

# Notas del Curso de Complejidad 2021-1

## 1 Máquinas de Turing

### 1.1 Introducción

En 1906 se declara el Problema 10 de Hilbert: ¿Existe un algoritmo que decida si una ecuación diofantina dada (con cualquier mínimo de variables) tiene solución (entera)?

En 1928 se genera el problema de Hilbert-Ackermann : ¿Hay un algoritmo que dado un enunciado, un conjunto de axiomas y un conjunto de reglas de inferencias, decida si el enunciado se puede deducir a partir de los axiomas mediante las reglas de inferencia?

Leibniz propuso el tener "un tipo de álgebra en el que todas las verdades de la razón pudieran reducirse a algún tipo de cálculo"

Un problema de decisión  $P$  lo podemos pensar como  $f : A \rightarrow \{0, 1\}$  donde  $A$  es el conjunto de instancias (o entradas) de  $P$ .  $f^y[0]$  conjunto de no-instancias  $f^y[1]$  conjunto de sí. instancias

### 1.2 Máquina de Turing

**Def** Una Máquina de Turing es una noneta ordenada  $M = (Q, \Sigma, \Gamma, \sqcup, \vdash, \delta, q_0, q_a, q_r)$

- $Q$  es un conjunto no vacío (de entrada)
- $\Sigma$  es un conjunto no vacío (alfabeto de entrada)
- $\Gamma$  es un conjunto no vacío (alfabeto de cinta)
- $\delta : Q' \times \Gamma \rightarrow Q \times \{L, R\}$  donde  $Q' = Q - \{q_0, q_r\}$
- $q_0 \in Q$  (estado inicial)
- $q_a \in Q$  (estado aceptación) Estado de paro
- $q_r \in Q$  (estado rechazo) Estado de paro
- $\sqcup \in \Gamma - \Sigma$  (símbolo en blanco)
- $\vdash \in \Gamma - \Sigma$  (marcador o delimitador izquierdo de inicio de cinta)

Para cualquier  $q \in Q$  existe  $p \in Q$  tal que  $\delta(q, \vdash) = (p, \vdash, R)$

Una configuración de la máquina de Turing se determina por:

- Estado  $q$
- Posición de la cabeza
- Contenido de la cinta  $uv$

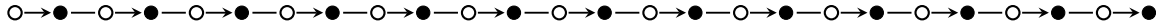
La configuración se ve como  $uqv$ ,  $q$  es el cabezal que se encuentra en la primera posición de  $v$ .

Ejemplo: Si  $M$  tiene entrada  $w$  su posición inicial es  $q_0 \vdash w$

La configuración  $\alpha$  da lugar a la configuración  $\beta$   $\alpha \rightarrow \beta$  y se define como sigue:

- $\alpha \rightarrow^0 \alpha$
- Diremos que  $\alpha \rightarrow^{n+1} \beta$  si existe una configuración  $\Gamma$  tal que  $\alpha \rightarrow^n \Gamma$  y  $\Gamma \rightarrow \beta$





- $\alpha \rightarrow^* \beta$  si  $\alpha \rightarrow^n \beta$  para algún  $n \in \mathbb{N}$

Una configuración e aceptación (o rechazo) es cualquiera en la que el estado sea  $q_a$  (ó  $q_r$ ). Éstas configuraciones son de paro.

Si  $T$  es una MT y  $w \in \Sigma^*$  se dice que  $T$  acepta  $w$  si  $q_0 \rightarrow^n xq_a y$  para cadenas  $x, y \in \Sigma^*$ .

El lenguaje de  $T$ ,  $L(T)$  es el conjunto  $L(T) = \{x \in \Sigma^* | T \text{ acepta a } x\}$ .

Si  $L \subseteq \Sigma^*$  es un lenguaje, se dice que  $T$  acepta a  $L$  si  $L(T) = L$ .

Los lenguajes regulares pueden ser reconocidos por un autómata finito determinista

**Algoritmo** Algoritmo para convertir de DFA (autómata finito determinista) a TM (máquina de Turing)

1. Copiar cada estado
2. Cambiar cada transición con etiqueta ' $x'$ ' por una con etiqueta ' $x/x, R'$ '
3. Agregar un estado de aceptación
4. Para cada estado final en el autómata, agregarle una transición  $\sqcup/\sqcup, R$

¿Cómo describimos las máquinas de Turing? Con pseudocódigo Ejemplo: Describimos la máquina de Turing que obtiene las potencias de dos como sigue:  $M = \text{"Con entrada } \vdash w:$

1. Recorre la cinta de izquierda a derecha, tachando un cero no y uno si
2. Si en 1 la cinta contenía un único cero, acepta
3. Si en 1 la cinta contenía más de un cero, y un número impar de ceros, entonces rechaza
4. Regresa el cabezal a extremo izquierdo de la cinta
5. Va a 1"

Las máquinas de Turing pueden computar funciones parciales.

Tenemos  $f : D \rightarrow C$   $D \subseteq \Sigma^*$   $C \subseteq \Gamma^*$ .

**Def.** Sea  $R$  una MT,  $k$  un natural y  $f$  una función parcial sobre  $(\Sigma^*)^k$  con valor en  $\Gamma^*$ . Decimos que  $T$  computa a  $f$  si para cada  $(x_1, \dots, x_k)$  en el dominio de  $f$ ,  $q_0 \vdash x_1 \sqcup x_2 \sqcup \dots \sqcup x_k \rightarrow^k q_0 \vdash f(x_1, \dots, x_k)$  y ninguna otra entrada que sea una  $k$ -ada de cadenas es aceptada por  $T$ .

Una función parcial  $f : (\Sigma^*)^k \rightarrow \Gamma^*$  es Turing-computable (computable) si hay una máquina de Turing que la computa.

Al fijar el contradominio y la aridad de la función, entonces una MT calcula una única función parcial.

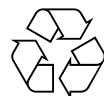
Una tercera opción del cabezal de la MT es quedarse en su lugar después de hacer algo (S).  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$ .

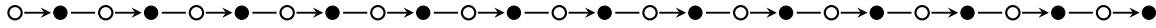
$M_1$  tiene "la opción"  $S$   $M_2$  no tiene "la opción"  $S$  pero reconoce el mismo lenguaje que  $M_1$  o computa la misma función.

Simulo una máquina de Turing que tiene la opción S con una máquina de Turing que no la tiene al poner un estado adicional, donde en las transiciones se escribe lo mismo que lea y se mueva a la derecha, luego se escriba lo mismo que lea y se mueva a la izquierda.

Algunas MT útiles que regresan el cabezal a donde estaba originalmente:

- NB (Next Blank)
- PB (Previous Blank)





- Copy ( $\vdash x \rightarrow x \sqcup x$ )
- Insert( $\sigma$ )
- "Borrar la cinta" (Ponemos un delimitador derecho y borramos todo a su izquierda)
- Compare ( $\vdash x \sqcup y$  acepta si  $x = y$  o rechaza si  $x \neq y$ )
- R (Reverse)

Copy  $\rightarrow$  NB  $\rightarrow$  R  $\rightarrow$  PB  $\rightarrow$  Equal (Esta máquina de Turing reconoce palíndromos)

### 1.3 Máquinas de Turing multicintas

Una MT multicinta es como una MT pero tiene  $k$  cintas de memoria. Su definición solo difiere respecto a una MT usual en la función de transición  $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$ .  $\delta(q_1, a_1, \dots, a_k) = (q_1, b_1, \dots, b_k, D_1, \dots, D_k)$   $D_i \in \{L, R, S\}$ .

