

Práctica 8

Si hay cambios raros a lo largo del TP es pq' es la 1^{ra} vez q' uso una tableta gráfica !!

Ej 1:

Era esto a lo q' hacia ref.

1. Un lenguaje \mathcal{L} es esparso si existe un polinomio p tal que $|\mathcal{L} \cap \{0, 1\}^n| \leq p(n)$ para todo $n \in \mathbb{N}$. Probar que todo lenguaje esparso está en P/poly .

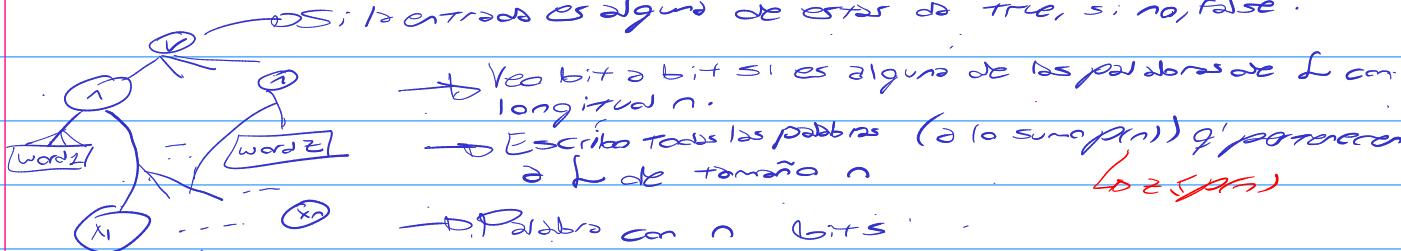
Idea: Para probar esto podría hacer una familia de circuitos los cuales Tengan cierta estructura la cual me permite demostrar que para una entrada x con $|x| = n$, $C_n = 1$ si $x \in L$ con L espero.

Voy a rescribir la def. de esparsa para entenderlo mejor:

Existe p un polinomio tq: $|L_n\{0,1^n\}| \leq p(n)$ para todo $n \in \mathbb{N}$.

Lo O sea, hay una cantidad polinomial de palabras de longitud n

Declaro entonces una filo de circuitos de la siguiente manera:



Estos circuitos tienen un tamaño polinomial, y a g' depende de la cantidad de polígonos de longitud n que hayan en \mathcal{L} (el cual sabemos g' es $p(n)$).

Desp. verif.
el exocarpio
(creo q' serán
n planas).

Son **polys** conjunciones (ver todos los polys de longitud n>1)

Luego poly disjunciones (veo q' la entrada coincide con algunas palabras de L con tamaño n)

Creería q' ésto es suficiente prueba de q' existe una familia de circuitos $(C_n)_{n \in \mathbb{N}}$ donde se compute todo L

~~Es más visual con Advice (resp. refiner)~~ (con L expresa)

Ej 2:

2. Probar que existen lenguajes fuera de P/poly.

Gracias @ezeg-uel-h por la corrección!

Ojo! En este ejercicio no se puede usar el argumento de cardinalidad. Los lenguajes en P/poly son infinitos no numerables (esto se puede ver esto de que las máq. q' deciden los lenguajes en P/Advice usan una función Advice arbitraria, por ende la cant. de lenguajes reconocibles en P/poly no es numerable)

Idea: Encararlo mediante un argumento de conteo

• Hay exactamente:

$$\# \text{Funciones}(n) = 2^{2^n}$$

✓ Cantidad de funciones del tipo $\{0,1\}^n \rightarrow \{0,1\}$.
Cada entrada de n bits tiene 0 ó 1 de salida y puedo elegir q' de 0 ó 1 en cada función.

• El número de circuitos con n entradas y tamaño $\langle S(n) \rangle$ se codifica con $O(S(n) \log(S(n)))$ bits por lo que:

$$\# \text{circuitos}(n) \leq 2^{\text{coste de } O(S(n) \log(S(n)))}$$

→ $O(S(n) \log(S(n)))$ sale de tener $S(n)$ puertas lógicas en el circuito y cada una se codifica con $\log S(n)$ bits.

• Para que cada función tuviera algún circuito de tamaño $\langle S(n) \rangle$ necesitaríamos:

$$\underbrace{2^{2^n}}_{\textcircled{1}} \leq \underbrace{2^{\text{coste de } O(S(n) \log(S(n)))}}_{\textcircled{2}} \times^*$$

① Cifra mínima de circuitos que harían falta (uno por función)

② Máxima cantidad de circuitos q' existen con ese tamaño.

