Guia 7 Complejidad Computacional 1 Cuatrimestre 2025

Juan DElia

Ejercicio 1 Probar que para todo oráculo A se tiene $P^A \subseteq NP^A \subseteq E^A$

$P^A \subseteq NP^A$:

Sea $\Pi \in P^A$ Existe M una maquina deterministica que corre en tiempo polinomial y tiene acceso al oráculo A.

$$x \in \Pi \iff M_A(x) = 1$$

Quiero ver que ese $\Pi \in NP^A$:

Es trivial que puedo simular a M_A con una maquina no deterministica con acceso al mismo oraculo A, N_A.

Entonces $P^A \subseteq NP^A$.

$NP^A \subseteq E^A$:

Tenemos $\Pi \in NP^A$, es decir Existe N una maquina no deterministica que corre en tiempo polinomial y tiene acceso al oráculo A.

$$x \in \Pi \iff N_A(x) = 1$$

Quiero ver que todo $\Pi \in E^A$.

Sabemos que podemos simular cualquier maquina no deterministica simulandola deterministicamente en tiempo exponencial. (y como ambos tienen acceso al mismo oraculo lo llaman y listo)

Ejercicio 2

Ver que sea satisfacible se puede hacer en NP, el problema es ver que sea la mas chica. Seria algo como:

$$\exists v \forall v'.\phi(v') = 1 \Rightarrow \phi(v) = 1 \wedge |v| < |v'|$$

y ademas chquear lo del bit i en 1, pero se ve que para ver que es la mas chica habria que ver que todas las demas valuaciones son mas grades. Asi que mientras la jerarquia polinomial no colapse creeriamos que no esta en NP.

Para resolverlo podemos hacer busqueda binaria en todas las asignaciones (son tiras de bits) y chequear que sea verdadera y con un oraculo ver que dada esa valuacion verdadera que no haya otra mas chica.

Defino la maquina M_Π y doy un algoritmo deterministico que resuelva LEX-SAT-BIT

Primero:

$$\Pi = \{\langle \phi, v \rangle \exists v'.v > v' \wedge \phi(v') = 1\}$$

($>$ chequeando lexicograficamente) Basicamente dada una v dice si existe una menor que la satisface. Se ve que es NP es solo un existe con cosas polinomiales $\Sigma_1^P = NP$ o tambien es facil dar un certificado (una valuacion mas chica que hace la verdadera a la formula)

Algoritmo:

```
M(x,i):
    res = 0
    busqueda bianaria de todas las valuaciones v de x:
        arrancamos "en el medio" y chequeamos si hay una mas chica
        que haga satisfacible a la formula y vamos haciendo busqueda binaria
        hasta encontrar la mas chica que satisface.
    //chequear que el bit i es 1
    if res[i] == 1:
        return 1
    else:
        return 0
```

Justificacion polinomial, en la busqueda binaria. Hay $2^{|x|}$ valuaciones posibles, como hacemos Binary search son $log(2^x)$ iteraciones, es decir es $O(|x|)$

Ejercicio 3 Probar que $NP \cup coNP \subseteq P^{NP}$

Analicemos cada caso por separado

$NP \subseteq P^{NP}$:

Sea $\Pi \in NP$ Veamos que $\Pi \in P^N P$

Hay que reconocer al lenguaje Π usando una maquina polinomial con acceso al oraculo NP, llamemosla M_{NP}. Este oraculo nos da "gratis" cualquier consulta sobre cualquier lenguaje de NP.

La maquina oracular puede reconocer Π, con entrada x, simplemente le pregunta al oraculo si x pertenece a Π.

Corre en O(1) Asi que es polinomial.

Ahora veamos $coNP \subseteq P^{NP}$:

Sea $\Pi \in coNP$ Veamos que $\Pi \in P^{NP}$

Si $\Pi \in coNP \Rightarrow \overline{\Pi} \in NP$. Y ademas $x \in \Pi \iff x \notin \overline{\Pi}$.

La maquina oracular con entrada x puede reconocer a Π de la siguiente manera: 1. Le pregunta al oraculo $x \in \overline{\Pi}$ esto lo hace en O(1) porque $\overline{\Pi} \in NP$ 2. devuelve la respuesta del oraculo negada, pues $x \in \Pi \iff x \notin \overline{\Pi}$

Ejercicio 8 Probar que $PSPACE^{PSPACE} = PSPACE$

Es trivial que $PSPACE \subseteq PSPACE^{PSPACE}$

Veamos que $PSPACE^{PSPACE} \subseteq PSPACE$

Sea $\Pi \in PSPACE^{PSPACE}$ quiero ver que $\Pi \in PSPACE$

Π es decidible con un oraculo con acceso a cualquier problema de pspace, es decir que hay una maquina M_{PSPACE} deterministica que corre en tiempo polinomial con acceso al oraculo y reconoce Π.

Simulemos a esa maquina con la maquina M_{PSPACE} con M, una maquina deterministica que usa espacio polinomial.