Las máquinas de Turing multicinta no tienen más poder que las de una cinta. Para simularlas en una máquina de Turing de una cinta podemos usar bloques como letras de alfabeto y manejamos la posición del cabezal como un puntito en las distintas letras. Le hace una pasada para ver los cabezales y luego realiza otra para hacer las transiciones. La memoria está en la estructura de los estados (el control finito). Otra manera es que los espacios pares simulen una cinta y los impares otra (es trivial generalizarlo a  $k$ ). Pero usaremos la siguiente:

**Teorema** Toda MT multicinta tiene una MT de una sola cinta equivalente.

Dem: Construimos, dada una MT multicinta  $M$  a la MT de una sola cinta  $S$  t.q. tiene los contenidos de las cintas de  $M$  en su cinta, separados por un símbolo especial  $\#$ , y usando versiones especiales de cada símbolo en  $\Gamma_M$  para denotar donde están los cabezales  $\dot{o}$

$S =$  Con entrada  $w_1, \dots, w_n$

1. Poner su cinta en el formato que representa las  $k$  cintas. La cinta formateada se ve como  $\vdash w_1 | \dots | w_n | \# | \sqcup | \# | \sqcup | \dots |$ .
2. Para simular un movimiento,  $S$  recorre la cinta desde  $\vdash$  donde hasta el último  $\#$ , para determinar los símbolos que están leyendo los  $k$  cabezales. Después  $S$  hace una segunda pasada para actualizar las cintas de acuerdo a la función de transición de  $M$ .
3. Si en cualquier punto,  $S$  mueve alguna de las cabezas virtuales hacia la almohadilla ( $\#$ ) de la derecha, en  $M$  entonces el cabezal correspondiente se movió a un espacio en blanco. Luego, basta con que se inserte un símbolo blanco en esa posición.

**Corolario** Si  $L$  es un lenguaje, entonces  $L$  es Turing-reconocible si y sólo si  $L$  es reconocido por alguna MT multicinta.

Dem: Toda MT es una MT multicinta.

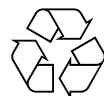
La otra implicación es el teorema anterior ■

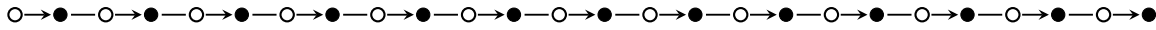
### 1.4 Máquinas de Turing no deterministas

Se diferencian de las deterministas está en que tenemos una nueva función de transición:

$$\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R, S\})$$

Ya no es un algoritmo, sino que ahora la máquina "adivina" qué transición tomar para llegar al resultado. Se dice que acepta si al menos un camino lleva a aceptación. Se parecen a los autómatas no-deterministas. No son implementables en la vida real. Son equivalentes a una MT (usando una MT que haga BFS exhaustivo sobre el árbol de posibles configuraciones) Es importante pensar en términos de que la máquina "adivina" la respuesta correcta (SOLO si existe). Toda MT es MTN.





Ejemplo: Un número en unario ¿Es compuesto? Aplica la siguiente combinación de MTs  $NB \rightarrow G$  (Guess)  $\rightarrow NB \rightarrow G \rightarrow PB \rightarrow M$  (Multiply)  $\rightarrow PB \rightarrow Equal$

¿Para qué sirven las MTN? Estas máquinas teóricas realizan búsquedas exhaustivas “rápidamente” Resuelven problemas lentos muy rápido. “Si tuviéramos una MTN, podríamos resolver un montón de problemas que nos causan dolores de cabeza en la vida real” Como por ejemplo factorización de números y polinomios, agente viajero, SAT, etc.

**Teorema** Toda MTN tiene una MTD equivalente.

Dem: Sea  $N$  una MTN, proponemos a  $M$  una MTD de 3 cintas.

- En la primera cinta está la entrada y nunca se modifica.
- La segunda cinta es una copia de la cinta de  $N$  y se usa para simular a  $N$
- La tercera cinta contiene las direcciones de los nodos del árbol de ejecución de  $N$  codificadas con  $\Sigma = \{1, \dots, b\}$  Existen direcciones inválidas.

$M =$  "con entrada  $w$

1. Copia la cinta 1 a la 2
2. Usa a la cinta 2 para simular a  $N$  con entrada  $w$  en una rama de su cómputo no determinista. Antes de cada transición, consulta la cinta 3 para determinar que decisión tomar entre las distintas posibilidades no deterministas. Si no hay más símbolos en la cinta 3 o se llega a una dirección inválida, aborta y va a 3. Si encuentra una configuración de rechazo va a 3. Si encuentra una configuración de aceptación, acepta
3. Reemplaza la cadena en la cinta 3 por la siguiente en el orden canónico de  $\{1, \dots, b\}^*$
4. Va a 1"

Si una MTN  $M$  se ejecuta hay 3 posibilidades:

1. acepta Configuración de paro
2. rechaza Configuración de paro
3. se cicla

**Corolario** Un lenguaje es reconocible si y sólo si alguna MTN lo reconoce.

Una MT que no se cicla es un decididor.

Un lenguaje  $L$  es Turing-decidible si existe un decididor que lo reconoce.

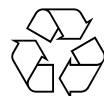
**Corolario** Un lenguaje es decidible si sólo si alguna MTN lo decide.

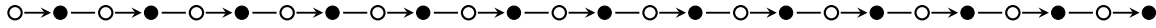
Dem. La ida es trivial. Para el regreso notemos que cada longitud de cadena es un nivel del árbol, así que mantenemos un booleano que es  $v$  si todas las ejecuciones han terminado (en la longitud actual) f.e.o.c. Cuando acaben las cadenas de longitud  $k$ , rechazamos si el booleano es  $v$ . En otro caso, vamos a la longitud  $k+1$ .

## 1.5 Enumeradores

Sea  $E$  una MT con  $k$  cintas  $k \in \mathbb{Z}$  y sea  $L \subseteq \Sigma^*$ . Decimos que  $E$  *enumera* a  $L$  si:

1. El cabezal de la primera cinta nunca se mueve a la izquierda, nunca escribe un espacio en blanco y nunca modifica un símbolo que ya escribió.





2. Para cualquier  $x \in L$  hay algún punto de la operación de E en la que la cinta 1 contiene  $x_1\#x_2\#\dots\#x_n\#x$  donde  $x_i \in L$  p.c  $1 \leq i \leq n$  y  $x_1, \dots, x_n, x$  son todas distintas. Si L es finito no se imprimen símbolos después de la última  $\#$ .

El lenguaje de E, es L, i.e.  $L(E) = L$ .

**Teorema:** Sea  $L \subseteq \Sigma^*$  un lenguaje. Entonces L es reconocible si y sólo si es enumerado por algún enumerador; además L es decidible si y sólo si es enumerado en orden canónico por algún enumerador.

Demostración: Supongamos primero que L es decidible y sea M una MT que lo decide. Proponemos a E dado por

E = "Ignora la entrada

1. Genera la siguiente cadena  $w$  en orden canónico de  $\Sigma^*$
2. Ejecuta M con entrada  $w$  e imprime a  $w$  si M acepta".

Como M es un decididor para L, E va a recorrer todas las cadenas en  $\Sigma^*$  y, en orden canónico, va a imprimir solamente las que están en L.

Supongamos que L es reconocido por M. Proponemos al enumerador E dado por: E = "Ignora la entrada

1. Para cada  $i \in \mathbb{N}$
2. Genera la siguiente cadena  $s_i$  en orden canónico y le agrega a la lista  $s_0, s_1, \dots, s_i$
3. Ejecuta M por  $i$  pasos con cada una de las entradas  $s_0$  hasta  $s_i$
4. Si M acepta alguna de las entradas y no la había aceptado en una ejecución anterior, la imprime."

Si tenemos  $s_i$  y  $s_k$  tal que  $i < k$ ,

Supongamos que E enumera a L. Proponemos a M para recorrer a L dado por: M = "Con entrada  $w$ :

1. Ejecuta E. Cada vez que se imprima una cadena, la compara con  $w$ .
2. Si  $w$  se imprime en algún momento acepta."

Supongamos que E enumera a L en orden canónico, Proponemos a M dado por M = "Con entrada  $w$ :

1. Ejecuta E. Cada vez que se imprima una cadena, la compara con  $w$ .
2. Si  $w$  se imprime en algún momento acepta.
3. Si la cadena impresa por E supera a  $w$  en orden canónico, rechaza." ■

## 1.6 Tesis Church-Turing

Turing, Church, Gödel et al formalizaron el concepto de algoritmo.

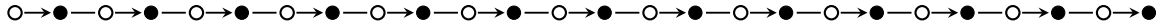
Los algoritmos desde el punto de vista intuitivo son equivalentes a las Máquinas de Turing.

Ejemplo:  $I : G$  es gráfica

Q: ¿Es G conexas?

M = "Con entrada  $\langle G \rangle$  con G una gráfica:





1. Elige el primer vértice de  $G$  y lo marca.
2. Repite el siguiente paso hasta que no se marquen vértices nuevos.
3. Para cada vértice en  $G$ , éste es marcado si es adyacente a algún vértice marcado.
4. Revisa si todos los vértices están marcados. En tal caso acepta, si no rechaza".

