Práctico 4 Complejidad Computacional

Julián Viera

jviera@fing.edu.uy
julviera44@gmail.com

4 de noviembre de 2021

Agenda

La clase Σ_2^p

La clase es Σ_2^p es el conjunto de los lenguajes L para los cuales existe una MT M de tiempo polinomial y un polinomio q de modo que:

$x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)} \ \forall v \in \{0,1\}^{q(|x|)} \ M(x,u,v) = 1$

$\forall x \in \{0,1\}^*$

Ejercicio 2 - Planteo del problema

Se considera el lenguaje:
$L = \{< \Phi, 1^B > / \Phi \in DNF \ y \ \exists \ \varphi \in CNF, \varphi$ equivalente a Φ y con a lo sumo B cláusulas $\}$.

Probar que $L \in \Sigma_2^p$

DNF : la fórmula es una disyunción (OR) de cláusulas, cada una de las cuales es una conjunción (AND) de literales.
CNF : la fórmula es una conjunción (AND) de cláusulas, cada una de las cuales es una disyunción (OR) de literales.

Prueba de que $L \in \Sigma_2^p$

$< \Phi, 1^B > \in L \Leftrightarrow \exists \varphi \; \forall Z \; M(\Phi, 1^B, \varphi, Z) = 1$
siendo

- φ fórmula *CNF* con las mismas variables de Φ y a lo sumo B cláusulas
- Z asignación de valores de verdad de las variables de Φ
- M es una *MT* que devuelve 1 si para la asignación Z, Φ y φ evalúan igual ($\Phi(Z) = \varphi(Z)$) y 0 en caso contrario

Sea n el número de variables de $\Phi \Longrightarrow |\varphi| = O(nB)$, pues cada cláusula de φ contiene a lo sumo n literales (si una cláusula tuviese más de n literales, dicha cláusula evaluaría siempre en TRUE pues contendría a un literal y a su negación y ser φ del tipo CNF, y entonces dicha cláusula se podría eliminar de φ), y además por hipótesis φ tiene a lo sumo B cláusulas.

Prueba de que $L \in \Sigma_2^p$

Asumimos que $|\varphi| \leq CnB$, donde C es una constante.
Tenemos que $| < \Phi, 1^B > | = kn + B$, donde $k \geq 1$ es proporcional al número de cláusulas de Φ.
$\Longrightarrow |\varphi| \leq C| < \Phi, 1^B > |^2$.
Por otro lado
$Z : x_1, x_2, ...x_n \rightarrow \{1, 0\}^n \Longrightarrow |Z| = n \leq C| < \Phi, 1^B > |^2$.
$\Longrightarrow$ tomamos $p(|x|) = C|x|^2$ y se verifican entonces los requisitos de los tamaños de los cuantificadores que exige la definición de Σ_2^p.
Como M puede decidir si ambas fórmulas evalúan a lo mismo en tiempo polinomial sobre $|x| = kn + B$, ya que su tiempo de ejecución es $O(kn + nB)$, se verifica que $L \in \Sigma_2^p$.

Las clases Σ_i^p, Π_i^p y PH

La clase es Σ_i^p es el conjunto de los lenguajes L para los cuales existe una MT M de tiempo polinomial y un polinomio q de modo que, $\forall x \in \{0,1\}^*$:
$x \in L \Leftrightarrow \exists u_1 \ \forall u_2 ... Q_i u_i \ \ M(x, u_1, u_2, ..., u_i) = 1$
$u_k \in \{0,1\}^{q(|x|)}$ $k \in 1...i$, $Q_i = \forall$ si i es par y $\exists$ si i es impar

$$\Pi_i^p = co\Sigma_i^p = \{\overline{L} : L \in \Sigma_i^p\}$$

La jerarquía polinomial (polynomial hierarchy) se define como

$$PH = \bigcup_{i=1}^{\infty} \Sigma_i^p$$

Ejercicio 3 - Planteo del problema

Probar que si $\Sigma_i^p = \Pi_i^p \Longrightarrow PH = \Sigma_i^p$

Esto implica que la jerarquía polinomial colapsa al i-ésimo nivel.

