mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$.
Supongamos que $\mathcal{L} \in \mathbf{AC}^d_{\text{nu}}$. Existe una familia de circuitos con $\wedge, \vee$ de *fan-in* arbitrario $(C_i)_{i\in\mathbb{N}}$ que decide $\mathcal{L}$,

$$|C_n| = O(n^c) \quad \text{y} \quad \text{prof}(C_n) = O(\log^d n).$$

Simulamos el *fan-in* arbitrario con un circuito con *fan-in* binario de profundidad $O(\log n)$.
Definimos $C'_n$ como el reemplazo en $C_n$ de todo nodo de *fan-in* arbitrario con el subcircuito de profundidad $O(\log n^c) = O(\log n)$.
Es claro que $(C'_i)_{i\in\mathbb{N}}$ decide $\mathcal{L}$.

## Proposición

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$. Por lo tanto, $\mathbf{AC}_{\text{nu}} = \mathbf{NC}_{\text{nu}}$.

## Demostración.

$\mathbf{NC}^d_{\text{nu}} \subseteq \mathbf{AC}^d_{\text{nu}}$ es trivial. Veamos $\mathbf{AC}^d_{\text{nu}} \subseteq \mathbf{NC}^{d+1}_{\text{nu}}$.
Supongamos que $\mathcal{L} \in \mathbf{AC}^d_{\text{nu}}$. Existe una familia de circuitos con $\wedge, \vee$ de *fan-in* arbitrario $(C_i)_{i\in\mathbb{N}}$ que decide $\mathcal{L}$,

$$|C_n| = O(n^c) \quad \text{y} \quad \text{prof}(C_n) = O(\log^d n).$$

Simulamos el *fan-in* arbitrario con un circuito con *fan-in* binario de profundidad $O(\log n)$.
Definimos $C'_n$ como el reemplazo en $C_n$ de todo nodo de *fan-in* arbitrario con el subcircuito de profundidad $O(\log n^c) = O(\log n)$.
Es claro que $(C'_i)_{i\in\mathbb{N}}$ decide $\mathcal{L}$.

- $C_n$ tiene profundidad $O(\log^d)$
- $C'_n$ tiene profundidad $O(\log n \cdot \log^d n) = O(\log^{d+1} n)$
- $|C'_n|$ es polinomial

□

# Uniformidad: las clases **NC** y **AC**

# Circuitos **P**-uniformes

Puede parecer extraño que lenguajes indecidibles estén en $\mathbf{P}_{/\mathbf{poly}}$. El origen de este fenómeno es la no uniformidad del modelo de circuitos.

## Definición

Una familia de circuitos $(C_n)_{n\in\mathbb{N}}$ es **P-uniforme** si existe una máquina determinística tal que

- $M$ corre en tiempo polinomial
- $M(1^n) = \langle C_n \rangle$

# Circuitos **P**-uniformes y **P**

**Teorema**

$\mathcal{L}$ es decidible por una familia **P**-uniforme de circuitos sii $\mathcal{L} \in \mathbf{P}$.

# Circuitos **P**-uniformes y **P**

**Teorema**

$\mathcal{L}$ es decidible por una familia **P**-uniforme de circuitos sii $\mathcal{L} \in \mathbf{P}$.

**Demostración.**

$\subseteq$ Sea $M$ una máquina determinística que corre en tiempo polinomial tal que para todo $n$, $M(1^n) = \langle C_n \rangle$ y para todo $x \in \{0,1\}^n$:

$$x \in \mathcal{L} \qquad \text{sii} \qquad C_n(x) = 1.$$

# Circuitos **P**-uniformes y **P**

**Teorema**

$\mathcal{L}$ es decidible por una familia **P**-uniforme de circuitos sii $\mathcal{L} \in \mathbf{P}$.

**Demostración.**

$\subseteq$ Sea $M$ una máquina determinística que corre en tiempo polinomial tal que para todo $n$, $M(1^n) = \langle C_n \rangle$ y para todo $x \in \{0,1\}^n$:

$$x \in \mathcal{L} \qquad \text{sii} \qquad C_n(x) = 1.$$

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

*Simular* $M(1^{|x|}) = \langle C \rangle$
*Evaluar* $C(x)$ *y devolver su salida*

# Circuitos **P**-uniformes y **P**

## Teorema

$\mathcal{L}$ es decidible por una familia **P**-uniforme de circuitos sii $\mathcal{L} \in \mathbf{P}$.

## Demostración.

$\subseteq$ Sea $M$ una máquina determinística que corre en tiempo polinomial tal que para todo $n$, $M(1^n) = \langle C_n \rangle$ y para todo $x \in \{0,1\}^n$:

$$x \in \mathcal{L} \qquad \text{sii} \qquad C_n(x) = 1.$$

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

*Simular* $M(1^{|x|}) = \langle C \rangle$
*Evaluar* $C(x)$ *y devolver su salida*

La simulación y la evaluación lleva tiempo polinomial.

# Circuitos **P**-uniformes y **P**

## Teorema

$\mathcal{L}$ es decidible por una familia **P**-uniforme de circuitos sii $\mathcal{L} \in \mathbf{P}$.

## Demostración.

$\subseteq$ Sea $M$ una máquina determinística que corre en tiempo polinomial tal que para todo $n$, $M(1^n) = \langle C_n \rangle$ y para todo $x \in \{0,1\}^n$:

$$x \in \mathcal{L} \qquad \text{sii} \qquad C_n(x) = 1.$$

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

*Simular* $M(1^{|x|}) = \langle C \rangle$
*Evaluar* $C(x)$ *y devolver su salida*

La simulación y la evaluación lleva tiempo polinomial.
Como $M'(x) = C_{|x|}(x) = \chi_{\mathcal{L}}(x)$, concluimos que $\mathcal{L} \in \mathbf{P}$.

# Circuitos **P**-uniformes y **P**

## Teorema

$\mathcal{L}$ es decidible por una familia **P**-uniforme de circuitos sii $\mathcal{L} \in \mathbf{P}$.

## Demostración.

$\subseteq$ Sea $M$ una máquina determinística que corre en tiempo polinomial tal que para todo $n$, $M(1^n) = \langle C_n \rangle$ y para todo $x \in \{0,1\}^n$:

$$x \in \mathcal{L} \qquad \text{sii} \qquad C_n(x) = 1.$$

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

*Simular* $M(1^{|x|}) = \langle C \rangle$
*Evaluar* $C(x)$ *y devolver su salida*

La simulación y la evaluación lleva tiempo polinomial. Como $M'(x) = C_{|x|}(x) = \chi_{\mathcal{L}}(x)$, concluimos que $\mathcal{L} \in \mathbf{P}$.

$\supseteq$ Seguir la demostración $\mathbf{P} \subseteq \mathbf{P}_{/\mathbf{poly}}$. La familia que se construye es, de hecho, **P**-uniforme. $\square$

# Circuitos **L**-uniformes

**Definición**

Una familia de circuitos $(C_n)_{n\in\mathbb{N}}$ es **L-uniforme** si existe una función $f : \{0,1\}^* \to \{0,1\}^*$ computable implícitamente en **L** tal que

$$f(1^n) = \langle C_n \rangle.$$

# Circuitos **L**-uniformes y **P**

**Teorema**

$\mathcal{L}$ es decidible por una familia de circuitos **L**-uniforme sii $\mathcal{L} \in \mathbf{P}$.

# Circuitos **L**-uniformes y **P**

**Teorema**

$\mathcal{L}$ es decidible por una familia de circuitos **L**-uniforme sii $\mathcal{L} \in \mathbf{P}$.

**Idea de la demostración.**

- $\subseteq$ es consecuencia de que $\mathbf{L} \subseteq \mathbf{P}$ y de que si $\mathcal{L}$ es decidible por una familia de circuitos **P**-uniforme entonces $\mathcal{L} \in \mathbf{P}$.
- $\supseteq$ Ver que en la demostración de $\mathbf{P} \subseteq \mathbf{P}_{/\mathbf{poly}}$ podemos construir los circuitos usando espacio logarítmico, gracias a que la máquina es *oblivious*.

