

# Práctica 4

## Variantes de las Máquinas de Turing e Indecibilidad

J. P. Abarca<sup>1</sup>   C. T. Apari<sup>1</sup>   C. A. Suca<sup>1</sup>   A. Vargas<sup>1</sup>

<sup>1</sup>Universidad Nacional del San Agustín. Facultad de Producción y Servicios.  
Escuela Profesional de Ciencias de la Computación  
Maestría en Ciencias de la Computación  
Docente: Dra. Marcela Quispe Cruz

20 de Agosto del 2021



## 1 Variantes de las Máquinas de Turing

- Máquinas de Turing Stay-Put
- Máquinas de Turing con cintas infinitas por ambos lados
- Máquinas de Turing Multicabezal
- Máquinas de Turing Multidimensional
- Máquinas de Turing con Múltiples Tracks
- Máquinas de Turing Multicinta
- Máquinas de Turing no Deterministas
- Enumeradores

## 2 Indecibilidad

- Metodo de la Diagonalización
- Un Lenguaje Indecidable
- Un Lenguaje Turing No-Reconocible
- Otros Problemas Indecidibles

## 3 Conclusiones

## 4 Referencias



## 1 Variantes de las Máquinas de Turing

- Máquinas de Turing Stay-Put
- Máquinas de Turing con cintas infinitas por ambos lados
- Máquinas de Turing Multicabezal
- Máquinas de Turing Multidimensional
- Máquinas de Turing con Múltiples Tracks
- Máquinas de Turing Multicinta
- Máquinas de Turing no Deterministas
- Enumeradores

## 2 Indecibilidad

- Metodo de la Diagonalización
- Un Lenguaje Indecidable
- Un Lenguaje Turing No-Reconocible
- Otros Problemas Indecidibles

## 3 Conclusiones

## 4 Referencias



# Variantes de Máquinas de Turing

Presentamos los diversos tipos o variantes de Máquinas de Turing, que surgen de la necesidad de flexibilidad para la solución de problemas.

Resulta que la definición de Máquina Turing (simple, normal o estándar) que analizamos es extremadamente robusta.

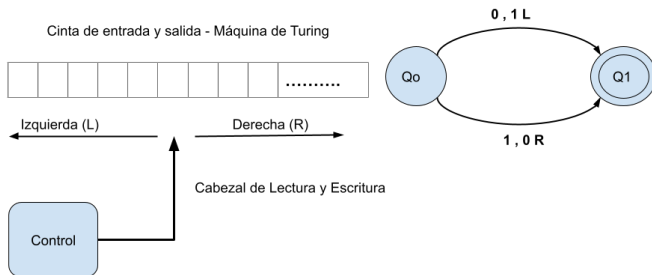
Podemos hacer uso de las variantes de máquinas de turing, o combinarlas para obtener máquinas aún más complejas, pero ello no implica que ofrecen mayor capacidad para describir lenguajes que la versión estándar.

Porque todos ellos tienen el mismo poder computacional.



# Máquina de Turing Estándar

- Determinista.
- Una sólo cinta.
- Un solo cabezal de lectura y escritura.
- Infinito a la derecha, limitado por la izquierda.
- función de transición:  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$



Esta variante de máquina de Turing, tiene la capacidad de mantener el cabezal en el mismo lugar de la cinta.

¿Podría esta característica permitir que las máquinas de Turing reconozcan lenguajes adicionales, aumentando así la potencia del modelo? Por supuesto que no, porque podemos convertir cualquier Máquina de Turing con la función de permanecer en su sitio, en una que no la tenga.

Lo hacemos reemplazando cada transición fija con dos transiciones: una que se mueve hacia la derecha y la segunda hacia la izquierda o una que se mueve a la izquierda y la segunda hacia la derecha.

Este pequeño ejemplo contiene la clave para mostrar la equivalencia de variantes de Máquinas de Turing con la Máquina Turing Estándar.



