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 $f : A \to \{0, 1\}$ 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)
- Σ es un conjunto no vacío (alfabeto de entrada)
- Γ es un conjunto no vacío (alfabeto de cinta)
- $\delta : Q' \times \Gamma \to Q \times \{L, R\}$ 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 α da lugar a la configuración β $\alpha \to \beta$ y se define como sigue:

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

- $\alpha \to^* \beta$ si $\alpha \to^n \beta$ 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 $q_0 \to^n xq_ay$ 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 = "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 creo, 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 $f : D \to C$ $D \subseteq \Sigma^*$ $C \subseteq \Gamma^*$.
Def. Sea R una MT, k un natural y f una función parcial sobre $(\Sigma^*)^k$ con valor en Γ^*. Decimos que T computa a f si para cada $(x_1, ..., x_k)$ en el dominio de f, $q_0 \vdash x_1 \sqcup x_2 \sqcup ... \sqcup x_k \to^k q_0 \vdash f(x_1, ..., x_k)$ y ninguna otra entrada que sea una k-ada de cadenas es aceptada por T.
Una función parcial $f : (\Sigma^*)^k \to \Gamma^*$ 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). $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$.
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 ($\vdash x \to x \sqcup x$)
- Insert(σ)
- "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 → NB → R → PB → 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 $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$. $\delta(q_1, a_1, ..., a_k) = (q_1, b_1, ..., b_k, D_1, ..., D_k)$ $D_i \in \{L, R, S\}$.
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 Γ_M para denotar donde están los cabezales $\dot{o}$
S = Con entrada $w_1, ..., w_n$

1. Poner su cinta en el formato que representa las k cintas. La cinta formateada se ve como $| \vdash |w_1|...|w_n|\#| \sqcup |\#| \sqcup |...|$.
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:
$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R, S\})$
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 → G (Guess) → NB → G → PB → M (Multiply) → PB → 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 $\Sigma = \{1, ..., b\}$ 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 $\{1, .., b\}^*$
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 $x_1\#x_2\#...\#x_n\#x$ donde $x_i \in L$ p.c $1 \leq i \leq n$ y $x_1, ..., x_n, x$ 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 Σ^*
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 Σ^* 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 si en orden canónico y le agrega a la lista $s_0, s_1, ..., s_i$
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:

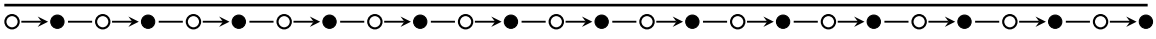
1. Ejecuta E. Cada vez que se imprima una cadena, la compara con w.
2. Si w se imprime en algún momento *acepta*."

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 Tésis Church-Turing

Turing, Church, Goëdel 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 conexa?
M= "Con entrada $< G >$ 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 $< M, w >$ 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

$A_{AFD} = \{< A, w > |$El AFD A 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 decidible.
Dem: Proponemos una MT M que decida a A_{AFD}
M = Con entrada $< A, w >$ 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 decidible
Dem: Proponemos una MTD N que decida a A_{AFN}
N = "Con entrada $< B, w >$ 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 ".

$A_{REX} = \{< R, w > |$R es una expresion regular que genera a w$\}$
Prop: A_{REX} es decidible
Dem: P = "Con entrada $< R, w >$ donde R es una regex y w una cadena:

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

$E_{AFD} = \{< A > |$A es un AFD con L(A) = $\emptyset\}$
Prop: E_{AFD} es decidible
Dem: T = "Con entrada $< A >$ 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, <u>*rechaza*</u>. Si no, <u>*acepta*</u>.

$EQ_{AFD} = \{< A, B > |$A y B son AFD tales que L(A)=L(B)$\}$
Prop: $C, L(C) = L(A) \triangle L(B)$
Dem: F = "Con entrada $< A.B >$ 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, <u>*acepta*</u>. Si no, <u>*rechaza*</u>■.

Función característica: El conjunto $S \subseteq A$ $\chi_S : A \to \{0,1\}$ $\chi_S(a) = 1$ si $a \in S$ 0 si $a \notin S$
$\omega = \{0, 1, 2, 3, ...\} = \mathbb{N}$ $(0, 1, 0, 1, ..)$ sería el conjunto de los impares.
$\mathcal{P}(\mathbb{N}) \sim^{\mathbb{N}} 2 = \{f : \mathbb{N} \to \{0, 1\}$
$\varphi : P(\mathbb{N}) \to^{\mathbb{N}} 2$ $A \mapsto \chi_A$ $\psi : 2 \to P(\mathbb{N})$ $f \mapsto \{i \in \mathbb{N} | f(i) = 1\}$
$\psi \cdot \varphi = 1_{P(\mathbb{N})}$ $\varphi \cdot \psi = 1_{P(\mathbb{N})}$
¿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: $f : \mathbb{N} \to^{\mathbb{N}} 2$ Si f fuera suprayectiva entonces toda sucesión binaria aparecería en la lista (sería imagen de alguien).
$g : \mathbb{N} \to 2$ $g(i) = 1 - f_i(i)$ $g = (1, 0, 1, 1, ...)$
Podemos llegar a contradicción porque: $\exists n \in \mathbb{N}$ $f_n(n) \neq g(n) = 1 - f_n(n)$! Por lo tanto, $\mathbb{N} \nsim \mathcal{P}(\mathbb{N})$
$\mathbb{N} \nsim \mathbb{N}_2$
"Nadie podrá expulsarnos de paraíso que Cantor ha creado"-Hilbert.