**Ejercicio**

Demostrarlo.

# Las clases $\mathbf{NC}_{\mathrm{nu}}$ y $\mathbf{AC}_{\mathrm{nu}}$ generadas $\mathbf{L}$-uniformemente

## Clase de complejidad: $\mathbf{NC}$

- $\mathbf{NC}^d$ es la clase de lenguajes decidibles por una familia $\mathbf{L}$-uniforme $(C_n)_{n\in\mathbb{N}}$ de circuitos tal que
  - $|C_n|$ es polinomial en $n$ y
  - $\mathrm{prof}(C_n) = O(\log^d n)$
- $\mathbf{NC} = \bigcup_{i\geq 0} \mathbf{NC}^i$

## Clase de complejidad: $\mathbf{AC}$

- $\mathbf{AC}^d$ se define igual que $\mathbf{NC}^d$ pero permitimos nodos $\wedge$ y $\vee$ con *fan-in* arbitrario.
- $\mathbf{AC} = \bigcup_{i\geq 0} \mathbf{AC}^i$

# Resultados ya vistos se adaptan a la uniformidad

**Proposición**

PARITY $\in \mathbf{NC}^1$.

**Proposición**

SUMA $\in \mathbf{AC}^0$.

**Proposición**

$\mathbf{NC}^d \subseteq \mathbf{AC}^d \subseteq \mathbf{NC}^{d+1}$. Por lo tanto, $\mathbf{AC} = \mathbf{NC}$.

**Observación**

$\mathbf{AC} = \mathbf{NC} \subseteq \mathbf{P}$

# Cómputo paralelo

Un problema tiene una **solución paralela eficiente** si puede ser resuelto para entradas de tamaño $n$ usando una computadora paralela con una cantidad polinomial ($n^{O(1)}$) de procesadores en tiempo polilogarítmico ($\log^{O(1)} n$).

**Teorema**

$\mathcal{L}$ tiene una solución paralela eficiente sii $\mathcal{L} \in \mathbf{NC}$.

Ejemplos de problemas en $\mathbf{NC}$

- en enteros: suma, multiplicación y división
- en matrices: suma y multiplicación, determinante, inversa, rango
- en grafos: camino mínimo, spanning tree mínimo
- ...

## Teorema

$\mathbf{NC}^1 \subseteq \mathbf{L}$.

## Teorema

$\mathbf{NC}^1 \subseteq \mathbf{L}$.

## Demostración.

Sea $\mathcal{L} \in \mathbf{NC}^1$ y sean

- $(C_n)_{n\in\mathbb{N}}$ una familia de circuitos tal que
  - $|C_n|$ es polinomial en $n$,
  - $\text{prof}(C_n) = O(\log n)$ y
  - $x \in \mathcal{L}$ sii $C_{|x|}(x) = 1$
- $M$ una máquina determinística que computa implícitamente en $\mathbf{L}$ la función $1^n \mapsto \langle C_n \rangle$

## Teorema

$\mathbf{NC}^1 \subseteq \mathbf{L}$.

## Demostración.

Sea $\mathcal{L} \in \mathbf{NC}^1$ y sean

- $(C_n)_{n\in\mathbb{N}}$ una familia de circuitos tal que
  - $|C_n|$ es polinomial en $n$,
  - $\text{prof}(C_n) = O(\log n)$ y
  - $x \in \mathcal{L}$ sii $C_{|x|}(x) = 1$
- $M$ una máquina determinística que computa implícitamente en $\mathbf{L}$ la función $1^n \mapsto \langle C_n \rangle$

Definimos la siguiente función $g(1^n, x, u)$ recursivamente:

- si $u$ es la codificación del $k$-ésimo $(1 \leq k \leq n)$ nodo de entrada de $C_n$, devolver $x(k-1)$
- si $u$ es la codificación de un nodo $*$ (con $* \in \{\wedge, \vee\}$) de $C_n$, con hijos $v_1, v_2$, devolver $g(1^n, x, v_1) * g(1^n, x, v_2)$
- si $u$ es la codificación de un nodo $\neg$ de $C_n$, con hijo $v$, devolver $\neg g(1^n, x, v)$

Tenemos

$$C_{|x|} = 1 \qquad \text{sii} \qquad g(1^{|x|}, x, u_0) = 1,$$

donde $u_0$ es la codificación de la salida (*sink*) de $C_{|x|}$.

Tenemos

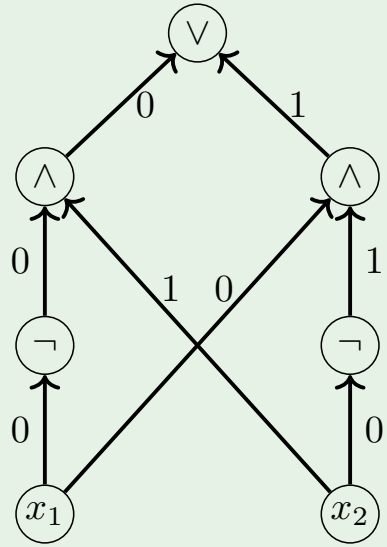
$$C_{|x|} = 1 \qquad \text{sii} \qquad g(1^{|x|}, x, u_0) = 1,$$

donde $u_0$ es la codificación de la salida (*sink*) de $C_{|x|}$.

Sea $n = |x|$.
Podemos computar $g(1^n, x, u_0)$ en espacio $O(\log n)$: para resolver la recursión necesitamos

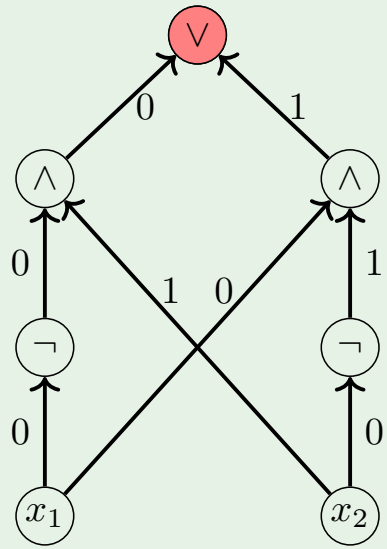
- llevar cuenta del camino desde la salida en $C_n$: $u_0 \to \cdots \to u_i$, con $i \in O(\log n)$. Cada camino se codifica con una palabra en $\{0,1\}^i$ (espacio $O(\log n)$):
  - 0 significa 'hijo izquierdo' o 'único hijo'
  - 1 significa 'hijo derecho'
- simular $M(1^n)$ para averiguar el tipo de nodo $u_i$: (espacio $O(\log n)$)

# Ejemplo: evaluación de un circuito




- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

# Ejemplo: evaluación de un circuito

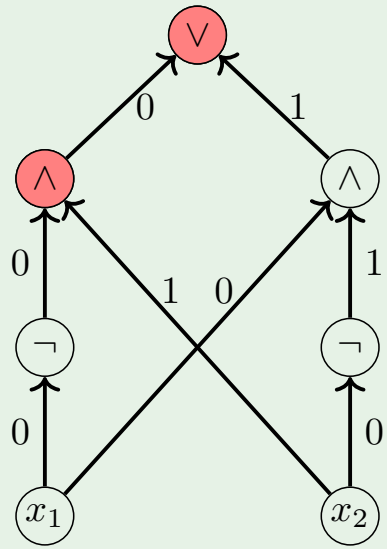



Estados:

$\varepsilon$

- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

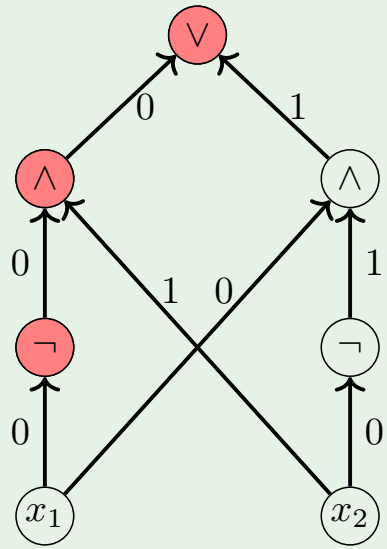
# Ejemplo: evaluación de un circuito




Estados:
$\varepsilon$
0

- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

# Ejemplo: evaluación de un circuito




Estados:

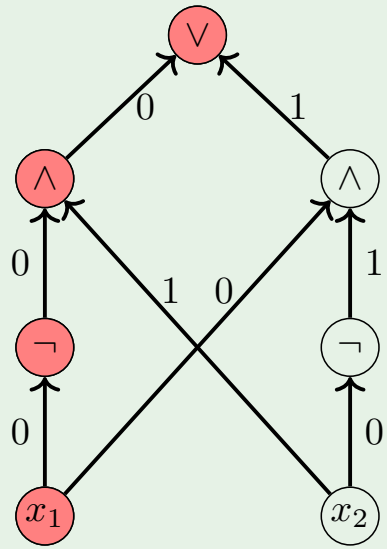
$\varepsilon$

0

00

- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

# Ejemplo: evaluación de un circuito




Estados:

$\varepsilon$

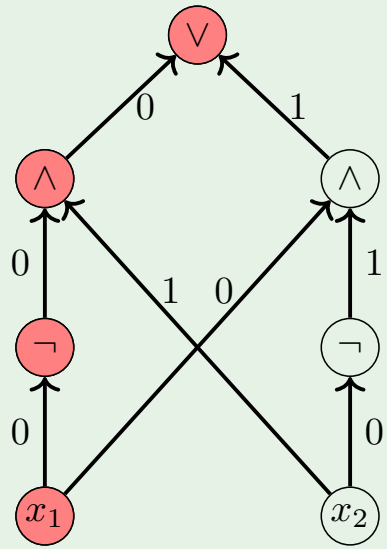
0

00

000

- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

# Ejemplo: evaluación de un circuito




Estados:

$\varepsilon$

0

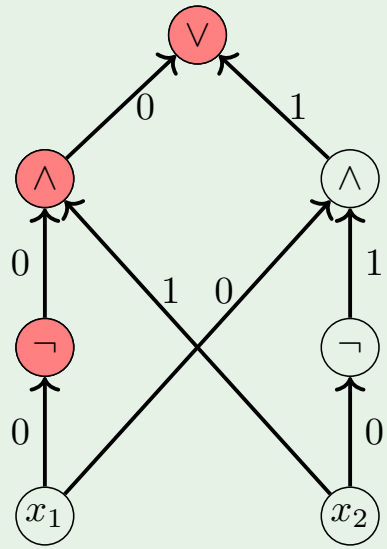
00

000

000[$x_1$]

- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

# Ejemplo: evaluación de un circuito




Estados:

$\varepsilon$

0

00

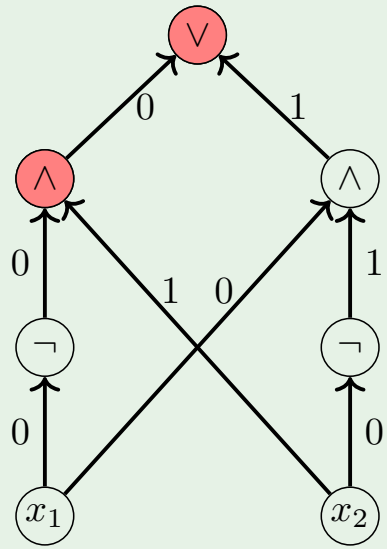
000

$000[x_1]$

$00[x_1]$

- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

# Ejemplo: evaluación de un circuito

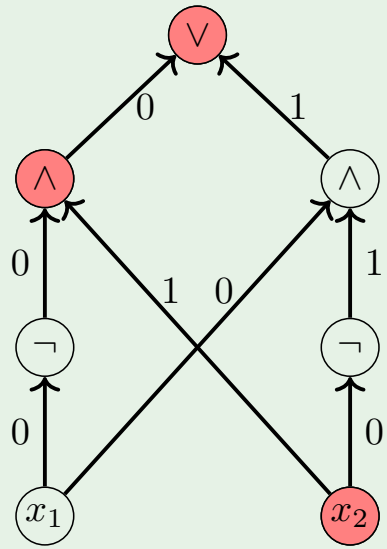



Estados:

$\varepsilon$
0
00
000
$000[x_1]$
$00[x_1]$
$0[\neg x_1]$

- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

# Ejemplo: evaluación de un circuito




Estados:

$\varepsilon$

0

00

000

$000[x_1]$

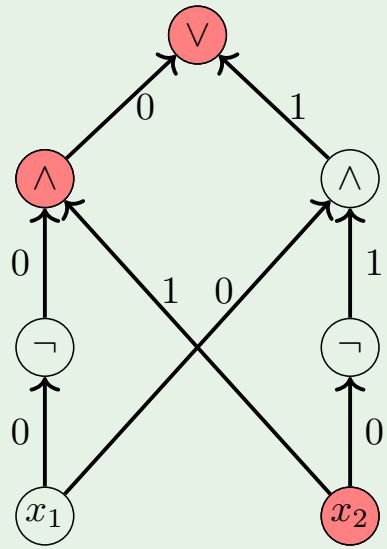
$00[x_1]$

$0[\neg x_1]$

$0[\neg x_1]1$

- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

# Ejemplo: evaluación de un circuito

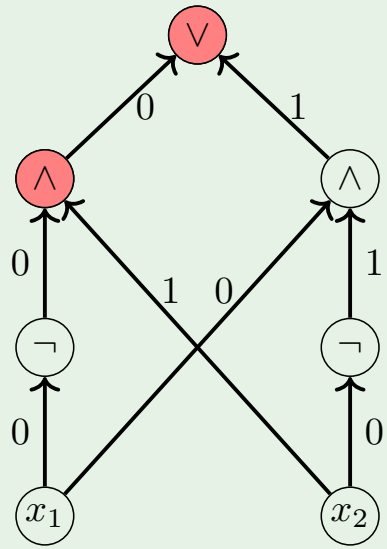



Estados:

$\varepsilon$
0
00
000
$000[x_1]$
$00[x_1]$
$0[\neg x_1]$
$0[\neg x_1]1$
$0[\neg x_1]1[x_2]$

- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

# Ejemplo: evaluación de un circuito

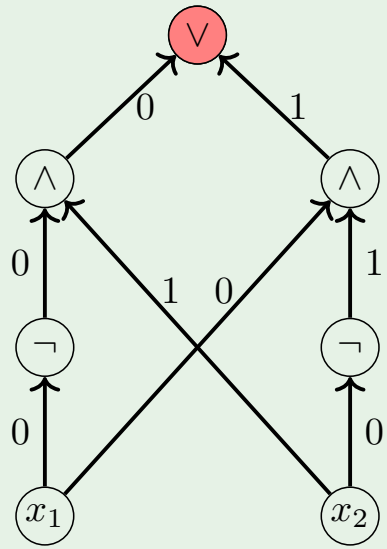



- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

Estados:

$\varepsilon$
0
00
000
$000[x_1]$
$00[x_1]$
$0[\neg x_1]$
$0[\neg x_1]1$
$0[\neg x_1]1[x_2]$
$0[\neg x_1][x_2]$

# Ejemplo: evaluación de un circuito

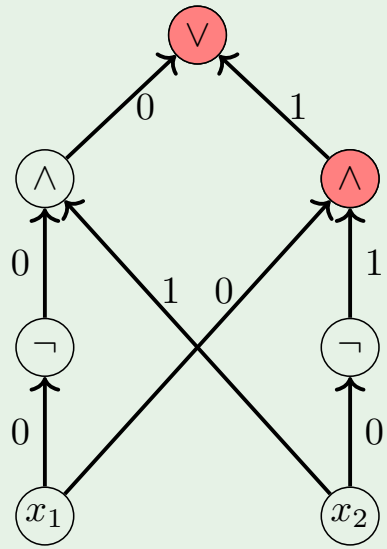



- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

Estados:

$\varepsilon$
0
00
000
$000[x_1]$
$00[x_1]$
$0[\neg x_1]$
$0[\neg x_1]1$
$0[\neg x_1]1[x_2]$
$0[\neg x_1][x_2]$
$[\neg x_1 \wedge x_2]$

# Ejemplo: evaluación de un circuito

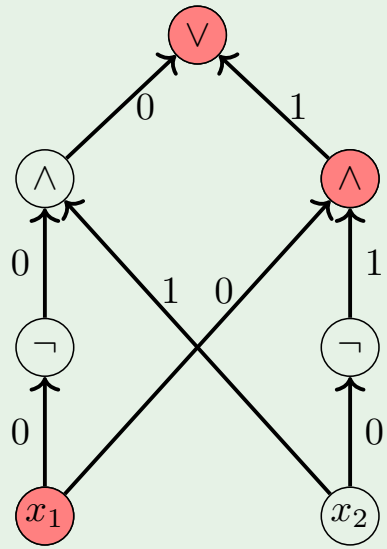



- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

Estados:

$\varepsilon$
$0$
$00$
$000$
$000[x_1]$
$00[x_1]$
$0[\neg x_1]$
$0[\neg x_1]1$
$0[\neg x_1]1[x_2]$
$0[\neg x_1][x_2]$
$[\neg x_1 \wedge x_2]$
$[\neg x_1 \wedge x_2]1$

# Ejemplo: evaluación de un circuito

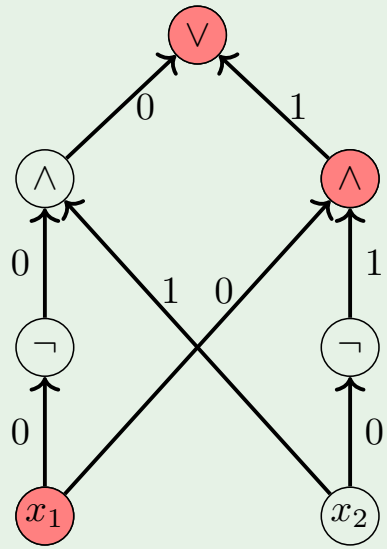



- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

Estados:

$\varepsilon$
$0$
$00$
$000$
$000[x_1]$
$00[x_1]$
$0[\neg x_1]$
$0[\neg x_1]1$
$0[\neg x_1]1[x_2]$
$0[\neg x_1][x_2]$
$[\neg x_1 \wedge x_2]$
$[\neg x_1 \wedge x_2]1$
$[\neg x_1 \wedge x_2]10$

# Ejemplo: evaluación de un circuito

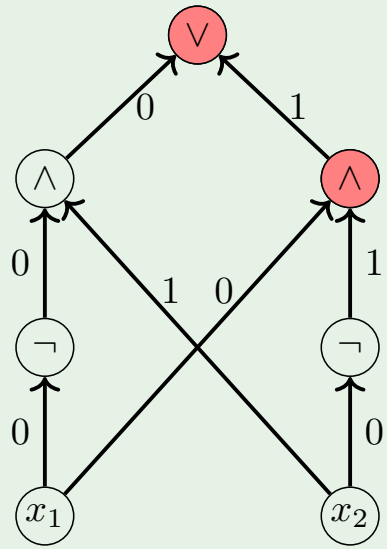



- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

Estados:

$\varepsilon$
$0$
$00$
$000$
$000[x_1]$
$00[x_1]$
$0[\neg x_1]$
$0[\neg x_1]1$
$0[\neg x_1]1[x_2]$
$0[\neg x_1][x_2]$
$[\neg x_1 \wedge x_2]$
$[\neg x_1 \wedge x_2]1$
$[\neg x_1 \wedge x_2]10$
$[\neg x_1 \wedge x_2]10[x_1]$

# Ejemplo: evaluación de un circuito

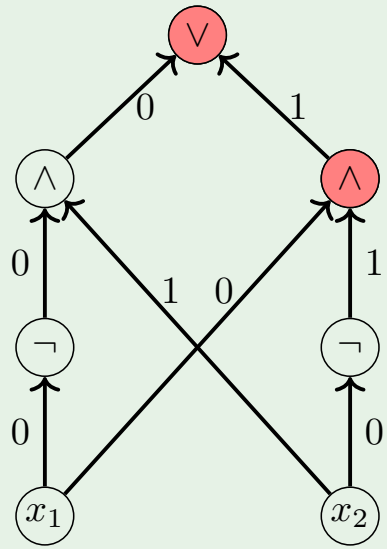



- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

Estados:

$\varepsilon$
$0$
$00$
$000$
$000[x_1]$
$00[x_1]$
$0[\neg x_1]$
$0[\neg x_1]1$
$0[\neg x_1]1[x_2]$
$0[\neg x_1][x_2]$
$[\neg x_1 \wedge x_2]$
$[\neg x_1 \wedge x_2]1$
$[\neg x_1 \wedge x_2]10$
$[\neg x_1 \wedge x_2]10[x_1]$
$[\neg x_1 \wedge x_2]1[x_1]$

# Ejemplo: evaluación de un circuito




- Es un recorrido DFS.
- Entre [ ] hay 1 bit que depende de los valores de $x_1$ y $x_2$. Ahí va calculando operaciones intermedias.
- Cada estado usa espacio $O(h)$, donde $h$ es la profundidad del circuito
- No entra todo el circuito en memoria: hay que ir pidiendo a $M$ los nodos a medida que se necesitan para no usar más que espacio logarítmico

Estados:

$\varepsilon$
$0$
$00$
$000$
$000[x_1]$
$00[x_1]$
$0[\neg x_1]$
$0[\neg x_1]1$
$0[\neg x_1]1[x_2]$
$0[\neg x_1][x_2]$
$[\neg x_1 \wedge x_2]$
$[\neg x_1 \wedge x_2]1$
$[\neg x_1 \wedge x_2]10$
$[\neg x_1 \wedge x_2]10[x_1]$
$[\neg x_1 \wedge x_2]1[x_1]$
$\ldots$

## Teorema

Sea $\mathcal{L} \in \mathbf{NSpace}(S(n))$. Existe una familia de circuitos $(C_n)_{n\in\mathbb{N}}$ y una máquina determinística $M$ tal que para todo $n$ y $x \in \{0,1\}^n$

- $x \in \mathcal{L}$ sii $C_n(x) = 1$
- $M(1^n) = \langle C_n \rangle$
- $C_n$ puede usar compuertas $\wedge$ y $\vee$ de *fan-in* arbitrario
- $|C_n| = 2^{O(S(n))}$
- $\text{prof}(C_n) = O(S(n))$

## Demostración.

Sea $N$ una máquina no-determinística que usa espacio $O(S(n))$ tal que $x \in \mathcal{L}$ sii existe un cómputo aceptador de $N(x)$.

## Demostración.

Sea $N$ una máquina no-determinística que usa espacio $O(S(n))$ tal que $x \in \mathcal{L}$ sii existe un cómputo aceptador de $N(x)$. Consideramos el grafo de configuraciones $G_{N,x}$, que tiene $m = 2^{c \cdot S(|x|)}$ nodos para alguna constante $c$.

## Demostración.

Sea $N$ una máquina no-determinística que usa espacio $O(S(n))$ tal que $x \in \mathcal{L}$ sii existe un cómputo aceptador de $N(x)$.
Consideramos el grafo de configuraciones $G_{N,x}$, que tiene $m = 2^{c \cdot S(|x|)}$ nodos para alguna constante $c$.
Sea $A_x \in \{0,1\}^{m \times m}$ la matriz de adyacencia de $G_{N,x}$

- $(A_x)_{ij} = 1$ sii $i$ evoluciona en un paso en $j$.
- $B_x = A_x \vee I$: $(B_x)_{ij} = 1$ sii $j$ es alcanzable desde $i$ en $\leq 1$ pasos.
- $(B_x^\ell)_{ij} = 1$ sii $j$ es alcanzable desde $i$ en $\leq \ell$ pasos.
  Consideramos una multiplicación $\otimes$ de matrices donde $\vee$ juega de $+$ y $\wedge$ juega de $\cdot$
  En general para $C, D \in \{0,1\}^{m \times m}$:

$$(C \otimes D)_{ij} = \bigvee_{1 \leq k \leq m} C_{ik} \wedge D_{kj}$$

  (circuito de profundidad constante gracias al *fan-in* arbitrario)

Representamos matrices de dimensión $m \times m$ en circuitos: cada nodo es una posición de la matriz. Pensemos en una matriz como una tira de $m^2$ nodos.

