

# Reducciones y primeras clases de complejidad

Semana (4)<sub>2</sub> = 100

Lógica para Ciencia de la  
Computación - IIC2213

Prof. Sebastián Bugedo

# Programa

Obertura

Primer acto

Máquinas universales

Un lenguaje indecidible

Intermedio

Segundo acto

Reducciones

Relaciones entre lenguajes

Epílogo

# Programa

## Obertura

Primer acto

Máquinas universales

Un lenguaje indecidible

## Intermedio

Segundo acto

Reducciones

Relaciones entre lenguajes

Epílogo

¿Cómo están?

On a 1-9 rubber duck scale, how are things going today?



# Recordatorio: Máquinas de Turing

## Definición

Un **máquina de Turing determinista** (TM) es una estructura

$$\mathcal{M} = (Q, A, q_0, F, \delta)$$

- $Q$  es un conjunto finito de **estados**
- $A$  es el alfabeto de **input**, tal que  $\sqcup \notin A$  está reservado
- $q_0$  es el estado **inicial**
- $F \subseteq Q$  es un conjunto finito de **estados finales** ( $F \neq \emptyset$ )
- $\delta : Q \times (A \cup \{\sqcup\}) \rightarrow Q \times (A \cup \{\sqcup\}) \times \{\leftarrow, \rightarrow\}$  es la **función parcial de transición**

# Lenguaje aceptado

Definición (lenguaje aceptado)

Sea  $\mathcal{M} = (Q, A, q_0, F, \delta)$  una MT. Definimos el **lenguaje aceptado** por  $\mathcal{M}$  como

$$L(\mathcal{M}) = \{w \in A^* \mid \mathcal{M} \text{ acepta } w\}$$

$L(\mathcal{M})$  reúne las de palabras que tienen una ejecución de aceptación

# Detención de una máquina de Turing

¿Toda MT puede usarse como un algoritmo en la práctica?

Nos interesa que **se detenga** en **todo input**

- No debe quedarse en loops infinitos
- Aunque se demoren, sabemos que **terminan** su ejecución para todo input

Distinguimos lenguajes según si se pueden aceptar con máquinas que se detienen en todo input

# Problemas y máquinas

## Definición (RE)

Un lenguaje  $L$  es **recursivamente enumerable (RE)** o **reconocible** si existe una MT  $\mathcal{M}$  tal que

- $L = L(\mathcal{M})$

¿Qué ejemplos de lenguajes RE ya conocemos?

# Problemas y máquinas

Definición (decidibilidad)

Un lenguaje  $L$  es **decidable** o **recursivo** si existe una MT  $\mathcal{M}$  tal que

- $L = L(\mathcal{M})$
- $\mathcal{M}$  se detiene en todo input

Si  $L$  no es decidable, entonces decimos que es **indecidible**

Dimos un argumento de cardinalidad para mostrar que efectivamente existen problemas **indecidibles**

# Playlist Unidad II y Orquesta



Playlist: LogiWawos #2

Además sigan en instagram:

@orquesta\_tamen

## Objetivos de la clase

- Demostrar un primer lenguaje indecidible
- Comprender el concepto de reducción entre lenguajes
- Demostrar que un lenguaje es indecidible mediante reducciones
- Demostrar resultados sobre decidibilidad
- Definir las primeras clases de complejidad
- Comprender las relaciones de reducciones que pueden existir entre lenguajes

# Programa

Obertura

Primer acto

Máquinas universales

Un lenguaje indecidible

Intermedio

Segundo acto

Reducciones

Relaciones entre lenguajes

Epílogo

# Problemas indecidibles

La argumentación de cardinalidad no es constructiva

- Sabemos que hay problemas indecidibles
- ¿Algún ejemplo?

Construiremos un lenguaje indecidible!!!

# Paréntesis: codificación de una máquina de Turing

Resulta natural representar una máquina por una palabra binaria

Dada una máquina cualquiera  $\mathcal{M} = (Q, A, q_1, F, \delta)$  con

- $Q = \{q_1, \dots, q_n\}$
- $A = \{0, 1\}$  (supuesto razonable)
- $F = \{q_{i_1}, q_{i_2}, \dots, q_{i_m}\}$ , para  $1 \leq i_1 < i_2 < \dots < i_m \leq n$

Definimos la **codificación** de  $\mathcal{M}$  como la palabra

$$\text{cod}(\mathcal{M}) = \text{cod}(Q) \cdot 00 \cdot \text{cod}(\delta) \cdot 00 \cdot \text{cod}(F)$$

¿Cómo codificamos  $Q, \delta, F$  en binario?