Prueba de que si $\Sigma_i^p = \Pi_i^p \Longrightarrow PH = \Sigma_i^p$

Probaremos que si $L \in \Sigma_{i+1}^p \Rightarrow L \in \Sigma_i^p$

Sea $L \in \Sigma_{i+1}^p$,

$\xrightarrow{\text{definición}} x \in L \Leftrightarrow \exists u_1 \ \forall u_2 ... Q_{i+1} u_{i+1} \quad M(x, u_1, u_2, ..., u_{i+1}) = 1$

$u_k \in \{0,1\}^{q(|x|)}$

Consideramos el lenguaje L' :

$< x, u_1 > \in L' \Leftrightarrow \forall u_2 \ \exists u_3 ... Q_{i+1} u_{i+1} \quad M(x, u_1, u_2, ..., u_{i+1}) = 1$

$u_k \in \{0,1\}^{q(|x|)}$

$\Rightarrow x \in L \Leftrightarrow < x, u_1 > \in L'$

Observamos que $L' \in \Pi_i^p \xrightarrow{\text{hipótesis}} L' \in \Sigma_i^p$

$\xrightarrow{\text{def}} < x, u_1 > \in L' \Leftrightarrow \exists u_2 \ \forall u_3 ... Q_i u_{i+1} \quad M'(x, u_1, u_2, ..., u_{i+1}) = 1$

$\Rightarrow < x, u_1 > \in L' \Leftrightarrow \exists u_1 \exists u_2 \ \forall u_3 ... Q_i u_{i+1} \quad M'(x, u_1, u_2, ..., u_{i+1}) = 1$

$\Rightarrow x \in L \Leftrightarrow \exists u_1 \circ u_2 \ \forall u_3 ... Q_i u_{i+1} \quad M'(x, u_1 \circ u_2, ..., u_{i+1}) = 1$

$\xrightarrow{\text{definición de } \Sigma_i^p} L \in \Sigma_i^p$

$u_1 \circ u_2 \in \{0,1\}^{2q(|x|)} \Rightarrow$ elegimos $q'(|x|) = 2q(|x|)$

El problema $\Sigma_i SAT$

$\Sigma_i SAT = < \varphi(u_1, u_2, ..., u_i > : \exists u_1 \ \forall u_2 ... Q_i u_i \quad \varphi(u_1, u_2, ..., u_i) = 1$
$Q_i = \forall$ si i es par y $\exists$ si i es impar
Cada u_i es un vector de variables booleanas.

Observación: $\Sigma_1 SAT = SAT$

Ejercicio 5 - Planteo del problema

Probar que si $\Sigma_i SAT$ es completo para Σ_i^p bajo reducciones de tiempo polinomial.

Para probar la completitud de $\Sigma_i SAT$ hay que probar 2 cosas:

1. $\Sigma_i SAT \in \Sigma_i^p$
2. Si $L \in \Sigma_i^p \Rightarrow L \leq_P \Sigma_i SAT$

Prueba de que $\Sigma_i SAT \in \Sigma_i^p$

Sea $L \in \Sigma_i^p$, se cumple que
$x \in L \Leftrightarrow \exists u_1 \ \forall u_2 ... Q_i u_i \ \ M(x, u_1, u_2, ..., u_i) = 1$
$u_k \in \{0,1\}^{q(|x|)} \ k \in 1...i \ , \ Q_i = \forall$ si i es par y $\exists$ si i es impar

De la definición de $\Sigma_i SAT$:
$\varphi(u_1, u_2, ..., u_i) \in \Sigma_i SAT \Leftrightarrow \exists u_1 \ \forall u_2 ... Q_i u_i \ \ \varphi(u_1, u_2, ..., u_i) = 1$
$Q_i = \forall$ si i es par y $\exists$ si i es impar

