Complejidad basada en Circuitos

IIC3242

Circuitos Booleanos

Definición

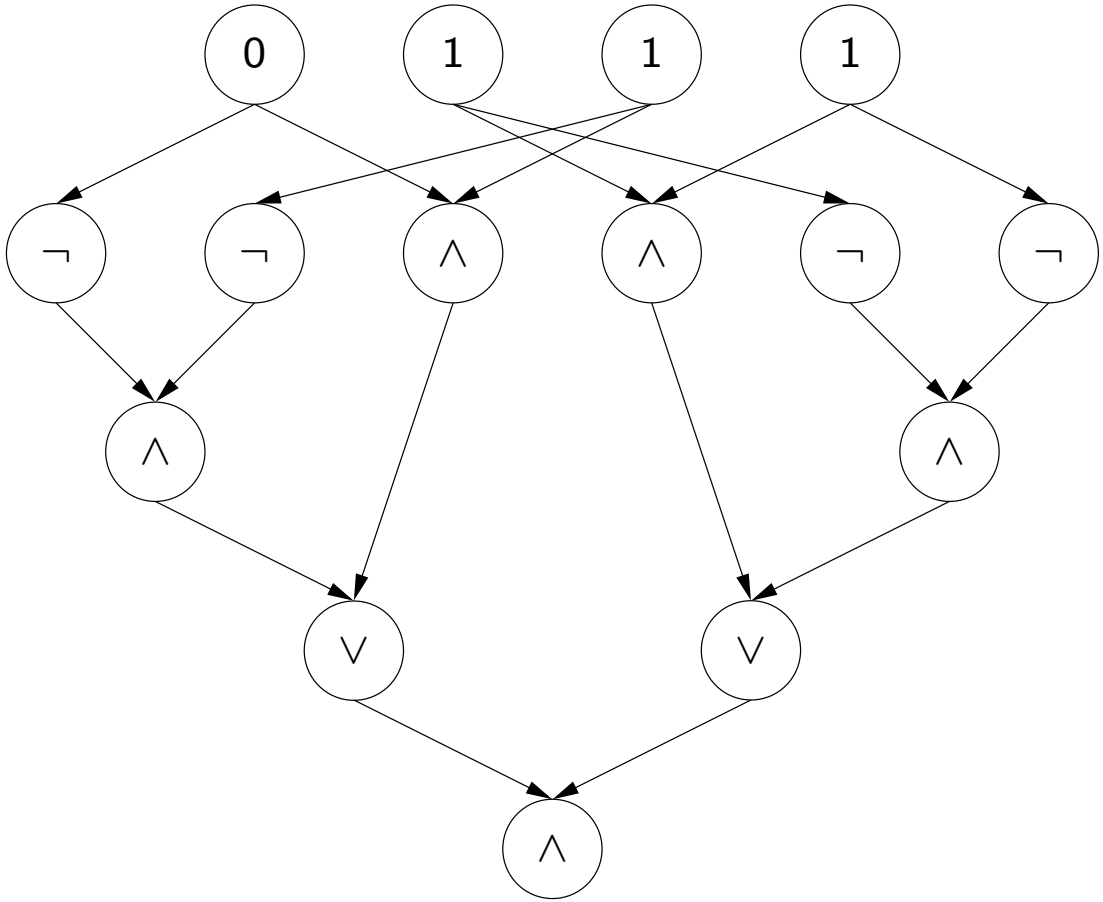
Un circuito Booleano es un grafo dirigido acíclico tal que:

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

Notación

- *Nodo de entrada: Nodo sin arcos de entrada*
- *Nodo interior: Nodo con arcos de entrada*
- *Nodo de salida: Nodo sin arcos de salida*

Circuito Booleano: Ejemplo

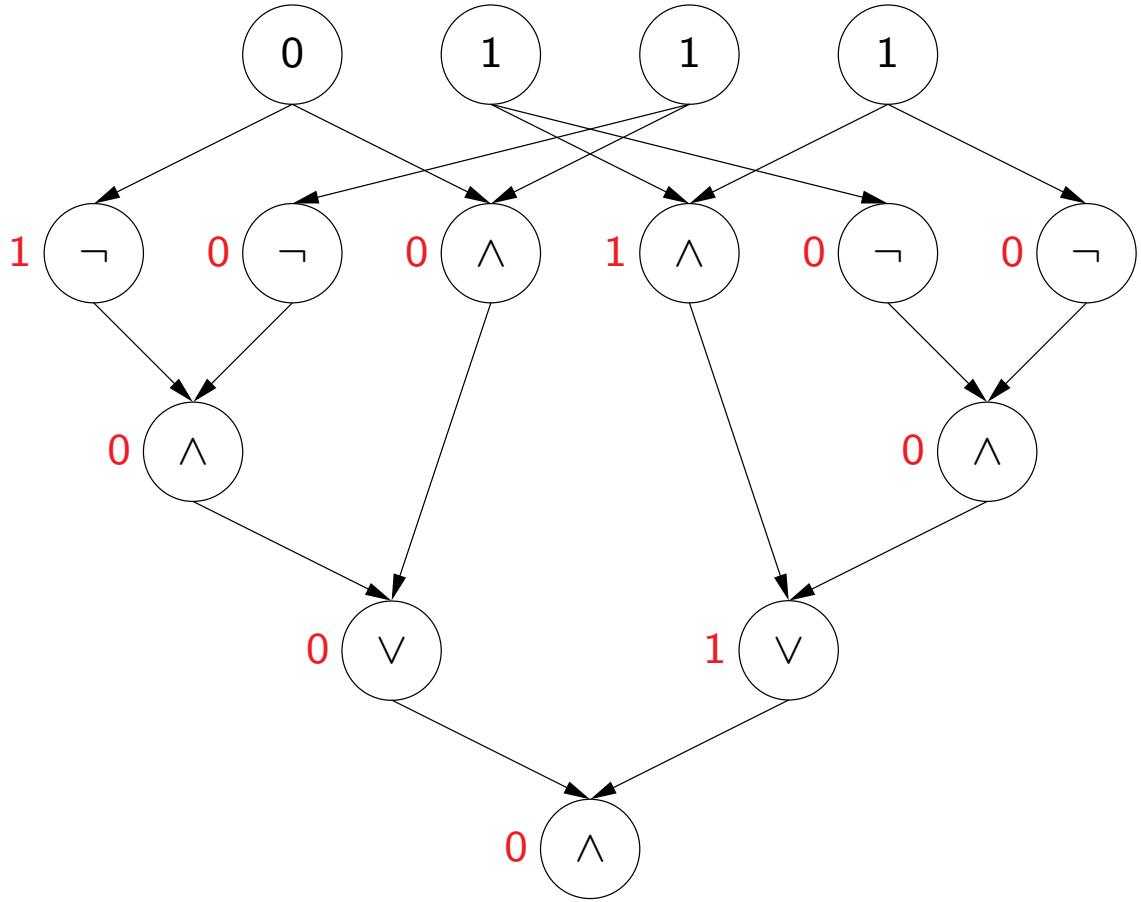


Circuitos Booleanos como funciones

Cada circuito Booleano puede ser interpretado como una función:

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

Circuitos Booleanos como funciones: Ejemplo



Circuitos Booleanos como funciones

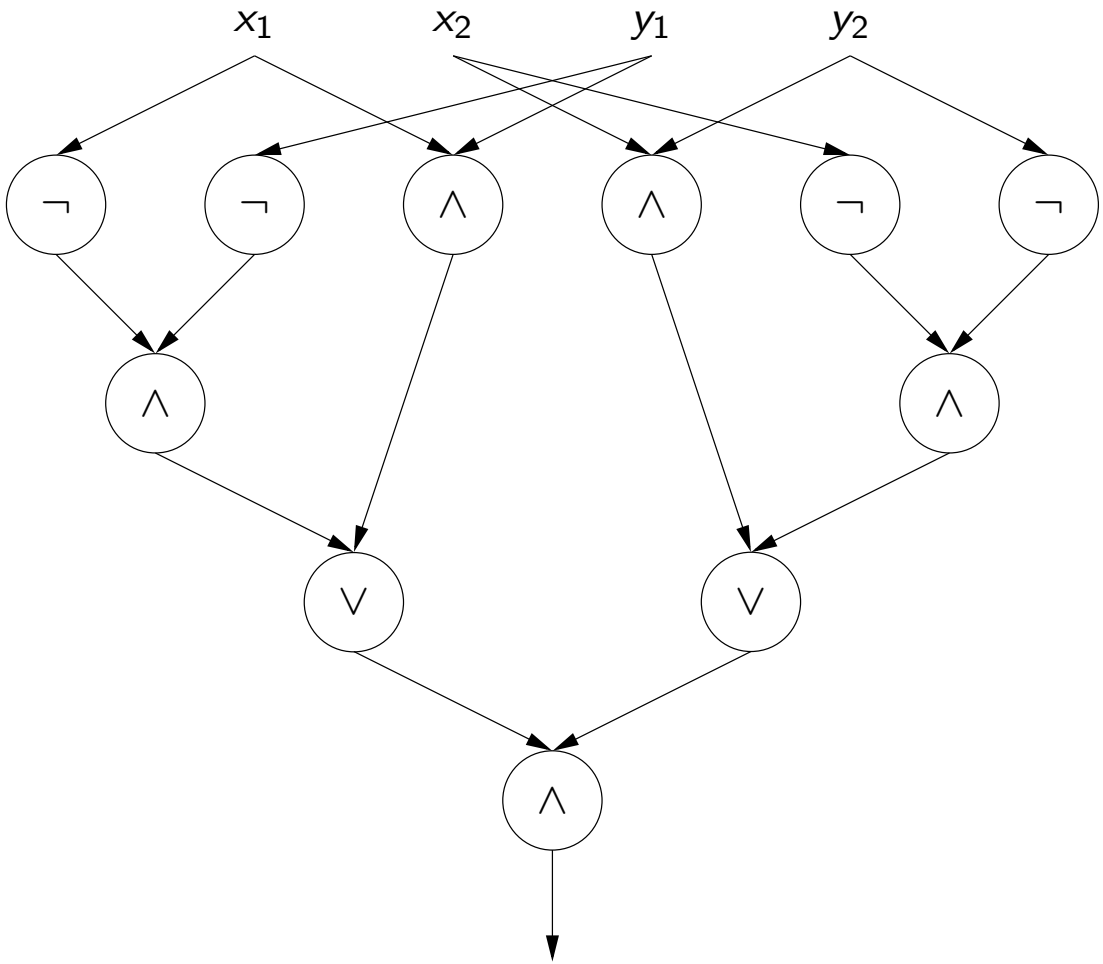
Circuito en el ejemplo anterior representa la función:

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

Para indicar la función que representa un circuito:

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

Circuito Booleano como función: Ejemplo



Circuitos Booleanos como funciones: Terminología

Definición

- *Tamaño de un circuito:* *Número de nodos con etiquetas* $\neg$, $\wedge$ *y* $\vee$
- *Profundidad de un circuito:* *Largo del camino más largo entre un nodo de entrada y el nodo de salida*

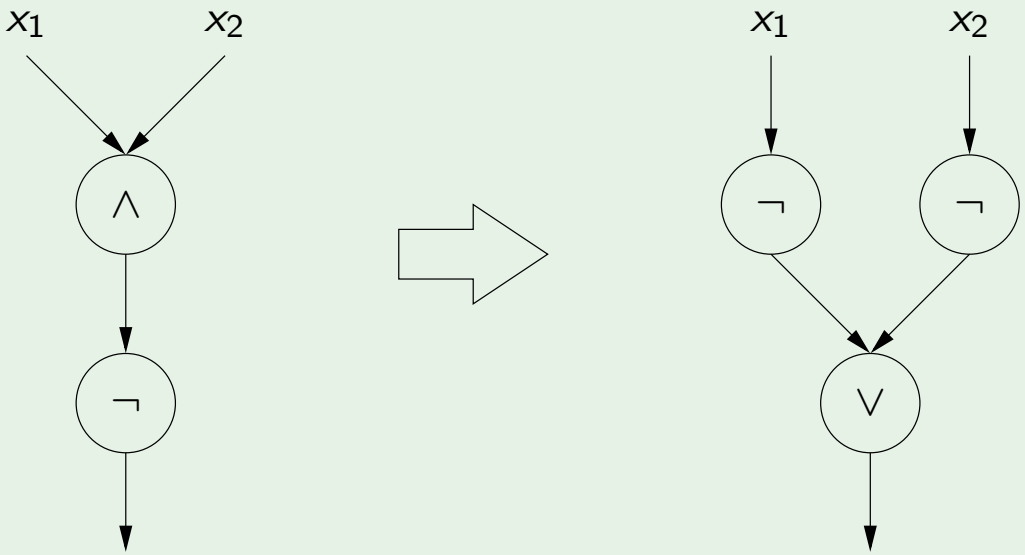
Ejemplo

El tamaño es 11 y la profundidad es 4 en el circuito de la transparencia anterior.

Circuitos Booleanos como funciones: Forma normal

Usando $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$ y $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$ podemos demostrar que cada circuito es equivalente a otro que sólo tiene negaciones en el primer nivel.

Ejemplo



Circuitos Booleanos como funciones: Forma normal

Aun más: El circuito generado tiene la misma profundidad que el circuito original y a lo más el doble de nodos.

De ahora en adelante: Suponemos que los circuitos sólo tienen negaciones en el primer nivel.

- Definimos el tamaño de un circuito sólo considerando el número de nodos con etiquetas $\wedge$ y $\vee$

Notación

- *tamaño*(C): *Tamaño del circuito* C
- *profundidad*(C): *Profundidad del circuito* C

Lenguajes aceptados por circuitos

Dado: Circuito $C(\bar{x})$ con n entradas

Definición

C define el lenguaje:

$$\{w \in \{0,1\}^n \mid C(w) = 1\}$$

Ejemplo

Para el caso anterior: $\{a_1 a_2 a_3 a_4 \in \{0,1\}^4 \mid a_1 = a_3 \text{ y } a_2 = a_4\}$

Lenguajes aceptados por circuitos

El lenguaje definido por un circuito siempre es finito.

- Tenemos que usar familias de circuitos para poder generar lenguajes infinitos

Definición

Una familia de circuitos $\{C_n\}_{n \geq 0}$ *acepta* $L \subseteq \{0,1\}^*$ *si:*

- C_n *tiene* n *entradas* (C_0 *es 0 ó 1*)
- *Para cada palabra* w *de largo* n: $w \in L$ *si y sólo si* $C_n(w) = 1$

Lenguajes aceptados por circuitos

La noción de aceptación que acabamos de definir es muy general.