# Paréntesis: codificación de una máquina de Turing

Codificación de  $Q = \{q_1, \dots, q_n\}$

- Representamos cada estado con su número en **unario**
- Separamos con ceros

$$\text{cod}(Q) = \underline{1} \ 0 \ \underline{1} \ \underline{1} \ 0 \ \dots \ 0 \ \underbrace{\dots \ 1}_{n \text{ veces}}$$

# Paréntesis: codificación de una máquina de Turing

Codificación de  $F = \{q_{i_1}, q_{i_2}, \dots, q_{i_m}\}$ , para  $1 \leq i_1 < i_2 < \dots < i_m \leq n$

- Estados en **unario**
- Separamos con ceros

$$\text{cod}(F) = \underbrace{1\cdots 1}_{i_1 \text{ veces}} 0 \underbrace{1\cdots 1}_{i_2 \text{ veces}} 0 \cdots 0 \underbrace{1\cdots 1}_{i_m \text{ veces}}$$

# Paréntesis: codificación de una máquina de Turing

## Codificación de $\delta$

- Símbolos legales en la cinta

$$\text{cod}(0) = 1 \quad \text{cod}(1) = 11 \quad \text{cod}(\_) = 111$$

- Direcciones

$$\text{cod}(\lhd) = 1 \quad \text{cod}(\rhd) = 11$$

- $k$ -ésima transición, e.g.  $t_k := \delta(q_i, 1) = (q_j, \_, \rhd)$

$$\text{cod}(t_k) = \underbrace{1\cdots 1}_{i \text{ veces}} 0 \underline{1 \ 1} 0 \underbrace{1\cdots 1}_{j \text{ veces}} 0 \underline{1 \ 1 \ 1} 0 \underline{1 \ 1}$$

- $\delta$  con  $m$  transiciones

$$\text{cod}(\delta) = \text{cod}(t_1) \cdot 0 \cdots 0 \cdot \text{cod}(t_m)$$

# Paréntesis: codificación de una máquina de Turing

Lo importante de esto:

Tenemos un método biúnivoco para representar máquinas como palabras

¿Qué ganamos con poder representar máquinas como palabras?

# Máquinas universales

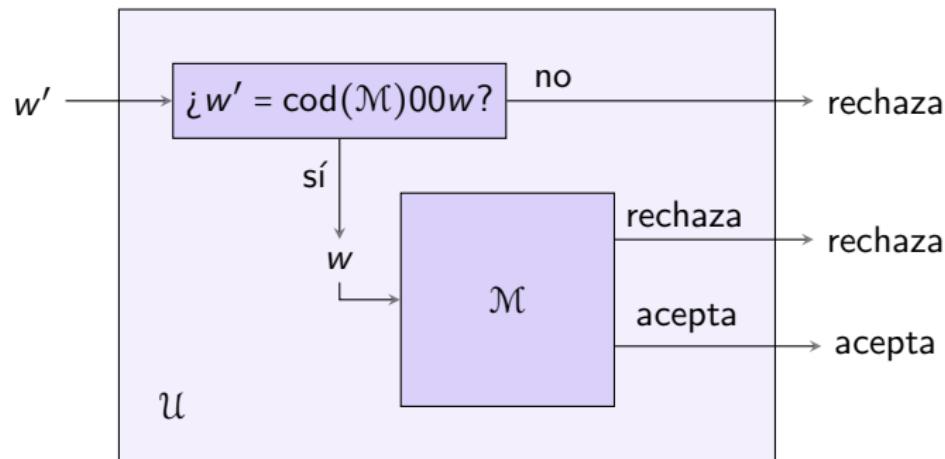
## Definición (máquinas universales)

Una máquina de Turing  $\mathcal{U}$  es una **máquina universal** si cumple que

- Si su entrada es de la forma  $\text{cod}(\mathcal{M})00w$ , para  $w \in A^*$  y  $\mathcal{M}$  máquina con alfabeto  $A$ , entonces
$$\mathcal{U} \text{ acepta } \text{cod}(\mathcal{M})00w \quad \text{si y solo si} \quad \mathcal{M} \text{ acepta } w$$
- Si su entrada no es de la forma  $\text{cod}(\mathcal{M})00w$ , entonces  $\mathcal{U}$  rechaza su entrada

¿Qué está haciendo  $\mathcal{U}$  cuando detecta una codificación de máquina?

# Máquinas universales



El módulo verificador de formato se conoce como **parser**

Usaremos esta idea para **simular** máquinas de Turing dentro de otras

# Programa

Obertura

Primer acto

Máquinas universales

Un lenguaje indecidible

Intermedio

Segundo acto

Reducciones

Relaciones entre lenguajes

Epílogo

# Decidibilidad vs Indecidibilidad

Pregunta central:

Dada una máquina  $\mathcal{M}$  y una palabra  $w$ ,

¿ $\mathcal{M}$  se detiene en  $w$ ?

# Lenguajes sobre máquinas

Dada una máquina  $\mathcal{M}$  y una palabra  $w$ , ¿ $\mathcal{M}$  se detiene en  $w$ ?

Problema: HALTING (PARADA)

Entrada: Máquina de Turing  $\mathcal{M}$ , palabra  $w$

Salida: ¿Se detiene  $\mathcal{M}$  con entrada  $w$ ?

¿Cómo se ve el lenguaje asociado a HALTING?

$$H = \{w' \in \{0,1\}^* \mid \text{existen } \mathcal{M} \text{ y } w \in A^* \text{ tales que } w' = \text{cod}(\mathcal{M})00w \text{ y } \mathcal{M} \text{ se detiene en } w\}$$

¿Es HALTING decidable?