# Stay-Put - Diagrama de Estados y Función de Transición

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$$

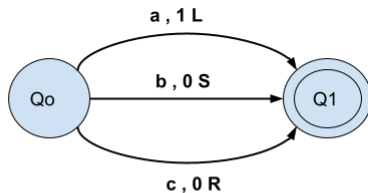
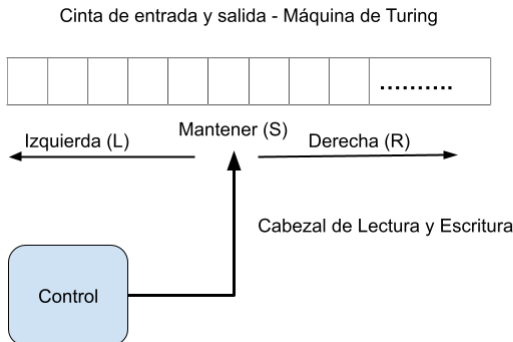


Figura: Máquina Turing Stay-Put y diagrama de estados



# Cintas infinitas por ambos lados

Una cinta infinita bidireccional de una Máquina de Turing, puede moverse indefinidamente en cualquier dirección.

Se puede demostrar que esto es equivalente a una Máquina de Turing de cinta infinita unidireccional mediante el siguiente argumento: Está claro que se pueden usar dos Máquinas de Turing de cinta infinita unidireccional para simular una cinta infinita bidireccional. Si estos están alineados “ uno al lado del otro ” y las dos cintas están intercaladas, entonces se puede construir una sola Máquina de Turing infinita unidireccional que sea equivalente.





# Cintas por ambos lados - Diagrama de Estados y Función de Transición

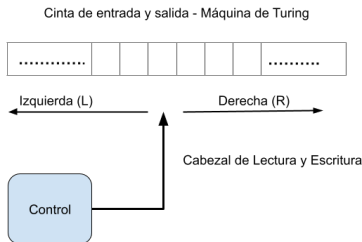


Figura: Máquina Turing cintas infinitas y diagrama de estados



Esta es una Máquina de Turing de una sola cinta con múltiples cabezales de lectura / escritura.

En cada estado solo se puede usar una cabeza.

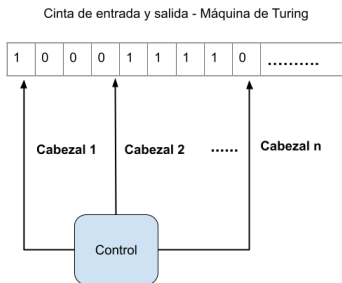
Por lo tanto, para  $k$  cabezas tienen una partición de los estados  $Q_1, Q_2, \dots, Q_k$  donde cada  $Q_i$  contiene el conjunto de estados que utilizan el  $i$ -ésimo cabeza.



# Multicabezal - Diagrama de Estados y Función de Transición

$$\delta : Q \times \{H1, H2, \dots, Hn\} \times (\Gamma \cup \{\Delta\}) \rightarrow (Q \cup \{h\}) \times (\Gamma \cup \{\Delta\}) \times \{R, L, S\}$$

Donde  $H1, H2, \dots, Hn$ , representan los cabezales.



**Figura:** Máquina Turing MultiCabezal y diagrama de estados



Este es un tipo de máquinas de Turing que tienen un control finito, un cabezal de lectura y escritura y una cinta bidimensional, extendiéndose hacia la derecha y hacia abajo infinitamente.

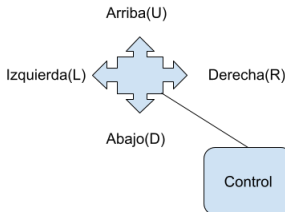
Para cualquier máquina de Turing de este tipo existe una máquina de Turing con una cinta unidimensional igualmente potente, es decir, la primera puede ser simulada por la segunda.



# Multidimensional - Diagrama de Estados y Función de Transición

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\}$$

## Cinta de entrada y salida 2 dimensiones - Máquina de Turing

[illegible]

v	1	v	2	3	h	4	5	6	v	...
---	---	---	---	---	---	---	---	---	---	-----

# Múltiples Tracks

Contienen varias pistas, pero solo un cabezal de cinta que lee y escribe en todas las pistas.

Aquí, un solo cabezal de cinta lee  $n$  símbolos de  $n$  pistas en un solo paso. Para cada Máquina Turing  $S$  de una sola pista, hay una máquina Máquina de Turing con múltiples tracks equivalente tal que  $L(S) = L(M)$ .



# Múltiples Tracks - Diagrama de Estados y Función de Transición

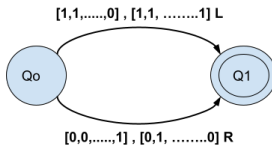
$$\delta(q_i, [x_1, x_2, \dots, x_n]) = (q_j, [y_1, y_2, \dots, y_n], L)$$

Cinta de entrada y salida - Máquina de Turing

Track 1	1	0	0	1			.....
Track 2	1	1	1	1			.....
	⋮	⋮	⋮	⋮	⋮	⋮	
Track n	0	0	0	1			.....