Ahora sabemos que si * falla significa que existen funciones q' que no tienen representación y, por lo tanto, el tamaño elegido no basta.

Para todo polinomio $S(n) = n^k$ nos queda que:

$$2^{\Theta(n^k \cdot \log(n))} \ll 2^{2^n} \rightarrow \text{Por lo cual no cumple } ^*.$$

Como * no se cumple \Rightarrow existen lenguajes fuera de P/poly \Rightarrow $\text{ALL} \neq \text{P/poly}$.

↓
Todas las lenguajes posibles.

Quedó espacio en blanco para la anterior corrección

Ej 3:

3. Definimos la clase P_{advice} como la clase de lenguajes que se pueden resolver en tiempo polinomial asumiendo que se cuenta con un consejo a para cada tamaño n de tamaño polinomial en n . Es decir, $\Pi \in P_{advice}$ si y solamente si existe una función $adv : \mathbb{N} \rightarrow \{0, 1\}^*$ y una máquina polinomial M tal que

$$x \in \Pi \iff M(x, adv(|x|)) = 1$$

Ver g' el adv. es el mismo
para todos cadenas de longitud.

donde aparte existe un polinomio p con $|adv(n)| \leq p(n)$ (es decir, el consejo es chico).

Probar que $P_{advice} = P/\text{poly}$.

Idea: Para \subseteq se pide ver medio fácil g' P_{advice}
tiene una máq. g' genera circuitos básicamente y habría
 g' ver g' estos son de altura poly .
Para \supseteq uso g' $adv(n) := C_n$ y luego g' la máq. simula
 C_n dado el input x (donde $|x|=n$), o sea $adv(n)$

Σ) $\text{P}_{\text{advice}} \subseteq \text{P/poly}$

$M :=$ Mág. det. polinomial

$\text{adv}(\cdot) :=$ Función de $\mathbb{N} \rightarrow \{0,1\}^*$ y $|\text{adv}(n)| \leq p(n)$. $\forall n \in \mathbb{N}$.

$x \in \Pi$ si $M(x, \text{adv}(|x|)) = 1$

Como $|\text{adv}(|x|)| \leq p(|x|)$ puedo insertar este advice como constante en el circuito C_n (con $|x|=n$). Esta práctica es válida pq' agrandaría a la suma polinomialmente el circuito, pues su tamaño es $\leq p(n)$. Además como la familia $(C_n)_{n \in \mathbb{N}}$ no tiene q' ser generable $M(n) \leq C_n$ & uniformemente puedo tratar info dependiente de n .

Luego, el funcionamiento de este C_n sería el mismo q' el de la M polinomial mencionada antes.

Ahora, ¿cómo sé que la simulación de esta M poly resulta en un circuito poly?

Dijera, esto demo es más difícil de lo anticipado, así q' lo voy a justificar 'con esto': Como sé q' $P \subseteq \text{P/poly}$ (Ver Teo 11) puedo decir q' cada mág. q' corre en tiempo poly está en P/poly , ya q' puedo interpretar q' cada mág. det. poly. con salida $\in \{1, 0\}$ obtenga un LEP (esta sale por def. de P básicamente).

Y

Σ) $\text{P/poly} \subseteq \text{P}_{\text{advice}}$

$(C_n)_{n \in \mathbb{N}}$:= Familia de circuitos de tamaño polinomial.

$x \in \Pi$ si $C_{|x|}(x) = 1$

Puedo crear a $\text{adv}(n)$ como la codificación de C_n , de manera q' la mág. M simule $C_{|x|}(x)$ polinomialmente.

¿Por qué y cómo lo hace poly?

M corre polynomialmente porque es evaluar un circuito.
O sea, se ve de la siguiente manera a máquina

$M \langle x, \langle c_{|x|} \rangle \rangle$:

Evaluó C_n con entrada x → Esto es porque se simula las computadoras lógicas codificadas en C_n . Esto sería \leq_{p} $\langle n, p(n) \rangle$ computadas a evaluar.

Como el advice cambia dependiendo del tamaño de la entrada
no hay problema en cambiar la configuración de circuito q' que
esté usando dependiendo de $|x|$.

✓

Ej 4:

4. Definimos $P/f(n)$ como la clase de problemas que se resuelven con un consejo de tamaño $f(n)$ (y entonces $P/\text{poly} = \bigcup_{k \in \mathbb{N}} P/n^k$). Probar que $P \neq P/L \cap R$.

Primero lo primero:

• ¿Qué es R ? R hace referencia a los problemas recursivos, o sea
todas las problemas computables.