# Lenguajes sobre máquinas

Para analizar  $H$ , definimos un problema relacionado más acotado

Problema: DIAGONAL

Entrada: Máquina de Turing  $\mathcal{M}$

Salida: ¿Se detiene  $\mathcal{M}$  en  $\text{cod}(\mathcal{M})$ ?

y su lenguaje

$$DG = \{w \in \{0, 1\}^* \mid \text{existe } \mathcal{M} \text{ tal que } w = \text{cod}(\mathcal{M}) \\ \text{y } \mathcal{M} \text{ se detiene en cod}(\mathcal{M})\}$$

¿Es DIAGONAL decidable? ¿Es RE?

# Lenguajes sobre máquinas: DIAGONAL

$DG = \{w \in \{0, 1\}^* \mid \text{existe } \mathcal{M} \text{ tal que } w = \text{cod}(\mathcal{M})$   
 $\quad \text{y } \mathcal{M} \text{ se detiene en cod}(\mathcal{M})\}$

## Demostración

Demuestre que  $DG$  es un lenguaje recursivamente enumerable (RE).



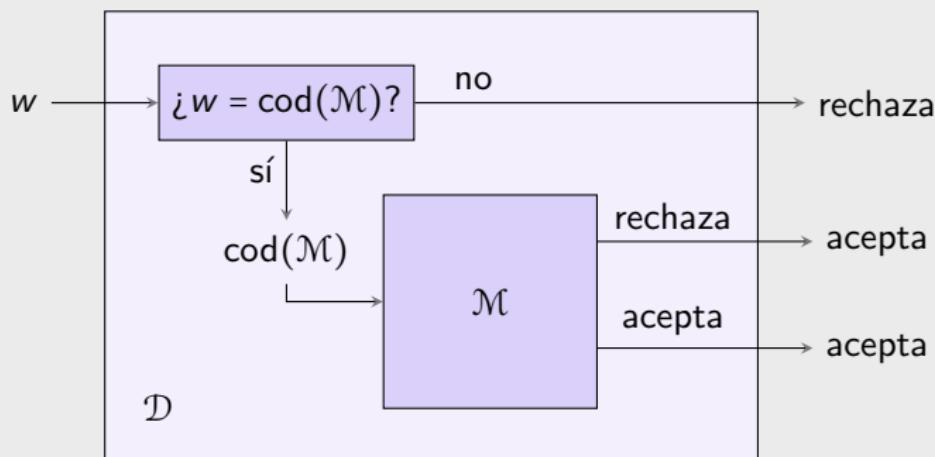
# Lenguajes sobre máquinas: DIAGONAL

## Demostración ( $DG$ es RE)

Sea  $\mathcal{D}$  una máquina tal que para si el input es  $w = \text{cod}(\mathcal{M})$

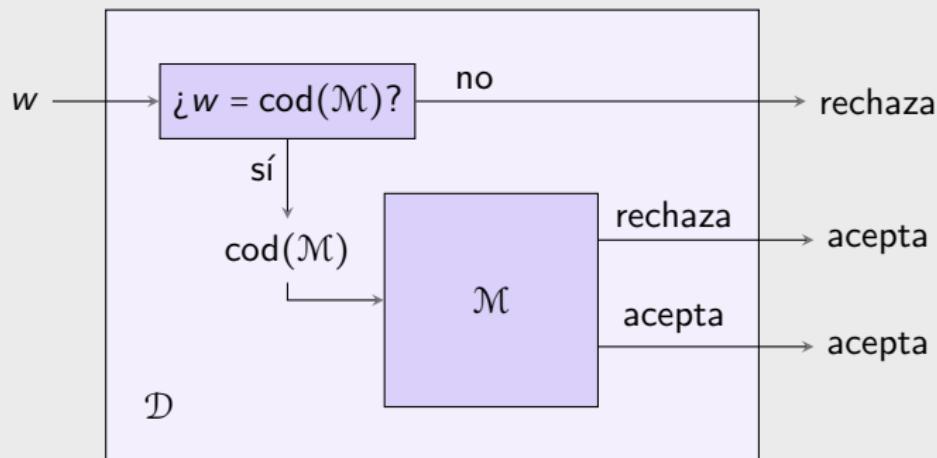
1. Simula  $\mathcal{M}$  con input  $\text{cod}(\mathcal{M})$
2. Si  $\mathcal{M}$  acepta/rechaza  $\text{cod}(\mathcal{M})$ , entonces  $\mathcal{D}$  acepta

Si no es de la forma  $w = \text{cod}(\mathcal{M})$ , lo rechaza



# Lenguajes sobre máquinas: DIAGONAL

Demostración ( $DG$  es RE)