L, R, S

Cabezal de Lectura y Escritura



Esta variante consta de múltiples cintas, cada una tiene su propio cabezal. Inicialmente la primera cinta contiene la entrada, las demás inician en blanco.

La función de transición se cambia para permitir leer, escribir y mover los cabezales en algunas o todas las cintas simultáneamente.

Las máquinas de Turing de cintas múltiples parecen ser más poderosas que las máquinas de Turing estándar, pero podemos demostrar que son equivalentes en potencia. Recuerde que dos máquinas son equivalentes si reconocen el mismo lenguaje.

**Teorema 3.13.** Cada máquina de Turing de varias cintas tiene una máquina de Turing de una sola cinta equivalente.



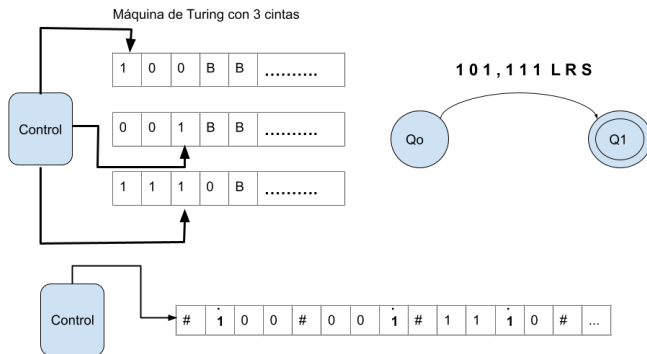


# Multicinta - Diagrama de Estados y Función de Transición

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$$

Donde K es el número de cintas.

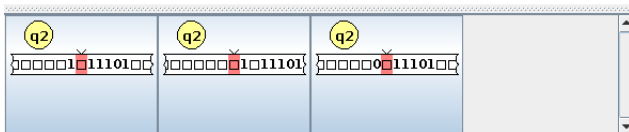
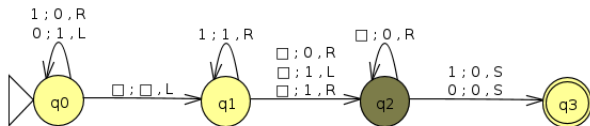
$$\delta(q_i, a_1, \dots, a_k) = (q_j, b_1, \dots, b_k, L, R, \dots, L)$$



# Máquinas de Turing no Deterministas

La máquina de Turing puede proceder con muchas posibilidades, la función de transición para una maquina de turing no deterministica [1] se da la siguiente forma:

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



# Máquinas de Turing no Deterministas

## Theorem

*Cada MT No determinista tiene un equivalente de MT determinista.*





# Máquinas de Turing no Deterministas

## Theorem

*Un lenguaje es turing reconocible si y solamente si alguna MT N reconoce esto.*

## Demostración.

Cualquier MT D es automaticamente una MT N. direccion izq. ☐

## Demostración.

Cualquier MT N tiene su equivalente de MT D. direccion der ☐



# Máquinas de Turing no Deterministas

## Definition

Sea  $L \subseteq \Sigma^*$  Decimos que una MT decide  $L$  si para toda  $w \in \Sigma^*$ , la MT decide  $w$

## Definition

Llamamos MT N decidor si todas las ramas tiene estado de parada sobre todos los ingresos de cadenas.



## Theorem

*Si una MT  $N$  siempre para sobre todas sus ramas de su computación, una MT  $D$  siempre también tendrá estado de parada sea de aceptación o rechazo.*

## Corollary

*Un lenguaje es decidible si y solamente si alguna MT  $N$  decide esto.*



Recordando...

## Definition

Lenguaje de Turing Decidible (Lenguaje Recursivo)

- $L$  es aceptada por una MT  $M$
- $M$  siempre para (se detiene)
- $M$  decide  $L$





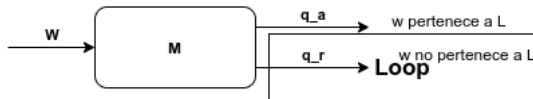
# Lenguajes Recursivos y Recursivamente Enumerables

Recordando...

