

Complejidad

Notación

- Uso de símbolo Γ (gamma) para referirse a alfabetos

• Conjunto de símbolos finitos

- Notación Γ^* al conjunto de todo lo que el Γ
- || se refiere a cont. de símbolos en la cadena
- Uso de Γ^n para referirse a los primeros n símbolos de Γ

Codificación en binario

- Mapeo a base binaria del alfabeto binario
- En realidad los resultados son independientes del alfabeto (P)
- Una expansión del alfabeto binario (E)

Notación de números

• $\lceil \log n \rceil = n$ codificado en binario

• $\lfloor \log n \rfloor = \log n$

• Codificación de ~~tupla~~ tuplas

• Una tupla $\sigma \in \{0,1\}^*$, $i = b_1 \dots b_n$, codificarse
 $\langle \sigma \rangle = b_1 b_2 \dots b_n b_n \langle \top \rangle$

• La idea es usar una codificación que permite auto-delimitar

• De este modo, la codina $\langle \top \rangle$ se codifica
 $\langle \top \rangle \langle \top \rangle$

• Cod de linter

• Misma idea, donde cada elemento σ_i se codifica como
 $\langle \sigma_i \rangle$, y al final de la lista se añade un \top

• Cod matriz

• codificarlos

• El tamaño de la codificación es de $\lceil \frac{1}{2}(2k+2) + 2 \rceil$ matriz

nº de filas fin de fila
2 líneas
por cada
elemento en
la representación

Problema de decisión

• Forma lógica de problema con funciones

• Un caso particular es el problema de decisión, que son
funciones booleanas:

$$f : \{0,1\}^* \rightarrow \{0,1\}$$

• Un lenguaje es un conjunto de palabras sobre un
alfabeto dado

• Cada función booleana f representa un lenguaje

$$L(f) = \{x : f(x) = 1\} \subseteq \{0,1\}^*$$

- y viceversa, todo lenguaje Y se puede representar por la función indicadora.

$$x_Y(x) = \begin{cases} 1 & \text{si } x \in Y \\ 0 & \text{si no} \end{cases}$$

Maquina y computo
(Entrada en bandeja, lo hace por los dígitos)

- Una maquina de Turing es un modelo abstracto del computo compuesto por:

- Un alfabeto $\Gamma \cup \Sigma$, en general $\Gamma = \{0, 1, \rangle, \square\}$
- Tiene $k \geq 3$ cintas. Una de ~~trabajo~~, otra de salida, y $k-2$ de trabajo
- Colocar, indicando donde se esta realizando el computo.
Al moverse a derecha y a izquierda, solo lectura / escritura para entrada / salida, despues de lo de piso trabajo
- Conjunto finito de estados, con dos distinguibles q_0 y q_f
- Instrucciones (en diazo se hace falso)

$$\delta: Q \times \Gamma^{k-1} \rightarrow \{\Gamma, L, S\} \times \Gamma^{k-2} \times \{\Gamma, L, S\} \times \{\emptyset, 0, 1\} \times Q$$

- | Es finitamente representable, y mediante una tabla de $|Q| \cdot |\Gamma^{k-1}|$ filas

- | Una vez que se llega a q_f no se sale

- Un ámbito en cuenta, podemos pensar a la máquina de Turing como triplón $M = (\Sigma, Q, S)$
- Una Configuración de una máquina M es una asignación de simblos del alfabeto a la cinta de cada una de las k cintas, y de posición particular a las cabezas.
- [Repasa este punto (página 8) de tu libro]
- Un cómputo de M a partir de $x \in \{0,1\}^*$ a una secuencia de configuraciones C_0, C_1, \dots donde C_0 es inicial, C_k final y C_{k+1} es la evolución de C_k que da S .

Términos computables

- Decimos que una máquina M (computa) $f : \mathbb{P}^* \rightarrow \mathbb{P}^*$ si para todo $x \in \mathbb{P}^*$, la máquina tiene un círculo que termina con $f(x)$ en la cinta de salida.
 - Todo función que tiene M que le computa es computable
- Una función parcial es una función que puede indefinirse en algunos puntos, notamos $f(x) \uparrow$ cuando esto ocurre y $f(x) \downarrow$ como contrario
 - Dicenmos dom(f) = $\{x : f(x) \downarrow\}$

• a) Decimos que una máquina M computa una función parcial computable si para todo x :

| Si $f(x) \downarrow$, entonces (lo mismo que para función total)

y decimos $M(x) \downarrow$

| Si $f(x) \uparrow$, la máquina es "nula"

~~Tiempo de~~

Tiempo de cálculo

• b) Decimos que M corre en tiempo $T(n)$ si para todo x hay un computador de longitud $\leq n$ que para f en tiempo $T(|x|)$ [?]

• c) Decimos que M computa f en tiempo $T(n)$ si para

• O grande

• M corre en $O(T(n))$ si $\exists C$ tal que para todo computador de x , sobre finito, el computador de longitud $\leq n$ para f en tiempo $C \cdot T(|x|)$

• Decimos f computable en $T(n)$ si —

• Decimos f decidible en tiempo $T(n)$ si $\exists g$ computable en tiempo $T(n)$

¶ Es el código binario de
una máquina reprogramable
(se faltó)

Writen de máquina

(T.C.)

- Una función es computable en tiempo si $T(n) \geq n$ y la función $1^n \rightarrow [T(n)]$ es computable en $\Theta(T(n))$?

! Proposición: Sea $f: \{0,1\}^* \rightarrow \{0,1\}^*$, y T una función T.C. y P un alfabeto. Si f es computable en tiempo $T(n)$ para $M = (P, Q, S)$ entonces lo es en ~~$\Theta(T(n))$~~
 ~~$O(\log |P| \cdot T(n))$~~ para una $M' = (\Sigma, Q', S')$



! Proposición: Si f es computable en $T(n)$ para M con $K \geq 3$ cintas, entonces lo es en $O(T(n^2))$ para una de cintas sencilla M' .

! Demo

! → Se indica el contenido de la cinta de M usando un alfabeto $\Gamma = \{\Delta, \square, 0, 1, \text{↑}, \text{↓}, \text{↔}, \text{○}, \text{█}\}$ con el símbolo subrayado indicando que M ha colgado de M otra tira de símbolos.

- !.) Como M viene en $T(n)$, no va a encontrar la subida
más allá de $K \cdot T(n)$. Cada par de codificadores tienen la información
de los cíntas.
- !.) Para ello, va alternando entre los cíntas correspondientes a
cada cinta de M .
- !.) M' con entradas x hace lo siguiente:
- ;.) Calcula $|x|=n$ y $T(n)$, (este es válido por $\exists T.C$)
- ;) Manda la cinta x hasta la $K \cdot T(n) + 1$ (bloque),
donde escribe \perp por uno de los extremos, otra vez
al principio.
- ;) ~~Manda~~ Limpia los extremos de M con sus esteras,
Corta a rasas y que no trate de limpiar
- ;) M' lanza las imágenes en desorden las cintas, recordando
mediante sus esteras que tipo de código cada cinta de
 M , y ~~extera~~ con la S de M , determina qué hace



- !.) Como M viene en $T(n)$ ^{M'}, no va a necesitar hacer uso de más de $K \cdot T(n)$ celdas para codificar todos los información de los cintas.
- !) Para ello, se alterna entre las celdas correspondientes a cada cinta de M .
- !). M' con entradas x hace lo siguiente:
 - ;) Calcula $|x|=n$ y $T(n)$, (estos son válidos para $x \in T.C$)
 - ;) Muestra la cinta $\xrightarrow{\text{y}} \xleftarrow{\text{y}} \dots \xrightarrow{\text{y}} \xleftarrow{\text{y}}$ hasta la $K \cdot T(n) + 1$ celda, donde escribe 1 por uno de límitado, otra de vez en el principio.
 - ;) ~~M~~ termina su estudio de M con sus ordenes, (esto es necesario ya que su trazo es finito)
 - ;) M' hace de igual forma lo deudas las cintas, recordando mediante sus ordenes qué stata llegando cada celda de M , y ~~síntesis~~ con la S de M , determine qué hacer en las nextas.



• Una máquina es oblivious si su código X y $i \in \mathbb{N}$

o) La posición de la cabecera del bit en el i -ésimo paso del computador con entrada X solo depende de f y de $|X|$

o) Las posiciones que computador con posición a patrón de X_i son polig

! Prop: Si f es computable en $T(n)$ entonces es en $O(T(n)^2)$ para un oblivious

! Demo: La idea es la misma que la anterior, nota que su código para la máquina recorre todos los $K T(n)$ estados, sin importar qué expectativa tiene X .