Idea: A ver, que $P \neq (P/L \cap R)$, o sea, tendría sentido que
 $P \subseteq P/L \cap R$, ya que tendría sentido que agregarle 1 bit de
advice no le saque poder de decisión (sería muy raro si no). Entonces,
el problema debe ser que $(P/L \cap R) \not\subseteq P$.

Entonces, por qué $(P/L \cap R) \not\subseteq P$? Debe haber un $\pi \in P$ tal que
 $\pi \in P/L \cap R$ y $\pi \notin P$, esto lo debemos probar ver por diagonalización.

Q: $(P/L \cap R) \not\subseteq P$ por diagonalización:

Enumero M_1, M_2, \dots a todos los máq. det. polinomiales

Construya $\pi \in P/L \cap R$ tal que:

- Para cada $n \in \mathbb{N}$ ($n = |x|$) difiere del lenguaje aceptado por M_n en
algun x de tamaño n usando el bit de consejo.

Visual:

	M_1	M_2	M_3	M_4	\dots
1	S_{M_1}	S_{M_2}	\dots		
2	S_{M_1}	S_{M_2}			
3					
4					
\vdots					

$S_{M_n} = \{c_i\}_{i=1}^n$ de salidas de todas las posibles entradas de M_n de tamaño n

○ \rightarrow \neg algún elemento de $S_{M_n}^i$ y lo pongo de advice (Medio q' esas estoy asumiendo)

Esto sucede si: por lo q' todas las máq. det. poly son de problemas visto en lo teórico/práctico de de decisión)

q' cualquier problema de decisión puede ser de salida normal y viceversa en tiempo poly

Ej. 5:

5. Probar que $NP = P$ si y solamente si $NP \subseteq P/\log(n)$.



Idea: Bueno acá va a haber q' probar la doble implicación (se viene largo)

$\Rightarrow NP = P \Rightarrow NP \subseteq P/\log(n)$

Si $NP = P$, entonces puedo probar q' $P \subseteq P/\log(n)$ y me basta, esto sale sencillamente porque si $L \in P$ y M decide L , puedo tener una M' q' le entre un advice q' no haga nada y luego q' la máquina funcione igual a como lo hacía M entonces M' decide L , y cumple con la def. de lengüaje en $P/\log(n)$:

$x \in L \iff M'(x, \underbrace{\text{adv}(|x|)})$

$|\text{adv}(n)| \leq \log(n)$

$\Leftarrow NP \subseteq P/\log(n) \Rightarrow NP = P$

*²Hecho más adelante !!

Como no lo uso puede ser cualquier cosa \tilde{O} de tamaño menor igual a $\log(n)$

Ej 6:

6. Probar que si $\text{NP} \not\subseteq \text{P/poly}$ entonces $\text{NP} \neq \text{P}$.

Sé q' $\text{P} \subseteq \text{P/poly}$ por lo cual si $\text{NP} \not\subseteq \text{P/poly}$, lógicamente $\text{NP} \neq \text{P}$.

Así q' es eso, me da miedo q' la rta. lleve 2 renglones.

✓

Ej 7: (tengo miedo, si fu lo explicó aparte a este 0.0)
↳ edita y con razón...

7. Probar que si $\text{EXP} \subseteq \text{P/poly}$ entonces $\Sigma_2^p = \text{EXP}$.

Ayuda: Probar que si $\Pi \in \text{EXP}$ y M es una máquina exponencial con Q estados que lo resuelve en $c2^{n^k}$ pasos entonces el lenguaje $\Pi_M = \{\langle x, i, t, p, q \rangle : i, t, p \leq c2^{|x|^k}, q \leq Q\}$, y en el timestep t el i -esimo bit de la memoria de M es 1, el puntero está en la posición p y la máquina está en el estado $q\}$ está en EXP . Usar el \exists para adivinar el circuito que resuelve Π_M , y luego el \forall para verificar que es el correcto.

$$\Sigma_2^p = \exists u, \forall u_1, M(x, u_1, u_2)$$

Ver después (el Teo 29 (Rarp-Lipton) crea q' ayuda a la idea)

Ej 8:

8. Probar que si $\text{PSPACE} \subseteq \text{P/poly}$ entonces $\text{PSPACE} = \Sigma_2^p \cap \Pi_2^p$.

Es similar a la prueba anterior ^^.^

Ej 9:

9. Probar que los lenguajes

- AND = $\{x_1 \dots x_n : \forall 1 \leq i \leq n, x_i = 1\}$
- OR = $\{x_1 \dots x_n : \exists 1 \leq i \leq n, x_i = 1\}$

están en NC^1 . Usar esto para probar que $\text{AC}^d \subseteq \text{NC}^{d+1}$ para todo $d \geq 0$.