## Definition

Lenguaje de Turing Reconocible (Lenguaje Recursivamente Enumerable)

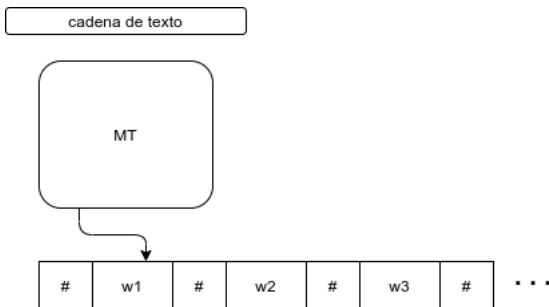
- $L$  es aceptada por una MT  $M$
- $M$  no siempre para (Loop)
- $M$  no decide  $L$



## Definition

Una lenguaje que es turing reconocible es también lenguaje recursivamente enumerable de ahí la existencia de una MT enumerable.

Una máquina enumerable E es aquella que imprime como output todas las cadenas aceptadas por una maquina de turing



- El enumerador E si no para puede imprimir infinitamente (recursivamente enumerable).
- Su lenguaje es una colección de cadenas que en algún momento se imprimen.
- las cadenas son generadas e impresas en cualquier orden
- las cadenas pueden imprimirse con repetición.



## Theorem

*Un lenguaje es turing reconocible si el enumerador  $E$  (MT) enumera este.*

## Demostración.

Un enumerador  $E$  que enumera un lenguaje  $A$ , si y solo si una MT  $M$  reconoce  $A$ .

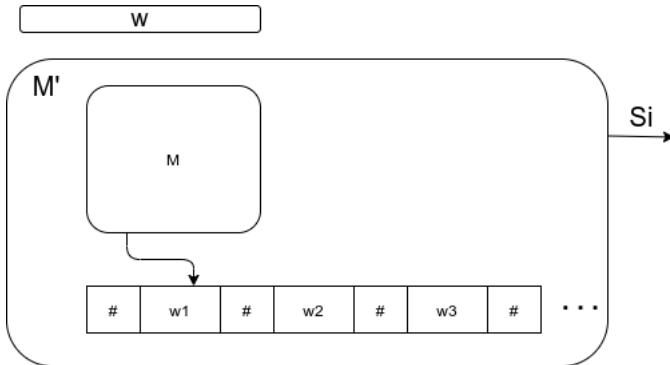
- $M$  toma entrada  $w$
- Corre Enumerador, y cada vez que imprime algo se compara con  $w$ .
- si  $w$  aparece en la salida de  $E$ , entonces acepta.



# Enumeradores - Demostracion Izq

## Theorem

*Sea  $L$  es recursivamente enumerable, si y solo si  $L = G(M)$  para una máquina enumerable.*



## Theorem

*Un lenguaje es turing reconocible si el enumerador  $E$  (MT) enumera este.*

## Demostración.

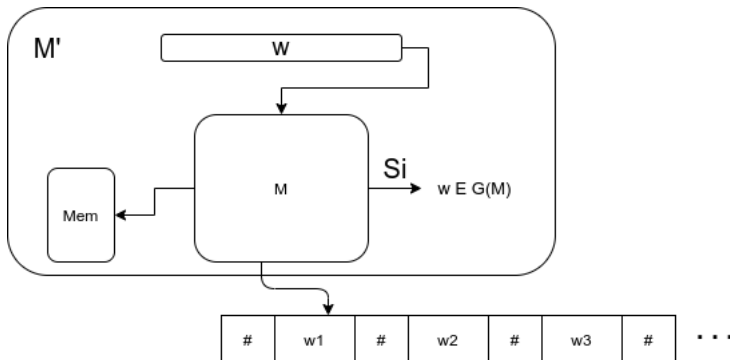
Si existe un MT  $M$  que reconoce  $A$ , se puede construir un enumerador  $E$  para  $A$ , Decir que  $s_1, s_2, \dots$  si lista de posibles strings de  $\Sigma^*$

- $E$  ignora el ingreso
- Repetir para  $i = 1, 2, \dots, m$
- Corre MT  $M$  para  $i$  pasos sobre cada ingreso  $s_1, s_2, \dots, s_i$
- Si acepta alguna computacion, se imprime la correspondiente  $s_i$ .