Proposición

Cada $L \subseteq \{0,1\}^*$ es aceptado por una familia de circuitos $\{C_n\}$.

Ejercicio

Demuestre la proposición.

Lenguajes aceptados por circuitos

Si queremos utilizar una familia de circuitos como un algoritmo tenemos que introducir una noción de **uniformidad.**

- Debe existir un algoritmo que genere los circuitos

Definición

Una MT determinista M genera una familia de circuitos $\{C_n\}$ si esta máquina con entrada 0^k genera C_k.

- *M genera C_k codificado como una palabra en $\{0,1\}^*$*

Clases de complejidad y circuitos

Es posible usar circuitos para definir clases de complejidad.

Definición (Clase de complejidad AC^i)

Para $i \geq 0$: Un lenguaje $L \subseteq \{0,1\}^*$ está en AC^i si existe una familia de circuitos $\{C_n\}$, una MT determinista M y una constante k tal que:

1. *L es aceptado por $\{C_n\}$*
2. *M genera $\{C_n\}$ y funciona en espacio logarítmico*
3. *profundidad$(C_n) \leq k \cdot (\log n)^i$*

AC^i: Algunos comentarios

No imponemos restricciones al número de arcos que salen de un nodo.

No imponemos restricciones al número de arcos que llegan a un nodo.

- Si usamos compuertas con dos entradas obtenemos las clases NC^i

Los problemas que pueden ser resueltos en alguna de estas clases son llamados paralelizables (o también muy paralelizables).

AC^i: Poder de computación

AC^0: Circuitos tienen profundidad constante.

Ejercicio

Muestre que la multiplicación de matrices esta en AC^0 ($+$ se interpreta como $\vee$ y $\cdot$ como $\wedge$)

AC^1: Circuitos tienen profundidad logarítmica. Esto nos permite resolver muchos problemas.

Ejercicio

1. Muestre que $L = \{w \in \{0,1\}^* \mid w \text{ tiene un número impar de } 1\text{s}\}$ está en AC^1
2. Muestre que el problema de verificar si dos nodos están conectados en un grafo está en AC^1

Otro ejemplo: Suma en AC^0

Vamos a demostrar que la suma puede ser calculada en AC^0, vale decir, con circuitos de profundidad constante.

Queremos construir un circuito $C(x_n, \ldots, x_1, y_n, \ldots, y_1)$:

- Recibe como entrada dos números binarios $\bar{x} = x_n \cdots x_1$ y $\bar{y} = y_n \cdots y_1$
- Tiene $n + 1$ salidas que corresponden a la suma de estos números: $z_{n+1} z_n \cdots z_1$

Otro ejemplo: Suma en AC^0

Notación

$$\begin{aligned} AND_i &= x_i \wedge y_i \\ OR_i &= x_i \vee y_i \\ XOR_i &= (x_i \wedge \neg y_i) \vee (\neg x_i \wedge y_i) \end{aligned}$$

Sea $r_0 = 0$ y r_i ($i \in [1, n]$) el i-ésimo resto en la suma de $\bar{x}$ y $\bar{y}$

Entonces para $i \geq 1$:

$$r_i = \mathrm{AND}_i \vee (\mathrm{OR}_i \wedge r_{i-1})$$

Otro ejemplo: Suma en AC^0

Desarrollando una vez obtenemos:

$$r_i \quad = \quad \text{AND}_i \vee (\text{OR}_i \wedge \text{AND}_{i-1}) \vee (\text{OR}_i \wedge \text{OR}_{i-1} \wedge r_{i-2})$$

Si seguimos desarrollando obtenemos:

$$\begin{aligned}
r_1 &= \text{AND}_1 \\
r_2 &= \text{AND}_2 \vee (\text{OR}_2 \wedge \text{AND}_1) \\
&\cdots \\
r_n &= \text{AND}_n \vee (\text{OR}_n \wedge \text{AND}_{n-1}) \vee (\text{OR}_n \wedge \text{OR}_{n-1} \wedge \text{AND}_{n-2}) \vee \cdots \\
&\qquad \cdots \vee (\text{OR}_n \wedge \text{OR}_{n-1} \wedge \cdots \wedge \text{OR}_2 \wedge \text{AND}_1).
\end{aligned}$$

Otro ejemplo: Suma en AC^0

Podemos usar los restos para definir el resultado de la suma:

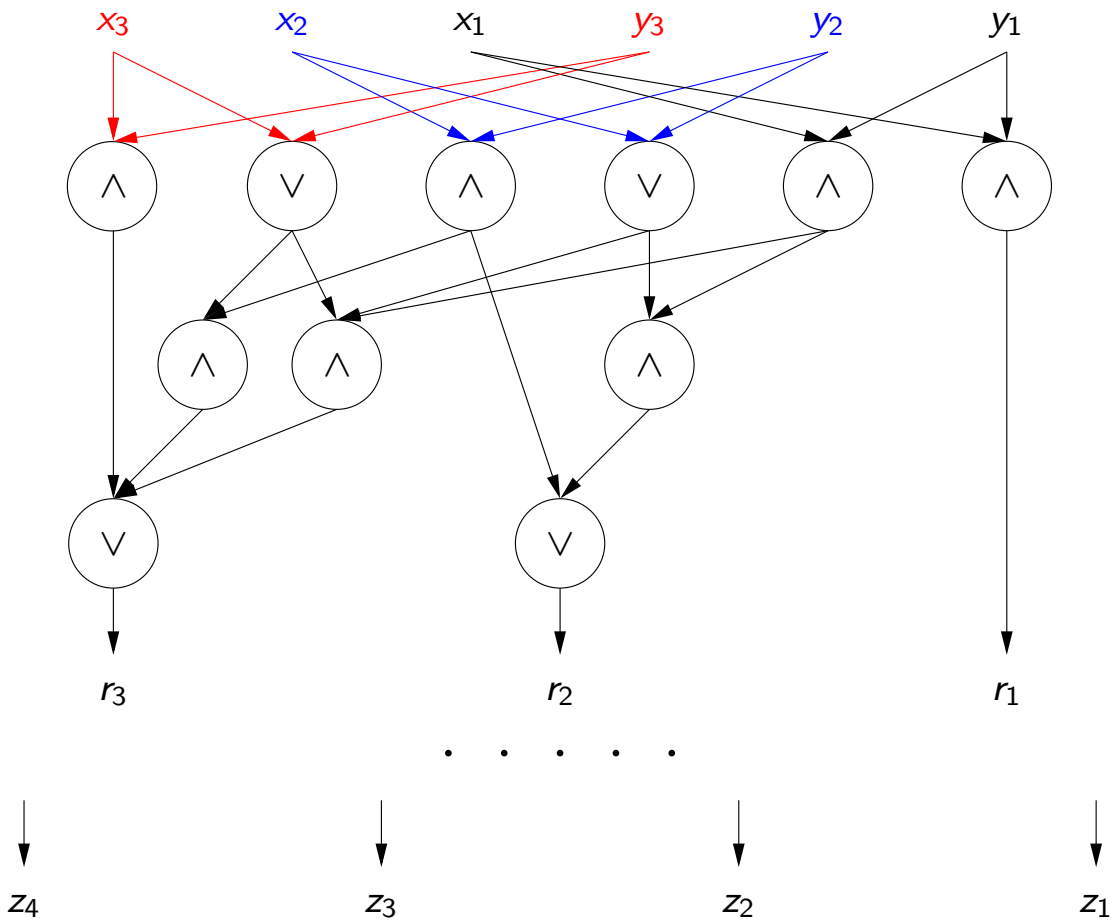
$$z_i \quad = \quad (\neg \mathrm{OR}_i \wedge r_{i-1}) \vee (\mathrm{XOR}_i \wedge \neg r_{i-1}) \vee (\mathrm{AND}_i \wedge r_{i-1})$$

En particular: $z_1 = \mathrm{XOR}_1$ y $z_{n+1} = r_n$.