! Prop: Bi-informante \Rightarrow computable en extendido en $T(n)$

Demo: Damos una máquina que usa alfabeto $\{1, 0\} \cup \{0, 1, \square\}^2$ y después la traducimos

Halt

$$f_{\text{halt}}(x) = \begin{cases} 1 & \text{si la } x\text{-ésima máquina con estrella } \\ & \text{termina} \\ 0 & \text{si no}\end{cases}$$

• Halt no es computable

Demo: supongamos que sí, entonces podemos definir una máquina M que ~~si $f_{\text{halt}}(x) = 1$ para todo x~~ $f_{\text{halt}}(x) = 1$ para todo x

Entonces, si $\exists M$ es la i máquinas, tenemos que
 $M(y) \leq t \Leftrightarrow \text{halt}(y) = 0 \Leftrightarrow M(y) \uparrow$

Maquina universal

• Si M_i es máquina tal que $\langle M \rangle i = i$, definimos

$$U \subseteq \{0,1\}^* \rightarrow \{0,1\}^* \text{ con:}$$

$$U(\langle i, x \rangle) = M_i(x)$$

$$(\text{notar que } U(\langle i, x \rangle) \downarrow \Leftrightarrow M_i(x) \downarrow)$$

• Teorema: existe una máquina U que computa

$U(i, x) \rightarrow$ opuesto, si $M_i(x)$ termina en t pasos,
 $U(i, x)$ termina en $c + \log t$ pasos, con c constante

• 1) Dado: se ide o plantea una máquina U , que tiene
3 cintas de trabajo que posee simbolo M , la primera
para x , la segunda simula la cinta de trabajo, y la tercera
para los estados.

• 2) Como la simulación no puede depender de la máquina
basta el trazo de pose cualquier máquina con más
de 1 cinta de trabajo o uno solo.

• 3) Esto ultimo hace que tiene $T(n)^2$ tiempo
(Revise la ley t !!)

• Entonces $\tilde{M}(\langle i, x, t \rangle) = 1 \Leftrightarrow M_i(x)$ termina en a lo más t pasos

P y NP

- Una máquina decide L si compute $x \in L$
- DTime($T(n)$)
- $\underline{P} = \bigcup_{n \in \mathbb{N}} \text{DTime}(n)$
 - ! $C_\infty = \{\langle G \rangle; G \text{ es conexo}\}$ está en P
- Proposición: podemos reemplazar una máquina que decide un lenguaje por una máquina más simple
- NP es la clase de lenguajes tal que existe un polinomio $p: \mathbb{N} \rightarrow \mathbb{N}$ y una máquina M tal q se poly y para todo x :
$$x \in L \Leftrightarrow \exists v \in \{0,1\}^{p(|x|)} \text{ tal que } M(x, v) = 1$$
- ! Demostrar: $P \subseteq NP$
 - Demo: definir una máquina M' que simula M con entrada x con entrada (x, λ)
- ! $\text{Independent} = \{(G, k); G \text{ tiene un conjunto ind de } k \text{ vértices}\}$ está en NP
 - Demo: Sea $G = (V, E)$, $V = \{0, \dots, |V|-1\}$. Necesitamos certificar que el conjunto de vértices es independiente, el tamaño es $O(|V| \log |V|)$
 - Como ~~la~~ $n = |G|, k \leq |V|$, claramente $k \log |V| \leq n \log n$
~~y~~ $\in O(n^2)$ y $\in O(n^2)$

La máquina verifica que el entero no es primo, y checa si el conjunto de primos

- Máquina no-determinística (la máquina del 8 sobre 2 una opción de respuesta) y tiene tres estados distintos, q_0, q_1, q_2, q_3 .
- Una configuración finita es $q_0 \# q_1 \# q_2 \# q_3$
- Un computador de N se pone de $x \in \{0,1\}^*$ (también q determinado)
- Un computador aceptador es uno en el que C_1 está en q_1
- Dicirte x si existe un computador aceptador a partir de x
- Obs: la máquina no-det(N) no computa función y no decide lenguaje
- Corre en tiempo $T(n)$: si todo computador tiene longitud a lo sumo $T(|x|)$
- Clase NDTIME($T(n)$)
- Dada una codificación de computador no-determinístico de t pasos con una codina $\{0,1\}^t$
 - Tiempo, puedes simular una NND con una M ditter en $O(2^t \cdot t)$ pasos

! Teorema: La def. de NP no es equivalente, o decir,
 $NP = \bigcup_{c \in N} NTIME(n^c)$

Demo:

\supseteq) Se illo a una cosa artificiala el computo aceptado de N , con como si el computo es de longitud $f(|x|)$, el M se le tomará poly. Luego, planteamos M tal que simula N , deviendo V para decidir qué componente de f S usó.

\subseteq) Unión entre N que inventa el artificial y dcp simula M .

! Teorema: Existe universal para la no determinística (NV) q no tiene overhead logarítmico $\boxed{?}$

Reducción polinomial

• $L \leq_p L'$ (Karp-reducible) si $\exists f$ poly tal que
 $x \in L \iff f(x) \in L'$

• NP-H y NP-C

! Si $L \leq_p L'$ y $L' \in P$, $L \in P$

• Teorema: $L \leq_p L$ o trivial

Demo: Supongamos que $L \leq_p L'$ va f y $L \leq_p L'$ va g
luego, $x \in L \iff f(x) \in L' \iff g(f(x)) \in L'$

Vamos que \oplus o \oplus es poly

Plantearnos para máquina que cumpla con la función
de \oplus es \rightarrow polinomial, para

1)

• Teorema: Si $NP-H \cap P \neq \emptyset$, entonces $P=NP$

Demo:

Si tenemos un lenguaje \mathcal{L} en la intersección,
entonces podemos reducir cualquier lenguaje \mathcal{L}' en NP_{DTA}

Mismo razonamiento, como \mathcal{L}_Y es poly, tenemos que

$\mathcal{L}_Y(f(x))$ es poly, por lo que \mathcal{L}_Y' tiene que ser

• Teorema: Si $\mathcal{L} \in NP-C$, $\mathcal{L} \in P (\Rightarrow) P=NP$

Demo:

$\Rightarrow) \mathcal{L} \in NP-C \cap P \Rightarrow NP-H \cap P \neq \emptyset$

$(\Leftarrow) P=NP \Rightarrow \mathcal{L} \in NP = P$

• TMSAT = $\{ \langle y, x, 1^n, 1^m \rangle \mid \exists v \in \{0,1\}^n \text{ M}_y(xv) = 1 \text{ en al menos } t \text{ posiciones}\}$

• Prop: $\exists NP-C$

Demo: \rightarrow darse que $\in NP$. Dado que además de $NP-H$,
plantearnos como reducir la función que, dado un lenguaje
 $\mathcal{L} \in NP$, sea el hecho que existe una máquina
M tal que termina en al menos $q(|x| + |b|)$

(poly). Con esto, simplemente metemos

$$\langle \mathcal{L} \rangle = \langle \langle M \rangle, x, 1^{P(|x|)}, 1^{q(|x| + P(|x|))} \rangle$$



Tenemos entonces,

$$f(x) \in \text{TM-SAT}(\models) \iff \text{CTM-SAT}$$

$$\Leftrightarrow \exists v \in \{0,1\}^{\mathbb{N}} P(|x|); M(x_v) = 1 \Leftrightarrow g(|x| + P(|x|))$$

$$\Leftrightarrow x \in \mathcal{Y}$$

✓

SAT

(repaso de introducción / notación / codificación por lora duda)

• SAT, 3SAT ∈ NP

Demostrar: definimos una máquina que es implementable
verificada que el satisfactorio tiene una solución
satisfactoria la fórmula φ .