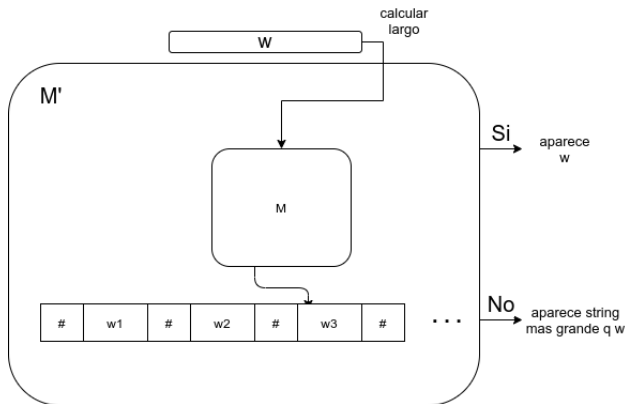
## Theorem

*Sea  $L$  es recursivamente enumerable, si y solo si  $L = G(M)$  para una máquina enumerable.*



## Theorem

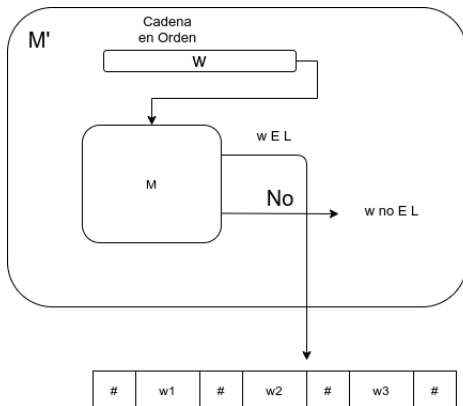
*Si  $L$  es recursivo, si y solo si  $L$  puede ser generado en orden por una maquina de turing*





## Theorem

*Si  $L$  es recursivo, si y solo si  $L$  puede ser generado en orden por una maquina de turing*



# Contenido

## 1 Variantes de las Máquinas de Turing

- Máquinas de Turing Stay-Put
- Máquinas de Turing con cintas infinitas por ambos lados
- Máquinas de Turing Multicabecal
- Máquinas de Turing Multidimensional
- Máquinas de Turing con Múltiples Tracks
- Máquinas de Turing Multicinta
- Máquinas de Turing no Deterministas
- Enumeradores

## 2 Indecibilidad

- Metodo de la Diagonalización
- Un Lenguaje Indecidable
- Un Lenguaje Turing No-Reconocible
- Otros Problemas Indecidibles

## 3 Conclusiones

## 4 Referencias



# Introducción

Turing escribió muchos programas, incluso trabajó en descifrar códigos para los ejércitos aliados durante la segunda guerra mundial. Rápidamente se dió cuenta de que algunos programas se 'colgaban' y se quedaban dando vueltas para siempre sin avanzar.

.

Si queremos escribir un programa "perfecto", que nunca se cuelgue, entonces una forma de hacerlo es probarlo con todas las entradas posibles, pero esto muchas veces no es práctico porque hay demasiadas combinaciones, pero además de no ser práctico, tiene un problema más de fondo y es que aún cuando transcurra mucho tiempo nunca podemos saber si el programa está calculando todavía o si se ha quedado 'colgado'

.

El principio de indecibilidad de Turing dice que no es posible escribir un programa que decida si otro programa cualquiera está correctamente escrito



# Teorema de la Indecibilidad

El teorema se basa en determinar si una máquina de Turing acepta una determinada cadena de entrada.

Lo llamamos  $A_{TM}$  por analogía con  $A_{DFA}$  y  $A_{CFG}$ . Pero, mientras  $A_{DFA}$  y  $A_{CFG}$  fueron decidibles,  $A_{TM}$  no lo es. Entonces:

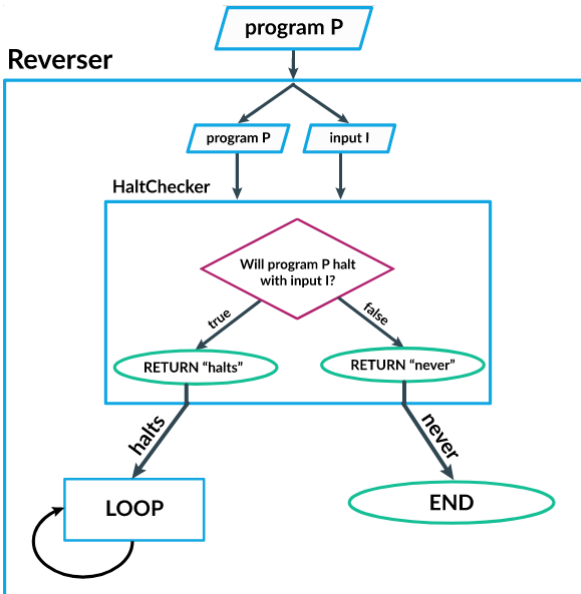
$$A_{TM} = \{\langle M, w \rangle \mid M \text{ es una TM y } M \text{ acepta } w\} \quad (1)$$