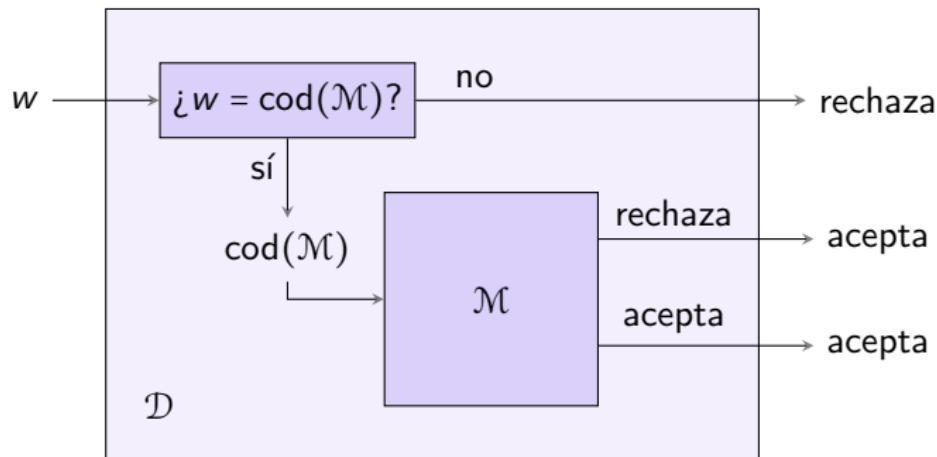
Las  $\text{cod}(\mathcal{M})$  para las que  $\mathcal{D}$  se detiene son **aceptadas**, i.e.

$$\mathcal{L}(\mathcal{D}) = DG$$

Como  $\mathcal{D}$  acepta  $DG$ , es un lenguaje RE. □

Demuestre de forma similar que  $H$  también es RE

# Lenguajes sobre máquinas: DIAGONAL



¿La máquina  $\mathcal{D}$  sirve para demostrar la decidibilidad de  $DG$ ?

# Problema DIAGONAL

## Teorema

El lenguaje  $DG$  es indecidible, es decir, no existe una máquina de Turing  $\mathcal{M}$  tal que

- $DG = L(\mathcal{M})$
- $\mathcal{M}$  se detiene en todos los inputs

# Problema DIAGONAL

## Teorema

El lenguaje  $DG$  es indecidible, es decir, no existe una máquina de Turing  $\mathcal{M}$  tal que

- $DG = L(\mathcal{M})$
- $\mathcal{M}$  se detiene en todos los inputs

## Demostración



# Problema DIAGONAL

## Demostración

Por contradicción, supongamos que existe una MT  $\mathcal{D}$  que se **detiene en todas las entradas** y tal que  $DG = L(\mathcal{D})$ .

Es decir, para toda MT  $\mathcal{M}$

- si  $\mathcal{M}$  se detiene en  $\text{cod}(\mathcal{M})$ , entonces  $\mathcal{D}$  acepta  $\text{cod}(\mathcal{M})$
- si  $\mathcal{M}$  NO detiene en  $\text{cod}(\mathcal{M})$ , entonces  $\mathcal{D}$  rechaza  $\text{cod}(\mathcal{M})$

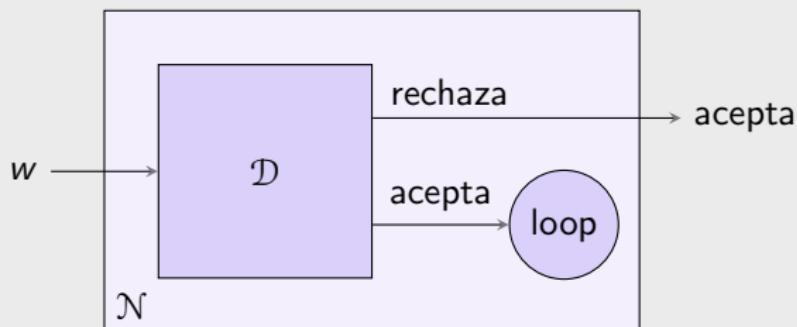
Además,  $\mathcal{D}$  rechaza toda entrada que no es la codificación de una MT.

Definiremos una nueva máquina  $\mathcal{N}$  a partir de  $\mathcal{D}$ .

# Problema DIAGONAL

Sea  $\mathcal{N}$  una MT que con entrada  $w$

- Simula  $\mathcal{D}$  con entrada  $w$
- Si  $\mathcal{D}$  acepta  $w$ , se entra en un loop
- Si  $\mathcal{D}$  rechaza  $w$ , se le acepta



# Problema DIAGONAL

Sea  $\mathcal{N}$  una MT que con entrada  $w$

- Simula  $\mathcal{D}$  con entrada  $w$
- Si  $\mathcal{D}$  acepta  $w$ , se entra en un loop
- Si  $\mathcal{D}$  rechaza  $w$ , se le acepta

Es decir,  $\mathcal{N}$  acepta los  $w$  rechazados por  $\mathcal{D}$ , y no se detiene en todas las demás entradas.