• Demostrar SAT ≤_p 3-SAT

Representación de funciones

booleanas

• Definición: para todo $F: \{0,1\}^l \rightarrow \{0,1\}$ existe una
fórmula booleana $\varphi_F(p_1, \dots, p_n)$ CNF tal que
 $v \models \varphi_F (\models) F(v) = 1 \wedge v \in \{0,1\}^l$. Ordeno, φ_F
se computa en tiempo polinomial a partir de $\langle F \rangle$ y
tiene complejidad $O(l^2)$

Demo:

Observe que $|(\mathcal{F})| \geq O(2^d)$, pues a la tabla de verdad, podemos entender cuándo es fórmula donde todos los entradas que valen 0 no alegamos que no se cumplen, o decir

$$\varphi_F = \bigwedge_{\substack{i: v_i(F) = 0 \\ i \in I}} \neg (\bigwedge_{j=1}^n p_j \wedge \bigwedge_{i: v_i(j) \neq 0} \neg p_j) = \bigwedge_{\substack{i: v_i(F) = 0 \\ i \in I}} (\bigvee_{j: v_i(j) = 1} p_j \vee \bigvee_{j: v_i(j) = 0} \neg p_j)$$

Así esto, una solución v satisface φ_F si y sólo si la gente de algunos de los que no le satisface. Como el tamaño de una solución es $O(d)$, y la cantidad de soluciones $\geq O(2^d)$, el tamaño de φ_F es $O(d \cdot 2^d)$

□

Corolario: Dado $F: \{0,1\}^l \rightarrow \{0,1\}^k$ existe φ_F tal que $\forall u \models \varphi_F \Leftrightarrow F(u) = v$. φ_F se computa en tiempo polinomial a partir de $\langle F \rangle$ y a de tamaño $O((l+k) \cdot 2^{(l+k)})$

Definición: Sea $G: \{0,1\}^{l+k} \rightarrow \{0,1\}$ donde $G(u,v) = 1 \Leftrightarrow F(u) = v$

mini con configuración

- Suponemos M det., sin círculo de rotida, con solo una de entrada y de trabajo
- La mini configuración en el paso i es una tripleta $Z_i = (a_i, b_i, c_i) \in \Sigma \times \Sigma \times Q$ donde:
 - a_i > el símbolo leído que se coloque en C_i
 - b_i > el trabajo
 - c_i > el estado de C_i
- Suponiendo que M es oblivious, tenemos que podemos calcular la siguiente función en ~~esta~~ pág:
 - $e(i, n) =$ posición de la cinta de estado en el i -ésimo paso del computo de M con entrada σ^n (en realidad la idea es con cualquier entrada arbitraria de longitud n)
 - $t(i, n) =$ idem pero para trabajo
 - $p_{\text{pas}}(i, n) = \max \{ j \leq i \mid t(j, n) = t(j, n) \} \cup \{ i \}$
- Queremos ahora considerar la evolución en un paso de C_{i-1} a C_i . Para conseguir Z_i necesitamos:
 - El trabajo y la cinta en el $i+1$ paso, dado por Z_{i+1}
 - La función S de M (nótese M fijo)
 - El anterior de la cinta de trabajo en $e(i, |x|)$
 - El anterior de la cinta de trabajo en el paso $p_{\text{pas}}(i, |x|)$ (la idea es usarlo para determinar el contenido actual ~~anterior~~ que instanciará en el paso)

an

- o Podemos ordenar los codificadores en una mini configuración
 $Z \in \mathbb{Z} \times \mathbb{Z} \times Q$ con $\{0,1\}^k$, $k = 2^{\log_2 |Q|}$

k depende solo de M . Para $i > 0$ definimos
 $F : \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^2 \rightarrow \{0,1\}^k$ de modo que:

$$F(\langle z_{i-1} \rangle, \langle z_{\text{par}(i, |x|)} \rangle, (x(e(i, |x|))) = \langle z_i \rangle)$$

y mandamos un 0^K todos los entradas invalidas para que
 F sea función.

- o Para el resultado anterior, existe $\phi_F(\bar{p}, \bar{q}, \bar{F}, \bar{s})$
donde p, q, s son variables libres que codifican la miniconfiguración
y \bar{F} codifica el símbolo de la entrada, donde
ordenamos \bar{s} en CNF y $\bar{s} \wedge \bar{a}, \bar{b}, \bar{d} \in \{0,1\}^k$, $c \in \{0,1\}^2$

$$\bar{a} \bar{b} \bar{c} \bar{d} \models \phi_F(\bar{p}, \bar{q}, \bar{F}, \bar{s}) \Leftrightarrow \bar{d} = F(\bar{a}, \bar{b}, \bar{c})$$

(recuerda que se calcula en tiempo poly y tiene complejidad
 $O((3k+2) 2^{3k+2})$)

Cook - Levin

(reparam la demo que se vió)

o Teorema: SAT \in NP-H

Algo Σ es NP-fijo, reparam $\Sigma \leq_p$ SAT.

Veremos que

$$X \in \Sigma \Leftrightarrow \exists u \in \{0,1\}^{P(|X|)} M(Xu) = 1$$

y podemos ver que M no tiene círculo de solido y es universal. Queremos entender cuál es una fórmula ϕ_X

\in CNF en tiempo polinomial tal que

$$X \in \Sigma \Leftrightarrow \phi_X \in \text{SAT}$$

• Veremos que M es fijo y tenemos que

$$X \in \Sigma \Leftrightarrow \exists u \in \{0,1\}^{P(|X|)} M(Xu) = 1 \Leftrightarrow \exists u \text{ y } \exists$$

Computador C $\Rightarrow C \in \{0,1\}^{t(|X|)}$ a partir de entrada Xu

tal que la salida es 1^*

• Tercero tenemos que este sucede si \exists configuración de los entradas $y \in \{0,1\}^{2^{n+4}}$ con $n = |X| + P(|X|)$ y una secuencia $z_0, \dots, z_m \in \{0,1\}^k$ (k depende de M) con $m = t(|X|)$ y ocurre que

1) y es círculo de entrada (y) empieza con x y sigue con $u \in \{0,1\}^*$

2) z_0 es la mini-configuración inicial correspondiente a M en entrada y

3) z_i evoluciona a z_{i+1}

4) z_m es una mini-wfng aceptadora (q.f y con 1 escritor)

• Entrada: Condición de representación con ℓ fórmulas

$$\gamma_1(y_0 \dots y_{2n+3}, b_0^0 \dots b_{m-k}^0, b_0^m \dots b_k^m)$$

Entrada

m mini-configuration

variables

de tamaño $\mathcal{O}(t(n))$ en CNF

$$y \text{ definir } f_x = \gamma_1 \wedge \gamma_2 \wedge \gamma_3 \wedge \gamma_4$$

• Queda justificar como definir las fórmulas

[1] La cinta de entrada (X) implica con x y sigue con $u \in \{0,1\}^*$

$$\gamma_1 = \gamma_0 \wedge \gamma_1 \quad (\text{codi de } \triangleright)$$

$$\wedge \wedge \gamma_j \wedge \gamma_{j+2} \wedge \gamma_{2j+3} \text{ si } X(j) = 1$$

$$\wedge \gamma_{j-1} \wedge \gamma_{j+2} \wedge \gamma_{2j+3} \text{ si } X(j) = 0$$

$$\wedge \wedge \gamma_j \oplus \gamma_{j+1} \quad (\text{entipicando tiene cualquier junt})$$

$$\wedge \gamma_{2n+2} \wedge \gamma_{2n+3} \quad (\text{codi de } \square)$$

Entonces $|\psi(\gamma_1)| = \mathcal{O}(n)$

[2] γ_0 es la miniconfiguración inicial ($\gamma_0 = (x(0), \square, g_0)$)

$$\gamma_2 = \gamma_0 \wedge b_0^0 \wedge b_1^0 \text{ si } X(0) = 1$$

$$\quad \quad \quad \wedge b_0^0 \wedge b_1^0 \text{ si no}$$

$$\wedge b_0^0 \wedge b_1^0 \quad (\square)$$

$$\wedge_{i=s+1 \dots k} \gamma_{b_i} \quad (\square)$$

$$|\psi_2| = \mathcal{O}(1)$$

[3] z_i evoluciona en un paso a z_{i+1}

Tenemos que la entrada es $y = \langle x_0 \rangle$ y recordar que tiene forma

$$z_i = F(\underbrace{Lz_{i-1}}_{K \text{ bits}}, \underbrace{\langle z_{\text{prev}(i,n)} \rangle}_{K \text{ bits}}, \underbrace{y_{\langle e(i,n) \rangle}}_{2 \text{ bits}})$$

Mirando que podemos obtener ese primerlo Φ de F equivalente,

donde ~~$\overline{a}, \overline{b}, \overline{c}$~~

$$\overline{a}, \overline{b}, \overline{c}, d \models \Phi_F \quad \text{y} \quad d = F(\overline{a}, \overline{b}, \overline{c})$$

Y luego, definir en primerlo $\gamma_3^{i,n}$ donde

$$\gamma_3^i = \Phi_F(\langle z_{i-1} \rangle, \langle z_{\text{prev}(i,n)} \rangle, y_{\langle e(i,n) \rangle}, \langle z_i \rangle)$$

de manera que cada uno se ve a si misma y la reducción
quiere de menos constante.