Recordar:

$$NC^d \subseteq AC^d \subseteq NC^{d+1}$$

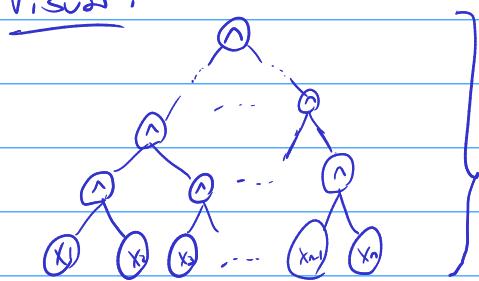
NC^1 : Lenguajes decidibles por una familia de circuitos L -uniformes
 $(C_n)_{n \in \mathbb{N}}$ tq' $|C_n|$ es poly en n y $\text{prof}(C_n) = O(\log n)$

Idea: Para probar esto debería hacer las familias de circuitos q' representan estos lenguajes.

$$\cdot AND = \{x_1, \dots, x_n : \forall L \leq i \leq n, x_i = 1\}$$

La familia de circuitos L -uniformes puede ser lo q' haga lo siguiente

Visual:



$O(\log n)$ (Es ir dividiendo el problema en 2 hasta q' que sea 1 (para casos impares se puede intuir una sol. sencilla))

Lógica

Entonces lo que haría (asumiendo $q' = 2^e$ pero funcionaría con menores modificaciones para el resto de casos) sería q' cada 2 nodos del nivel $i-1$ q' no tienen flechas de salida los conecta con un \oplus y continúo hasta q' no queden nodos de nivel $i-1$ sin flechas de salida

\oplus nivel $\log(n)$

Comentario al margen:



Creo q' = hago $\oplus\oplus\oplus\oplus\oplus\oplus\oplus\oplus\oplus$ de tantas

$$\cdot OR = \{x_1, \dots, x_n : \exists 1 \leq i \leq n, x_i = 1\}$$

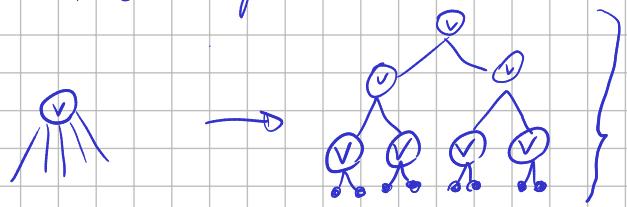
Es muy similar a la anterior solución pero con \vee

Si, acabo de descubrir q' hay distintos tipos de hojas !!

Ahora qrg $AC^d \subseteq NC^{d+1}$ para todo $d \geq 0$

Idea: Esto debería salir viendo q' q' cada operación q' se haga en AC^d puede reproducirse de forma q' q' quede en NC^{d+1}

Para hacer lo planteado en la Idea comienza definiendo el \bigcirc :



Esto lleva a q' por cada \bigcirc arbitrario pero se tiene un árbol de \bigcirc no arbitrario de altura a lo sumo $\log(n)$

Entonces esta sería la forma para representarlo y toma $\log(n)$ por cada \bigcirc arbitrario, por lo cual queda a lo sumo $O(\log^d(n))$ de profundidad

Para definir el \bigcirc es la misma idea q' para el \bigcirc .

Luego para el Caso de \bigcirc no es necesario, pues no tendría sentido un \bigcirc arbitrario.

Ej 10:

10. Probar que $NC^1 \subseteq L$.

(Es el Teo 33)

Idea: Tomo un $L \in NC^1$ y pruebo q' se encuentra en L.

$\pi \in NC^1$ sii π es decidible por una familia de circuitos q' son L-uniformes, $|C_n| \leq \text{poli en } n$ y $\text{prof}(C_n) = O(\log(n))$.

$x \in \pi$ sii $C_{l(x)}(x) = 1$

QVQ $\pi \in L$, o sea, q' π sea decidible por una máq. det. M q' corra en espacio logarítmico.

$M(x)$
 Se puede hacer q' la familia de circuitos es L-uniforme.

1- Con M' genera el i-ésimo bit del circuito $C_{l(x)}$ ingresando $\langle x, i \rangle$

2- Voy evaluando recursivamente el circuito C_n con entrada x usando M'

Es similar, pero en lo teórico esto mejor la demo.