Como  $\mathcal{N}$  es una MT, cabe preguntarse si  $\mathcal{D}$  la acepta como entrada.

$$\begin{aligned}\mathcal{D} \text{ acepta } \text{cod}(\mathcal{N}) &\Leftrightarrow \mathcal{N} \text{ se detiene en } \text{cod}(\mathcal{N}) \quad (\text{def. de } \mathcal{D}) \\ &\Leftrightarrow \mathcal{N} \text{ acepta } \text{cod}(\mathcal{N}) \quad (\text{def. de } \mathcal{N}) \\ &\Leftrightarrow \mathcal{D} \text{ rechaza } \text{cod}(\mathcal{N}) \quad (\text{def. de } \mathcal{N})\end{aligned}$$

Esta contradicción significa que no existe tal  $\mathcal{D}$ . Por lo tanto,  
DIAGONAL es indecidible. □

# Problema HALTING

## Teorema (Halting)

El lenguaje  $H$  es indecidible, es decir, no existe una máquina de Turing  $\mathcal{M}$  tal que

- $H = L(\mathcal{M})$
- $\mathcal{M}$  se detiene en todos los inputs

Demostraremos el teorema usando que DIAGONAL es indecidible

# Programa

Obertura

Primer acto

Máquinas universales

Un lenguaje indecidible

**Intermedio**

Segundo acto

Reducciones

Relaciones entre lenguajes

Epílogo

# Programa

Obertura

Primer acto

Máquinas universales

Un lenguaje indecidible

Intermedio

Segundo acto

Reducciones

Relaciones entre lenguajes

Epílogo

# La noción de reducción

Definimos a los problemas como objeto de estudio

Nos gustaría poder concluir propiedades de un problema a partir de otros problemas

Nos proponemos “*transformar*” problemas entre sí

# La noción de reducción

**Problema:**

Multiplicación de números

$$3 \times 4$$



**Se reduce a:**

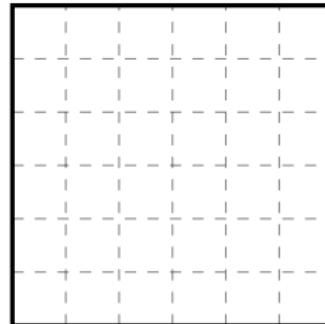
Suma de números

$$4 + 4 + 4$$

# La noción de reducción

**Problema:**

Calculo del área de un cuadrado



**Reducción:**

Multiplicación de dos números

base × altura

# La noción de reducción

Definición en chileno, 1.0

Una **reducción** transforma un **problema A** en un **problema B** tal que si encontramos un algoritmo para **B** entonces tenemos un algoritmo para **A**

**multiplicación**    se reduce a    **suma**

# La noción de reducción

Definición en chileno, 2.0

Una **reducción** transforma un **lenguaje**  $L_A$  en un **lenguaje**  $L_B$  tal que si encontramos un algoritmo para **B** entonces tenemos un algoritmo para **A**

$L_A$     se reduce a     $L_B$

*Si tengo un algoritmo para resolver  $L_B$  entonces tengo un algoritmo para resolver  $L_A$ .*

## La noción de reducción

Si queremos transformar instancias de un lenguaje a otro...

... necesitamos más que solo **decidir** cosas con máquinas

Necesitamos máquinas que **calculen**...

¡Pero eso ya podemos hacerlo! jj

# Funciones computables

Definición (máquinas que calculan)

Una MT  $\mathcal{M}$  **calcula una función**  $f : A^* \rightarrow A^*$  si

- Recibe entrada  $w \in A^*$
- Se detiene y acepta  $w$
- Al detenerse, el contenido de su cinta es exclusivamente  $f(w)$

Definición (funciones computables)

Una función  $f : A^* \rightarrow A^*$  es **computable** si existe una máquina de Turing  $\mathcal{M}$  que calcula  $f$

Podemos pensar en su cinta como el output

# Funciones computables

## Ejemplo

Sea  $g : \{0, 1\}^* \rightarrow \{0, 1\}^*$  una función definida como  $g(w) = w0000$ .

Sea  $\mathcal{M} = (Q, A, q_1, F, \delta)$  con  $Q = \{q_1, q_2, q_3, q_4, q_f\}$ ,  $A = \{0, 1\}$ ,  $F = \{q_f\}$  y  $\delta$  definida por

$$\begin{array}{lll} \delta(q_1, 0) & = & (q_1, 0, \triangleright) \\ \delta(q_1, 1) & = & (q_1, 1, \triangleright) \\ \delta(q_1, \_) & = & (q_2, 0, \triangleright) \end{array} \quad \begin{array}{lll} \delta(q_2, \_) & = & (q_3, 0, \triangleright) \\ \delta(q_3, \_) & = & (q_4, 0, \triangleright) \\ \delta(q_4, \_) & = & (q_f, 0, \triangleright) \end{array}$$

Para toda entrada  $w$ ,  $\mathcal{M}$  acepta  $w$  y el contenido de su cinta es  $g(w)$ . Por lo tanto,  $g$  es computable.

¿Toda función es computable?

# Reducciones de Turing

Definición (reducciones de Turing)

Dados lenguajes  $L_1, L_2$  con alfabeto  $A$ , decimos que  $L_1$  es

**Turing-reducible** a  $L_2$  si existe una función computable  $f : A^* \rightarrow A^*$  tal que para todo  $w \in A^*$

$$w \in L_1 \quad \text{si y solo si} \quad f(w) \in L_2$$

En tal caso, llamamos a  $f$  una **reducción de Turing** de  $L_1$  a  $L_2$  y denotamos

$$L_1 \leq_T L_2$$

Intuición:  $L_2$  es al menos tan difícil como  $L_1$

# Formalización de la intuición

## Teorema

Sean  $L_1, L_2$  lenguajes tales que  $L_1$  es Turing-reducible a  $L_2$ .