M decide A

## 1.7 Máquina Universal de Turing

U: "Con entrada  $\langle M, w \rangle$  con  $M$  una MT

1. Simula a M con entrada a  $w$
2. Acepta si M acepta, rechaza si M rechaza.

## 2 Problema de aceptación

$A_{AFD} = \{ \langle A, w \rangle \mid \text{El AFD } A \text{ acepta a la cadena } w \}$

Problema  $A_{AFD}$

I: Un AFD A y una cadena  $w$

Q: ¿A acepta a  $w$ ?

**Prop:**  $A_{AFD}$  es decidable.

Dem: Proponemos una MT  $M$  que decida a  $A_{AFD}$

$M$  = Con entrada  $\langle A, w \rangle$  con a un AFD y  $w$  una cadena:

1. Simula A con entrada  $w$ .
2. Si la simulación termina en un estado de aceptación, acepta. Si no, rechaza■

**Prop:**  $A_{AFN}$  es decidable

Dem: Proponemos una MTD N que decida a  $A_{AFN}$

$N$  = "Con entrada  $\langle B, w \rangle$  con B un AFN y  $w$  una cadena:

1. Construye una AFD A equivalente a B.
2. Ejecuta M con entrada  $w$ .
3. Acepta si M acepta y rechaza si M rechaza".

$A_{REX} = \{ \langle R, w \rangle \mid R \text{ es una expresion regular que genera a } w \}$

**Prop:**  $A_{REX}$  es decidable

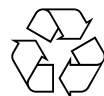
Dem: P = "Con entrada  $\langle R, w \rangle$  donde R es una regex y  $w$  una cadena:

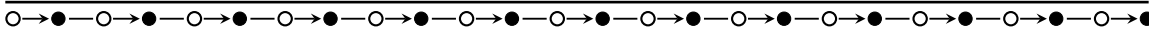
1. Construye un AFN A equivalente a R.
2. Ejecuta N con entrada  $\langle A, w \rangle$
3. Responde lo mismo que N".

$E_{AFD} = \{ \langle A \rangle \mid A \text{ es un AFD con } L(A) = \emptyset \}$

**Prop:**  $E_{AFD}$  es decidable

Dem: T = "Con entrada  $\langle A \rangle$  donde A es un AFD:





1. Marca el estado inicial de A.
2. Repite hasta que no se marquen estados nuevos
3. Marca todos los estados que tengan transición desde un estado marcado.
4. Si algún estado de aceptación está marcado, rechaza. Si no, acepta.

$EQ_{AFD} = \{ \langle A, B \rangle \mid A \text{ y } B \text{ son AFD tales que } L(A) = L(B) \}$

**Prop:**  $C, L(C) = L(A) \triangle L(B)$

Dem: F = "Con entrada  $\langle A, B \rangle$  donde A y B es un AFD:

1. Construye a C tal que  $L(C) = L(A) \triangle L(B)$
2. Ejecuta a T con entrada C
3. Si T acepta, acepta. Si no, rechaza■.

Función característica: El conjunto  $S \subseteq A$   $\chi_S : A \rightarrow \{0, 1\}$   $\chi_S(a) = 1$  si  $a \in S$  0 si  $a \notin S$

$\omega = \{0, 1, 2, 3, \dots\} = \mathbb{N}$   $(0, 1, 0, 1, \dots)$  sería el conjunto de los impares.

$\mathcal{P}(\mathbb{N}) \sim^{\mathbb{N}} 2 = \{f : \mathbb{N} \rightarrow \{0, 1\}\}$

$\varphi : \mathcal{P}(\mathbb{N}) \rightarrow^{\mathbb{N}} 2$   $A \mapsto \chi_A$   $\psi : 2 \rightarrow \mathcal{P}(\mathbb{N})$   $f \mapsto \{i \in \mathbb{N} \mid f(i) = 1\}$

$\psi \cdot \varphi = 1_{\mathcal{P}(\mathbb{N})}$   $\varphi \cdot \psi = 1_{P(\mathbb{N})}$

¿La potencia de los naturales es biyectable con las sucesiones binarias?  $\mathbb{N} \sim \mathcal{P}(\mathbb{N})$  ¿Los naturales y la potencia de los naturales no son biyectables?  $\mathbb{N} \sim \mathbb{N}_2$

Si fueran biyectables podrías tener:  $f : \mathbb{N} \rightarrow^{\mathbb{N}} 2$  Si f fuera suprayectiva entonces toda sucesión binaria aparecería en la lista (sería imagen de alguien).

$g : \mathbb{N} \rightarrow 2$   $g(i) = 1 - f_i(i)$   $g = (1, 0, 1, 1, \dots)$

Podemos llegar a contradicción porque:  $\exists n \in \mathbb{N}$   $f_n(n) \neq g(n) = 1 - f_n(n)$ ! Por lo tanto,  $\mathbb{N} \not\sim \mathcal{P}(\mathbb{N})$

$\mathbb{N} \sim \mathbb{N}_2$

"Nadie podrá expulsarnos de paraíso que Cantor ha creado"-Hilbert.

**Prop.** Existen lenguajes no reconocibles.

Dem: Si  $\Sigma$  es contable, entonces  $\Sigma^*$  es numerable. Sea  $\mathcal{M} = \{ \langle M \rangle \mid M \text{ es una MT} \} \subset \Sigma^*$   
 $|\mathcal{M}| \leq \aleph_0$

Si  $\mathcal{L}$  es el conjunto de todos los lenguajes sobre el alfabeto  $\Sigma$ , afirmamos que  $|\mathcal{L}| = |\mathbb{N}_2|$

Por tanto,  $|\mathcal{M}| < |\mathcal{L}|$  Es decir hay más lenguajes que MT. Así, debe haber lenguajes no reconocibles por MT alguna.

$\varphi : \mathcal{L} \rightarrow \mathbb{N}_2$  Sea  $S_0, S_1, S_2, \dots$  el orden canónico de  $\Sigma^*$ , y sea  $\mathcal{L}$  un lenguaje sobre  $\Sigma^*$  ■

¿En general  $S \sim P(S)$ ? No

Sea  $f : S \rightarrow P(S)$ . Veamos que f no es suprayectiva. Proponemos  $T = \{s \in S \mid s \notin f(s)\}$  Si  $s \in T$  entonces  $s \notin f(s)$  ( $T \cup (S - T) = S$ ) Si  $s \notin T$  entonces  $s \in f(s)$  Por tanto  $T \neq f(s)$   $T \notin \text{Im}(f)$  ■

$A, E, EQ$   $A_{TM} = \{ \langle M, w \rangle \mid M \text{ es una MT que acepta a la cadena } w \}$

**Obs.**  $A_{TM}$  es reconocible. U = Con entrada  $\langle M, w \rangle$  donde M es una MT

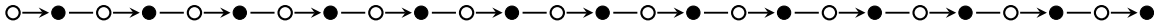
1. Simula a M con entrada w
2. Si M acepta, acepta, si M rechaza, rechaza

**Prop.**  $A_{TM}$  no es decidable Dem. Supongamos que sí, y sea R un decididor de  $A_{TM}$   $R(\langle M, w \rangle) = \{ \text{acepta} \mid M \text{ acepta a } w, \text{rechaza} \mid M \text{ rechaza a } w \}$  Construimos una MT D que usa a R como subrutina D = "Con entrada  $\langle M \rangle$  dada M es una MT:



¿Realmente necesitas imprimir esta hoja?