Ej. 11:

11. Decidir si las clases AC^k y NC^k están cerradas por unión, intersección y complemento.

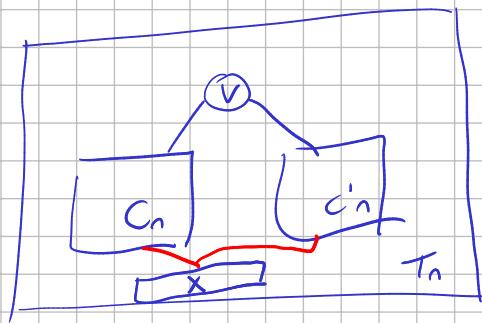
Unión AC^k :

Si tengo $L \in AC^k$ y $\pi \in AC^k \Rightarrow L \cup \pi \in AC^k$.

Sé que L y π tienen sus respectivas familias de circuitos $(C_n)_{n \in \mathbb{N}}$ y $(C'_n)_{n \in \mathbb{N}}$.

Entonces puedo formar la familia de circuitos de la siguiente manera:

- $(T_n)_{n \in \mathbb{N}}$: Es la familia de circuitos q' para cada n se pone a correr C_n y C'_n y se si alguno de estos da 1



→ Que los circuitos estén en paralelo no le agrega altura al circuito.

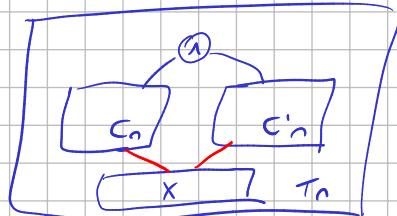
Considerar las líneas rojas como una conexión de todos los bits de X a los circuitos.

Unión NC^k :

Es el mismo concepto q' para la unión de NC^k .

Intersección AC^k y NC^k :

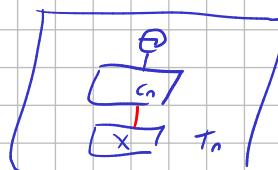
Es similar al concepto de la unión, pero en vez de un V es un \oplus , para indicar q la palabra está en ambas lenguajes.



Complemento AC^k :

Si $L \in AC^k$ tal q' sea decidido por la familia de circuitos $(C_n)_{n \in \mathbb{N}}$

Luego, $\overline{L} = \{x : x \notin L\}$, entonces si el circuito $C_{\overline{x}}(x) = 1$ la nueva familia de circuitos $(T_n)_{n \in \mathbb{N}}$ por cada $C_{\overline{x}}(x)$ devuelve $\overline{C_{\overline{x}}(x)}$.



Complemento NC^K:

Misma idea q' para AC^K.

Y

Ej 12:

12. Dadas dos matrices $A, B \in \{0, 1\}^{n \times n}$ definimos el producto booleano como

$$(A \cdot B)_{ij} = \bigvee_{k=1}^n (A_{ik} \wedge B_{kj})$$

→ Opción de la conjunción
 de elementos de la fila i
 de A y col; de B
 (uno a uno)

Considerar el lenguaje

- $PBM = \{\langle A, B, n, i, j \rangle : A, B \in \{0, 1\}^{n \times n}, 0 \leq i, j < n, (A \cdot B)_{i,j} = 1\}$

Probar que: \vdash Chequea q' en cierta posición del producto booleano de A y B
+ tengo un 1.

a) $PBM \in AC^0$.

b) El lenguaje

- $EBM = \{\langle A, n, k, i, j \rangle : A \in \{0, 1\}^{n \times n}, k \leq \log n, (A^{2^k})_{ij} = 1\}$

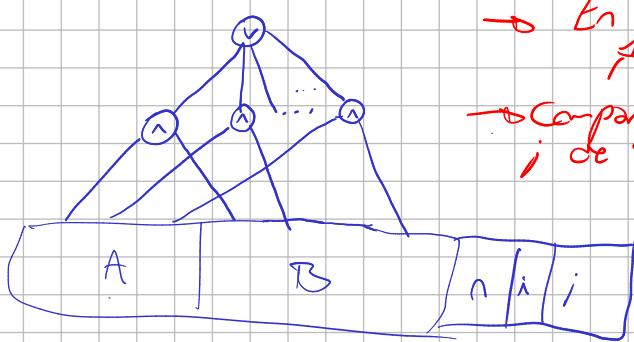
está en AC^1 .

c) Concluir que $NL \subseteq AC^1$.

→ Calculo q' esta potencia
es con productos booleanos

a) $PBM \in AC^0$

Idea: Para q' esté en AC^0 tiene q' tener una familia de circuitos (Cn) q' decida PBM y q' sea L-uniforme tal q' $|Cn| \leq \log n$ y $poly(n) = c \cdot n$.