1. Si  $L_2$  es decidable, entonces  $L_1$  es decidable
2. Si  $L_1$  es indecidible, entonces  $L_2$  es indecidible

## Ejercicio (propuesto)

Demuestre el teorema.

# HALTING es indecidible

Con las reducciones de Turing y este último resultado podemos demostrar que HALTING es indecidible!

## Demostración ( $H$ indecidible)

Se demostró que  $DG$  es un lenguaje indecidible. Recordemos que

$$DG = \{w \in \{0, 1\}^* \mid \text{existe } \mathcal{M} \text{ tal que } w = \text{cod}(\mathcal{M}) \text{ y } \mathcal{M} \text{ se detiene en cod}(\mathcal{M})\}$$

y el lenguaje  $H$  es

$$H = \{w' \in \{0, 1\}^* \mid \text{existen } \mathcal{M} \text{ y } w \in A^* \text{ tales que } w' = \text{cod}(\mathcal{M})00w \text{ y } \mathcal{M} \text{ se detiene en } w\}$$

Buscamos una reducción de Turing de  $DG$  a  $H$ .

# HALTING es indecidible

## Demostración ( $H$ indecidible)

$$DG = \{w \in \{0, 1\}^* \mid \text{existe } \mathcal{M} \text{ tal que } w = \text{cod}(\mathcal{M}) \text{ y } \mathcal{M} \text{ se detiene en cod}(\mathcal{M})\}$$

$$H = \{w' \in \{0, 1\}^* \mid \text{existen } \mathcal{M} \text{ y } w \in A^* \text{ tales que } w' = \text{cod}(\mathcal{M})00w \text{ y } \mathcal{M} \text{ se detiene en } w\}$$

Consideramos la función  $g : \{0, 1\}^* \rightarrow \{0, 1\}^*$  dada por

$$g(w) = w00w$$

Es claro que  $g$  es computable y a continuación probaremos que efectivamente es reducción de Turing de  $DG$  a  $H$ .

# HALTING es indecidible

## Demostración ( $H$ indecidible)

( $\Rightarrow$ ) Sea  $w \in DG$ .

$$\begin{aligned} w \in DG &\Rightarrow w = \text{cod}(\mathcal{M}) \text{ con } \mathcal{M} \text{ que se detiene en su código} \\ &\Rightarrow \text{cod}(\mathcal{M})00\text{cod}(\mathcal{M}) \in H \\ &\Rightarrow g(\text{cod}(\mathcal{M})) \in H \\ &\Rightarrow g(w) \in H \end{aligned}$$

( $\Leftarrow$ ) Sea  $w \notin DG$ .

$$\begin{aligned} w \notin DG &\Rightarrow w = \text{cod}(\mathcal{M}) \text{ con } \mathcal{M} \text{ que no detiene en su código} \\ &\quad \text{o } w \text{ no es codificación de una máquina} \\ &\Rightarrow w00w \notin H \\ &\Rightarrow g(\text{cod}(\mathcal{M})) \notin H \\ &\Rightarrow g(w) \notin H \end{aligned}$$

Es decir,  $g$  es reducción de Turing de  $DG$  a  $H$ . Como  $DG$  es indecidible, concluimos que  $H$  es indecidible.



## Paréntesis: tuplas en lenguajes

Cuando sea conveniente, podremos expresar palabras de un lenguaje binario como **tuplas**...

... no olvidar que la estructura de la tupla se puede representar de forma binaria, por lo que no es un problema

Esto evitará poner 0's en exceso y clarificará qué representa el lenguaje

# Paréntesis: tuplas en lenguajes

## Ejemplo

Podemos reescribir  $H$  según

$$H = \{(\mathcal{M}, w) \mid w \in A^* \text{ y } \mathcal{M} \text{ máquina tal que } \mathcal{M} \text{ se detiene en } w\}$$

Cualquier palabra que no cumple la estructura pedida, es rechazado.  
Es decir, en este caso no pertenecen a  $H$

- Tuplas  $(\mathcal{M}, w)$  donde  $\mathcal{M}$  no se detiene en  $w$
- Palabras que no corresponden al formato de tupla  
máquina-palabra

Notemos que esta forma se parece a la de las cajas

# Paréntesis: tuplas en lenguajes

## Ejemplo

Podemos reescribir  $H$  según

$$H = \{(\mathcal{M}, w) \mid w \in A^* \text{ y } \mathcal{M} \text{ máquina tal que } \mathcal{M} \text{ se detiene en } w\}$$

Esta forma se asemeja a la representación como problema de decisión

Problema: HALTING (PARADA)

Entrada: Máquina de Turing  $\mathcal{M}$ , palabra  $w$

Salida: ¿Se detiene  $\mathcal{M}$  con entrada  $w$ ?