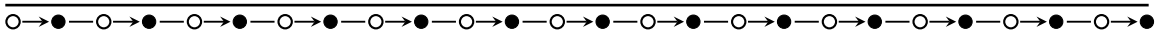
Prop. Existen lenguajes no reconocibles.
Dem: Si Σ es contable, entonces Σ^* es numerable. Sea $\mathcal{M} = \{< M > | Mesuna MT\} \subset Sigma^*$
$|\mathcal{M} \leq \aleph_0$
Si $\mathcal{L}L$ es el conjunto de todos los lenguajes sobre el alfabeto Σ, afirmamos que $|\mathcal{L}| = |\mathbb{N}_2|$
Por tanto, $|\mathcal{M}| < |\mathcal{L}|$ Es decir hay más lenguajes que MT. Así, debe haber lenguajes no reconocibles por MT alguna.
$\varphi : \mathcal{L} \to \mathbb{N}_2$ Sea $S_0, S_1, S_2, ...$ el orden canónico de $Sigma^*$, y sea $\mathcal{L}$ un lenguaje sobre $Sigma^*$ ■

¿En general $S \sim P(S)$? <u>*No*</u>
Sea $f : S \to P(S)$. Veamos que f no es suprayectiva. Proponemos $T = \{s \in S | s \notin f(s)\}$ 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 $T \neq f(s)$ $T \notin Im(f)$■

A, E, EQ $A_{TM} = \{< M\omega > |$M es una MT que acepta a la cadena$\omega\}$
Obs. A_{TM} es reconocible. U = Con entrada $< M, w >$ donde M es una MT

1. Simula a M con entrada w
2. Si M acepta, <u>*acepta*</u>, si M rechaza, <u>*rechaza*</u>

Prop. A_{TM} no es decidible Dem. Supongamos que sí, y sea R un decididor de A_{TM} $R(< M, w >) = \{$<u>*acepta*</u>si M acepta a w, <u>*rechaza*</u>si M rechaza a w$\}$ Construimos una MT D que usa a R como subrutina D = "Con entrada $< M >$ dada M es una MT:



1. Ejecuta R con entrada $< M, < M >>$

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

$D(< M >) = \{$*acepta*si M no acepta a $< M >$; *rechaza*si M acepta a $< M >\}$
$D(< D >) = \{$*acepta*si D rechaza a $< D >$; *rechaza*si D acepta a $< D >\}$! D *acepta* a D si y sólo si D *rechaza* a D! ■

Def.: Un lenguaje L sobre Σ es $co - reconocible$ si $\Sigma(\Sigma^* - L)$ es reconocible.
Prop. Un lenguaje es decidible si y sólo si es reconocible y co-reconocible.
Dem: Sea A un lenguaje. Si A es decidible, 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 $\overline{\wedge}A$ 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 decidible pero no lo es ■

$H_{TM} = \{< M, w > |$M 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 $< M, w >$, donde M es una MT:

1. Ejecuta a R con entrada ¡M,w¿

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}! ■
$E_{TM} = \{< M > |L(M) = \emptyset\}$
Prop E_{TM} no es decidible.
Dem. Por contradicción. Supongamos que sí es decidible y sea R un decididor para E_{TM}. Construimos a S un decididor para A_{TM} dado por: S : "Con entrada $< M, w >$ 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 $< M_1 >$

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 decididor para A_{TM}■
Entonces E_{TM} no es decidible
$REG_{TM} = \{< M > | M$es una Mt y L(M) es regular$\}$
Prop. REG_{TM} es indecidible
Dem: Por contradicción. Sea R un decididor para REG_{TM} Proponemos la MT S dada por: S = "
Con entrada $< M, w >$, 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 $< M_2 >$

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.
$I_1 \rightarrow^{M_2} I_2$ 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: $< M >$ con M una MT Pregunta: ¿L(M) = $\emptyset$?
Problema 2 $EQ_{TM} = \{< M_1, M_2 > | M_1$ y M_2 son MT y $L(M_1) = L(M_2)\}$ Entrada: $< M_1, M_2 >$ 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 $< M > \rightarrow^A < M, M_2 >$ 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 $< M >$ donde M es una MT:

1. Ejecuta R con entrada $< M, M_2 >$ 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 $< M, M_2 >$ que dado que $L(M_2) = \emptyset$, entonces $L(M) = \emptyset$. Si R rechaza a $< M, M_2 >$ entonces $L(M) \neq \emptyset$ luego, S acepta a $< M >$ si y sólo si $L(M) \neq \emptyset$. Por lo tanto S decide a E_{TM}!

Def. Una función $f : \Sigma^* \to \Sigma^*$ 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 $f_:\Sigma^* \to \Sigma^*$ 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 decidible, entonces A es decidible. 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 decidible, entonces B no es decidible.
PropH_{TM} es indecidible 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 = "Con entrada $< M, w >$ 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 $< M', w >$"

Si $< M, w > \in A_{TM}$, entonces $< M', w > \in H_{TM}$.
Si $< M, w > \notin A_{TM}$, entonces $< M', w > \notin H_{TM}$■