Para lo demotivo anteriormente, este primerlo tiene tiempo

$$O((3k+2)2^{3k+2}) = O(1) \quad \text{y teniendo en cuenta esto, llega}$$

$$\Phi |\gamma_3| = O(m) = O(t(n))$$

[4] z_m a aceptar (misma idea que [2])

Problemas NP-C

- Problema ENP-C (repara demo) $\vdash \vdash$
- CAMINAM, TSP, KnapSack ENP-C \square [?]
- CONP = \exists ; $\exists \in$ NP (una \forall en vez de \exists)
 - Problema PC \subseteq NP \cap CONP
 - Problema P = NP \Rightarrow NP = CONP
 - Problema TAUT \in CONP-C

EXPTIME y

NEXP

- EXP = $\bigcup_{c>0}$ DTIME(2^{n^c})
- NEXP = $\bigcup_{c>0}$ NTIME(2^{n^c})

- Problema NP \in EXP

- Teorema: si $P = NP$ entonces $EXP = NEXP$

Demo: Consideremos un lenguaje $L \in NEXP$, tenemos entonces que N es una máquina ND que decide L en $c \cdot 2^{n^c}$. Entonces $L_{\text{pol}} = \{ \langle X, 1^{2^{1+X^c}} \rangle : x \in L \}$. Sabemos que $L \in NP$, para ello, definimos una máquina N' tal que:

si los entradas X no \rightarrow de la forma $\langle X, 1^{2^{1+X^c}} \rangle$, rechaza. Caso contrario, simula N por $c \cdot 2^{n^c}$ pasos, y devolver el resultado.

$$\frac{-2}{3}x + 0 + \frac{2}{3}z = 0 \Rightarrow -\frac{2}{3}x = -\frac{2}{3}z \quad (\Leftarrow) x = z$$

$$\frac{1}{3}x + \frac{1}{3}z = 0 \Rightarrow x = -z$$

$$192/31$$

$$Dx + w_L x = Dx - w_L x - w_U x + wb$$

$$(\Leftarrow) w_L x + w_L x + w_U x = wb$$

Como $N' \in NP = P$, tenemos ahora que $L_{\text{pad}} \in EXP$

Para esto, definiremos una máquinas M' que con entrada x hace lo siguiente:

$$\text{Compute } y = \langle x \rangle_{\text{pad}}^{2^{|x|C}} \in O(2^{|x|C})$$

Una L pad (y) O Polinomial en $|y|$

M' claramente $\in DTIME(O(2^{|x|C^4}))$

Replicación de claves de
comprobación

(replicar o unir)

• Vemos o comprobamos ahora que hay infinitas máquinas i
tal que una máquina $M = M_i$ ($\langle M_i, i \rangle$)

• Teorema: Si f, g son TC, y $f(n) \log(f(n)) = o(g(n))$
 entonces $\text{DTIME}(f(n)) \subsetneq \text{DTIME}(g(n))$

• Demo: Lo idem a definir una máquina D que
 con entrada x haga lo siguiente:

a) Calcula $n = |x|$

b) Calcula $t = g(n)$

($\cup(g(n))$)

c) Simula $U(x, x)$ para b veces t por $(\cup(g(n)))$

d) Si termina el bucle, devolver 0

→ Con palabras, negar lo anterior

Entonces $g(D) \in \text{DTIME}(g(n))$, respondiendo que admite

$\in \text{DTIME}(g(n))$, que M entregar tal que $g(M) = 2(D)$

y \Rightarrow tal que existe m pose el cual si $|x| \geq m$ termina en
 a los n pasos $C_f(|x|)$ para \rightarrow

Vehicula a D, si simulara $M_x(|x|)$ tiene que existe
 una constante c' tal que, pose un n_1 y $|x| \geq n_1 \geq n_0$

la simulación se acaba para:

$$c'(c_f(|x|) \log(c_f(|x|))) \leq c_f(|x|) \log(f(|x|)) \leq g(|x|)$$

Así, elegimos un X suficientemente grande tal que $|x| \geq n_1$,

y $M_X = M$ (lo cual es válido con la nueva codificación)

Entonces entregar que $U(x, x)$ termina en a los n pasos $g(|x|)$

poro, ya lo que sabemos que no es termina U. Así, tenemos

$$= 1 - M_X$$

$$\Sigma(x) = 1 - U(x, x) = 1 - M_X(x) \quad \text{obr.}$$



o [Teorema] si f , g son TC, $f(i+1) = o(g(i))$,
entonces $\text{NTIME}(f(n)) \subseteq \text{NTIME}(O(g(n)))$

Demostración: definiremos la máquina N de N :

o) Si x no tiene punto $1^i 0^j$, rechazar.

o) Si $|y| < g(i)$:

o) Si $N_i(x_0) \neq N_i(x_1)$ por $g(|x|)$ para $O(g(|x|))$

o) Si alguna no termina, rechazar.

o) Si la otra termina, aceptar si la otra acepta.

o) Si $|y| = g(i)$:

o) Si $N_i(1^i 0)$ sigue el código codificado para y o $O(g(|x|))$

o) Si termina, aceptar si no acepta.

o) Si $|y| > g(i)$, rechazar.

Ahora $N \in \text{NTIME}(O(g(n)))$.

Supongamos $N \in \text{NTIME}(f(n))$, luego, $\exists N' : L(N) = L(N')$

y un n_0 tal que para $|x| \geq n_0$ N' termina en x en $\text{time}(f(|x|))$ para x .

Como $f(i+1) = o(g(i))$, para i suficientemente grande

tenemos $f(i) \leq o(g(i))$. Luego, para i suficientemente

grande tal que $N_i = N'$ y $N_i(x_0) \neq N_i(x_1)$ terminan

en $O(g(|x|))$ para x . Entonces

$1^i 0 \in L(N)$ si $1^i 0^j y \in L(N) \Rightarrow L(N) = L(N')$, $\forall y \in \{0,1\}^{g(i)}$

si $1^i 0^j y \in L(N)$, $\forall y \in \{0,1\}^{g(i)}$

si $N_i(1^i 0)$ rechaza todo lo contrario de y

si $1^i 0 \notin L(N) = L(N')$

(LA DIFER PENDIENTE)

Espacio usado por un
computador

(Voy a recordar que los resultados que he dado hasta ahora
son abridos)

- Espacio usado s. const de celdas (sin tener la entrada)
- Espacio $S(n) / M$ una espaciación $S(n) / f$ computable en espacio
- $\text{SPACE}(S(n)) / \text{NSPACE}(S(n))$

Proposition $\text{DTIME}(S(n)) \subseteq \text{SPACE}(S(n))$

O lo mismo que $S(n)$ celdas

- $\text{SPACE} \subseteq \text{NSPACE}$

Gráfico de configuración

- Sea M una máquina que dada espaciación $S(n)$, define una G_{M} el grafo de config. donde cada config. tiene la información de los cíntas de trabajo hasta $S(|X|)$ celdas, y hay un enlace de C a C' si C' es la celdas desde C
- Prop: Cada vértice de $S(n) \geq \log n$ se configura con $\leq S(n)$ lados.

Demos: respondió, pero lo más difícil es que la cantidad de entradas no se configura con

• Si $S(n) \leq \text{SC}$, entonces $\text{SPACE}(S(n)) \subseteq \text{DTIME}(2^{O(n)})$

PSPACE/NPSPACE

• $NP \subseteq PSPACE$

• Tenemos: Si $f(n)$ son SC ($f = O(g)$) entonces
 $\text{SPACE}(f(n)) \subseteq \text{PSPACE}[g(n)]$

(repara QBF y codificación)

• $TQBF = \{\langle \varphi \rangle \text{ for QBF } \varphi\}$

• $TQBF \in PSPACE$

Nota: La idea es que vemos reduciendo la fórmula, si el cuantificador \exists existe, retenemos los resultados $x=0/1$ y el cuantificador en 0 y en 1 y vemos si sigue de 1 cuantificador en 0 y en 1 y vemos si sigue de 1 en el mundo inverso. Si ese para todos lo hace true si no de non 1.

Como lo obtenemos del otro en el teorema H , y en cada una copiamos la fórmula con el primer cuantificador, que es espacio $O(1|\varphi| |\varphi|)$

• Una QBF generalizada se puede poner en prenexo en tiempo polinomial



• CHECKQBF = { $\langle \varphi, v \rangle$; φ es QBF generalizada y los m variables libres, $v \in \{0,1\}^m$ y $v \models \varphi$ }

• CheckQBF $\leq T$ QBF

Damos C y c para el generalizado a prenexo (φ'), y después reemplazar las variables libres (φ'')

• Nos queremos mostrar que es una especie

$S(n) \geq \log n$ y para cualquier configuración C .