La notación de tuplas nos será más clara en varios momentos

# Reducciones y lenguajes indecidibles

## Ejercicio

Demuestre que el siguiente lenguaje es indecidible

$$DD = \{(\mathcal{M}_1, \mathcal{M}_2) \mid \mathcal{M}_1 \text{ se detiene con entrada } \text{cod}(\mathcal{M}_1) \text{ y} \\ \mathcal{M}_2 \text{ se detiene con entrada } \text{cod}(\mathcal{M}_2)\}$$



# Reducciones y lenguajes indecidibles

Un par de preguntas espirituales

- ¿Todos los problemas indecidibles son igual de difíciles?
- ¿Se pueden comparar todos los lenguajes difíciles?

# Programa

Obertura

Primer acto

Máquinas universales

Un lenguaje indecidible

Intermedio

Segundo acto

Reducciones

Relaciones entre lenguajes

Epílogo

# Relaciones entre lenguajes

La herramienta de reducción nos permite estudiar los lenguajes como objetos

Para formalizar esta idea, nos centraremos en colecciones de lenguajes

Llamaremos **clase de complejidad** a un conjunto de lenguajes que comparten algún criterio

¿Qué clases de complejidad conocemos?

# Relaciones entre RE y R

¿Cómo se relacionan los lenguajes recursivamente enumerables (RE) y decidibles (R)?

Definiremos las clases de complejidad

$$R = \{L \mid L \text{ es decidable}\}$$

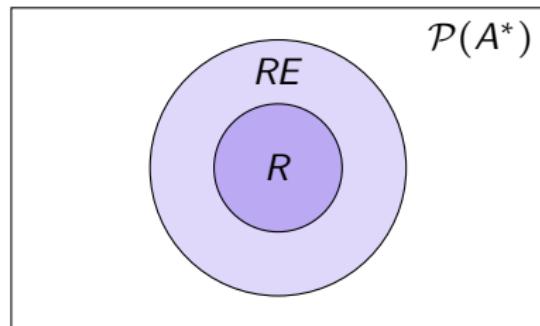
$$RE = \{L \mid L \text{ es recursivamente enumerable}\}$$

¿Cómo se ve un diagrama de conjuntos de las clases RE y R?

# Relaciones entre RE y R

## Teorema

Si un lenguaje es decidable, entonces es recursivamente enumerable



Por argumentos de conteo, no todo lenguaje indecidible es RE

# Reducciones en RE

En el mundo RE también tenemos resultados sobre reducciones

Teorema

Sean  $L_1, L_2$  tales que  $L_1 \leq_T L_2$ . Si  $L_2$  es RE, entonces  $L_1$  es RE

Ejercicio (propuesto)

Demuestre el teorema.

# Lenguajes complementarios

## Notación

Dado un lenguaje  $L$  sobre alfabeto  $A$ , denotamos su complemento

$$\bar{L} = A^* \setminus L$$

Dada una clase  $W$ , definiremos la clase  $\text{co}W$  según

$$\text{co}W = \{L \mid \bar{L} \in W\}$$

Con esto, definimos

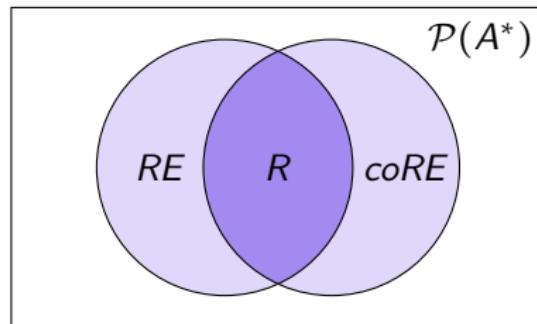
$$\text{coRE} = \{L \mid \bar{L} \text{ es recursivamente enumerable}\}$$

¿Cómo se ve el diagrama de RE, R y co-RE?

# Una nueva caracterización de $R$

## Teorema

Si un lenguaje  $L$  y su complemento  $\bar{L}$  son RE, entonces  $L$  es decidible



El teorema nos entrega otra forma de demostrar que  $L$  es decidible

# Una nueva caracterización de $R$

## Teorema

Si un lenguaje  $L$  y su complemento  $\bar{L}$  son RE, entonces  $L$  es decidible

## Demostración



# Una nueva caracterización de $R$

## Demostración

Como  $L$  y  $\bar{L}$  son RE, existen máquinas  $\mathcal{M}$  y  $\bar{\mathcal{M}}$  tales que aceptan  $w$  si y solo si pertenece al lenguaje correspondiente.