1. Ejecuta R con entrada  $\langle M, \langle M \rangle \rangle$

2. Si R acepta, rechaza; si R rechaza, acepta

$D(\langle M \rangle) = \{\text{aceptasi M no acepta a } \langle M \rangle; \text{rechazasi M acepta a } \langle M \rangle\}$

$D(\langle D \rangle) = \{\text{aceptasi D rechaza a } \langle D \rangle; \text{rechazasi D acepta a } \langle D \rangle\}$ ! D acepta a D si y sólo si D rechaza a D! ■

**Def.:** Un lenguaje L sobre  $\Sigma$  es co-reconocible si  $\Sigma(\Sigma^* - L)$  es reconocible.

**Prop.** Un lenguaje es decidable si y sólo si es reconocible y co-reconocible.

Dem: Sea A un lenguaje. Si A es decidable, entonces es reconocible. Sea M un decididor para A. M = "Con entrada w:

1. Ejecuta M con entrada w

2. Si M acepta, rechaza Si M rechaza, acepta

$L(M') = A$  Si A es reconocible y co-reconocible. Proponemos una MT M que decide a A que usa a  $M_1$  y  $M_2$  que reconocen a A y  $\bar{A}$  respectivamente. M = "Con entrada w:

1. Simula en paralelo a  $M_1$  y  $M_2$  con entrada w

2. Si  $M_1$  acepta, acepta

3. Si  $M_2$  acepta, rechaza ■

**Cor:** El complemento de  $A_{TM}$  no es reconocible.

Dem:  $A_{TM}$  es reconocible. Si su complemento fuera reconocible, entonces  $A_{TM}$  sería decidable pero no lo es ■

$H_{TM} = \{\langle M, w \rangle \mid M \text{ es una MT y M se detiene con entrada } w\}$

**Prop:**  $H_{TM}$  es indecible

Dem: Supongamos que R decide  $H_{TM}$  Proponemos la MT S, dada por: S = " Con entrada  $\langle M, w \rangle$ , donde M es una MT:

1. Ejecuta a R con entrada  $\langle M, w \rangle$

2. Si R rechaza, rechaza

3. Simula a M con entrada w

4. Si M acepta, acepta, si no, rechaza"

Claramente, S decide a  $A_{TM}$ ! ■

$E_{TM} = \{\langle M \rangle \mid L(M) = \emptyset\}$

**Prop**  $E_{TM}$  no es decidable.

Dem. Por contradicción. Supongamos que sí es decidable y sea R un decididor para  $E_{TM}$ . Construimos a S un decididor para  $A_{TM}$  dado por: S : "Con entrada  $\langle M, w \rangle$  donde M es una MT:

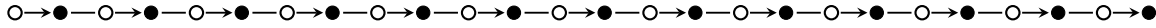
1. Construye a la MT  $M_1$  dada por:  $M_1$  = "Con entrada x:

(a) Si  $x \neq w$ , rechaza

(b) Si  $x = w$ , corre a M con entrada w y acepta si M lo hace".







2. Ejecuta R con entrada  $\langle M_1 \rangle$
3. Si R acepta, rechaza. Si R rechaza, acepta.

El lenguaje de  $M_1$  es  $|w|$  o  $\emptyset$  dependiendo de si M acepta o no a w respectivamente. Por tanto, S es un decidor para  $A_{TM}$  ■

Entonces  $E_{TM}$  no es decible

$REG_{TM} = \{ \langle M \rangle \mid M \text{ es una MT y } L(M) \text{ es regular} \}$

**Prop.**  $REG_{TM}$  es indecidible

Dem: Por contradicción. Sea R un decidor para  $REG_{TM}$  Proponemos la MT S dada por: S = " Con entrada  $\langle M, w \rangle$ , donde M es una MT:

1. Construye a la MT  $M_2$  dada por:  $M_2 = "$  Con entrada x:
  - (a) Si x es de la forma  $0^2$  acepta
  - (b) Si no, ejecuta M con entrada w y acepta a x si M acepta a w".
2. Ejecuta R con entrada  $\langle M_2 \rangle$
3. Si R acepta, acepta. Si R rechaza, rechaza.

Como entonces S decide a  $A_{TM}$  ■

## 2.1 Reducibilidad

Tenemos dos problemas: Problema 1 (Entrada:  $I_1$  Q:  $Q_1$ ) Problema 2(Entrada:  $I_2$  Q:  $Q_2$ )

Si tenemos un algoritmo  $M_1$  para resolver Prob 2, Y un algoritmo  $M_2$  que tiene una instancia  $I_1$  del Prob 1 y la transforma en una instancia  $I_2$  del Prob 2, de tal forma que  $I_2$  es una sí-instancia del Prob. 2 si y sólo si  $I_1$  es una sí-instancia del Prob 1, entonces

Proponemos  $M_3$  que resuelve Prob 1.

$I_1 \xrightarrow{M_2} I_2$  Si  $M_1$  responde sí, tenemos que  $I_2$  es una sí-instancia  $\rightarrow I_1$  es una sí-instancia Si  $M_1$  responde no, tenemos que  $I_2$  es una no-instancia  $\rightarrow I_1$  es una no-instancia

Si Prob. 1 se reduce a Prob. 2 y hay un algoritmo para resolver Prob. 2, entonces hay un algoritmo para resolver el problema 1.

Si sabemos que no hay un algoritmo para resolver Prob. 1 y podemos reducir Prob. 1 a Prob. 2, entonces no hay un algoritmo para resolver al Prob. 2

Problema 1  $E_{TM}$  Entrada:  $\langle M \rangle$  con M una MT Pregunta:  $L(M) = \emptyset$ ?

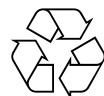
Problema 2  $EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ y } M_2 \text{ son MT y } L(M_1) = L(M_2) \}$  Entrada:  $\langle M_1, M_2 \rangle$  con  $M_1$  y  $M_2$  MT Pregunta:  $L(M_1) = L(M_2)$ ?

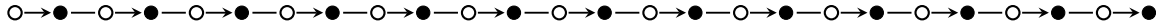
Vamos a mostrar que el problema 1 se reduce al problema 2  $\langle M \rangle \rightarrow^A \langle M, M_2 \rangle$  donde  $M_2$  es una MT que rechaza a todas sus entradas. Si  $L(M) = \emptyset$  entonces  $L(M) = L(M_2)$  Si  $L(M) \neq \emptyset$  entonces  $L(M) \neq L(M_2)$

**Prop.**  $EQ_{TM}$  es indecidible.

Dem: Por contradicción. Suponemos que R es una MT que decide a  $EQ_{TM}$ . Proponemos una MT S. S = "Con entrada  $\langle M \rangle$  donde M es una MT:

1. Ejecuta R con entrada  $\langle M, M_2 \rangle$  donde  $M_2$  es una MT que rechaza todas las entradas
2. Si R acepta, acepta. Si R rechaza, rechaza. Entonces si R acepta a  $\langle M, M_2 \rangle$  que dado que  $L(M_2) = \emptyset$ , entonces  $L(M) = \emptyset$ . Si R rechaza a  $\langle M, M_2 \rangle$  entonces  $L(M) \neq \emptyset$  luego, S acepta a  $\langle M \rangle$  si y sólo si  $L(M) \neq \emptyset$ . Por lo tanto S decide a  $E_{TM}$ !





**Def.** Una función  $f : \Sigma^* \rightarrow \Sigma^*$  es computable si existe una MT M que con cualquier entrada  $w$  se detiene con una reducción  $f(w)$  en su cinta

**Def.** Un lenguaje A es reducible si por funciones (muchas o una) al len. B,  $A \leq_M B$  si existe una función computable  $f: \Sigma^* \rightarrow \Sigma^*$  tal que para cada  $w \in \Sigma^*$ ,  $w \in A$  si y solo si  $f(w) \in B$

**Teo.** Si  $A \leq_M B$  y B es decidable, entonces A es decidable.

Dem: Sea M un decididor para B y sea f la reducción de A a B. Proponemos a N. N = "Con entrada w:

1. Calcula  $f(w)$
2. Ejecuta M con entrada  $f(w)$
3. Responde lo mismo que M "

Si  $w \in A$ , por la def. de reducción,  $f(w) \in B$ , y por tanto M acepta a  $f(w)$  y N acepta a w. Si  $w \notin A$ ,  $f(w) \notin B$ , y M rechaza a  $f(w)$  y N rechaza. Luego, N decide a A ■

**Cor** Si  $A \leq_M B$  y A no es decidable, entonces B no es decidable.

**Prop**  $H_{TM}$  es indecible.

Dem: Veamos que el problema del paro se reduce al problema de la aceptación  $A_{TM} \leq H_{TM}$   
Proponemos a F tal que  $F = \text{"Con entrada } \langle M, w \rangle \text{ donde M es una MT:}$

1. Construye la MT M' dado que: M' = "Con entrada x:
  - (a) Corre a M con entrada x
  - (b) Si M acepta, acepta
  - (c) Si M rechaza, se cicla
2. Devuelve  $\langle M', w \rangle$

Si  $\langle M, w \rangle \in A_{TM}$ , entonces  $\langle M', w \rangle \in H_{TM}$ .