Con entrada x se representa con $C \cdot S(1|x|)$ bits. Dado

$x \in \{0,1\}^t$, se puede computar en poly tiempo la

$S(n)$ una fórmula $f_{M,x}(j,t)$ en CNF

tal que $\langle C \rangle \models \varphi_{M,x}(S,t)$ si

hay un solo flecho de C a C' en $G_{M,x}$ y j es poly en S

• La idea es la siguiente: hacia la fórmula φ el

stack de una barra vertical, extender la fórmula

haciendo punto del "toda" se mantiene igual, salvo

la vertiginosa del adyacente (izquierda/derecha según el movimiento). De modo, la cinta de rodar, y lo que viene

en la cinta de entrada o que la entrada es

en la rule x

Hemos visto para cada posible verificación $\mathcal{P}(n)$ que
 como todo lo posible esto es una cantidad constante de
 bits, tenemos que ~~esta~~ verificación es una cantidad constante de
 segundos para cada uno.

o) Tarea: $\vdash \text{TQBF} \in \text{PSPACE}$

Demostración: Sea $\mathcal{L} \in \text{PSPACE}$ y M tal que decide \mathcal{L}
 en espacio $\mathcal{S}(n)$. Sean C_0 y C_f las configuraciones correspondientes
 a la inicial y a la única final (notar que esto
 lo podemos hacer porque es un único estado aceptado)

- Definir la fórmula $\varphi_i(\bar{s}, \bar{t})$ ($\bar{s}, \bar{t} \in \Sigma^{\mathcal{S}(n)}$) tal que $\langle C_i \rangle \vdash \varphi_i(\bar{s}, \bar{t})$ si y
 sólo si C_i es una configuración entre C_0 y C_f
 que es un resultado de n pasos longitud i entre C_0 y C_f
 en el grafo. ~~que~~
- Como el cont de vértices en el grafo es $2^{\mathcal{S}(n)}$ tiene
 $x \in \mathcal{L} \Leftrightarrow \langle C_0 \rangle \langle C_f \rangle \vdash \varphi_{2^{\mathcal{S}(n)}}(\bar{s}, \bar{t})$
 $\Leftrightarrow \langle \varphi_{2^{\mathcal{S}(n)}}(\bar{s}, \bar{t}), \langle C_0 \rangle \langle C_f \rangle \rangle \in \text{CHECKER}$
- Entonces $\mathcal{L} \leq_p \text{CHECKER} \leq_p \text{TQBF}$
- Otra definición de la fórmula de menor recorrido:
 $f_0(\bar{s}, \bar{t}) = \bar{s} = \bar{t} \vee \varphi_{M, x}(\bar{s}, \bar{t})$
- $\varphi_i(\bar{s}, \bar{t}) = \exists \bar{U} \forall \bar{V} \bar{U} \bar{V} ((\bar{U} = \bar{s} \wedge \bar{V} = \bar{t}) \vee (\bar{U} = \bar{F} \wedge \bar{V} = \bar{N})) \rightarrow \varphi_{i-1}(\bar{U}, \bar{V})$

Teorema: (Montaña) $\text{NPSPACE}(\lambda(n)) \subseteq \text{PSPACE}(\lambda(n)^2)$

o) Tiempo PSPACE = NPSPACE

Demo:

o) De modo similar a lo anterior, definimos una máquina

M con $x \in \text{NPSPACE}(\lambda(n))$ y N una ND tal que

que M decide. Tenemos que $x \in \lambda (=) \exists$ columnas de

longitud $\geq \lambda(n). c$ donde $C_0 \neq C_f$

o) Como estás en cuenta, definimos una función recursiva Dread tal que dado 1 ní tiempo sea columna de longitud i^2

$\text{Dread}(C, C', i)$:

n: $i=0$, devolver 1 si $(C \neq C' \wedge C \neq 0) \vee C' = 0$

Otro caso configuración C'' (aún recién revisar)

$j = \text{Dread}(C, C'', i-1)$

K = $\text{Dread}(C', C'', i-1)$

n: $j = n = 1$ devolver 1

dentro 0



Empieza logarítmico

- $NL \subseteq P$
- Problema EVENCL!
- $\text{PATH} = \{(G, s, t)\};$ hay un camino de s a t en G ?
- PATE NL (no se robe L)
Definir \exists una N tal que:
 $\text{curr} = s; \text{count} = 0$
mientras $\text{count} < |V|:$
~~curr~~ $\text{next} = \text{mejor vecino curr}$
 $\text{si } (\text{curr}, \text{next}) \notin E^{(NL)}, \text{pon a } q_{\text{no}}$
~~curr~~
 $\text{si } \text{next} = t, \text{pon a } q_{\text{si}}$
 $\star \text{curr} = \text{next}$
 $\cdot m+1;$
pon a q_{no}
- ~~La~~ next para cada lenguaje (nodos no visitados)
- NLL-H repetir $\in \leq_p$

Una idea: definir una p de tel que

$$p = \begin{cases} a \in X \in Q^* \\ b \in w \end{cases}$$

con $a \in Q, b \in Q$

CIL

- Una función es computable implicantemente en L si $\exists p$ polig tal que $\forall x, |f(x)| \leq p(x)$ y $\{ \langle x, i \rangle ; f(x)(i) = 1 \} \subseteq \omega$ están en L . En otras, la función $\langle x, i \rangle \mapsto \{ f(x)(i) \mid i \in |f(x)|\}$ está en L .
- f es trájico - f computable si $\exists M$ tal que para espacio $O(\log n)$, más allá de cierto límite. Cada vez que escribimos a trájico
- Prueba equivalencia $\boxed{\text{! ! !}}$
- Sean $f, g : CIL$, entonces $g \circ f \in CIL$
 - La idea es usar sumandos de f , pero elementos divididos al final de $f(x)$ que rechazar, para ello, usa sigma variables. Sea $M_g(\langle x, i \rangle) = o_g(x)(i)$ y $M_f(x, i) = f(x)(i)$ plantea M
 - $M(x_i)$
 - $j = 1$
 - $b = \max_{i \leq j} M_f(x, i)$
 - $k = \text{último parcial escrita}$
 - número $M_g(M_f(x))$ para decidir si el límite que requiere (repaso diagonal)

- $y L$ es L -reducible si y no es $f_i, i \in \{1, 2\} \cap L$
tal que $\forall x \in L \Rightarrow f(x) \in y$
- Dibujar que se transmite $y (L \leq y \wedge y' \in L \Rightarrow y' \in L)$

~~Y~~ \rightarrow ~~PATH~~ \in NL-C
y \rightarrow ~~imposible~~

- $\overline{\text{PATH}} \in \text{NL-C}$, $\overline{\text{PATH}} \in \text{CONL-C}$

Demo:

- Sea $L \in \text{NL}$, N máquina que responde $O(\log n)$
y $L(N) = L$, tenemos que con $f(x) = \langle G_{x,N}, C_0, C_f \rangle$
 $x \in L \Rightarrow$ hay un camino de C_0 a C_f
 $\Rightarrow \langle G_{N,x}, C_0, C_f \rangle \in \text{PATH}$

→ Recordar que cada configuración ocupa $C \cdot \log n$ bits
y que $g \vee g' \models C$ en N

- Primero observamos que f recibe la mitad de adyacencia
de $G_{x,N}$ de tamaño $O(2^{\log n+1} \times 2^{\log(n)}) = O(|x|^2C)$
por lo que lo rebaja a de tamaño poly

- Después, cuando le pides el i -ésimo bit, implementa
determina la configuración correspondiente a esa posición
y cuando se determina si hay una flecha o no