→ En alguna pos. ambas son 1 → Retorna L.

→ Compara 1 a 1 la fila i de A con la columna j de B.

→ Fila i de A, col; de B, elementos de 1 a n.

me indica qué posiciones debo tomar

* Justificación de q' es L-uniforme más abajo

b) $EBM \in AC^1$

Esto sale relativamente fácil al tener a), $K \leq \log n$ y $(A^{2^K})_{ij} = 1$.

$A^{2^K} = ((A^2)^2)^{2^{K-1}}$ veces
y q' $A^2 = A \cdot A^T$ (y también $X^2 = X \cdot X$ con $X = A^c$ con $c \leq K$).

Luego, tengo un circuito de AC^0 q' se repite a lo sumo $\log n$ veces, por lo cual sé que $EBM \in AC^1$ pues es $\log n$ veces un circuito de AC^0 , entonces quedan $\log n \times \text{ctte}$ (en este caso ctte)

C) $NL \subseteq AC^1$

Idea:

Sé q' PATH es NL-completo, o sea q' si $PATH \in AC^1$, entonces $NL \subseteq AC^1$

Cómo resuelva PATH con una familia de circuitos $(C_n)_{n \in \mathbb{N}}$; $|C_n| \leq \text{poly de } n$; $C_{\text{prof}}(C_n) \leq \log n$

Ojo, a ver, si yo tengo una matriz de adyacencia representando el grafo, entonces el producto booleano me permite obtener el siguiente resultado de qué nodos están conectados con cuales otras con a lo sumo 2 aristas de distancia.

Bueno, siguiendo la idea planteada anteriormente podemos usar EBM (el cual si q' está en AC^1) con la entrada de $\langle A, n, k, s, t \rangle$ donde A es la matriz de adyacencia de la entrada de PATH, s es el starting point de PATH, t es el terminal y n es la cantidad de nodos.

Visual (ejemplo):

$$\begin{array}{c}
 \text{Grafo: } \begin{array}{c} \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \end{array} \quad \text{A} = \begin{array}{c} \begin{array}{ccccc} s & 1 & 2 & 3 & t \\ \hline 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 1 & 0 \\ 3 & 0 & 1 & 1 & 1 \\ t & 0 & 0 & 1 & 1 \end{array} \end{array} \\
 \xrightarrow{\cdot A} \begin{array}{c} \begin{array}{c} \text{---} \\ \text{A} \\ \text{---} \end{array} \quad \begin{array}{c} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \end{array} \end{array} = \begin{array}{c} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \end{array} \\
 \text{Res: } (1 \wedge 0) \vee (1 \wedge 0) \vee (0 \wedge 1) \vee (1 \wedge 0) \vee (0 \wedge 1) = 0 \quad \downarrow \cdot A \\
 \begin{array}{c} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \end{array} = \begin{array}{c} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \end{array}
 \end{array}$$

Sé q' luego de las multiplicaciones a A , van a aparecer todas las conexiones entre nodos q' estén a lo sumo a una distancia de cada nodo.

Como debo ver todos los caminos de a lo sumo n distancia, debo tener A^n para ver si se conectan de alguna forma los nodos. Entonces el k que voy a estar utilizando es $k = \log n$, pues $A^n = A^{\log n}$.

Ahora mi pregunta es, esto q' estoy planteando de la multiplicación de la matriz de adyacencia, ¿tiene sentido o solo me autoconvencí de q' Funciona?

Vamos a hacer una demo informal al respecto:

Si yo tengo A' me muestra las conexiones entre nodos directas. Ahora bien, si yo tengo A^k , ¿qué significa?

Por cada $Z \cdot Y$ uno obtiene en cada posición $(Z \cdot Y)_{ij} = \sum_{k=1}^n Z_{ik} Y_{kj}$.

Si uno lleva el anterior caso a matrices con significado de conexiones de nodos, nos quedamos con que en la posición ij está en 1 si el nodo i está conectado a algún nodo k y el nodo j está conectado al mismo nodo (Estamos viendo $A \cdot A$ por lo tanto para $A^k \cdot A^k$ también).

Por último, esto se encuentra en AC^1 , pues sería básicamente usar el circuito de EBM con la entrada $\langle A, n, k, s, t \rangle$ descrita anteriormente.

*' Ej 12.2: Justificación de L-Uniformidad

Cuál es mi $f: \{0,1\}^k \rightarrow \{0,1\}^*$ computable implícitamente que no crea el circuito?

