Práctica 8

Ej 1:

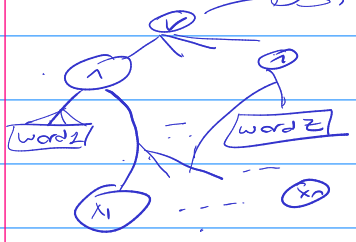
Consultar, una ayu me dijo q' era así pero no estaba segura.

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 permita demostrar que para una entrada x con $|x|=n$, $C_n == 1$ si $x \in L$ con L esparso.

Voy a reescribir la def. de esparso para entenderlo mejor:
$\exists p$. p es un polinomio tq' $|L \cap \{0,1\}^n| \leq p(n)$ para todo $n \in \mathbb{N}$.
↳ O sea, hay una cantidad polinomial de palabras de longitud n

Declaro entonces una Flia de circuitos de la siguiente manera:



→ Si la entrada es alguna de estas da true, si no, false.
→ Veo bit a bit si es alguna de las palabras de L con longitud n.
→ Escribo todas las palabras (a lo sumo $p(n)$) q' pertenecen a L de tamaño n (a lo sumo $p(n)$)
→ Palabra con n bits.

Estos circuitos tienen un tamaño polinomial, ya q' dependen de la cantidad de palabras de longitud n q' haya en L (el cual ya sabemos q' es $p(n)$).

Dep. veré el exacto poly (creo q' sería $n \cdot p(n)$).

Son poly conjunciones (ver todas las palabras de longitud n de L) Luego poly disyunciones (veo q' la entrada coincida con alguna palabra de L con tamaño n.)

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

★ Es más visual con advice (dsp. rehago)

Ej 2:

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

Idea: Sale por diagonalización. La cantidad de lenguajes en P/poly es enumerable y sabemos q' la cant. de lenguajes es no numerable.

Más que nada debemos justificar q' la cant. de circuitos máq. poly con consejo es enumerable, esto se ve en q' se tiene una codificación para cada máq. y por lo cual cada una es codificable finitamente y por ende enumerable

Como no se puede enumerar lo no numerable necesariamente existen lenguajes q' no están en P/poly.

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, $\Pi \in \mathsf{P}_{advice}$ si y solamente si existe una función $adv : \mathbb{N} \to \{0,1\}^\star$ y una máquina polinomial M tal que

Ver q' el adv. es el mismo para toda cadena de = longitud.

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

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

Probar que $\mathsf{P}_{advice} = \mathsf{P/poly}$.

Idea: Para $\subseteq$ se puede ver medio fácil pq' Padvice tiene una máq. q' genera circitos básicamente y habría q' ver q' estos son de altura poly.
Para $\supseteq$ uso q' adv(n) := C_n y luego q' la máq. simule C_n dado el input x (donde |x| = n), o sea adv(n)

⊆) $P_{advice} \subseteq P/poly$

M := Máq. det. polinomial
adv( ) := Función de $\mathbb{N} \to \{0,1\}^*$ y $|adv(n)| \leq p(n)$. $\forall n \in \mathbb{N}$.

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

Como $|adv(|x|)| \leq p(|x|)$ 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 uniformemente puedo hardcodear info dependiente de n.

$M(n) = \langle C_n \rangle$ con M det.

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?
Bueno, esta demo es más difícil de lo anticipado, así q' lo voy a justificar con esto: Como sé q' $P \subseteq P/poly$ (Ver TeoLL) puedo decir q' cada máq. q' corre en tiempo poly está en P/poly, ya q' puedo interpretar q' cada máq. det. poly. con salida $\in \{1, 0\}$ decidía un $L \in P$ (esto sale por def. de P básicamente).

⊇) $P/poly \subseteq P_{advice}$

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

$$x \in \Pi \quad \text{sii} \quad C_{|x|}(x) = 1$$

Puedo crear a adv(n) como la codificación de C_n, de manera q' la máq. M simule $C_{|x|}(x)$ polinomialmente.
¿Por qué y cómo lo hace poly?

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

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

Evalúa C_n con entrada x → Esto es poly pq' es simular las compuertas lógicas codificadas en C_n. Esto sería aprox. $\leq n \cdot p(n)$ compuertas a evaluar.

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

✓

Ej 4:

4. Definimos $\mathsf{P}/f(n)$ como la clase de problemas que se resuelven con un consejo de tamaño $f(n)$ (y entonces $\mathsf{P/poly} = \bigcup_{k\in\mathbb{N}} \mathsf{P}/n^k$). Probar que $\mathsf{P} \neq \mathsf{P}/1 \cap \mathsf{R}$.

Primero lo primero:

- Qué es R? R hace referencia a los problemas recursive, o sea todos los problemas computables.

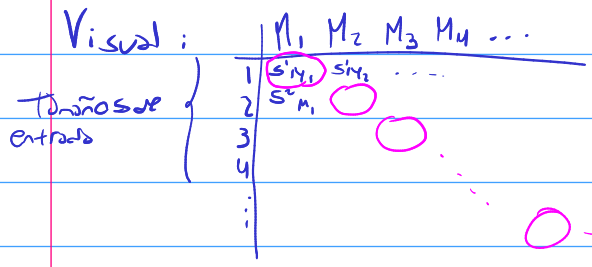
Idea: A ver, q.v.q. $P \neq (P/1 \cap R)$, o sea, tendría sentido q' $P \subseteq P/1 \cap R$, ya q' tendría sentido q' agregarle 1 bit de advice no le saque poder de decisión (sería muy raro si no) Entonces, el problema debe ser q' $(P/1 \cap R) \notin P$.
Entonces, por qué $(P/1 \cap R) \notin P$? Debe haber un Π tal que $\Pi \in P/1 \cap R$ y $\Pi \notin P$, esto lo debo poder ver por diagonalización

Q.v.Q $(P/1 \cap R) \notin P$ por diagonalización:

Enumero $M_1, M_2, \ldots$ a todas las máq. det. polinomiales
Construya $\Pi \in P/1 \cap R$ tal que:

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

Visual:



$\hat{S}^1_{M_n}$ = {Cjto de salidas de todas las posibles entradas de M_n de tamaño n}

○ → Luego algún elemento de $\hat{S}^1_{M_n}$ y lo pongo de advice (Medio q' acá estoy asumiendo que todas las máq. det. poly son de problemas de decisión)

Esto sucede así por lo vista en lo teórico/práctico de 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)

⇒) NP = P ⇒ 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 lenguaje en P/log(n):

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

$|adv(n)| \leq \log(n)$

← Como no lo uso puede ser cualquier cosa ü de tamaño menor o igual a log(n)

Ej 6:

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

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

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

Ej 7: (tengo miedo, cifu la explicó a parte a este 0.0)

7. Probar que si EXP $\subseteq$ P/poly entonces Σ_2^p = EXP.
Ayuda: Probar que si $\Pi \in$ 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.