Si  $\langle M, w \rangle \notin A_{TM}$ , entonces  $\langle M', w \rangle \notin H_{TM}$  ■

$A \leq_M B$  B decible entonces A decible A indecible entonces B indecible

$A_{TM} \leq H_{TM}$

$E_{TM} \leq EQ_{TM}$   $\langle M \rangle \rightarrow \langle M, M_2 \rangle$ ,  $M_2$  rechaza todas las codificaciones

$\langle M \rangle \in E_{TM} \leftrightarrow L(M) = \emptyset \leftrightarrow L(M) = L(M_2) \leftrightarrow \langle M_1, M_2 \rangle \in EQ_{TM}$

$E_{TM}$  es indecible  $\overline{E_{TM}}$  es indecible  $\langle M, w \rangle M_1 = \text{"Con entrada x}$

1. Si  $x \neq w$ , rechaza
2. Ejecuta M con entrada w
3. Si M acepta, acepta

$L(M_1) = \emptyset$  si  $w \notin L(M)$  |  $w \in L(M)$

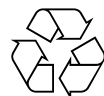
$\langle M, w \rangle \rightarrow \langle M_1 \rangle$

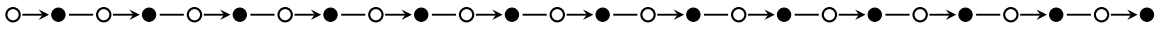
$\langle M_1, w \rangle \in A_{TM} \leftrightarrow M \text{ acepta a } w \leftrightarrow L(M_1) \neq \emptyset \leftrightarrow \langle M_1 \rangle \notin E_{TM}$

**Obs** L es decible si y sólo si  $\overline{L}$  complemento es decible.

Dem: Si R decide a L proponemos a S dada por: S = "Con entrada w

1. Ejecuta R con entrada w
2. Si R acepta, rechaza y si R rechaza, acepta





**Obs** Si  $A \leq_M B$  entonces  $\overline{A} \leq_M \overline{B}$ . Dem:  $f$  computable tal que  $w \in A$  si y sólo si  $f(w) \in B$  luego  $w \notin A$  si y sólo si  $f(w) \notin B$

**Obs** Si  $A \leq_M B$  y  $B$  es reconocible, entonces  $A$  es reconocible.  $R$  es un reconocedor  $f(w) \in B$  si y sólo si  $w \in A$ .  $S =$  "Con entrada  $w$

1. Calcular  $f(w)$
2. Ejecutar  $R$  con entrada  $f(w)$
3. Responder lo mismo que  $R$

$S(w) = R(f(w))$   
 $\overline{A_{TM}} \leq \overline{E_{TM}} \Rightarrow \overline{A_{TM}} \leq E_{TM}$   
 $\overline{E_{TM}}$  es reconocible  
 $\overline{A_{TM}} \leq_M \overline{E_{TM}} \stackrel{!}{\Rightarrow} A_{TM} \leq_M E_{TM}$ ? No  
 $\overline{A_{TM}} \leq_M \overline{E_{TM}} \Rightarrow \overline{E_{TM}}$  no es reconocible !

**Prop**  $EQ_{TM}$  no es reconocible ni co-reconocible.

$A_{TM} \leq EQ_{TM}$   $A_{TM} \leq EQ_{TM}$  Dem:  $F =$  "Con entrada  $\langle M, w \rangle$ , donde  $M$  es una MT

1. Construimos las MT  $M_1$  y  $M_2$  dadas por  $M_1 =$  "Con entrada  $x$ :

(a) Rechaza

$M_2 =$  "Con entrada  $x$ :

(a) Ejecuta a  $M$  con entrada  $w$ . Si  $M$  acepta, acepta

2. Devuelve  $\langle M_1, M_2 \rangle$

Notemos que  $L(M_1) = \emptyset$  y  $L(M_2) = \Sigma^*$  si  $M$  acepta a  $w$  o  $\emptyset$  si  $M$  no acepta a  $w$ .

Si  $\langle M, w \rangle \in A_{TM}$ , entonces  $L(M_2) = \Sigma^*$  y  $\langle M_1, M_2 \rangle \in \overline{EQ_{TM}}$ .

Si  $\langle M_1, M_2 \rangle \in \overline{EQ_{TM}}$ , entonces  $L(M_2) \neq L(M_1)$ , entonces  $L(M_2) = \Sigma^*$  y por tanto  $M$  acepta a  $w$  y  $\langle M, w \rangle \in A_{TM}$

Luego  $F$  calcula una reducción  $A_{TM} \leq_M \overline{EQ_{TM}}$

Si hacemos que  $M_1$  acepte en lugar de rechazar, tenemos que: Si  $\langle M, w \rangle \in A_{TM}$ , entonces  $L(M_2) = \Sigma^*$  y  $\langle M_1, M_2 \rangle \in EQ_{TM}$ .

Si  $\langle M_1, M_2 \rangle \in EQ_{TM}$ , entonces  $L(M_2) \neq L(M_1)$ , entonces  $L(M_2) = \Sigma^*$  y por tanto  $M$  acepta a  $w$  y  $\langle M, w \rangle \in A_{TM}$

### 3 Complejidad

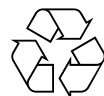
Vamos a trabajar únicamente con máquinas de Turing decidibles

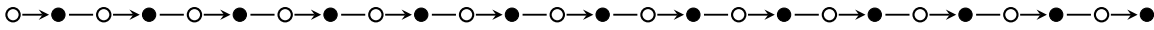
Sea  $M$  una MT que se detiene en todas sus entradas. La complejidad (en tiempo) de  $M$  es la función  $f : \mathbb{N} \rightarrow \mathbb{N}$  tal que  $f(n)$  es el mejor tiempo que puede tardarse  $M$  en ejecutarse con una entrada de longitud  $n$ . Si  $f(n)$  es el tiempo de ejecución de  $M$ , diremos que  $M$  corre en tiempo  $f(n)$  o que  $M$  es de tiempo  $f(n)$ .

Sea  $t : \mathbb{N} \rightarrow \mathbb{R}$  una función. La clase de complejidad en tiempo  $\text{TIME}(t(n))$  es el conjunto de todos los lenguajes decidibles en tiempo  $\mathcal{O}(t(n))$  por una MT

$L = \{0^n 1^n\}$   $n \in \mathbb{N}$   $M =$  "Con entrada  $w$ :

1. Recorre la cinta y si encuentra un 1 a la derecha de un 0, rechaza ( $\mathcal{O}(n)$ )
2. Repite mientras haya 0 y 1 en la cinta





3. Recorre la cinta(entrada) y borra el primer 0 y el primer 1. ( $\mathcal{O}(n) \cdot \frac{n}{2}$ )

4. Si sobra algún 0 o algún 1 en la cinta rechaza. Si no, acepta". ( $\mathcal{O}(n)$ )

$M \in TIME(n^2)$  M es de tiempo  $\mathcal{O}(n^2)$   $L \in TIME(t(n))$

$TIME(n) \subseteq TIME(n^2)$

$M_2$  (Máquina de Turing con 2 cintas)= "Con entrada w:

1. Recorre la entrada y rechaza si hay un 0 a la derecha de 1. ( $\mathcal{O}(n)$ )

2. Repite mientras haya 0 y en 1 en la cinta ( $\mathcal{O}(\log n)$ )

3. Revisa si el número de 0 y 1 es par o impar. Si es impar, rechaza ( $\mathcal{O}(n)$ )

4. Recorre la cinta borrando un 0 si y uno no y un 1 si y uno no, empezando por el primero. ( $\mathcal{O}(n)$ )

5. Si hay 0 o 1 en la cinta, rechaza. Si no, acepta" ( $\mathcal{O}(n)$ )

$\mathcal{O}(n) + \mathcal{O}(\log n) \cdot 2 \cdot \mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n \log n)$

$M_2$  corre en tiempo  $\mathcal{O}(n \log n)$  ó  $\Theta(n \log n)$

$L \in TIME(n \log n)$   $L \in TIME(n^2)$

$M_3$  (Máquina de Turing con 3 cintas)= "Con entrada w:

1. Recorre la entrada y copia los 0 en la cinta 2 y los 1 en la cinta 3 ( $\mathcal{O}(n)$ )

2. Repite mientras haya 0 en la cinta 2 y 1 en la cinta 3 ( $\mathcal{O}(n)$ )

3. Borra un 0 en la cinta 2 y un 1 en la cinta 3. ( $\mathcal{O}(1)$ )

4. Si los 0 se acaban antes que los 1, o viceversa, rechaza. De otro modo acepta". ( $\mathcal{O}(n)$ )

$M_3$  corre en tiempo  $\mathcal{O}(n)$

$L \in TIME(n \log n)$  Cambiando el modelo  $\rightarrow L \in TIME(n)$

**Teo** Sea  $t(n)$  una función con  $t(n) \geq n$  Entonces toda MT multicinta de tiempo  $t(n)$  tiene una MT de una sola cinta equivalente y de tiempo  $\mathcal{O}(t(n)^2)$

Dem: Utilizamos la simulación de una MT multicinta M con una MT de una sola cinta S. Para simular un paso de M, S recorre la entrada y ubica el símbolo que lee cada cabezal virtual; esto lleva tiempo la longitud de la cadena actualmente en su cinta. Esta longitud es la suma de las longitudes de los contenidos de todas las cintas de M. Cada cinta de M contiene a lo más  $t(n)$  símbolos (pues se ejecuta por a lo más tiempo  $t(n)$ ) En total, este paso se lleva tiempo  $k \cdot t(n) \in \mathcal{O}(t(n))$  Después, recorre nuevamente la cinta, actualizando las acciones de M. Quizá, haciendo k shifts en el proceso. Este recorrido junto con los shifts se lleva tiempo  $k \cdot \mathcal{O}(t(n))$  que es  $\mathcal{O}(t(n))$ . Así, S se tarda  $\mathcal{O}(t(n))$  en simular un paso de M, y tiene que simular  $t(n)$  pasos, luego el tiempo total es  $\mathcal{O}(t(n)^2)$  ■.

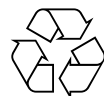
N es una MTN  $t(n) \rightarrow$  existe M MTD equivalente  $\mathcal{O}(2^{t(n)})$

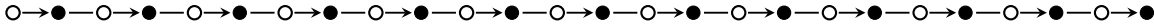
Toda MT multicinta  $M_1$  tiene una MT  $M_2$  de una cinta tal que si  $M_1$  corre en un tiempo  $t(n)$ , entonces  $M_2$  es de tiempo  $\mathcal{O}(t(n)^2)$

**Def** Sea  $t: \mathbb{N} \rightarrow \mathbb{N}$  tal que  $t(n) \geq n$  para cada  $n \in \mathbb{N}$ . Decimos que la MTN N corre un tiempo  $t(n)$  si  $t(n)$  es el máximo número de pasos que debe realizar N para detenerse con una entrada de longitud n. N es un decidor (Todas las ramas terminan).

M simula a N

1. cinta w - entrada de N





2. cinta de trabajo
3. generar dirección

**Teo** Si  $N$  es una MTN de tiempo  $t(n)$  que es un decidor, donde  $t(n) \geq n$  p.c.  $n \in \mathbb{N}$ , entonces existe una MTD,  $D$ , tal que  $D$  es equivalente a  $N$  y es de tiempo  $2^{\mathcal{O}(t(n))}$ .

**Dem:** Consideremos la simulación de  $N$  por una MTD de 3 cintas vista anteriormente. Con una entrada de longitud  $n$ , cada rama de la ejecución no determinista de  $N$  tiene longitud a lo más  $t(n)$ . Además cada nodo en el árbol tiene a lo más  $b$  hijos. Hay a lo más  $b^{t(n)}$  hojas en el árbol y a lo más, el mismo número de vértices internos, i.e. hay  $\mathcal{O}(b^{t(n)})$  vértices en el árbol de ejecución de  $N$ . El tiempo que toma a  $M$  iniciar en la raíz y simular una rama de la ejecución de  $N$  hasta un vértice dado, es a lo más  $t(n)$ . Luego, hay  $\mathcal{O}(b^{t(n)})$  vértices en el árbol y simular a  $M$  para llegar a cada uno de ellos en tiempo  $t(n)$ , i.e., la simulación completa toma tiempo  $t(n) \cdot \mathcal{O}(b^{t(n)}) = \mathcal{O}(b^{t(n)}) = \mathcal{O}(2^{t(n)}) = 2^{\mathcal{O}(t(n))}$ . Como  $M$  tiene 3 cintas, la podemos simular con una MT de una sola cinta en tiempo  $2^{2\mathcal{O}(t(n))} = 2^{\mathcal{O}(2t(n))} = 2^{\mathcal{O}(t(n))}$  ■

**Def** La clase  $P$  es la clase de lenguajes decible por una MTD de una sola cinta en tiempo polinomial, i.e.  $P = \bigcup_{t \in \mathbb{N}} TIME(n^t)$

**Def** La clase  $NP$  es la clase de lenguajes decible por una MTN de una sola cinta en tiempo polinomial, i.e.  $NP = \bigcup_{t \in \mathbb{N}} NTIME(n^t)$

$NTIME(t(n))$  = Todos los lenguajes  $L$  tales que hay una MTN de tiempo  $t(n)$  que decide a  $L$ .

$NP$  es Non-deterministic Polynomial, i.e. una MT no determinista lo puede decidir en tiempo Polinomial.

$PATH\{< D, s, t > \mid \text{hay una } st\text{-trayectoria (camino que no repite vértices) dirigida en } D\}$

**Prop:**  $PATH \in P$

**Dem:** Proponemos a la MT  $M$  dada por  $M = \text{"Con entrada } \langle D, s, t \rangle \text{ donde } D \text{ es una digráfica y } s \text{ y } t \text{ son vértices de } D:$

1. Marca  $S$ .
2. Repite hasta que no marquen vértices nuevos.
3. Revisa cada flecha de  $D$ . Si hay alguna cuya cola esté marcada y cuya cabeza no esté marcada, marca la cabeza.
4. Si  $t$  está marcado, acepta. Si no, rechaza

Los pasos 1 y 4 se ejecutan una única vez y toman tiempo  $\mathcal{O}(n)$ . A lo más, el paso 3 se repite  $|V|(es\mathcal{O}(n))$  veces. Cada ejecución de 3 es  $\mathcal{O}(n)$ . Luego, todas las ejecuciones de 3 se pueden hacer en  $\mathcal{O}(n^2)$ .

Así,  $M$  es de tiempo  $\mathcal{O}(n^2)$  ■

$RELPRIME = \{ \langle x, y \rangle \mid x \text{ y } y \text{ son primos relativos} \}$

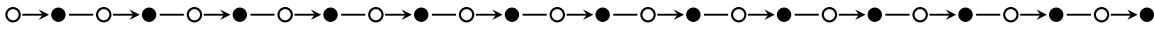
**Prop:**  $RELPRIME \in P$

**Dem:** Proponemos a  $E$  una MT dada por

$E = \text{"Con entrada } \langle x, y \rangle \text{ donde } x \text{ y } y \text{ son enteros positivos}$

1. Repite hasta que  $y = 0$
2. Asigna  $x \leftarrow x(mod y)$
3. Intercambia  $x$  y  $y$
4. Devuelve  $x$





Proponemos a M dada por:

M = "Con entrada  $\langle x, y \rangle$  donde  $x$  y  $y$  son enteros positivos

1. Ejecuta E con entrada  $\langle x, y \rangle$
2. Si E devuelve 1 acepta. Si no, rechaza

Como cada paso individual se puede hacer en tiempo polinomial, basta ver cuantas veces se repiten los pasos 2 y 3.

Si  $\frac{x}{2} \geq y$ ,  $x \bmod y < y \leq \frac{x}{2}$  Si  $\frac{x}{2} \leq y$ ,  $x \bmod y = x - y < \frac{x}{2}$

Cada dos vueltas del ciclo, el valor de  $x$  y el de  $y$  disminuyen a la mitad. Luego, estos pasos se realizan  $\mathcal{O}(\log(n))$  veces ■

$EQ_{LLC}$  es indecidible.

Sí-certificado. Algoritmo para verificar el certificado (simple).

Un *verificador* para el lenguaje A es una MT decididora V tal que  $A = \{w | V \text{ acepta } \langle w, c \rangle \text{ p.c. cadena } c\}$  La complejidad de V se mide en términos de la longitud de  $w$ , luego, un verificador polinomial para A es un verificador que corre en tiempo polinomial respecto a  $|w|$ .