¿Qué hace mi máq. det. trabajo-L computable?

Algoritmo: $\langle L \rangle$

- Crea u_0 como compuerta de $\textcircled{1}$ *representa la salida*. g' va a tener un Fan-in de γ_2 nodos (los cuales van a ser desde u_{n+1} hasta $u_{n+1+\gamma_2}$) y lo pongo en la salida.
- Crea u_1, \dots, u_n (los nodos de entrada) y los voy poniendo en la salida.
- Crea $u_{n+1}, \dots, u_{n+1+\gamma_2}$ donde son compuertas $\textcircled{1}$ conectadas a el nodo u_i y al u_{i+n} (con $n+1 \leq i \leq n+1+\gamma_2$) (Fan-in 2) y las voy calando en la salida.

*' Ej 5: $\text{NP} \subseteq P/\log n \Rightarrow \text{NP} = P$.

Si tengo $g' \text{NP} \subseteq P/\log n$, es equivalente a decir $g' \text{SAT}$ (g' es NP-completo) $\in P/\log n$.

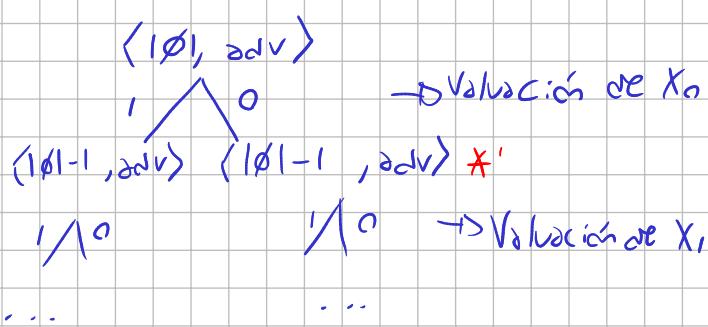
Luego, tengo M det. g' computa en tiempo poly y g' usa un advice de tamaño $\log n$.

¿Cómo hago una M det. poly. para ver $g' \text{SAT}$ está en P ?

Si tengo un adv. de $\log(n)$ podrías generar en M todos los advices posibles. Ya que son $2^{\log n}$ combinaciones, o sea n posibilidades (lo cual es lineal respecto a la entrada) el generarlos a todos.

¿Cómo sé cuál es el advice que me sirve?

Si el advice es para encontrar la valúación correcta de la siguiente manera



Visto en lo teórico g' está en logspace (Prop. 37). Llamo a la máq. g' que hace esto.

Luego, para ver g' tiene sentido la valúación g' retorna $C\langle \langle \emptyset, \text{adv} \rangle \rangle$, la evalúo en \emptyset y la devuelva.

$\phi(C\langle \langle \emptyset, \text{adv} \rangle \rangle)$ (Esto es medio rami)

Por última, ¿Qué es el '*' que puse? Bueno, para cada longitud de entrada el advice es distinto, esto se soluciona sencillamente con padding sobre la fórmula \emptyset .

Ej 13:

13. En este ejercicio se demuestra que el lenguaje

- MAJORITY = $\{x : x \text{ tiene mas 1s que 0s}\}$

está en NC^1 .

- Diseñar un circuito NC^0 que haga lo siguiente: dados 3 números de n bits x, y, z devuelve dos números u, v tales que $x + y + z = u + v$. **Ayuda:** Tomar $v_{i+1}u_i = x_i + y_i + z_i$.
- Probar que el lenguaje $\mathcal{L} = \{\langle s_1, s_2, \dots, s_k, r \rangle : s_i, r \subseteq \{0, 1\}^*, s_1 + s_2 + \dots + s_k = r\}$ está en NC^1 .
- Probar $\text{MAJORITY} \in \text{NC}^1$.

Tengo sueña, q' sea lo q' dios quiera ^1

•

Bonus track: Repasando conceptos q' debemos saber ^11
No me voy sin entender el Teorema de jeroglífico de tiempos (Teo 24).

Si f, g son construibles en tiempo y $f(n) \cdot \log(n) = O(g(n))$, entonces
 $DTime(f(n)) \subseteq DTime(g(n))$

$DTime$ $\{x\}$ longitud tarda en tiempo $g(n)$ en calcularse.

Calculo $n = |x|$ y $t = g(n)$

Simulo $U(\{x, x\})$ por t pasos $\rightarrow O(g(n))$ $\underbrace{g(n)}_{g(n) \text{ pasos}}$

Si $U(\{x, x\})$ no terminó en $\geq t$ pasos: ret 0

Si: no: ret $1 - U(\{x, x\})$ \rightarrow lo contrario a la salida de la máquina

D corre en $O(g(n))$ (se ve clarito), o sea $L(D)$ (el lenguaje decidido por D) $\subseteq DTime(g(n))$.