Podemos utilizar las definiciones de r_i y z_i para construir el circuito.

- ¿Cuál es la profundidad del circuito?
- Veamos un ejemplo: $C(x_3, x_2, x_1, y_3, y_2, y_1)$

Otro ejemplo: Suma en AC^0



Un ejemplo final: Lenguajes regulares

Teorema

Si L es un lenguaje regular, entonces $L \in NC^1$.

Ejercicio

Demuestre el teorema.

- Demuestre primero que el lenguaje generado por la expresión regular $(01)^*$ está en NC^1
- Extienda esta demostración al caso general

Circuitos de tamaño polinomial

Si $L \in \mathrm{AC}^i$: Existe una familia de circuitos de tamaño polinomial que acepta L.

- Aun más: Tenemos que $L \in \mathrm{PTIME}$ por la condición de uniformidad.

¿Qué clase de problemas pueden ser aceptados por familias de circuitos de tamaño polinomial?

- Vamos a estudiar este problema.
- Lo interesante de este estudio: **No vamos a considerar la condición de uniformidad.**

Circuitos de tamaño polinomial: Primer resultado

Vamos a empezar por demostrar que casi todos los lenguajes necesitan de familias de circuitos de tamaño exponencial.

- Como no tenemos la condición de uniformidad, esto incluye hasta los lenguajes indecidibles.

Proposición

Cada lenguaje $L \subseteq \{0,1\}^*$ *es aceptado por una familia de circuitos* $\{C_n\}$ *tal que tamaño*$(C_n) \leq 2^n + 1$.

Ejercicio

Demuestre la proposición.

Circuitos de tamaño polinomial: Primer resultado

Ya demostramos que todo lenguaje puede ser aceptado por una familia de circuitos de tamaño exponencial.

- Ahora vamos a demostrar que casi ningún lenguaje puede ser aceptado por una familiar de circuitos de tamaño polinomial

Notación

Para $n \geq 1$*:*

$$\mathcal{F}_n = \{f \mid f \text{ es una función Booleana con } n \text{ entradas}\}$$

$$\mathcal{P}_n = \{C \mid C \text{ es un circuito con } n \text{ entradas tal que } \text{tamaño}(C) \leq 2^{\frac{n}{4}}\}$$

Circuitos de tamaño polinomial: Primer resultado

De hecho, vamos a demostrar algo más fuerte que lo enunciado en la transparencia anterior:

- Casi ninguna función Booleana con n entradas puede ser aceptada por un circuito de tamaño a lo más $2^{\frac{n}{4}}$

Proposición

$$\lim_{n \to \infty} \frac{|\mathcal{P}_n|}{|\mathcal{F}_n|} = 0$$

Circuitos de tamaño polinomial: Primer resultado

Demostración: Tenemos que existen a los más $(2 \cdot 2^{k+2 \cdot n})^k$ circuitos con n entradas y $k \geq 1$ nodos con etiquetas $\wedge$ ó $\vee$.

- ¿Por qué?

Por lo tanto: Existen a los más $(k+1) \cdot (2 \cdot 2^{k+2 \cdot n})^k$ circuitos con n entradas y $\ell \in \{0, \ldots, k\}$ nodos con etiquetas $\wedge$ ó $\vee$.

Considerando $k = 2^{\frac{n}{4}}$, obtenemos que:

$$|\mathcal{P}_n| \quad \leq \quad (2^{\frac{n}{4}} + 1) \cdot (2 \cdot 2^{2^{\frac{n}{4}} + 2 \cdot n})^{2^{\frac{n}{4}}}$$

Circuitos de tamaño polinomial: Primer resultado

Por lo tanto, para n suficientemente grande se tiene que:

$$\begin{aligned} |\mathcal{P}_n| &\leq 2^{1+\frac{n}{4}+(1+2\cdot n)\cdot 2^{\frac{n}{4}}+2^{\frac{n}{2}}} \\ &\leq 2^{2^{\frac{n}{2}}+2^{\frac{n}{2}}+2^{\frac{n}{4}}\cdot 2^{\frac{n}{4}}+2^{\frac{n}{2}}} \\ &= 2^{4\cdot 2^{\frac{n}{2}}} \end{aligned}$$

Así, dado que $|\mathcal{F}_n| = 2^{2^n}$, concluimos que:

$$0 \leq \lim_{n\to\infty} \frac{|\mathcal{P}_n|}{|\mathcal{F}_n|} \leq \lim_{n\to\infty} \frac{2^{4\cdot 2^{\frac{n}{2}}}}{2^{2^n}} = \lim_{n\to\infty} \frac{1}{2^{2^{\frac{n}{2}}\cdot(2^{\frac{n}{2}}-4)}} = 0$$

□

Circuitos de tamaño polinomial y PTIME

Sabemos que:

$$\bigcup_{i \in \mathbb{N}} \mathrm{AC}^i \subseteq \mathrm{PTIME}$$

¿Es esta inclusión propia?

- Esta pregunta está abierta, pero se cree que la respuesta es sí
- En todo caso, sí se conocen relaciones entre PTIME y las familias de circuitos de tamaño polinomial

Circuitos de tamaño polinomial y PTIME

Teorema

Cada problema en PTIME es aceptado por una familia de circuitos de tamaño polinomial.

Demostración: Es importante notar que $L \subseteq \{0,1\}^*$.

Circuitos de tamaño polinomial y PTIME: Demostración

Como $L \in \text{PTIME}$, existe una MT determinista $M = (Q = \{q_0, \ldots, q_m\}, \Sigma = \{0, 1\}, \Gamma = \{0, 1, \text{B}, \vdash\}, q_0, \delta, F)$ tal que:

- M tiene una cinta de lectura/trabajo
- M para en todas las entradas
- $L = L(M)$
- $t_M(n)$ es $O(n^c)$, donde $c > 0$ es un número natural

Circuitos de tamaño polinomial y PTIME: Demostración

Sin perdida de generalidad suponemos que:

- $F = \{q_m\}$
- Para cada $a \in \Gamma$, se tiene que $\delta(q_m, a)$ no está definido

¿Por qué podemos suponer esto?

Circuitos de tamaño polinomial y PTIME: Demostración

Sea $n \geq 1$ y $L_n = \{w \in L \mid |w| = n\}$.

Tenemos que construir un circuito $C_n(x_1, \ldots, x_n)$ tal que para toda palabra w de largo n:

$$w \in L_n \quad \text{si y sólo si} \quad C_n(w) = 1$$

- ¿Qué pasa con el caso $n = 0$?

C_n va a codificar una fórmula proposicional que representa el funcionamiento de M con una entrada de largo n.