Ejemplo:  $L = \{\langle G \rangle | \text{G es k-coloreable}\}$  V = "Con entrada  $\langle G, c \rangle$  donde G es una gráfica y c es una k-coloración de G:

1. Revisa que cada vértice aparezca una única vez en c, si no, rechaza  $\mathcal{O}(|V|^2)$
2. Para cada arista e de G, verifica si los extremos de e tienen colores distintos. Si no, rechaza  $\mathcal{O}(|E|)$
3. Revisa que haya a lo más k-colores en c. Si no, rechaza.  $\mathcal{O}(|V|)$
4. Acepta

El certificado es una cadena, verificador es un algoritmo.

Ejemplos de problemas con sus verificadores:

k-col  $\langle G, \text{coloración} \rangle$

ciclo hamiltoniano  $\langle G, \text{ciclo hamiltoniano} \rangle$

graf conexa  $\langle G, \text{árbol generador} \rangle$

sat  $\langle \phi, \tau(\text{asignación de verdad}) \rangle$

**Teo** Un lenguaje L está en NP si y sólo si tiene un verificador polinomial.

**Def** Un verificador para un lenguaje A es una MT decididora tal que  $A = \{w | V \text{ acepta } \langle w, c \rangle \text{ p.a. cadena } c\}$

HAMPATH =  $\{\langle D, s, t \rangle | \text{Des una digráfica, s y t vértices de D y tales que existe una s-t trayectoria hamiltoniana}\}$

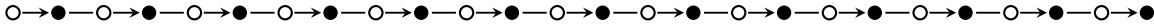
**Prop.** HAMPATH  $\in NP$

Dem: Proponemos una MTN N tal que:

N = "Con entrada  $\langle D, s, t \rangle$  donde D es una digráfica,  $s, t \in V_D$ :

1. Escribe una lista de enteros  $p_1, \dots, p_{|V|}$  cada número en la lista se elige de manera no determinista entre 1 y  $|V|$   $\mathcal{O}(|V|)$
2. Si hay repeticiones en la lista rechaza  $\mathcal{O}(|V|^2)$
3. Revisa si  $p_1 = s$  y  $p_{|V|} = t$ , si no, rechaza  $\mathcal{O}(|V|)$
4. Para cada i entre 1 y  $|V|-1$ , revisa que  $(p_i, p_{i+1})$  sea una flecha en D. Si no rechaza  $\mathcal{O}(|V|^t)$   $t \in \mathbb{N}$





### 5. Acepta

N corre en tiempo polinomial respecto a  $|V|$ , por lo tanto  $\text{HAMPATH} \in \text{NP}$  ■

**Teorema** Un lenguaje L está en NP si y sólo si tiene un verificador polinomial

Dem: Si V es un verificador polinomial para L, proponemos la MTN N que decide a L dada por: Supongamos que V es de tiempo  $n^t$  N: "Con entrada w de longitud n:

1. Elige de manera no determinista una cadena c de longitud a lo más  $n^t$
2. Ejecuta V con entrada  $\langle w, c \rangle$
3. Si V acepta, acepta. Si no, rechaza  $\mathcal{O}(n^t)$

Si L es decidido por la MTN N y la rama de la ejecución no determinista está codificada por C, entonces proponemos a V dada por:  $V = \text{"Con entrada } \langle w, c \rangle$

1. Simula a N con entrada w usando cada símbolo de c como la siguiente elección en la ejecución no determinista de N que acepta a w.  $\mathcal{O}$  = El tiempo que tarda N en decidir a w
2. Si esta rama de la ejecución de N acepta, acepta. Si no, rechaza ■  $\mathcal{O}(1)$

$L \in \text{NP}$   $L = \{w \mid \exists V \text{ acepta } \langle w, c \rangle \text{ para alguna cadena } c\}$  V verificador polinomial para L:

1. "Adivinar" una solución
2. Verificar que la solución es correcta

Si G es una gráfica, un clan de G es un subconjunto C de V tal que  $G[C]$  (gráfica inducida) es completa. Un k-clan es un clan de cardinalidad k. El máximo entero k tal que G tiene un k-clan es el número de clan de G  $\omega_G$

CLAN =  $\{\langle G, k \rangle \mid G \text{ es una gráfica con un k-clan}\}$

(Mapping reducibility) Existe una MT que computa  $f: \Sigma^* \rightarrow \Sigma^*$  en tiempo polinomial,  $\mathcal{O}(n^k)$  tal que  $A \leq_m B$ ,  $w \in A$  si y sólo si  $f(w) \in B$ . M decide B en tiempo polinomial. M se ejecuta con entrada  $f(w)$ , en  $|f(w)|$  en  $\mathcal{O}(n^k)$ , luego su tiempo de ejecución es  $\mathcal{O}(|f(w)|^l) = \mathcal{O}((n^k)^l) = \mathcal{O}(n^{k \cdot l})$  F con entrada w computa  $f(w)$  en tiempo  $\mathcal{O}(|w|^k) = \mathcal{O}(n^k)$

**Def.** Sean A y B lenguajes. Decimos que A es reducible en tiempo polinomial mediante funciones a B,  $A \leq_p B$  si existe una función computable en tiempo polinomial  $f: \Sigma^* \rightarrow \Sigma^*$  tal que, para cada  $w \in \Sigma^*$  se tiene:  $w \in A$  si y sólo si  $f(w) \in B$ . Se dice que f es una reducción polinomial de A en B.

**Prop.** Si  $A \leq_p B$  y  $B \in P$ , entonces  $A \in P$  Dem. Sea M la MT que decide a B en tiempo polinomial, y sea f la reducción polinomial de A en B. Proponemos a S una MT que decide a A, dada por: S: "Con entrada w:

1. Calcula  $f(w)$
2. Ejecuta M con entrada  $f(w)$  y responde lo mismo que M".

Para ver que S decide a A, consideremos  $w \in A$ . Entonces  $f(w) \in B$  y por lo tanto M acepta a  $f(w)$ . Luego, S acepta a w. Recíprocamente si S acepta a w, entonces M acepta a  $f(w)$ , i.e.  $f(w) \in B$  y como f es una reducción,  $w \in A$ . Se sigue que S decide a A. Claramente el paso 1 se realiza en tiempo polinomial or la definición de reducción polinomial y el paso 2 también, pues la composición de polinomios es un polinomio. (Si  $f(w)$  es  $\mathcal{O}(n^k)$  M con entrada  $f(w)$  tarda  $\mathcal{O}(n^k)^l = \mathcal{O}(n^{k \cdot l})$ )

Si  $A \leq_p B$  y  $A \notin P$  entonces  $B \notin P$

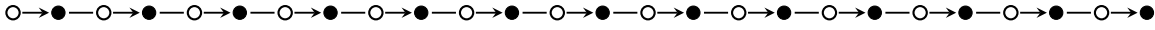
$\text{SAT} = \{\langle \varphi \rangle \mid \varphi - \varphi \text{ es una fórmula booleana satisfacible}\}$

Una literal es una variable proporcional o su negación.

Una clausula es una disyunción con literales.

Una fórmula booleana está en FNC si es una conjunción de disyunciones.





$SAT_{FNC} = \{ \langle \varphi \rangle \mid \varphi \text{ es una función booleana en FNC satisfacible} \}$

$SAT_{FNC} \leq_P SAT$

Una fórmula está en 3-FNC si es una conjunción de clausulas de tamaño 3.

$3SAT = \{ \langle \varphi \rangle \mid \varphi \text{ es una fórmula booleana satisfacible en 3-FNC} \}$

$CLAN = \{ \langle G, k \rangle \mid G \text{ es una gráfica con un clan de cardinalidad } k \}$

**Prop.**  $3SAT \leq_P CLAN$  Modela el problema de la izquierda con el de la derecha. Sea  $\varphi \in 3FNC$ , entonces  $\varphi = \bigcap_{i=1}^k (a_i \cup b_i \cup c_i)$  ( $f(\langle \varphi \rangle) = \langle G, k \rangle$ ) Para cada clausula en  $\varphi$  creamos 3 vértices en  $G$ , etiquetados con las literales correspondientes en la clausula ( $V = \bigcup_{i=1}^k \{a_i, b_i, c_i\}$ ) Cualesquiera dos vértices de  $G$  son adyacentes, excepto:

1. Vértices correspondientes a la misma tripleta (De la misma clausula)
2. Vértices con etiquetas que generan una contradicción, e.g.  $x_i$  y  $\overline{x_i}$

Veamos que  $\langle \varphi \rangle \in 3SAT$  si y sólo si  $\langle G, k \rangle \in CLAN$ .