Supongamos q' $L(D) \in DTime(f(n))$.

Sea M det. q' decide $L(D)$, o sea $L(M) = L(D)$, y existe n_0 tal q'
si $x \in \{0, 1\}^*$ y $|x| \geq n_0$. \rightarrow por la definición de O .

M con entrada x termina en \geq la suma $c \cdot f(n)$ pasos.

\rightarrow Por def. de O y q' M corre en $O(f(n))$

$U(\{x, x\})$ simula $M_x(x)$ tal q' precise \geq la suma $c' \cdot (c \cdot f(n)) \cdot \log(c \cdot f(n))$ pasos (pq' en def. es $K \cdot n \cdot \log(n)$ con K constante dependiente de la máquina y q' tarda la máquina normalmente.)

Existen ctos d , $n_1 \geq n_0$ tal q' $\forall n \geq n_1$ tenemos:

$$d \leq d \cdot f(n) \log(f(n)) \leq g(n)$$

La otra q' tenía que ser más tonta q' una dada.

Tomando x , tal q' $|x| \geq n_1$, y $M_x = M$. La existencia de este x está garantizada por la nueva codificación de máq. que elegimos.

Tenemos $\geq U(\{x, x\})$ q' termina en \geq la suma $g(|x|)$ pasos.

Luego, $D(x) = 1 - M_x(x) = 1 - M(x)$. Pero esto es abs. pq' $L(D) = L(M)$

Siempre termina, así q' da eso.

Demuestra esto q' hay un L tal q' está en $DTime(g(n))$ y no está en $DTime(f(n) \log(f(n)))$.

y

Si f, g son construibles en espacio y cumplen $g' f(n) = o(g(n))$, entonces:

$\text{Space}(f(n)) \subsetneq \text{Space}(g(n))$ (Ferenguiá espacio)

Existe una enumeración M_1, M_2, \dots de máq. det. g' corre en tiempos $\langle f(n) \rangle$

Defino $L = \{ \langle M_i \rangle : M_i \text{ rechaza } \langle M_i \rangle \text{ en espacio } \langle f(1 \langle M_i \rangle) \rangle \}$

Si $L \in \text{Space}(f(n))$, hay una máq. M_K g' decide L en espacio $\langle f(n) \rangle$.

Evaluamos

$\langle M_K \rangle \in L \iff M_K \text{ rechaza } \langle M_K \rangle \iff M_K \text{ acepta } \langle M_K \rangle$.

$L \in \text{Space}(g(n))$ pq' lo q' hace es:

Generar $\langle M_i \rangle$

Simular $M_i(\langle M_i \rangle)$ en espacio $f(n)$

Ver si: acepta o no.

\Rightarrow Se puede hacer en espacio $O(g(n))$ si: $f(n)$ es construible y $f(n) = o(g(n))$

Por último, existe $L \in \text{Space}(g(n))$ tq' $L \notin \text{Space}(f(n))$.

$\text{NPspace}(S(n)) \subseteq \text{Space}(S(n))$, o sea $\text{NPspace}(S(n)) = \text{Space}(S(n))$

Con una gráfica de configuraciones busco en espacio $S(n)$ si: llega a la config final en $2^{O(S(n))}$ pasos

$\begin{pmatrix} \text{---} & \text{---} & \text{---} & \dots & \text{---} \\ \text{---} & \text{---} & \text{---} & \dots & \text{---} \\ \text{---} & \text{---} & \text{---} & \dots & \text{---} \end{pmatrix}$

Es la negación de una diagonal de todos los longuitos en $\text{Space}(f(n))$.
Y esto está en $\text{Space}(g(n))$

$\text{NPspace}(S(n)) \subseteq \text{DTire}(2^{O(S(n))})$

Karp-Lipton: $S: \text{NP} \subseteq \text{P/poly} \Rightarrow \text{PH} = \Sigma_2^P$

Sé q' tengo una máq det tq' con $|\text{adv}| \leq p(n) \Rightarrow$

Computo todos los adv.

Chueco q' este bien: D (con q' se q' $\exists (C) \in \text{NP}$ poly - tq' devuelva una valación de la fórmula q' se satisface).

$q(C(q(\text{adv})))$

Me da una valación

Immerman-S: $\text{PAT} \in \text{NL}$ -completo, por lo tanto $\text{PATH} \in \text{NL-C}$.