Circuitos de tamaño polinomial y PTIME: Demostración

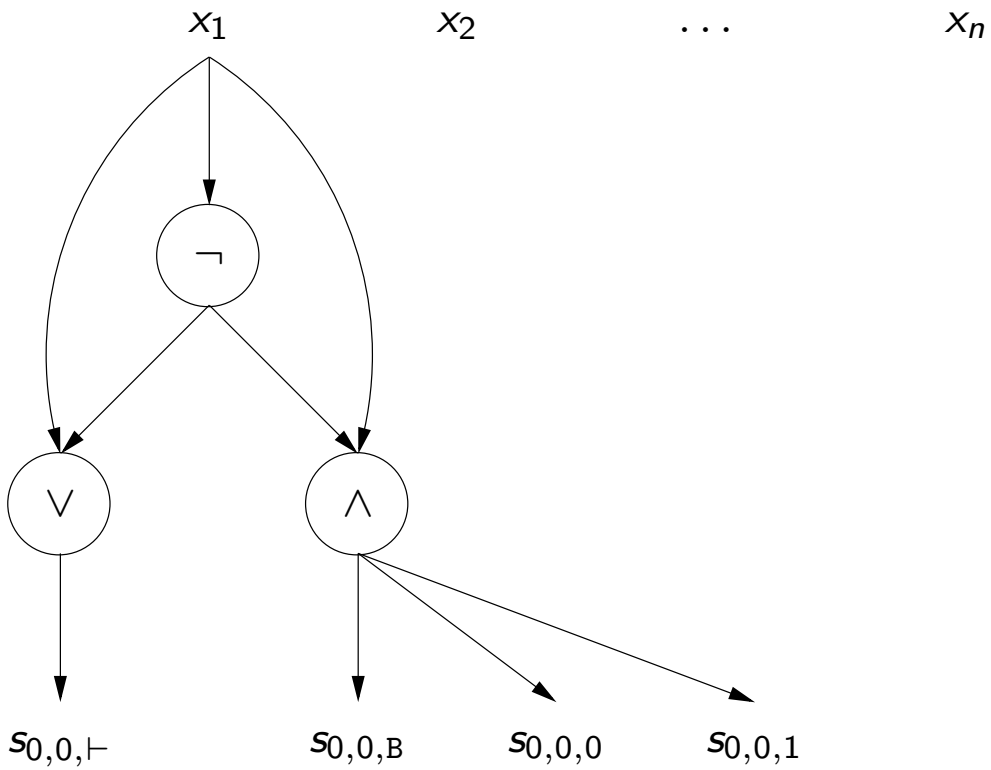
En la construcción de C_n vamos a hacer referencia a las siguientes variables proposicionales:

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

Estas variables han sido usadas varias veces en el curso para codificar el funcionamiento de una MT.

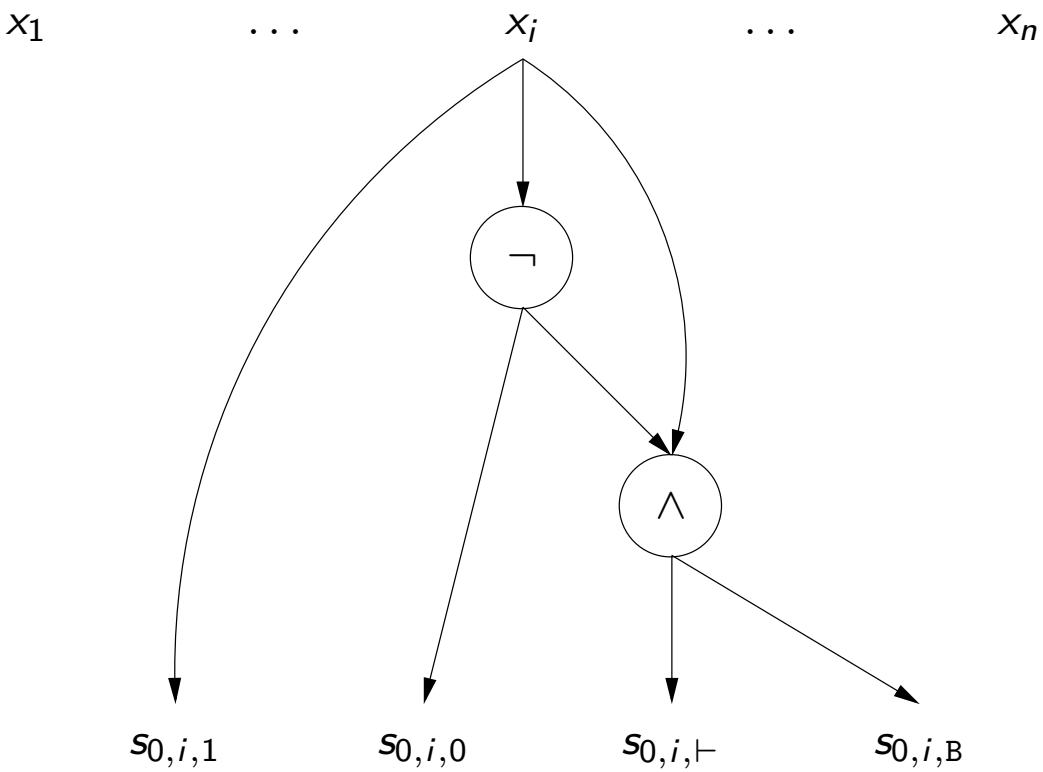
Circuitos de tamaño polinomial y PTIME: Demostración

Codificación de la configuración inicial:



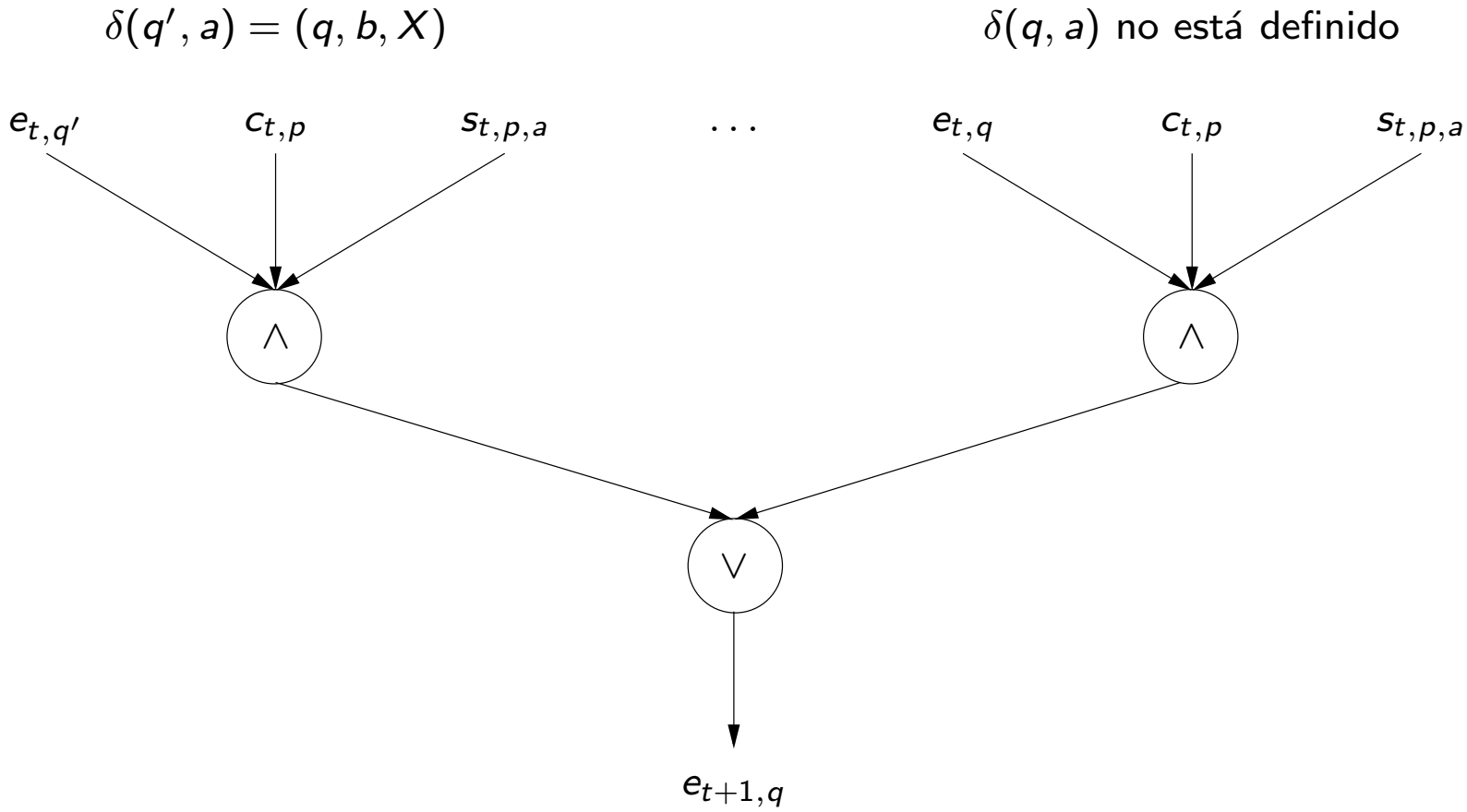
Circuitos de tamaño polinomial y PTIME: Demostración