Si  $\langle \varphi \rangle \in 3SAT$  existe  $\tau$  asignación de verdad que satisface a  $\varphi$ . Sea  $\{x_i\}_{i \in 1 \dots k}$  un conjunto de literales de  $\varphi$  tal que  $\tau(x_i) = 1$  y  $x_i$  pertenece a la  $i$ -ésima cláusula. Notemos que al recibir todas las  $x_i$  el valor 1 bajo  $\tau$ , tenemos que no es posible encontrar a una literal y a su negación en  $\tau$ . Se sigue que en  $G$  los vértices correspondientes a  $\{x_i\}_{i=1}^k$  es un clan de tamaño  $k$ . Luego  $\langle G, k \rangle \in CLAN$

Si  $\langle G, k \rangle \in CLAN$ , entonces  $G$  tiene un clan de tamaño  $k$ ,  $S \subseteq V$ . Por la construcción de  $G$  no hay dos vértices de  $S$  en la misma tripleta, y no hay dos vértices de  $S$  tales que sus etiquetas sean contradictorias, por lo tanto podemos asignar 1 a cada literal correspondiente a un vértice en  $S$ . Si llamamos  $\tau$  a alguna asignación obtenida al extender la asignación antes descrita a todas las literales de  $\varphi$ , entonces  $\tau$  satisface a  $\varphi$  (pues le asigna el valor 1 al menos una variable en cada clausula).

Así,  $f$  es una reducción  $|V| = 3k |E| \cdot O(k^2)$

$STABLE = \{ \langle G, E \rangle \mid G \text{ es una gráfica con un conjunto independiente de tamaño } k \}$

$CLIQUE =$  Todos son adyacentes dos a dos

$CLIQUE \leq_P STABLE$   $\langle G, k \rangle \mapsto \langle H, l \rangle$   $H = \overline{G}$   $l = k$

$\langle G, k \rangle \in CLIQUE$ , entonces  $G$  tiene un clan de tamaño  $k$ , digamos  $S$ . En  $\overline{G}$   $S$  es un conjunto independiente de tamaño  $k$ . Si  $\langle \overline{G}, k \rangle \in STABLE$  entonces  $\overline{G}$  tiene un conjunto independiente de tamaño  $k$  y por tanto  $\overline{G} = G$  tiene un clan de tamaño  $k$ , entonces  $\langle G, k \rangle \in CLIQUE$

**Def** Sea  $G$  una digráfica. Un subconjunto  $S \subseteq V$  es convexo si todos los vértices de cualquier geodésica (trayectoria de longitud mínima) entre pares de vértices distintos están contenidos en  $S$ . Usualmente, por conjunto convexo nos referimos a uno distinto de  $V$ .

$CONVEX = \{ \langle G, k \rangle \mid G \text{ tiene un conjunto convexo de tamaño } k \}$

$CLIQUE \leq CONVEX$   $\langle G, k \rangle \mapsto \langle H, l \rangle$

$H$  se obtiene de  $G$  al añadir dos vértices nuevos,  $x$  y  $y$ , y hacerlos adyacentes a cada vértice en  $G$ , pero no entre sí.

Si  $G$  tiene un  $k$ -clan  $S$ ,  $S \cup \{x\}$  es un clan, y por tanto es un conjunto convexo. Por tanto,  $H$  tiene un  $(k+1)$ -conjunto convexo

$S \subseteq V_H$  es conexa.

$S \subseteq V_G$

Afirmamos  $S$  es un clan. De otro modo,  $x \in S$  Si  $x \in S$ , entonces  $y \notin S$ . Nuevamente  $S$  es un clan, pues de otro modo  $y \in S$

Por lo tanto  $G$  tiene un  $k$ -clan ■

$|V_H| = |V_G| + 2$   $|E_H| = |E_G| + 2|V_G|$

$CLIQUE \leq_P CONVEX$

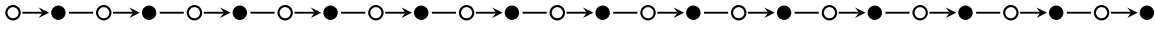
$DConvex = \{ \langle D, k \rangle \mid D \text{ es una digráfica con un conjunto convexo de tamaño } k \}$

**Prop.**  $CLIQUE \leq DConvex$

La gráfica será:







- Bipartita
- Sin flechas simétricas
- Con cuello 6 (ciclo más corto será de longitud 6)

Dem:

Gadgets Por cada vértice de la gráfica hay un hexágono y para cada arista de la gráfica se unirá el vértice  $x$  del primer hexágono al  $y$  del segundo hexágono y el  $x$  del segundo hexágono al  $y$  del primer hexágono.

La estructura de control será una trayectoria dirigida

$$|V_G| = 6|V_G| + 4 \quad |A_G| = 8|V_G| + 2|E_G| + 3$$

Obs.

- Si  $n \in V_G$  y  $C$  un convexo de  $D$   $|C| \geq 2$ , tal que  $C \cap V_{Hu} \neq \emptyset$ , entonces  $V_{Hu} \in C$
- Sea  $C$  un convexo de  $D_1$   $|C| \geq 2$ . Si  $z_i \in C$ , p.a.  $i \in \{1 \dots 4\}$  entonces  $C = V_0$
- Sean  $u, r \in V_G$  t.q.  $d(u, r) \geq 2$ , si  $C$  es un convexo en  $D$  t.q.  $V_{Hu} \cap C \neq \emptyset \neq V_{Hu} \cap C$ , ent.  $C = V_D$
- Si  $S$  es un clan de  $G$ , ent.  $C = \cup_{u \in S} Hu$  es un convexo en  $D$ . Si  $G$  tiene un  $k$ -clan, entonces  $D$  tiene un  $6k$  conjunto convexo.

Un lenguaje  $A$  es NP-completo si:

- $A \in NP$
- Para cada  $B \in NP$ ,  $B \leq_P A$

Si hubiera un algoritmo polinomial para resolver  $A$ , entonces tenemos un algoritmo polinomial para resolver cualquier problema en NP.

Si  $L$  es NP-c y tenemos al lenguaje  $A$  t.q.

- $A \in NP$
- $L \leq_P A$ , entonces  $A$  es NP-c

$4-col = \{ \langle G \rangle \mid G \text{ es una gráfica 4-coloreable} \}$

**Prop**  $4-col \leq_P SAT$

$G \mapsto \varphi$

$\varphi$  es satisfacible si y solo si  $G$  es 4-coloreable

$x_{vc}$  El vértice  $v$  recibe el color  $c$

$C = \{a, b, c, d\}$

$\varphi_{color} = \bigwedge_{v \in V_G} (x_{va} \wedge \overline{x_{vb}} \wedge \overline{x_{vc}} \wedge \overline{x_{vd}}) \vee (\overline{x_{va}} \wedge x_{vb} \wedge \overline{x_{vc}} \wedge \overline{x_{vd}}) \vee (\overline{x_{va}} \wedge \overline{x_{vb}} \wedge x_{vc} \wedge \overline{x_{vd}}) \vee (\overline{x_{va}} \wedge \overline{x_{vb}} \wedge \overline{x_{vc}} \wedge x_{vd})$

$\varphi_{propia} = \bigwedge_{uv \in E_G} (\neg(x_{ua} \wedge x_{sa}) \wedge \neg(x_{ub} \wedge x_{sb}) \wedge \neg(x_{uc} \wedge x_{sc}) \wedge \neg(x_{ud} \wedge x_{sd}))$

$\varphi = \varphi_{color} \wedge \varphi_{propia}$

Si  $G$  es 4-col, entonces  $\varphi$  satisfacible.  $\varphi$  satisfacible, entonces  $G$  es 4-col.

$k$ -coloracion se puede reducir a SAT en tiempo lineal

$A$  es NP-completo

$SubsetSum \in NP$

$SubsetSum \leq_P A$

$M$  decide a  $A$  en tiempo polinomial

**Teo** Cook-Levin SAT es NP-completo

Dem: Sea  $L \in NP$ , entonces existe  $N$  una MTN que decide a  $L$ , el tiempo de ejecución de  $N$  es

$\mathcal{O}(n^k)$  p.a.  $k \in \mathbb{Z}^+$



¿Realmente necesitas imprimir esta hoja?