Representamos matrices de dimensión $m \times m$ en circuitos: cada nodo es una posición de la matriz. Pensemos en una matriz como una tira de $m^2$ nodos.
Sea $n = |x|$. $C_n$ hace esto:

- calcula $A_x$: profundidad constante.
- calcula $B_x$: ídem.
- calcula $D_x = B_x^m = B_x^{2^{c \cdot S(n)}}$: profundidad $O(\log m) = O(\log 2^{c \cdot S(n)}) = O(S(n))$.
  Ejemplo: $B_x^{2^3} = ((B_x \cdot B_x) \cdot (B_x \cdot B_x)) \cdot ((B_x \cdot B_x) \cdot (B_x \cdot B_x))$
- la salida es $(D_x)_{1,2}$
  (suponemos que la configuración inicial es 1 y la final es 2)

Representamos matrices de dimensión $m \times m$ en circuitos: cada nodo es una posición de la matriz. Pensemos en una matriz como una tira de $m^2$ nodos.
Sea $n = |x|$. $C_n$ hace esto:

- calcula $A_x$: profundidad constante.
- calcula $B_x$: ídem.
- calcula $D_x = B_x^m = B_x^{2^{c \cdot S(n)}}$: profundidad $O(\log m) = O(\log 2^{c \cdot S(n)}) = O(S(n))$.
  Ejemplo: $B_x^{2^3} = ((B_x \cdot B_x) \cdot (B_x \cdot B_x)) \cdot ((B_x \cdot B_x) \cdot (B_x \cdot B_x))$
- la salida es $(D_x)_{1,2}$
  (suponemos que la configuración inicial es 1 y la final es 2)

Tenemos

- $C_n(x) = 1$ sii $N$ acepta $x$

Representamos matrices de dimensión $m \times m$ en circuitos: cada nodo es una posición de la matriz. Pensemos en una matriz como una tira de $m^2$ nodos.
Sea $n = |x|$. $C_n$ hace esto:

- calcula $A_x$: profundidad constante.
- calcula $B_x$: ídem.
- calcula $D_x = B_x^m = B_x^{2^{c \cdot S(n)}}$: profundidad $O(\log m) = O(\log 2^{c \cdot S(n)}) = O(S(n))$.
  Ejemplo: $B_x^{2^3} = ((B_x \cdot B_x) \cdot (B_x \cdot B_x)) \cdot ((B_x \cdot B_x) \cdot (B_x \cdot B_x))$
- la salida es $(D_x)_{1,2}$
  (suponemos que la configuración inicial es 1 y la final es 2)

Tenemos

- $C_n(x) = 1$ sii $N$ acepta $x$
- $\text{prof}(C_n) = O(S(n))$

Representamos matrices de dimensión $m \times m$ en circuitos: cada nodo es una posición de la matriz. Pensemos en una matriz como una tira de $m^2$ nodos.
Sea $n = |x|$. $C_n$ hace esto:

- calcula $A_x$: profundidad constante.
- calcula $B_x$: ídem.
- calcula $D_x = B_x^m = B_x^{2^{c \cdot S(n)}}$: profundidad $O(\log m) = O(\log 2^{c \cdot S(n)}) = O(S(n))$.
  Ejemplo: $B_x^{2^3} = ((B_x \cdot B_x) \cdot (B_x \cdot B_x)) \cdot ((B_x \cdot B_x) \cdot (B_x \cdot B_x))$
- la salida es $(D_x)_{1,2}$
  (suponemos que la configuración inicial es 1 y la final es 2)

Tenemos

- $C_n(x) = 1$ sii $N$ acepta $x$
- $\text{prof}(C_n) = O(S(n))$
- $|C_n| = 2^{O(S(n))}$

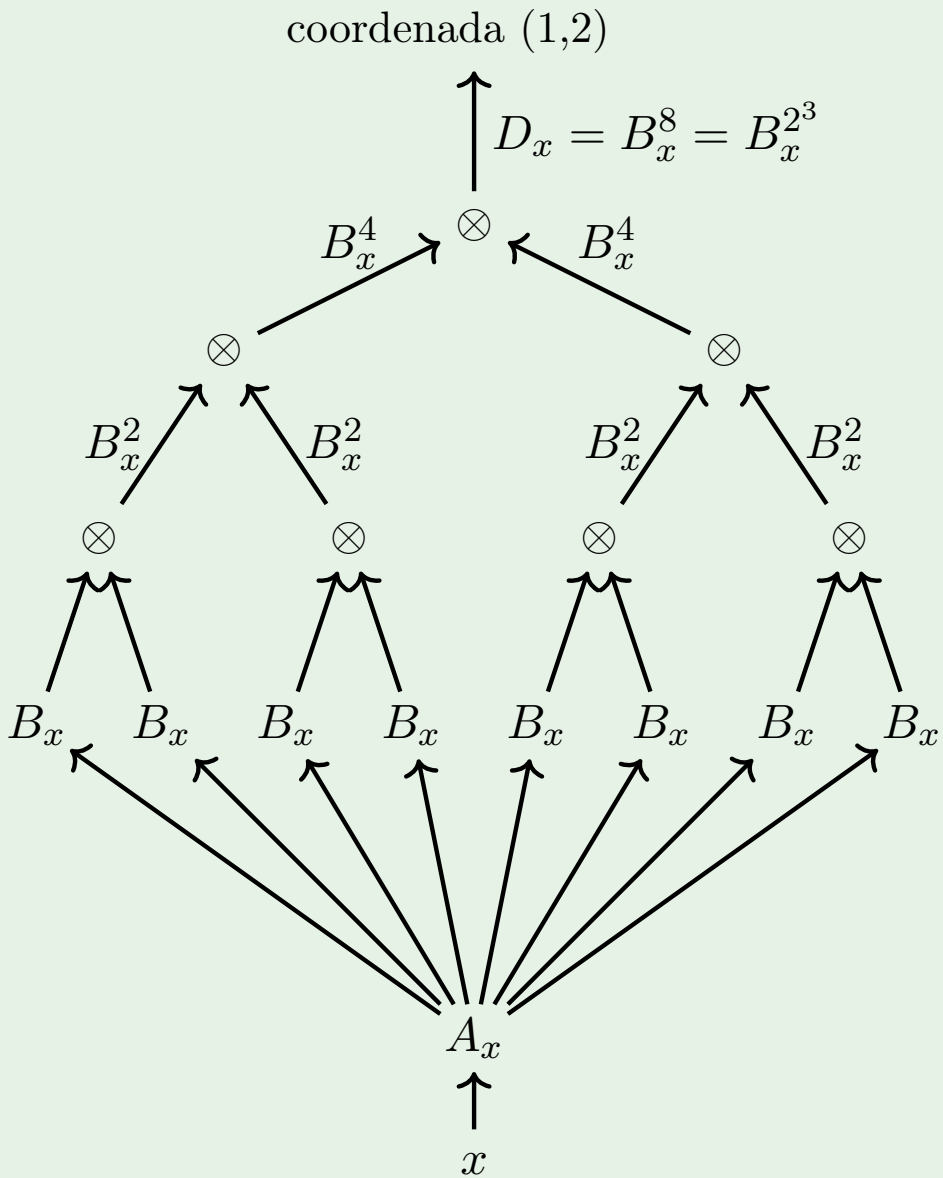
Representamos matrices de dimensión $m \times m$ en circuitos: cada nodo es una posición de la matriz. Pensemos en una matriz como una tira de $m^2$ nodos.
Sea $n = |x|$. $C_n$ hace esto:

- calcula $A_x$: profundidad constante.
- calcula $B_x$: ídem.
- calcula $D_x = B_x^m = B_x^{2^{c \cdot S(n)}}$: profundidad $O(\log m) = O(\log 2^{c \cdot S(n)}) = O(S(n))$.
  Ejemplo: $B_x^{2^3} = ((B_x \cdot B_x) \cdot (B_x \cdot B_x)) \cdot ((B_x \cdot B_x) \cdot (B_x \cdot B_x))$
- la salida es $(D_x)_{1,2}$
  (suponemos que la configuración inicial es 1 y la final es 2)

Tenemos

- $C_n(x) = 1$ sii $N$ acepta $x$
- $\text{prof}(C_n) = O(S(n))$
- $|C_n| = 2^{O(S(n))}$
- La construcción es uniforme, es decir, existe una máquina determinística $M$ tal que $M(1^n) = \langle C_n \rangle$

# Ejemplo del circuito para $c \cdot S(n) = 3$




- $|x| = n$
- $|A_x| = |B_x| = |D_x| = (2^{c \cdot S(n)})^2 = 2^{O(S(n))}$
- En total, el circuito tiene $2^{O(S(n))}$ nodos y profundidad $O(S(n))$

## Corolario

$\mathbf{NL} \subseteq \mathbf{AC}^1$.

## Corolario

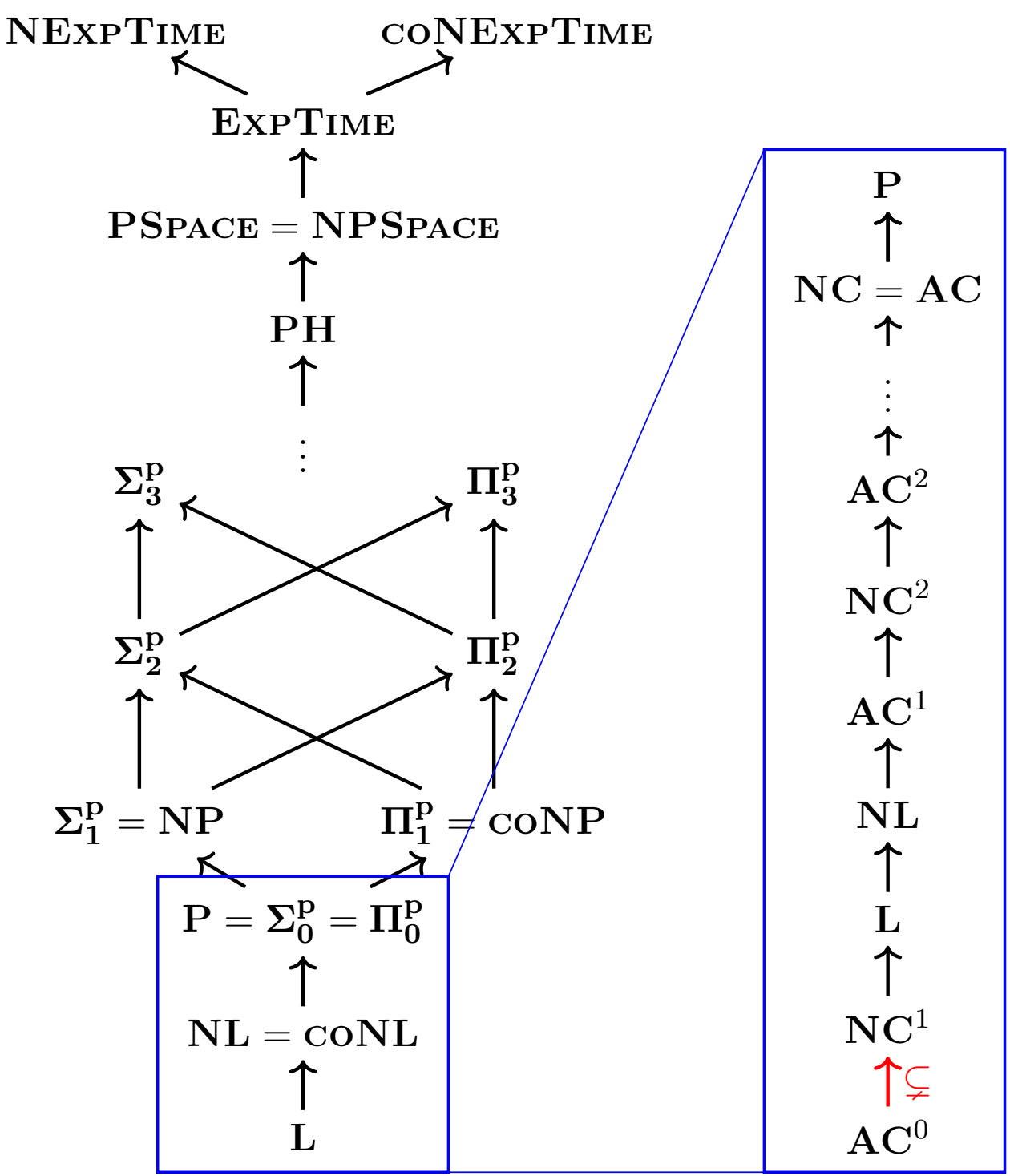
$\mathbf{NL} \subseteq \mathbf{AC}^1$.

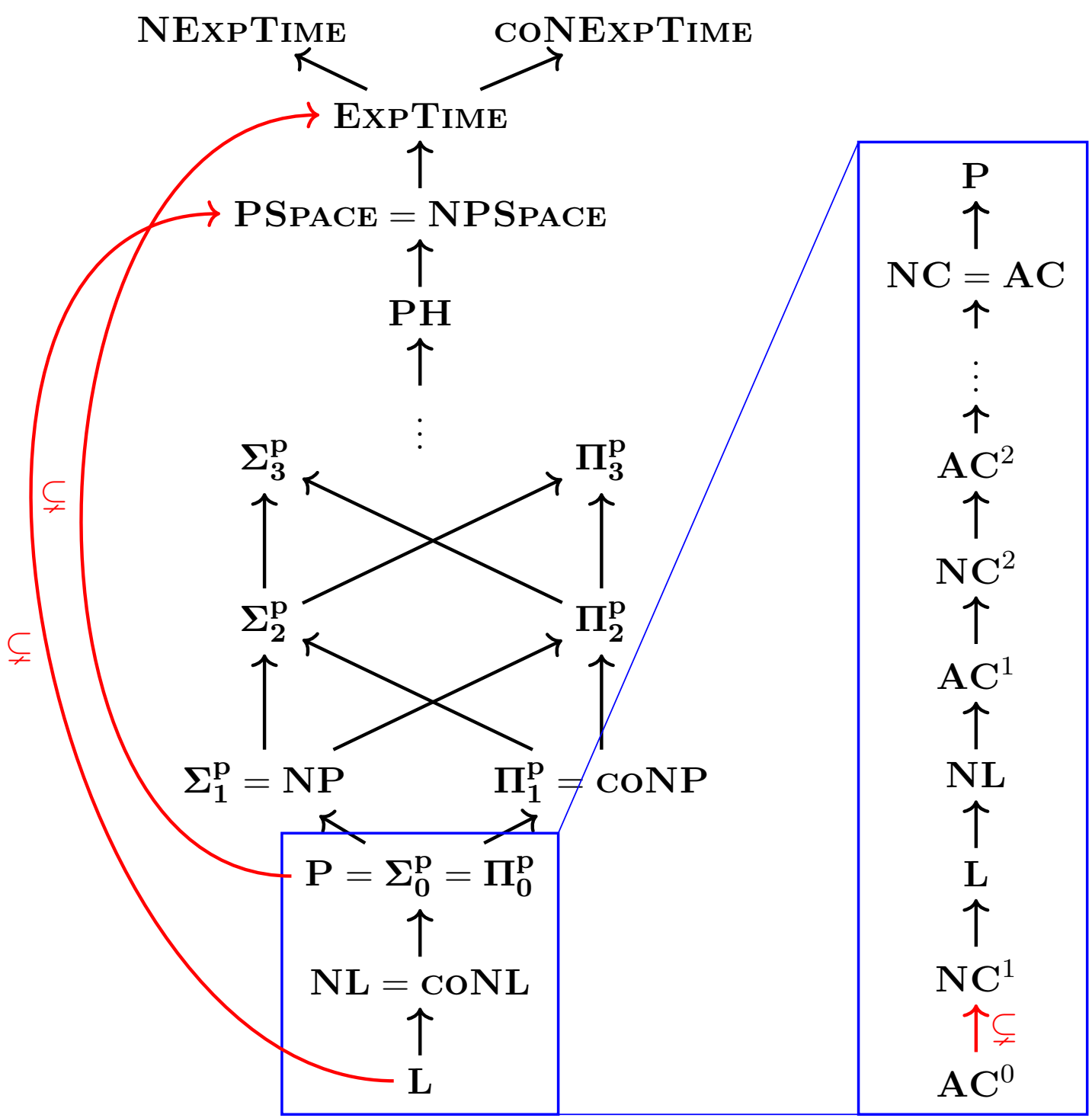
## Demostración.