Codificación de la configuración inicial:



Circuitos de tamaño polinomial y PTIME: Demostración

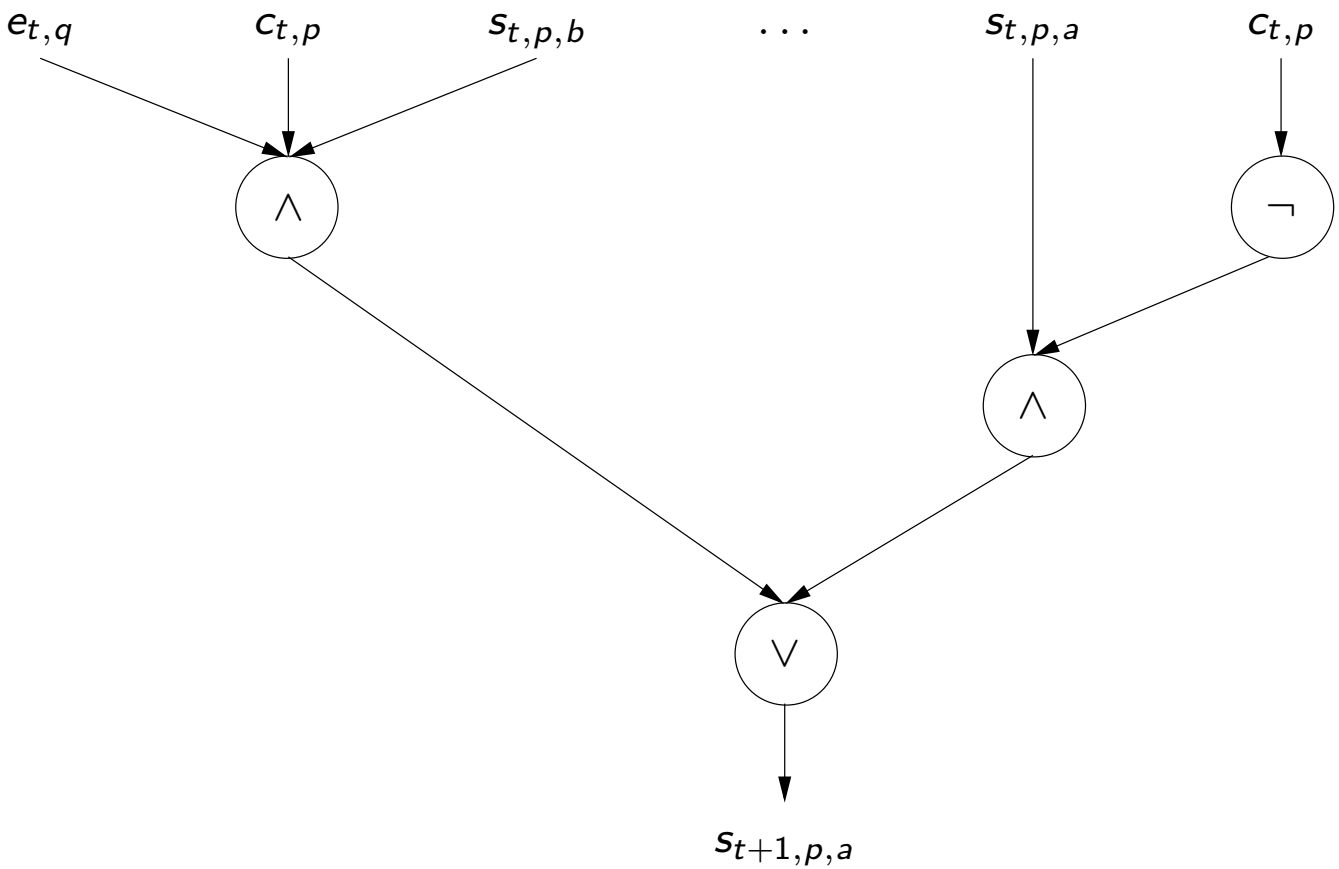
Función de transición:



Circuitos de tamaño polinomial y PTIME: Demostración

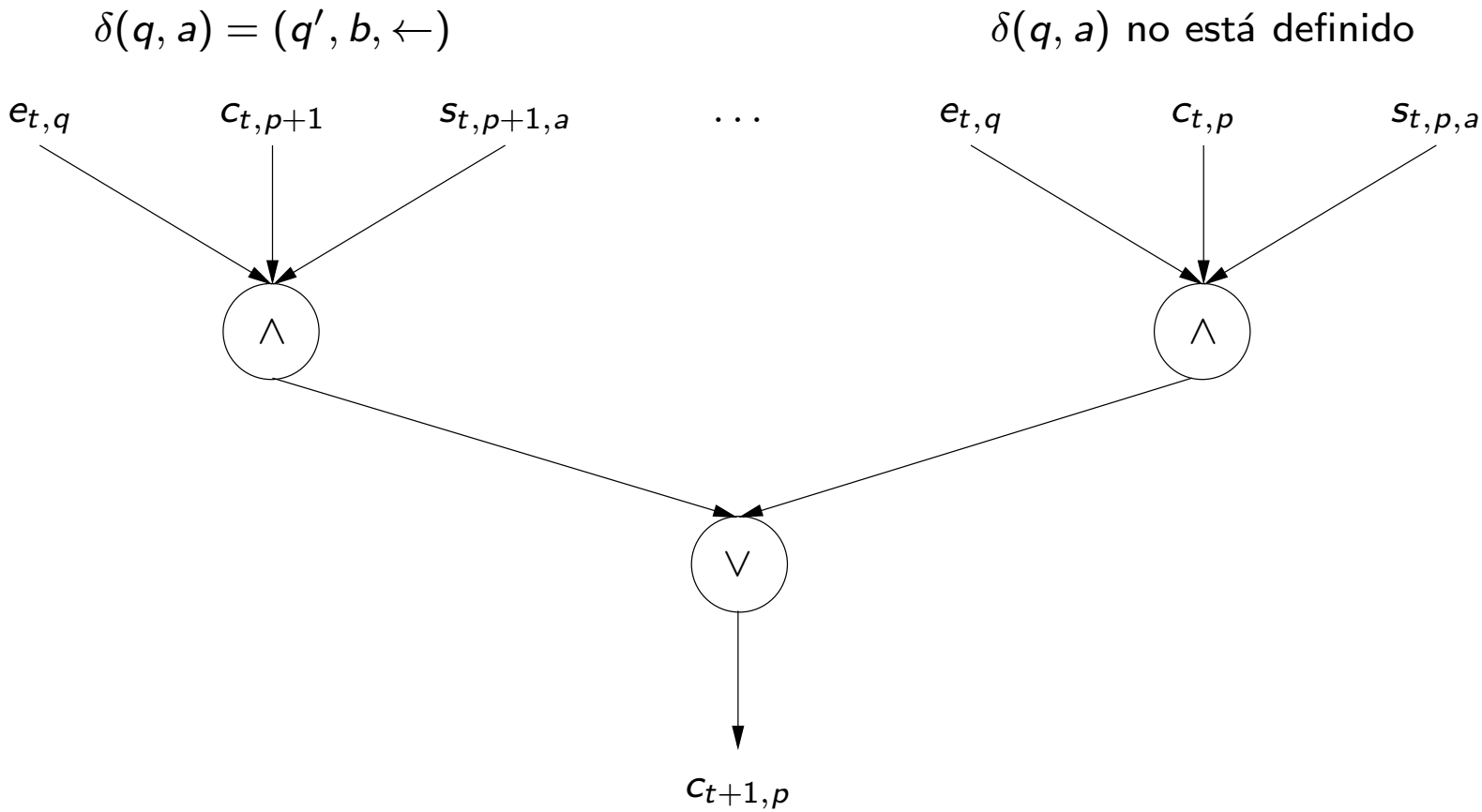
Función de transición:

$\delta(q, b) = (q', a, X)$ cabeza lectora no está en la posición p



Circuitos de tamaño polinomial y PTIME: Demostración

Función de transición:



Circuitos de tamaño polinomial y PTIME: Demostración

Salida del circuito C_n:

$$e_{t_M(n),q_m}$$

Para terminar hay que probar que $C_n(w) = 1$ si y sólo si $w \in L_n$.

- ¿Cómo se hace esto?

Circuitos de tamaño polinomial y PTIME: Comentarios

¿Cuál es la profundidad de los circuitos construidos en la demostración?

- ¿Puede cada una de estas familias ser considerada un algoritmo?
- ¿Pueden ser consideradas como buenos algoritmos paralelos?

¿Son todos los problemas de la clase PTIME paralelizables?

La clase PTIME/poly

Sea PTIME/poly la clase de los lenguajes aceptados por familias de circuitos de tamaño polinomial.

En las transparencias anteriores demostramos que:

Teorema

PTIME $\subseteq$ *PTIME/poly*

¿Qué sucede con la otra dirección?

- ¿Es cierto que PTIME/poly $\subseteq$ PTIME?
- Vamos a demostrar que esta inclusión no es cierta

La clase PTIME/poly y los lenguajes indecidibles

Proposición

Existe un lenguaje indecidible que pertenece a PTIME/poly.

De la proposición, obtenemos como corolario que:

Corolario

$PTIME \subsetneq PTIME/poly$

La clase PTIME/poly y los lenguajes indecidibles: Demostración

Sea $\{M_n\}_{n\geq 0}$ una enumeración de las MTs deterministas y con una cinta de lectura/trabajo.

Usamos esta enumeración para definir el siguiente lenguaje:

$$L = \{w \in \{0,1\}^* \mid \text{existe } i \geq 0 \text{ tal que } w = 0^i \text{ y } M_i \text{ se detiene con entrada } \varepsilon\}$$

Se tiene que L es indecidible y $L \in$ PTIME/poly.

- ¿Por qué?

La clase PTIME/poly y los lenguajes poco densos

Es posible demostrar una propiedad más fuerte que la considerada anteriormente.

- El lenguaje indecidible construido en la transparencia anterior estaba en PTIME/poly porque contenía *muy pocas* palabras
- Es posible generalizar esta idea

Definición

Un lenguaje L es poco denso si existe un polinomio $p(n)$ tal que para todo $n \geq 0$:

$$|\{w \in L \mid |w| = n\}| \quad \leq \quad p(n)$$

La clase PTIME/poly y los lenguajes poco densos

Teorema

Cada lenguaje poco denso pertenece a PTIME/poly.

Ejercicio

Demuestre el teorema.

PTIME/poly, NP y la jerarquía polinomial

Vimos que las familias de circuitos de tamaño polinomial pueden aceptar todos los lenguajes poco densos.

- En particular, pueden aceptar algunos lenguajes indecidibles.

Una pregunta natural es que consecuencias tiene el hecho de que estas familias acepten lenguajes complejos.

- Por ejemplo, ¿qué consecuencias tendría el hecho de que NP este contenido en PTIME/poly?
- Vamos a dar una respuesta precisa a esta pregunta.

PTIME/poly, NP y la jerarquía polinomial

Teorema (Karp-Lipton)

Si $NP \subseteq PTIME/poly$, entonces $PH = \Sigma_2^P$.

Demostración: Sea $\text{SAT}_{\bot,\top}$ una versión extendida de SAT donde las fórmulas proposicionales pueden incluir los símbolos $\bot$ y $\top$.

- $\bot$ representa el valor 0, y $\top$ representa el valor 1

Es fácil demostrar que $\text{SAT}_{\bot,\top}$ es NP-completo.

- De hecho $\text{SAT}_{\bot,\top}$ es NP-hard ya que la identidad es una reducción de SAT en $\text{SAT}_{\bot,\top}$

Teorema de Karp-Lipton: Demostración

Por hipótesis, sabemos que existe una familia de circuitos $\{C_n\}_{n \geq 0}$ de tamaño polinomial que acepta $\text{SAT}_{\bot,\top}$.

- Existe un polinomio p tal que tamaño$(C_n) \leq p(n)$ para todo $n \in \mathbb{N}$

Notación

Usamos φ para denotar tanto una fórmula proposicional como la palabra en $\{0,1\}^*$ que la representa.

- *$|\varphi|$ es el largo de la palabra que representa a la fórmula proposicional φ*

Entonces tenemos que:

- φ es satisfacible si y sólo si $C_{|\varphi|}(\varphi) = 1$

Teorema de Karp-Lipton: Demostración

El siguiente lema es fundamental en la demostración.

Lema

Existe una función H computable en tiempo polinomial tal que para toda fórmula proposicional φ:

$\varphi \in SAT_{\bot,\top}$ *si y sólo si*

$H(\varphi, C_{|\varphi|})$ *es una valuación que satisface a* φ.

¿Cómo se demuestra el lema?

- Vamos a ver un algoritmo polinomial que calcula una función H con la propiedad mencionada en el lema

Teorema de Karp-Lipton: Demostración

En la demostración del lema usamos esta notación:

- $\alpha[\frac{p}{\top}]$: Fórmula que resulta de reemplazar p por $\top$ en α (se define de la misma forma para $\bot$)

function $H(\varphi$: fórmula proposicional, C: circuito)

$\sigma := \emptyset$

while φ contiene al menos una letra proposicional **do**

seleccione una letra p desde φ

if $C_{|\varphi|}(\varphi[\frac{p}{\top}]) = 1$

$\varphi := \varphi[\frac{p}{\top}]$

$\sigma := \sigma \cup \{p \mapsto 1\}$

else

$\varphi := \varphi[\frac{p}{\bot}]$

$\sigma := \sigma \cup \{p \mapsto 0\}$

return σ

Teorema de Karp-Lipton: Demostración

Vamos a demostrar que $\Pi_2^P \subseteq \Sigma_2^P$.