Tomamos:

- $x = \varphi$
- $q(|x|) = |x| = |\varphi|$
- M es una MT que evalúa φ en tiempo polinomial en $|\varphi|$

$\xRightarrow{\text{definición de } \Sigma_i^p} \Sigma_i SAT \in \Sigma_i^p$

Prueba de que si $L \in \Sigma_i^p \Rightarrow L \leq_P \Sigma_i SAT$

Sea $L \in \Sigma_i^p$, se cumple que:
$x \in L \Leftrightarrow \exists u_1 \ \forall u_2 ... Q_i u_i \ \ M(x, u_1, u_2, ..., u_i) = 1$
$u_k \in \{0,1\}^{q(|x|)} \ k \in 1...i$, $Q_i = \forall$ si i es par y $\exists$ si i es impar

Sea $L^*/x \in L^* \Leftrightarrow \exists u_1 \circ u_2 \circ ... \circ u_i \ \ M(x, u_1, u_2, ..., u_i) = 1$
$u_k \in \{0,1\}^{q(|x|)} \ k \in 1...i$
Es claro que $L^* \in NP$, pues un verificador para L^* es M y un certificado es $u_1 \circ u_2 \circ ... \circ u_i$

Como SAT es NP-completo $\Rightarrow L^* \leq_P SAT$
$\xRightarrow{\text{Teorema de Cook}} \exists f : \ x \xrightarrow{f} \varphi_x(u_1, u_2, ...u_i)$
f polinomial en $|x \circ u_1 \circ u_2 \circ ... \circ u_i|$ y se verifica que:
$M(x, u_1, u_2, ..., u_i) = 1 \Leftrightarrow \varphi_x(u_1, u_2, ...u_i) = 1$
$\Rightarrow \exists u_1 \ \forall u_2 ... Q_i u_i \ M(x, u_1, u_2, ..., u_i) = 1 \Leftrightarrow$
$\exists u_1 \ \forall u_2 ... Q_i u_i \ \varphi_x(u_1, u_2, ...u_i) = 1$
$\Rightarrow \Sigma_i SAT$ es Σ_i^p-completo

Ejercicio 6 - El lenguaje $\exists USAT$

Se considera el lenguaje:
$\exists USAT = \{< \varphi / \ \exists x \in \{0,1\}^n \ \exists! y \in \{0,1\}^m \ \varphi(x,y) = 1\}$.
φ fórmula booleana
$\exists!$: " existe exactamente uno"
$x = (x_1, x_2, ..., x_n)$
$y = (y_1, y_2, ..., y_m)$

Por ejemplo $\varphi(x, y_1, y_2) = (x \wedge y_1 \wedge \overline{y_2}) \vee (\overline{x} \wedge y_1) \in \exists USAT$ porque la asignación $x = 1$ hace que la única asignación para y que hace verdadera la fórmula es $y_1 = 1$ y $y_2 = 0$.

Ejercicio 6 - Planteo del problema

Probar que $\exists USAT$ es $\Sigma_2^p - completo$

Hay que probar entonces que:

1. $\exists USAT \in \Sigma_2^p$
2. $\exists USAT$ es $\Sigma_2^p - hard$

Prueba de que $\exists USAT \in \Sigma_2^p$

$\varphi(x, y) \in \exists USAT \Leftrightarrow \exists x_1 \circ y_1 \ \forall y_2 \quad M(\varphi, x_1, y_1, y_2) = 1$

Se define la MT M de forma que:

$M(\varphi, x_1, y_1, y_2) = 1 \Leftrightarrow \varphi(x_1, y_1) = 1 \wedge (\varphi(x_1, y_2) = 0 \text{ si } y_2 \neq y_1)$