- $x \in NL \Leftrightarrow \exists p \text{ y } M$ (un círculo sólido de centro }
corrido } que
 - $x \in Q \Leftrightarrow \exists v \in \{0,1\}^{P(|x|)} M(x,v) = 1$
 - $M \in SPACE(\log n)$
 - (el centro de entrada no caerá en el espacio)
- Yacimiento: PATH $\in NL$
 - Vamos a ver la definición de NL con certificado
 - Para probar una de las entradas preliminar que no se rechace
 - en época logarítmica vamos a considerar el certificado
 - $A_i = \{v; v \geq \text{desde } r \text{ en } s \text{ es una } i\text{-paso}\}$
 - Con esto en cuenta, el certificado va a ser Z

$$Z = \langle Z_{|A_1|=a_1}, \dots, Z_{|A_n|=a_n}, Z_{t \notin A_n} \rangle$$
 - La idea del certificado $Z_{|A_i|=a_i}$ es comprobar que el
tamaño de A_i sea realmente a_i , para ello, queremos que
estos certificados nos convügen que el tamaño de A_i sea realmente
 a_i , para lo que hay que entrar sobre, para tener la
verdad, si el mismo pertenece a A_i o no. Si es así, los
puntos del certificado v

$$Z_{|A_i|=a_i} = \langle (1, z_1), \dots, (n, z_n) \rangle$$

nº nro

-) Dado $Z_i = \sum_{v \in A_i} v$ y el vértice visible
 y para comprobar de si el vértice de A_i puede ser contado
 todo lo que el certificado sea válido
-) De punto de $Z_{v \in A_i}$ es sólo una simplemente una
 recurrencia de nodos (v_0, \dots, v_k) donde la máquina
 tiene que ser que $v_0 = s$, $v_k = v$ y $k \leq i$, lo que se
 puede en espacio logarítmico
-) De punto de $Z_{v \notin A_i}$ es una recurrencia donde se
 tiene todo los nodos pertenecientes a A_{i-1} con el certificado
 correspondiente, de manera que puede comprobar que el vértice de
 la lista sea $|A_{i-1}|^{(k=|A_{i-1}|)}$, que los certificados de pertenencia sea
 válido, y que $v \notin v_j$, en espacio logarítmico. El
 certificado $\rightarrow (y \notin v \notin N(v_j))$
-) $Z_{v \notin A_i} = \langle (v_1 | Z_{v \in A_{i-1}}), \dots, (v_k | Z_{v \in A_{i-1}}) \rangle$
-) El certificado final es lo mismo que el último, solo que
 no considera los vértices.
-) Notar que la verificación siempre está ordenada por si no
 se alcanza el espacio para verificar el certificado

- $\text{CONL} = \text{NL}$

- Si $S(n)$ es computable en espacio lineal de $S(n) \geq \log n$

$$\text{NSPACE} = \text{CONSPACE}$$

Se juzga por polinomial

- Clasificar si el problema requiere forma en que pueda ser expresado en Lógica de primer orden

- Σ_i^P, Π_i^P : Clase de lenguajes \mathcal{L} tales que existe M det. poly. g polinomial q. tal que

$$\exists u, \phi \{0,1\}^{g(|x|)} \rightarrow \{0,1\}^{|u|} \quad Q_i \in \{0,1\}^{O(|x|)} \quad M(x, u, \dots, u)$$

Colocarlos como Π_{i+1}^P ($i+1$ alternancia)

- $\Sigma_0^P = P$

- $P = \bigcup \Sigma_i^P$

- $\Pi_i^P = \{q; \mathcal{L} \in \Sigma_i^P\}$

- No pueden seguir \exists y \forall

- $\Sigma_i^P \subseteq \Sigma_{i+1}^P \quad \Lambda \subseteq \Pi_{i+1}^P$ (idem Π_i^P)

- $P = NP$ se puede generalizar a $\Sigma_i^P = \Sigma_{i+1}^P$

- Teorema: $P = NP \Rightarrow P \vdash$ (cálculo)

Dem. por inducción en i

Si $i \geq 1$ podemos $P = NP \Rightarrow \Sigma_i^P, \Pi_i^P \subseteq P$

- Caso $i=1$ es trivial para tener $P = NP$
- Caso $i > 1$

Sea $\varphi \in \Sigma_{i+1}^P$, luego $\exists M \text{ poly}$
 $\exists u_1 \xrightarrow{i \text{ alternación}} M(x, u_1, \dots) = 1 (\Leftrightarrow x \in \varphi)$

Para encontrar uno φ' tal que $\langle x, u_1 \rangle \in \varphi'$

$$(\Leftarrow) \forall u_2 \dots u_{i+1} M(x, u_1, \dots) = 1$$

$$(\Leftarrow) \varphi' \in \Pi_{i+1}^P \subseteq P$$

~~Por lo tanto que $\langle x, u_1 \rangle \in \varphi' \Rightarrow x$~~

Luego, existe M' tal que $M'(x, u_1) = 1$

esto implica que $\exists u_1 : M'(x, u_1) = 1$

$$(\Leftarrow) x \in \varphi \quad (\Leftarrow) \varphi \in NP = P \quad \boxed{\square}$$

• Título el caso general $\boxed{\square}$

• Si los Σ_i^P y Π_i^P están cerrados hacia abajo por \subseteq_P
lo ideal es definir una M donde encierra directamente
la $f(x)$ o la ruta del computador

Problema $\Sigma_i^P - C$

- Si existe $\mathcal{L} \in \text{PH-C}$, entonces $\text{Cologra}(\mathcal{L})$ es la idea de que reducir todo a \mathcal{L} es NP-hard .
- Probamos $\text{PH} \subseteq \text{PSPACE}$ $\boxed{\vdash}$
 - Si $\text{PH}_{\text{Cologra}} = \text{PH} \neq \text{PSPACE}$ no
- $\Sigma_i^{\text{SAT}} = \{ (\varphi, \vec{y}, \vec{Q}) \mid \varphi \in QBF \text{ de la forma } \exists \vec{y}_1 \forall \vec{y}_2 \dots \Omega_i \vec{y}_i \varphi(\vec{y}, \vec{y})$ donde $\vec{y} \models \varphi$
- Para $i > 0$, $\Sigma_i^{\text{SAT}} \in \Sigma_i^P$
 - Demostrar que para cada máquina que toma entrada $(\vec{x}, u_1, \dots, u_i)$ y verifica que $|u_k| \geq |\vec{y}_k|$ que si $(\varphi, \vec{y}, \vec{Q})$ satisface, y que los u_j lo verifiquen.
 - Supongamos $\vec{y}_1 \models \varphi$
 - $\exists \vec{y} \in \Sigma_i^{\text{SAT}} \iff \exists \vec{y} \in \{0,1\}^k \quad \vec{y} \models \varphi \wedge \vec{y} \models \vec{Q}(\varphi, u_1, \dots, u_i) \wedge \vec{y} \models \vec{y}_1$
- $\Sigma_i^{\text{SAT}} \in \Sigma_i^P - C$
 - Demostrar como a una tupla \vec{c} para referirnos a la variable que hacen referencia al valor de \vec{x} en la fórmula φ_x de la demo de SAT-C para el certificado, y \vec{t}_j a la j -ésima minicondición.
 - Considerarse que hay $T(n)$ de esto, y cada uno de tamaño constante.

Problema $\Sigma_i^P - C$

- Si existe $\mathcal{L} \in P \neq C$, entonces $C = \emptyset$
La idea es que reducir todo a ser \mathcal{L}
- Probar $PH \subseteq PSPACE$ $\boxed{\vdash \dashv}$
 - Si $PH_{\text{Colognes}} \Rightarrow PH \not\subseteq PSPACE$
no
- $\Sigma_i^P SAT = \{ (\varphi; \varphi, Q) \} F$ de la forma $\exists \bar{y}_1 \forall \bar{y}_2 \dots Q(\bar{y}_1 \bar{y}_2 \dots)$
donde $\models \vdash \varphi$
- Para $i > 0$, $\Sigma_i^P SAT \in \Sigma_i^P$
Demostración: se hace una máquina que tiene entrada
 $(\bar{x}, u_1, \dots, u_i)$ y verifica que $|u_k| \geq |\bar{y}_k|$ que
 $\varphi(\bar{y} \models Q) F$ alterne, y que la u_i lo retenga
Luego \bar{u}_1
 $\exists \bar{y} \models \varphi \in \Sigma_i^P SAT \Leftrightarrow \exists \bar{y} \models \varphi, |u_k| \geq |\bar{y}_k| \dots n(\varphi, u_1, \dots, u_i) = 1$
- $\Sigma_i^P SAT \in \Sigma_i^P - C$
Demostración: toma a un par tupla \bar{t} para representar
a la variable que hacen referencia al valor de X en
la fórmula φ_X de la denso de SAT. \bar{t} por el certificado,
y \bar{t}_j a la j-ésima miniconfiguración
Considerarse que hay $T(n)$ de esta, y cada uno de
los términos constante

γ_1 opinando que la codificación de x no borre el y que
 C tiene $\in \{0,1\}^x$, γ_2 que z_0 es lo inicial, γ_3 que
 z_j reducida a z_{j+1} y γ_4 que z_m sea final
 $M(xu) = 1(F) \quad \overline{u} = \overline{f} \overline{e}, \overline{g_0} \dots \overline{g_m} \quad \overline{\gamma_1 \wedge \gamma_2 \wedge \gamma_3 \wedge \gamma_4}$
 $(\Rightarrow) \overline{u} \models \forall^u (11213) \Rightarrow 4$

La idea es en la última
 (reparar !!)