- De esto se deduce que $\Pi_2^P = \Sigma_2^P$ y, entonces, $\text{PH} = \Sigma_2^P$. ¿Por qué?

Sea $L \in \Pi_2^P$.

- Vamos a demostrar que $L \in \Sigma_2^P$

Dado que $L \in \Pi_2^P$, existe un lenguaje $L_1 \in \text{NP}$ y un polinomio q tal que para todo $x \in \{0,1\}^*$:

$x \in L$ si y sólo si $\forall y \in \{0,1\}^*$ tal que $|y| = q(|x|)$, se tiene que $x\#y \in L_1$

Teorema de Karp-Lipton: Demostración

Como $\text{SAT}_{\bot,\top}$ es NP-completo, existe una reducción g de L_1 en $\text{SAT}_{\bot,\top}$.

- Suponemos que g es computable en tiempo $r(n)$, donde r es un polinomio
- También suponemos que si $|w_1| = |w_2|$, entonces $|g(w_1)| = |g(w_2)|$. ¿Por qué podemos suponer esto?

Concluimos que para todo $x \in \{0,1\}^*$:

$x \in L$ si y sólo si $\forall y \in \{0,1\}^*$ tal que $|y| = q(|x|)$,
se tiene que $g(x\#y) \in \text{SAT}_{\bot,\top}$

Teorema de Karp-Lipton: Demostración

Combinando lo anterior con el lema, concluimos que para todo $x \in \{0,1\}^*$:

$x \in L$ si y sólo si $\forall y \in \{0,1\}^*$ tal que $|y| = q(|x|)$,

se tiene que $H(g(x\#y), C_{|g(x\#y)|})$

es una valuación que satisface a $g(x\#y)$

Teorema de Karp-Lipton: Demostración

Es importante notar que:

$$\begin{aligned} \text{tamaño}(C_{|g(x\#y)|}) &\leq p(|g(x\#y)|) \\ &\leq p(r(|x\#y|)) \\ &= p(r(|x| + |y| + 1)) \\ &= p(r(|x| + q(|x|) + 1)) \end{aligned}$$

Importante: Suponemos que $p(n_1) \leq p(n_2)$ si $n_1 \leq n_2$

- ¿Por qué podemos suponer esto?

Teorema de Karp-Lipton: Demostración

Combinando lo anterior con el lema, concluimos que para todo $x \in \{0,1\}^*$:

$x \in L$ si y sólo si

$\exists$ circuito C tal que C tiene $|g(x\#0^{q(|x|)})|$ entradas,

tamaño$(C) \leq p(r(|x| + q(|x|) + 1))$ y

$\forall y \in \{0,1\}^*$ tal que $|y| = q(|x|)$

se tiene que $H(g(x\#y), C)$

es una valuación que satisface a $g(x\#y)$

Se puede verificar si $H(g(x\#y), C)$ es una valuación que satisface a $g(x\#y)$ en tiempo polinomial.

- Concluimos que $L \in \Sigma_2^P$. ¿Por qué? □

Una consecuencia del Teorema de Karp-Lipton

Corolario

Si existe un lenguaje poco denso que es NP-completo, entonces $PH = \Sigma_2^P$.

Demostración: Suponga que $L \subseteq \{0,1\}^*$ es poco denso y NP-completo.

Vamos a probar que de esto se concluye que NP $\subseteq$ PTIME/poly.

- Por el Teorema de Karp-Lipton: PH $= \Sigma_2^P$

Una consecuencia del Teorema de Karp-Lipton

Sea $L^\star$ un lenguaje definido como:

$$L^\star = \{0^n 1 w \mid w \in L \text{ and } n \geq 0\}$$

Como L es NP-completo, se tiene que $L^\star$ es NP-completo.

- ¿Por qué?

Como L es poco denso, se tiene que $L^\star$ es poco denso.

- Sabemos que $|\{w \in L \mid |w| = n\}| \leq p(n)$
- Por lo tanto:

$$|\{w \in L^\star \mid |w| = n\}| \leq \sum_{i=0}^{n-1} p(n - (i+1))$$

Una consecuencia del Teorema de Karp-Lipton

Como $L^\star$ es poco denso: Existe una familia de circuitos $\{C_n\}_{n\geq 0}$ de tamaño polinomial que acepta $L^\star$.

- Vamos a usar esto para probar que NP $\subseteq$ PTIME/poly

Sea $L' \subseteq \{0,1\}^*$ un lenguaje en NP.

Como L es NP-completo, existe $f : \{0,1\}^* \to \{0,1\}^*$ tal que:

- Para cada $w \in \{0,1\}^*$: $w \in L'$ si y sólo si $f(w) \in L$
- f puede ser calculada en tiempo $q(n)$, donde q es un polinomio

Una consecuencia del Teorema de Karp-Lipton

Sea $g : \{0,1\}^* \to \{0,1\}^*$ definida como:

$$g(w) \quad = \quad 0^{q(|w|)-|f(w)|}1f(w)$$

Se tiene que:

- Para cada $w \in \{0,1\}^*$: $w \in L'$ si y sólo si $g(w) \in L^\star$
- g puede ser calculada en tiempo polinomial
- Para cada $w \in \{0,1\}^*$: $|g(w)| = q(|w|) + 1$
  - En particular: Si $|w_1| = |w_2|$, entonces $|g(w_1)| = |g(w_2)|$

Una consecuencia del Teorema de Karp-Lipton

Entonces: Existe una familia de circuitos $\{D_n\}_{n\geq 0}$ de tamaño polinomial que acepta L'.

- Existe una familia de circuitos $\{E_n\}_{n\geq 0}$ de tamaño polinomial que calcula g.
  - ¿Por qué?
- Cada circuito D_n se construye componiendo E_n con $C_{q(n)+1}$.
  - ¿Cómo se hace esto?

Concluimos que $L' \in \text{PTIME/poly}$. □

Comentarios finales

En este capítulo nos concentramos en estudiar las clases AC^i ($i \geq 0$) y PTIME/poly.

- Vimos que $\bigcup_{i \geq 0} \mathrm{AC}^i \subseteq \mathrm{PTIME} \subsetneq \mathrm{PTIME/poly}$

En particular, estudiamos varias propiedades fundamentales de PTIME/poly.

Y vimos que la pregunta ¿$\bigcup_{i \geq 0} \mathrm{AC}^i \subsetneq \mathrm{PTIME}$? está abierta.

- Pero ¿qué se sabe sobre la estructura de las clases AC^i?
- Esto es lo último que vamos a ver en este capítulo

La estructura de las clases AC^i

Teorema

$AC^0 \subseteq LOGSPACE \subseteq NLOGSPACE \subseteq AC^1$

Ejercicio

Demuestre el teorema.

Teorema (Furst-Saxe-Sipser)

$AC^0 \subsetneq LOGSPACE$

Idea de la demostración: $L = \{w \in \{0,1\}^* \mid w$ tiene un número impar de 1s$\}$ está en LOGSPACE $\setminus$ AC^0.