Sea $\mathcal{L} \in \mathbf{NSpace}(\log n)$. Existe una familia de circuitos $(C_n)_{n\in\mathbb{N}}$ y una máquina determinística $M$ tal que para todo $n \geq 1$ y $x \in \{0,1\}^n$

- $x \in \mathcal{L}$ sii $C_n(x) = 1$
- $M(1^n) = \langle C_n \rangle$
- $C_n$ puede usar compuertas $\wedge$ y $\vee$ de *fan-in* arbitrario
- $|C_n| = 2^{O(\log n)} = n^{O(1)}$
- $\text{prof}(C_n) = O(\log n)$

Solo falta ver que la familia $(C_n)_{n\in\mathbb{N}}$ es $\mathbf{L}$-uniforme.
Alcanza con ver que $1^n \mapsto \langle C_n \rangle$ es computable implícitamente en $\mathbf{L}$ por $M$. □


NExpTime
coNExpTime
ExpTime
PSpace = NPSpace
PH
$\Sigma_3^p$
$\Pi_3^p$
$\Sigma_2^p$
$\Pi_2^p$
$\Sigma_1^p = NP$
$\Pi_1^p = coNP$
$P = \Sigma_0^p = \Pi_0^p$
NL = coNL
L
P
NC = AC
$AC^2$
$NC^2$
$AC^1$
NL
L
$NC^1$
$\subsetneq$
$AC^0$


NExpTime
coNExpTime
ExpTime
PSpace = NPSpace
PH
$\Sigma_3^p$
$\Pi_3^p$
$\Sigma_2^p$
$\Pi_2^p$
$\Sigma_1^p = NP$
$\Pi_1^p = coNP$
$P = \Sigma_0^p = \Pi_0^p$
NL = coNL
L
⊆
⊆
P
NC = AC
$AC^2$
$NC^2$
$AC^1$
NL
L
$NC^1$
⊆
$AC^0$

# **P**-completitud

Clase 12

- Las clases $\mathbf{NC}_{\text{nu}}$ y $\mathbf{AC}_{\text{nu}}$
- Uniformidad: las clases **NC** y **AC**
- **P**-completitud

# NC vs P

- ¿$\mathbf{NC} = \mathbf{P}$ o $\mathbf{NC} \subsetneq \mathbf{P}$? Es una pregunta abierta.
- ¿Cualquier problema factible tiene solución paralela eficiente?
- Sabemos que $\mathbf{NC}^1 \subsetneq \mathbf{PSpace}$ porque $\mathbf{NC}^1 \subseteq \mathbf{L}$ y $\mathbf{L} \subsetneq \mathbf{PSpace}$.
- No sabemos $\mathbf{NC} \stackrel{?}{=} \mathbf{P}$.
- Ni siquiera sabemos $\mathbf{NC}^1 \stackrel{?}{=} \mathbf{PH}$.

# NC vs P

- ¿$\mathbf{NC} = \mathbf{P}$ o $\mathbf{NC} \subsetneq \mathbf{P}$? Es una pregunta abierta.
- ¿Cualquier problema factible tiene solución paralela eficiente?
- Sabemos que $\mathbf{NC}^1 \subsetneq \mathbf{PSpace}$ porque $\mathbf{NC}^1 \subseteq \mathbf{L}$ y $\mathbf{L} \subsetneq \mathbf{PSpace}$.
- No sabemos $\mathbf{NC} \stackrel{?}{=} \mathbf{P}$.
- Ni siquiera sabemos $\mathbf{NC}^1 \stackrel{?}{=} \mathbf{PH}$.

**Clase de complejidad: P-completo**

$\mathcal{L}$ es **P-completo** si $\mathcal{L} \in \mathbf{P}$ y $\mathcal{L}' \leq_\ell \mathcal{L}$ para todo $\mathcal{L}' \in \mathbf{P}$.

Si apostamos a que $\mathbf{NC} \neq \mathbf{P}$, entonces los problemas **P-completos** son los que, a pesar de ser factibles, no tienen una solución paralela eficiente.

# Un problema **P-completo**

**Problema: Evaluación de un circuito**

$$\mathsf{CIRC\text{-}EVAL} = \{\langle C, x\rangle : \begin{array}{l} C \text{ es un circuito con } n \text{ entradas con única} \\ \text{salida, } x \in \{0,1\}^n, \text{ y } C(x) = 1. \end{array} \}$$

# Un problema **P-completo**

## Problema: Evaluación de un circuito

$$\text{CIRC-EVAL} = \{\langle C, x\rangle : \begin{array}{l} C \text{ es un circuito con } n \text{ entradas con única} \\ \text{salida, } x \in \{0,1\}^n, \text{ y } C(x) = 1. \end{array} \}$$

## Proposición

CIRC-EVAL es **P-completo**.

# Un problema **P-completo**

## Problema: Evaluación de un circuito

$$\text{CIRC-EVAL} = \{\langle C, x\rangle : \begin{array}{l} C \text{ es un circuito con } n \text{ entradas con única} \\ \text{salida, } x \in \{0,1\}^n, \text{ y } C(x) = 1. \end{array} \}$$

## Proposición

CIRC-EVAL es **P-completo**.

## Demostración.

Es claro que CIRC-EVAL $\in \mathbf{P}$.

# Un problema **P-completo**

## Problema: Evaluación de un circuito

$$\text{CIRC-EVAL} = \{\langle C, x\rangle : \begin{array}{l} C \text{ es un circuito con } n \text{ entradas con única} \\ \text{salida, } x \in \{0,1\}^n, \text{ y } C(x) = 1. \end{array} \}$$

## Proposición

CIRC-EVAL es **P-completo**.

## Demostración.

Es claro que CIRC-EVAL $\in \mathbf{P}$.
Supongamos $\mathcal{L} \in \mathbf{P}$ y sea $M$ una máquina determinística que corre en tiempo polinomial y $\mathcal{L}(M) = \mathcal{L}$. Ya vimos que $\mathcal{L}$ es decidible por una familia **L**-uniforme de circuitos.

# Un problema **P-completo**

## Problema: Evaluación de un circuito

$$\mathsf{CIRC\text{-}EVAL} = \{\langle C, x\rangle : \begin{array}{l} C \text{ es un circuito con } n \text{ entradas con única} \\ \text{salida, } x \in \{0,1\}^n, \text{ y } C(x) = 1. \end{array} \}$$

## Proposición

CIRC-EVAL es **P-completo**.