Máquina con oráculo

- Si $M^X(x)$ termina y a lo largo del cálculo hace consultas $\gamma_1 \dots \gamma_j$ a u lo puede reemplazar por $M^Y(x)$ y todo funciona

- P^X, NP^X

- Si $X \in P$, $P^X = P$

Demo: $P^X \subseteq P$

Simulamos el computador o p^X y cada llamada al oráculo lo reemplazamos por el computador de la M correspondiente a X .

Decidir si lo mismo dice cantidad polinomial de llamadas al oráculo en P^X , y si lo mismo lo entiende también si se le da un input polinomial.

- $\text{EXP}^{\text{COM}} = \{ \langle M, X, 1^n \rangle : \text{la máquina dada } M \text{ con entrada } X \text{ detallada } 1^n \text{ termina lo mismo en } 2^n \}$

$\text{EXP} \subseteq \text{P}^{\text{EXP}^{\text{COM}}}$

Sea $L \in \text{EXP} = \text{DTIME}(2^n)$ tenemos que existe un díjito tal que, para todo $X, |X| \geq K$ se cumple que termina en 0 o lo mismo de $2^{|X|-K}$ para y que lo termina para K suficientemente grande termina en 0 o lo mismo $2^{|X|-K+1}$ para

Algoritmo interno $M^{\text{EXP}^{\text{COM}}}(X)$ con

Si $|X| \leq K$, devolver 100 si $X \in L(01)$

Si no, calcular si $\langle M, X, 1^{|X|-K+1} \rangle \in \text{EXP}^{\text{COM}}$

□

$\text{NP}^{\text{EXP}^{\text{COM}}} \subseteq \text{EXP}$

similar al anterior □

$\text{NP}^{\text{EXP}^{\text{COM}}} = \text{P}^{\text{EXP}^{\text{COM}}}$

Baker, Gill, Solovay

Teorema: Existen máquinas A y B tales que $P^A = \text{NP}^A$ y $P^B \neq \text{NP}^B$

o) Definirnos U_B , para cualquier B, como la caja que contiene todos los que existe un elemento en B de igual longitud:

$$U_B = \{ 1^n : \exists x \in B, |x|=n \}$$

| Al tiene que $U_B \in NP$, por que B , para
que N^B le muestre que.
lo introduce.

| Si no tiene la parte i^n , rechaza
dando un X de tamaño n .

le pregunta si el número n pertenece o no

o) Otra forma de definir un B tal que $U_B \notin P^B$. Para
ello, recordemos primero q hay infinitas soluciones $M = M_i$,
de modo a definir $B = U_B$; y $(n_i)_{i \in N}$ tal que

$$| B_0 = \emptyset, n_0 = 1$$

$$| B_i \subseteq \Sigma^{i+1} \text{ y } n_i < n_{i+1}$$

$$| X \in B_i \Rightarrow |X| \leq n_i \text{ (para q es finito)}$$

o) Pero luego esto lleva a disponer de memoria que por cada
 i , si M_i^B corre en tiempo polinomial, entonces tiene la de una
equivalencia en cadena M_i , o equivalentemente que el lenguaje M_i
que corre en tiempo polinomial decide U_B . En definitiva

| $M_i^B(i^{n_i})$ es igual (al tiempo 2^{n_i-1}) a $M_i^B(i^{n_i})$ o en
tiempo (por que una cadena por q, y tienen misma resultado,
y no pas una ni la otra no pasa)

| M_i^B acepta i^{n_i} en tiempo $2^{n_i-1} (\Rightarrow B_i \text{ no contiene cadena}$
de longitud $i^{n_i} (\Rightarrow i^n \notin U_B))$

• Ahora construimos B_i y n_i . Supongamos que ya tenemos $\forall k \in \mathbb{N}_K \exists n_k \in \mathcal{B}_{i-1}$. Sea m_k el m^o de los longitudes de las cadenas consultadas por $M_{k-1}^{B_{i-1}}(1^n_k)$ (notar q si la misma est^a en D_K) al tiempo $2^{n_{k-1}}$. Definimos $n_i = (\lceil \max\{1, m_k \rceil)$.
 Podremos definir B_i , similar a $M_i(1^{n_i})$ por 2^{n_i-1} para

| Si M_i consulta al oráculo por un X de longitud menor o igual a n_i , responder lo mismo que " $X \in \mathcal{B}_{i-1}$ "

| Si M_i consulta por una de longitud $\geq n_i$, le responder que no

• Si M_i acepta 1^{n_i} en la hora 2^{n_i-1} para , tenemos $\mathcal{D}_i = \mathcal{D}_{i-1}$, por lo que no es o haber cadena de longitud n_i en \mathcal{D}_0 . Si M_i rechaza o no termina la pregunta, preguntar una cadena X de longitud n_i q solvete existir para simular la máquina por 2^{n_i-1} para ~~sin d^r d^r~~ y definir $\mathcal{D}_i = \mathcal{D}_{i-1} \cup \{X\}$ o bien que no se ha visto anteriormente q nunca consultó por una cadena de esta longitud q^o que $M_K \notin \mathcal{D}_i$ y n_{i+1} sea el m^o de la longitud consultada anteriormente. Con esto tenemos que M_K acepta 1^{n_k} q^o no hay cadena de su longitud en \mathcal{D} .

(Ver el último parrafo)

Jerarquía polinomial de NP

en orde alfabético

$$P^C = \bigcup_{X \in C} P^X$$

• Si X GC-completo, $P^C = P^X$

• Teorema: Para $i \geq 1$, $\sum_i^D = NP^{\sum_i^D}$

• \subseteq) Sea $z \in \sum_{i+1}^D$. Existe $\exists M$ tal que
 $x \in z \Leftrightarrow \exists u_1 \dots \exists u_{i+1} M(x, u_1, \dots, u_{i+1}) = 1$

Definimos z' tal que $\langle x, u_1 \rangle \in z' \Leftrightarrow \forall u_2 \dots \forall u_{i+1} M' = 1$

$\Rightarrow z' \in \sum_i^D \Leftrightarrow z' \subseteq \sum_i^D$
No existe N la máquina ND tal que en ordenador z' .
Entonces $z' \in NP^{\sum_i^D}$

Correlativo $(x, u_1) \in z'$

Entonces $z(N^{z'}) = z \wedge z' \subseteq_p \sum_i^D \Leftrightarrow z \in NP^{\sum_i^D}$

$z \in NP^{\sum_i^D} = N^{\sum_i^D}$

\supseteq) Sea $z \in N^{\sum_i^D}$ y sea $N^{\sum_i^D}$ una ND

que decide z . ~~o lo hace de un computador~~ $x \in z$ si y solo si
 $\exists c \in \{0, 1\}^{t(|x|)}$ un computador satisface de $N^{\sum_i^D}$.

Si lo hace de este computador, N la máquina hace o

lo hace una cantidad polinomial de corridas del ordenador

$$(p_j = \exists u_1 \forall \dots \exists u_i \forall \forall j (u_1, \dots, u_i))$$

• Revisa repartición $\gamma_j \in \{0,1\}$ para cada consulta

$\exists v_i$ tal que $\overline{w_j} = \overline{\gamma_j(v_i)}$

$$\overline{v_i} = \overline{\gamma_j} \wedge \overline{\gamma_j} \in \underbrace{\text{FAE}}_{1-2 \text{ alternativas}} \Rightarrow Y_j(\overline{v_i}) =$$

$\exists v_i$ tal que $\overline{w_j} = \overline{\gamma_j(v_i)}$

$\overline{w_j} \notin \overline{\gamma_j} \wedge \overline{\gamma_j} \in \underbrace{\text{FAE}}_{i-1 \text{ alternativa}}$

o si no, tienen que $\overline{x} \neq \overline{y}$

$x \in \mathcal{Y}(\overline{z}) \Leftrightarrow \exists c \exists r_j \exists (\overline{r}_j) \exists (\overline{v}_j) \exists (\overline{w}_j) \forall (\overline{v}_1, \overline{v}_2) \forall (\overline{r}_1, \overline{r}_2)$

$$\overline{Q}(\overline{v}_j) \wedge \overline{Q}(\overline{w}_j)$$

verificable } tal que $N^{\sum_{i=1}^{k+1} p_i}$ acepte x significa el computo en y
 en poly } para cada consulta $j = 1 \dots k$

o bien lo repartir a 1 y $F(Y_j(\overline{v}_1, \dots, \overline{v}_j))$

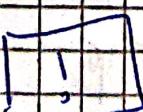
o bien a 0 y $F(Y_j(\overline{w}_1, \dots, \overline{w}_j))$



$$\Delta_1^P = P$$

$$\Delta_{i+1}^P = P^T$$

o bien SAT



Circuito booleano

- Operar sólo con n entrada y única salida con componentes \neg, \wedge, \vee .