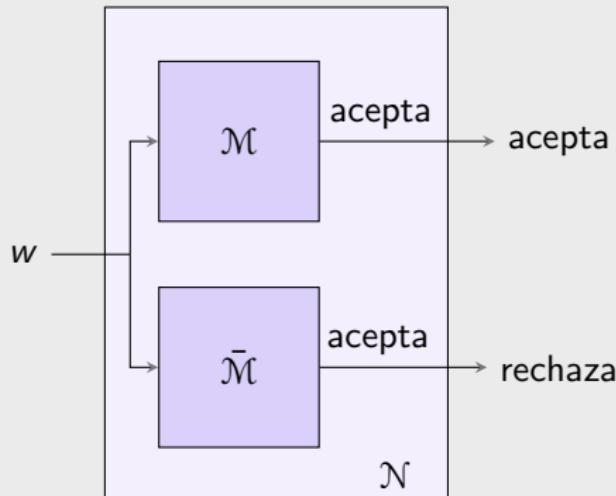
Sea  $\mathcal{N}$  una MT que con entrada  $w$

- Simula **en paralelo** a  $\mathcal{M}$  y  $\bar{\mathcal{M}}$  con entrada  $w$
- Si  $\mathcal{M}$  acepta  $w$ , se acepta
- Si  $\bar{\mathcal{M}}$  acepta  $w$ , se le rechaza

La ejecución se hace instrucción por instrucción hasta que alguna de las dos máquinas acepte.

# Una nueva caracterización de $R$

## Demostración



Esta máquina se detiene en todo input y  $L(N) = L$ , por lo que  $L$  es decidible. □

# Lenguajes indecidos no RE

En este punto ya podemos dar un ejemplo de **lenguaje indecidible y que no es RE**

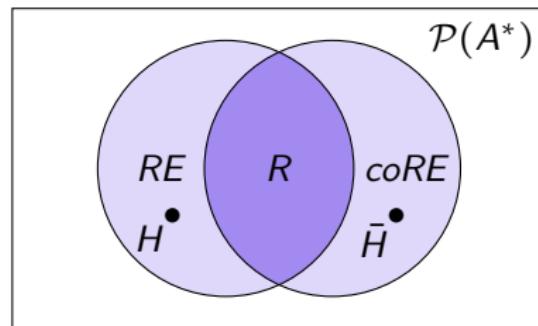
Corolario

1.  $\bar{H}$  es indecidible y no recursivamente enumerable.
2.  $\bar{H}$  no es reducible a  $H$

¿Qué significa que un lenguaje no sea reducible a otro?

¿Cuál es más difícil?

# El panorama hasta ahora



¿Todos los lenguajes en una clase son reducibles entre sí?

## Hacia dónde vamos

Estudiaremos cómo se relacionan los problemas de una misma clase de complejidad

... para ello motivaremos el concepto de **completitud**

*¿Qué problema representa a una clase?*

*¿Qué problema es el más difícil de una clase?*

Próxima clase: clases de complejidad y problemas completos

# Programa

Obertura

Primer acto

Máquinas universales

Un lenguaje indecidible

Intermedio

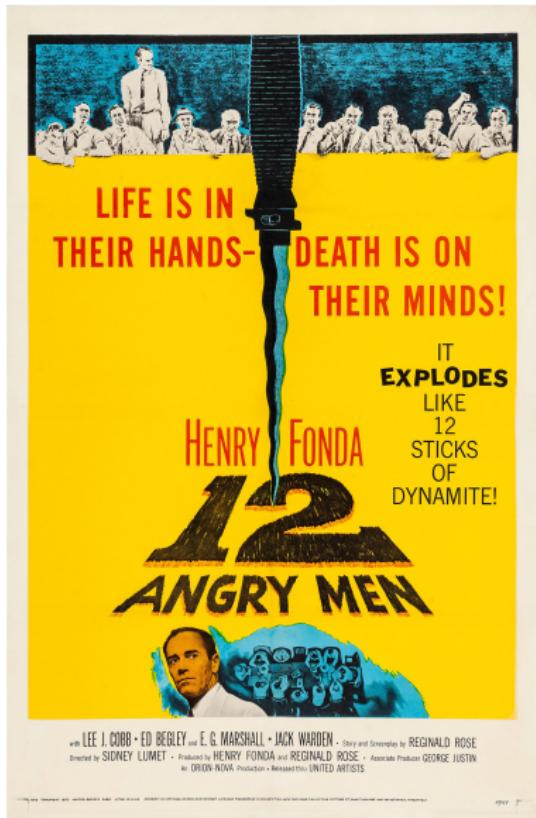
Segundo acto

Reducciones

Relaciones entre lenguajes

Epílogo

# Actividad Espiritual Complementaria #1 (última semana)



With LEE J. COBB • ED BEGLEY and E. G. MARSHALL • JACK WARDEN • Story and Screenplay by REGINALD ROSE  
Directed by SIDNEY LUMET • Produced by HENRY FONDA and REGINALD ROSE • Associate Producer GEORGE JUSTIN  
An ORION-NOVA Production • Presented by UNITED ARTISTS

## Objetivos de la clase

- Demostrar un primer lenguaje indecidible
- Comprender el concepto de reducción entre lenguajes
- Demostrar que un lenguaje es indecidible mediante reducciones
- Demostrar resultados sobre decidibilidad
- Definir las primeras clases de complejidad
- Comprender las relaciones de reducciones que pueden existir entre lenguajes

¿Qué aprendí hoy? ¿Comentarios?

Ve a

**www.menti.com**

Introduce el código

**8452 8073**



O usa el código QR