M va a ser igual a M_{PSPACE} pero va a reemplazar cada llamada al oraculo por el llamado a la maquina que reconozca el lenguaje al que le esten preguntando al oraculo. Como el oraculo es de PSPACE la "maquina" a la que llamamos en su reemplazo tambien es PSPACE, es decir que usa espacio polinomial.

Como los llamados nuevos que hace solo usan espacio polinomial (PSPACE es cerrada por composicion), sigue siendo PSPACE.

Podemos simular cualquier maquina de $PSPACE^{PSPACE}$ con una maquina PSPACE sin oraculo. En conclusion $PSPACE^{PSPACE} = PSPACE$

Ejercicio 9 Encontrar una clase C tal que $C^C \neq C$

Lo hicimos en clase con C = E.

Veamos que $E \neq E^E$

Por jeraquia de tiempos sabemos que $E \subsetneq 2E$ (2E doble exponencial).

Probemos que $2E \subseteq E^E$ asi podemos ver que E esta estrictamente incluido en E^E:

$$E \subsetneq 2E \subseteq E^E$$

Sea $\Pi \in 2E \Rightarrow \Pi \in TIME(2^{2^{n^k}})$.

Definimos $\Pi_{pad}\{x01^{2^{|x|^k}}\}$

Entonces $\Pi \in 2E$ se puede resolver en E^E con entrada x como:

1. Escribir $|x|01^{2^{x^k}}$ (esto toma tiempo exponencial)
2. pedirle al oraculo el resultado de Π_pad con x.

De esta manera podemos simular cualquier lenguaje de 2E con una maquina $E^E \Rightarrow 2E \subseteq E^E$

Finalmente $E \subsetneq E^E \Rightarrow E \neq E^E$

Ejercicio 10 Probar que $NP^{NP \cap coNP} = NP$

La inclusion $NP \subseteq NP^{NP \cap coNP}$ es trivial, el oraculo es gratis.

Veamos que $NP^{NP \cap coNP} \subseteq NP$

Para esto hay que ver una maquina no deterministica con oraculo $NP \cap coNP$ que corre en tiempo polinomial puede ser simulada por una maquina no determinista que corre en tiempo polinomial sin oraculo.

Notemos que si $\Pi \in NP \cap coNP \Rightarrow \Pi \in NP \wedge \overline{\Pi} \in NP$

Es decir existe un certificado c y un verificador V_1 tq $V_1(\text{x}) = 1 \iff x \in \Pi$

y

existe un certificado c y un verificador V_2 tq $V_2(\text{x}) = 1 \iff x \in \overline{\Pi} \iff x \notin \Pi$

Puedo simular los llamados de la maquina con oraculo sin usar oraculos con entrada x de la siguiente manera:

Simulo Maquina oracular: si hay una consulta al oraculo: adivina r respuesta del oraculo (1 o 0) si r = 1 (segun la query x pertenece): adivinar un certificado C si V_1(xc) rechaza, rechazar (para que pertenezca tiene que existir un cert para V_1) si r = 0 (analogo): adivinar un certificado C si V_2(xc) rechaza, rechazar (para que **no** pertenezca tiene que existir un cert para V_2)

Esto es una maquina no determinista polinomial.

Entonces podemos decir que cualquier maquina no deterministica con oraculo $NP \cap coNP$ que corre en tiempo polinomial, puede ser simulada por una maquina no determinista que corre en tiempo polinomial sin oraculo.

$NP^{NP \cap coNP} = NP$

Ejercicio 11 Dada una clase C, se define $low(C) = \{\Pi \subseteq \Sigma : C^{Pi} = C\}$ **Probar que** $low(NP) = NP \cap coNP$

Primero notemos que low(NP) = $\{\Pi \subseteq \Sigma : NP^{\Pi} = NP\}$

Veamos ambas contenciones.

Primero $NP \cap coNP \subseteq low(NP)$:

Esta es casi automatica.

Sea $\Pi \in NP \cap coNP$ para que este en low tendriamos que ver que $NP^{NP \cap coNP} = NP$.

Esto ya lo demostramos en el ejercicio anterior.

Veamos $low(NP) \subseteq NP \cap coNP$

Ahora tomamos un $\Pi \in low(NP) \iff NP^{\Pi} = NP$ (Hipotesis).

Veamos que pertenecza a NP y coNP por separado.

NP:

Es trivial que $\Pi \in NP^{\Pi}$ pues sea cual sea Π lo puedo decidir "gratis" con el oraculo.

Ademas por hipotesis $NP = NP^{Pi}$:

$\Pi \in NP^{\Pi} \Rightarrow \Pi \in NP$

coNP:

Tambien podemos decir que $\overline{\Pi} \in NP^{\Pi}$. Se puede decidir al complemento de Π negando la salida del oraculo dada cualquier entrada.

$\overline{\Pi} \in NP^{\Pi} \Rightarrow \overline{\Pi} \in NP \Rightarrow \Pi \in coNP$