Tomamos $q(|\varphi|) = |\varphi|$

$x_1 \in \{0,1\}^n \ y_1 \in \{0,1\}^m \ y_2 \in \{0,1\}^m$

$\Rightarrow |x_1 \circ y_1| \leq q(|\varphi|), \ |y_2| \leq q(|\varphi|)$

$\xrightarrow{\text{definicion}} \exists USAT \in \Sigma_2^p$

Prueba de que si $L \in \Sigma_2^p \Rightarrow L \leq_P \exists USAT$

Sabemos que $\Sigma_2 SAT$ es completo para Σ_2^p.
Es suficiente entonces probar que $\Sigma_2 SAT \leq_P \exists USAT$.
$\Sigma_2 SAT : \varphi(x, y) \ / \ \exists u_1 \ \forall u_2 \ \varphi(u_1, u_2) = 1$
Consideramos la siguiente transformación f de $\Sigma_2 SAT$ a $\exists USAT$:
$\varphi(x, y) \overset{f}{\Rightarrow} \psi(x, y)$
$f : \psi(x, y) = \overline{\varphi(x, y) \wedge (y_1 \vee y_2 ... \vee y_m)} \vee \overline{\varphi(x, 0)}$
f se puede computar en tiempo polinomial en $|\varphi|$.
Hay que probar que $\varphi \in \Sigma_2 SAT \Leftrightarrow \psi \in \exists USAT$

Correctitud de la reducción propuesta ($\Longrightarrow$)

Ⓗ $\varphi(x, y)$ es una instancia SI de $\Sigma_2 SAT$

Ⓣ $\psi(x, y)$ es una instancia SI de $\exists USAT$

Sea $\varphi \in \Sigma_2 SAT \Rightarrow \exists x\ \forall y\ \varphi(x, y) = 1$

$\Rightarrow \varphi(x, 0) = 1 \Rightarrow \overline{\varphi(x, 0)} = 0$

Como $\varphi(x, y) = 1\ \forall y$, la única asignación de y que hace verdadera a $\psi = \varphi(x, y) \wedge (y_1 \vee y_2 ... \vee y_m) \vee \overline{\varphi(x, 0)}$ es $y_1 = ... = y_m = 0$

$\Rightarrow \exists x\ \exists! y\ \psi(x, y) = 1$

Correctitud de la reducción propuesta ($\Longleftarrow$)

Ⓗ $\psi(x, y)$ es una instancia SI de $\exists USAT$
Ⓣ $\varphi(x, y)$ es una instancia SI de $\Sigma_2 SAT$
Sea
$\psi \in \exists USAT \Rightarrow \exists x\ \exists! y\ \overline{\varphi(x, y) \wedge (y_1 \vee y_2 ... \vee y_m)} \vee \overline{\varphi(x, 0)} = 1$
$\Rightarrow \overline{\varphi(x, 0)} = 0$ pues de lo contrario habría más de una asignación para y que satisface ψ (toda asignación para y lo haría)
$\Rightarrow \varphi(x, 0) = 1$
La asignación $y_1 = y_2 = ... = y_m = 0$ satisface $\psi \Rightarrow$ por hipótesis, es la única asignación para y que satisface ψ.
Para cualquier otra asignación de $y \neq 0$, debe ser necesariamente $\varphi(x, y) = 1$, pues de lo contrario se viola la hipótesis de que para ese x hay un único y que hace verdadera a ψ (si $\varphi(x, y) = 0,\ \psi$ evalúa en 1) .
$\Rightarrow \exists x\ \forall y\ \varphi(x, y) = 1$

Máquinas con oráculo

Una MT con oráculo es una MT M que tiene una cinta de lectura-escritura especial, denominada cinta oráculo, y tres estados especiales q_{query}, q_{yes} y q_{no}.
Para ejecutar M se especifica además una lenguaje $O \in \{0,1\}^*$ que es usado como **oráculo** para M. Siempre que M entra en el estado q_{query}, si en la cinta oráculo se encuentra la consulta q la MT M pasa al estado q_{yes} si $q \in O$ y al estado q_{no} si $q \notin O$.
La salida de la MT M con entrada x y oráculo O se denota como $M^O(x)$.