## Demostración.

Es claro que CIRC-EVAL $\in \mathbf{P}$.
Supongamos $\mathcal{L} \in \mathbf{P}$ y sea $M$ una máquina determinística que corre en tiempo polinomial y $\mathcal{L}(M) = \mathcal{L}$. Ya vimos que $\mathcal{L}$ es decidible por una familia **L**-uniforme de circuitos.
Existe una función $f : \{0,1\}^* \to \{0,1\}^*$ computable implícitamente en **L** tal que $f(1^n) = \langle C_n \rangle$.

$$x \in \mathcal{L} \quad \text{sii} \quad C_{|x|}(x) = 1 \quad \text{sii} \quad \langle f(1^{|x|}), x\rangle \in \mathsf{CIRC\text{-}EVAL}.$$

$x \mapsto \langle f(1^{|x|}), x\rangle$ es computable implícitamente en **L**.
Entonces $\mathcal{L} \leq_\ell$ CIRC-EVAL. □

# **P**-completitud

## Ejercicio

Si $\mathcal{L} \in \mathbf{NC}$ y $\mathcal{L}' \leq_\ell \mathcal{L}$, entonces $\mathcal{L}' \in \mathbf{NC}$.

# **P**-completitud

## Ejercicio

Si $\mathcal{L} \in \mathbf{NC}$ y $\mathcal{L}' \leq_\ell \mathcal{L}$, entonces $\mathcal{L}' \in \mathbf{NC}$.

## Teorema

Supongamos que $\mathcal{L}$ es **P-completo**.

1. $\mathcal{L} \in \mathbf{NC}$ sii $\mathbf{P} = \mathbf{NC}$
2. $\mathcal{L} \in \mathbf{L}$ sii $\mathbf{P} = \mathbf{L}$

# P-completitud

## Ejercicio

Si $\mathcal{L} \in \mathbf{NC}$ y $\mathcal{L}' \leq_\ell \mathcal{L}$, entonces $\mathcal{L}' \in \mathbf{NC}$.

## Teorema

Supongamos que $\mathcal{L}$ es **P-completo**.

1. $\mathcal{L} \in \mathbf{NC}$ sii $\mathbf{P} = \mathbf{NC}$
2. $\mathcal{L} \in \mathbf{L}$ sii $\mathbf{P} = \mathbf{L}$

## Ejercicio

Probar el ítem 2.

## Demostración del ítem 1.

$\Leftarrow$ es trivial.
Veamos que si $\mathcal{L} \in$ **P-completo** y $\mathcal{L} \in \mathbf{NC}$ entonces $\mathbf{P} \subseteq \mathbf{NC}$ (ya sabemos que $\mathbf{NC} \subseteq \mathbf{P}$).

## $\mathcal{L} \in$ **P-completo** y $\mathcal{L} \in$ **NC**, entonces **P** $\subseteq$ **NC** (idea)

Como $\mathcal{L} \in \mathbf{NC}$, existe $e$ y una familia **L**-uniforme $(C_n)_{n\in\mathbb{N}}$ de circuitos tal que $|C_n| = O(n^e)$, $\text{prof}(C_n) = O(\log^e n)$, y para todo $x \in \{0,1\}^n$

$$C_n(x) = 1 \qquad \text{sii} \qquad x \in \mathcal{L}.$$

## $\mathcal{L} \in$ **P-completo** y $\mathcal{L} \in$ **NC**, entonces **P** $\subseteq$ **NC** (idea)

Como $\mathcal{L} \in \mathbf{NC}$, existe $e$ y una familia $\mathbf{L}$-uniforme $(C_n)_{n\in\mathbb{N}}$ de circuitos tal que $|C_n| = O(n^e)$, $\text{prof}(C_n) = O(\log^e n)$, y para todo $x \in \{0,1\}^n$

$$C_n(x) = 1 \qquad \text{sii} \qquad x \in \mathcal{L}.$$

Sea $\mathcal{L}' \in \mathbf{P}$. Existe $f' : \{0,1\}^* \to \{0,1\}^*$ computable implícitamente en $\mathbf{L}$ tal que para todo $x \in \{0,1\}^n$,

$$x \in \mathcal{L}' \qquad \text{sii} \qquad f'(x) \in \mathcal{L}.$$

Existe $c$ tal que $|f'(x)| \leq c \cdot n^c$. Por simplicidad, supongamos que $|f'(x)| = c \cdot n^c$.

## $\mathcal{L} \in$ **P-completo** y $\mathcal{L} \in$ **NC**, entonces **P** $\subseteq$ **NC** (idea)

Como $\mathcal{L} \in \mathbf{NC}$, existe $e$ y una familia $\mathbf{L}$-uniforme $(C_n)_{n\in\mathbb{N}}$ de circuitos tal que $|C_n| = O(n^e)$, $\text{prof}(C_n) = O(\log^e n)$, y para todo $x \in \{0,1\}^n$

$$C_n(x) = 1 \qquad \text{sii} \qquad x \in \mathcal{L}.$$

Sea $\mathcal{L}' \in \mathbf{P}$. Existe $f' : \{0,1\}^* \to \{0,1\}^*$ computable implícitamente en $\mathbf{L}$ tal que para todo $x \in \{0,1\}^n$,

$$x \in \mathcal{L}' \qquad \text{sii} \qquad f'(x) \in \mathcal{L}.$$

Existe $c$ tal que $|f'(x)| \leq c \cdot n^c$. Por simplicidad, supongamos que $|f'(x)| = c \cdot n^c$.
Como $\mathbf{L} \subseteq \mathbf{NC}$, existe $d$ y una familia $\mathbf{L}$-uniforme $(C'_n)_{n\in\mathbb{N}}$ de circuitos tal que $|C'_n| = O(n^d)$, $\text{prof}(C'_n) = O(\log^d n)$, y para todo $x \in \{0,1\}^n$ $C'_n(x) = f'(x)$ (En realidad $\mathbf{L} \subseteq \mathbf{NC}$ lo vimos para lenguajes pero vale funciones computables implícitamente en $\mathbf{L}$, como $f'$.)

## $\mathcal{L} \in$ **P-completo** y $\mathcal{L} \in$ **NC**, entonces **P** $\subseteq$ **NC** (idea)

Como $\mathcal{L} \in \mathbf{NC}$, existe $e$ y una familia $\mathbf{L}$-uniforme $(C_n)_{n\in\mathbb{N}}$ de circuitos tal que $|C_n| = O(n^e)$, $\text{prof}(C_n) = O(\log^e n)$, y para todo $x \in \{0,1\}^n$

$$C_n(x) = 1 \qquad \text{sii} \qquad x \in \mathcal{L}.$$

Sea $\mathcal{L}' \in \mathbf{P}$. Existe $f' : \{0,1\}^* \to \{0,1\}^*$ computable implícitamente en $\mathbf{L}$ tal que para todo $x \in \{0,1\}^n$,

$$x \in \mathcal{L}' \qquad \text{sii} \qquad f'(x) \in \mathcal{L}.$$

Existe $c$ tal que $|f'(x)| \leq c \cdot n^c$. Por simplicidad, supongamos que $|f'(x)| = c \cdot n^c$.
Como $\mathbf{L} \subseteq \mathbf{NC}$, existe $d$ y una familia $\mathbf{L}$-uniforme $(C'_n)_{n\in\mathbb{N}}$ de circuitos tal que $|C'_n| = O(n^d)$, $\text{prof}(C'_n) = O(\log^d n)$, y para todo $x \in \{0,1\}^n$ $C'_n(x) = f'(x)$ (En realidad $\mathbf{L} \subseteq \mathbf{NC}$ lo vimos para lenguajes pero vale funciones computables implícitamente en $\mathbf{L}$, como $f'$.) Entonces

$$x \in \mathcal{L}' \quad \text{sii} \quad f'(x) \in \mathcal{L} \quad \text{sii} \quad C_{c\cdot n^c}(f'(x)) = 1 \quad \text{sii} \quad C_{c\cdot n^c}(C'_n(x)) = 1$$

La composición de los circuitos $C$ y $C'$ tiene tamaño polinomial, profundidad polilogarítmica, y se construyen uniformemente a partir de $1^n$. Esto prueba que $\mathcal{L}' \in \mathbf{NC}$. □