• El tamaño $|C| = \# \text{ vértex}$

$P/poly$

FAM $S(n)$

- Una $S: N \rightarrow N$, una función de circuito de tamaño $S(n)$ es una recurrencia $(C_n)_{n \in N}$ donde C_n es un circuito con n entrada y $|C_n| \leq S(n)$

- Una FAM $(S(n))$ decide Σ si para todo $n \geq 1$ y todo $x \in \{0,1\}^n$, $x \in \Sigma$ si $C_n(x) = 1$

- ~~Por~~ Toda Σ se puede uniformizar

• $\text{SIZE}(S(n))$

de S nodes

- Codificamos a un circuito en la forma $TS \log S$ bits

- Es equivalente pensar circuito que representa función

$$\{0,1\}^n \rightarrow \{0,1\}^m$$

• $\mathcal{P}/poly = \bigcup \text{SIZE}(n^c)$

- Feferman's $P \subseteq P_{/\text{poly}}$
) AT (we'll discuss)
- Feferman $P \not\subseteq P_{/\text{poly}}$
Dene: go where give total language using the $P_{/\text{poly}}$ you can be given to decideable function

Karp - Lipton

- Proposición: Sea $(C_n) \in N$ una familia de circuitos de tamaño $\leq S(n)$ tal que $C_n(\varphi) = \chi_{\{f \models \varphi\}}$. Entonces existe $F \in AM$ $S(n)$ $(C'_n)_{n \in N}$ de tamaño poly en $S(n)$ con n válido tal que $\varphi(C'_n(\varphi)) = \chi_{\{f \models \varphi\}}$
- Demoz: Una vez self-reducibility de SAT
 - $\varphi(X_1, \dots, \neg X_n) \in SAT$, entonces $\varphi(X_2, \dots, \neg X_n)$ también
- Si $NP \subseteq P/poly$, $\Sigma_2^P = PH$
 - Demoz: Vamos a probar $\Pi_2^P \subseteq \Sigma_2^P$
 - Como $NP \subseteq P/poly$, existe un polinomio p y un circuito $(C_n)_{n \in N}$ de tamaño $p(n)$ tal que para toda fórmula φ $\varphi \in SAT \Leftrightarrow \varphi \models \exists \vec{x} \in \{0,1\}^n \models \varphi(\vec{x})$
 - $\varphi \in SAT \Leftrightarrow \exists \vec{x} \models \varphi \models \exists \vec{y} \in \{0,1\}^n \models \varphi(\vec{y})$
 - $\Rightarrow C_n(\varphi) = 1$
 - $\forall \vec{x} \exists \vec{y} \models \varphi(\vec{x}, \vec{y}) \Leftrightarrow \forall \vec{x} \exists \vec{u} \in \{0,1\}^n \exists \vec{v} \in \{0,1\}^n \models \varphi(\vec{u}, \vec{v})$
 - $\Leftrightarrow \forall \vec{u} \in \{0,1\}^n (\exists \vec{v} \in \{0,1\}^n \models \varphi(\vec{u}, \vec{v}))$
 - $\underbrace{\quad}_{f_n \in SAT}$
 - $\Rightarrow \forall \vec{u} C_n(\varphi_{\vec{u}}) = 1 \Leftrightarrow \forall \vec{u} f_n(C'_n(\varphi_{\vec{u}})) = 1$
 - $\exists \vec{u} \in \{0,1\}^{p(n)} \not\models \varphi_{\vec{u}}(R_n(\varphi_{\vec{u}}))$

Tamaño y profundidad

de circuito

$$f: \{0,1\}^n \rightarrow \{0,1\}$$

- $\text{MinDg}(f) = \min \{ |C| : C \text{ tiene un entodo } \forall \bar{x} \in C(\bar{x}) = f(\bar{x}) \}$

- $\text{MinDg}(f) = O(n^2)$ (esta super)

• Para todo $n > 1$ existe $f: \{0,1\}^n \rightarrow \{0,1\}$ tal que

$$\text{minDg}(f) \geq \frac{2^n}{4^n}$$

Demostración: considerar de $f = 2^{n/2}$ y $C_n = 2^{n/2} \log 5$

- Profundidad es la distancia desde entodo a salida

NC_{NU} y AC_{NU}

- NC_{NU}^d es la clase de lenguajes decidibles por una FAM (C_n) tal que $|C_n| \geq \text{poly in } n$ y $\text{prof}(C_n) = O(\log d n)$

$$\text{NC}_{NU} \supseteq \bigcup_{d \geq 0} \text{NC}_{NU}^d$$

• AC_{NU} viene poco con hacerlo ordinario

- Parity $\in \text{NC}_{NU}^1, \notin \text{AC}^0$

$$\Rightarrow \text{Parity} \in \text{AC}^0 \subset \text{NC}^1$$

- Numa $\in \text{AC}^0$

$$C_1 = 0, C_{i+1} = \bigvee_{j \geq i \geq 1} ((x_j \wedge y_j) \wedge \bigwedge_{k \geq i+1} (x_k \vee y_k))$$

• Teorema: $A \in \text{C}^d \subseteq N \cup \text{d}^{d+1}$

Demo: simulación por \vdash en orígenes con circuito de
tamaño $\log(n)$

Uniformizado

• Una FAM (C_n) es P-uniforme si $\exists M$ poly
tal que $M(\overline{1^n}) = \langle C_n \rangle$

• Teorema: \vdash es decidible por una familia P-uniforme
 $\forall i \in \mathbb{N} \vdash C_i$

$\Rightarrow (\Leftarrow)$ Podemos definir M tal que simula $M'(\overline{1^n}) = C_n$
y después evalúa el circuito en su interior

(\Leftarrow) En la demo de $P/\text{poly} \supseteq P$ descriptiva comprobamos
que los circuitos uniformes

• ~~Teorema~~ Una familia es L-uniforme si existe una
función $f: \mathbb{N} \rightarrow \mathbb{N}$ tal que ~~$f(\overline{1^n}) = \langle C_n \rangle$~~ $f(\overline{1^n}) = \langle C_n \rangle$

\Rightarrow Podemos que \vdash es decidible por L-uniforme (\Leftarrow) $\vdash \in P$ $\boxed{\vdash}$

- NC^d y AC^d tienen psw algor 2-uniformes
- $NC \subseteq P$
- Un problema tiene una solución paralela eficiente si posee una red de paralelo para entregar el resultado en donde un computador paralelo con una const poly ($O(n^{0.1})$) de procesar en tiempo polilogarítmico
- L tiene solución paralela eficiente si $L \in NC$
- $NC^1 \subseteq L$
 - Denotación: Sea $L \in NC^1$, enton $\exists FAM(C_n)$
 $|C_n| = p(n)$, $pnf(C_n) = \log n$, y M Computa
 $1^n \mapsto \langle C_n \rangle$. Definir $g(1^n, x, u)$ como:
 - Si $u \rightarrow$ la codificación de uno entero, devolver $x_{(k-1)}$
 $(1 \leq k \leq n)$
 - Si u es etiqueta, devolver $g(1^n, x, u_1) * g(1^n, x, u_2)$
 - Una idea \rightarrow hacer DFS, psw queriendo de manejo "eficiente" el recorrido en el circuito, y se que no podemos tener el circuito completo porque \nrightarrow de tenerlo polinomial

- Teorema: Sea $\lambda \in \text{NSPACE}(S(n))$. Existe una familia de circuitos $(C_n)_{n \in \mathbb{N}}$ con compuertas \wedge y \vee de fan-in arbitrario y una máquina de Turing M tal que para todo $n \geq 1$, $x \in \{0,1\}^n$:
 - $x \in \lambda \Leftrightarrow C_n(x) = 1$, $M(i^n) = \langle C_n \rangle$, $|C_n| = O(S(n))$
 - y $\text{prof}(C_n) = O(S(n))$

○ Demo: Considera el grafo de configuración de $N, G_{N,V}, \lambda$. Sea A la matriz de adyacencia, definimos $D_x = A_x \succcurlyeq I$ y tenemos B_x : $j=1$ si j es alcanzable desde i en $2^{\log(\lambda)}$ pasos, cero (iden para $(D_x)^l$) si no, luego $D_x^{2^{\log(\lambda)}} \succcurlyeq I_2$ es lo requerido. Pues tenemos un circuito, tenemos operaciones exponenciales las cuales logran profundidad $\log(2^{\log(\lambda)}) = O(\log(n))$