Ejercicio 7 - Planteo del problema

Probar que si $NP = NP^{NP}$ entonces se cumple $coNP \subseteq NP$

Un camino de prueba posible

Consideraremos el problema $\overline{SAT}$.

Se probará que $\overline{SAT} \subseteq P^{SAT}$:
Dada una instancia x de $\overline{SAT}$, se define una MT determinista M que invoca al oráculo SAT con x y M devuelve 0 si el oráculo SAT devuelve 1 y M devuelve 1 si la respuesta del oráculo SAT es 0.
$\Rightarrow \overline{SAT} \subseteq P^{SAT}$.

Como SAT es NP-completo, $\Rightarrow P^{SAT} = P^{NP}$.
$\Rightarrow \overline{SAT} \subseteq P^{NP} \subseteq NP^{NP}$.
Como $\overline{SAT}$ es coNP-completo, $\Rightarrow coNP \subseteq NP^{NP}$.
Por hipótesis, $NP = NP^{NP} \Rightarrow coNP \subseteq NP$.

Ejercicio 8 - Planteo del problema

Sea A un algoritmo que resuelve el problema de decisión de isomorfismo de grafos, en el que dados dos grafos G_1 y G_2 devuelve "si" si los grafos son isomorfos y "no" en caso contrario. Mostrar que existe una MT polinomial B con oráculo tal que con entrada $< G_1, G_2 >$, B^A devuelve un isomorfismo entre G_1 y G_2 en caso de existir alguno y devuelve "no son isomorfos" en caso contrario.

Este es un caso en que un problema de búsqueda (devolver un isomorfismo) se reduce a un problema de decisión (decidir si dos grafos son isomorfos o no).

Idea de una posible solución

La *MT B* inicialmente invoca al oráculo *A* para saber si los grafos de entrada son isomorfos o no. Si *A* devuelve 0, *B* devuelve "no". Si *A* devuelve "1", *B* va eligiendo una a una las aristas de G_1 y busca para cada una de ellas la arista correspondiente de G_2 en el isomorfismo.
Para ello elige una arista cualquiera $e1$ de G_1 y la saca de G_1, obteniendo $G_1' = G_1 - \{e1\}$. Luego saca de a una las aristas de G_2, obteniendo un grafo G_2' e invocando al oráculo *A* con $< G_1', G_2' >$ hasta que éste devuelva "si", momento en que se identifica la arista $e2$ de G_2 que se corresponde con $e1$ en el isomorfismo. Se repite todo el procedimiento a partir de los grafos isomorfos G_1' y $G_2' = G_2 - \{e2\}$.
Este algoritmo se basa en la propiedad de que si 2 grafos son isomorfos, al quitarles un par de aristas correspondientes en el isomorfismo, los grafos resultantes siguen siendo isomorfos.

Un algoritmo para B

Si $A(G_1, G_2) = 0$ devolver "no son isomorfos"
sino
    Mientras $E(G_1)$ no vacía
        elegir arista e_1 de G_1
        encontre = 0;
        Mientras encontre == 0
            elegir arista e_2 no marcada de G_2
            Si $A(G_1\text{-}\{e_1\}, G_2\text{-}\{e_2\})=1$
                $ISO(e_1)= e_2$
                $G_1=G_1\text{-}\{e_1\}$
                $G_2=G_2\text{-}\{e_2\}$
                encontre = 1
            sino
                marcar(e_2)
            Fin Si
        Fin Mientras
        Desmarcar todas las aristas de G_2
    Fin Mientras
    Devolver el vector ISO de aristas correspondientes
Fin Si

Orden del algoritmo propuesto

Sea m el número de aristas de ambos grafos.
Orden del bucle exterior: $\mathcal{O}(m)$
Orden del bucle interior: $\mathcal{O}(m)$
$\Longrightarrow$ el algoritmo es $\mathcal{O}(m^2) = \mathcal{O}(n^4)$
$\Longrightarrow$ B es polinomial