$U =$  Entrada  $\langle M, w \rangle$ , donde  $M$  es una MT y  $w$  es una cadena:

- ① Simulando  $M$  en una entrada  $w$
- ② Si  $M$ , entra en un estado de aceptación, 'acepta';  
Si  $M$ , entra en un estado de rechazo, 'rechazada';



# Flujo de un algoritmo Indecidable



# Metodo de la Diagonalización

En 1873 Georg Cantor estaba preocupado con el problema de medir los tamaños de conjuntos infinitos. Si tenemos dos conjuntos infinitos, ¿cómo podemos saber si uno es más grande que el otro o si son del mismo tamaño? Para conjuntos finitos, es sencillo, ya que simplemente debemos contar los elementos en un conjunto finito, y el número resultante es su tamaño. Pero si tratamos de contar los elementos de un conjunto infinito ¡nunca terminaremos!

Entonces, Cantor propuso una solución a este problema. Observó que dos los conjuntos finitos tienen el mismo tamaño si los elementos de un conjunto se pueden emparejar con el elementos del otro conjunto. Este método compara los tamaños de dos conjuntos infinitos sin tener que contarlos.



# Definición de Diagonalización

Suponga que tenemos los conjuntos  $A$  y  $B$  y una función  $f$  de  $A$  a  $B$ . Supongamos que  $f$  es uno a uno si nunca asigna dos elementos diferentes al mismo lugar, es decir, si  $f(a) \neq f(b)$  siempre que  $a \neq b$ . Se dice que  $f$  está en todos los elementos de  $B$ , es decir, si para cada  $b \in B$  hay un  $a \in A$  tal que  $f(a) = b$ . Tal que  $A$  y  $B$  son de igual tamaño si está de uno a uno, en la función  $f : A \rightarrow B$ . Una función esta representada uno sobre uno se llama correspondencia. En una correspondencia cada elemento de  $A$  se asigna a un elemento único de  $B$  y cada elemento de  $B$  tiene un elemento único de  $A$  mapeado. Una correspondencia es simplemente una forma de emparejar los elementos de  $A$  con los elementos de  $B$ .



# Emparejamiento I

Sea  $N$  el conjunto de números naturales  $\{1, 2, 3, \dots\}$  y sea  $E$  el conjunto de números naturales pares  $\{2, 4, 6, \dots\}$ . Usando la definición de Cantor de tamaño, podemos ver que  $N$  y  $E$  tienen el mismo tamaño. La correspondencia  $f$  mapeando  $N$  a  $E$  es simplemente  $f(n) = 2n$

$n$	$f(n)$
1	2
2	4
3	6
$\vdots$	$\vdots$

Figura: Emparejamiento de Números Pares





# Emparejamiento II

Si decimos que  $Q = \{\frac{m}{n} | m, n \in N\}$  sea el conjunto de números racionales positivos,  $Q$  parece ser mucho mayor que  $N$ . Sin embargo, estos dos conjuntos son del mismo tamaño según nuestra definición. Damos una correspondencia con  $N$  para mostrar que  $Q$  es contable. Una forma sencilla de hacerlo es enumerar todos los elementos de  $Q$ . Luego emparejamos el primer elemento de la lista con el número 1 de  $N$ , el segundo elemento de la lista con el número 2 de  $N$ , y así sucesivamente.

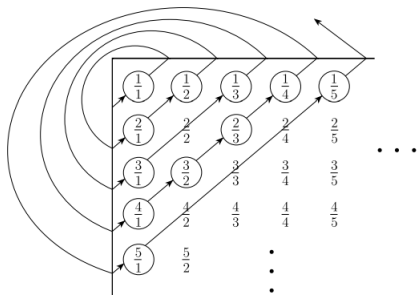


Figura: Emparejamiento de Divisibilidad



Supongamos una tabla infinita en la que:

- En la primera fila se colocan las cadenas de  $(0+1)^*$  en orden canónico
- En la primera columna se colocan las Maquinas de Turing ordenadas por el orden canónico de sus codificaciones
- *Para* :  $i, j \geq 1$ 
  - $(i, j) = 1$  si la cadena  $j$  es aceptada por la maquina que se codifica en binario como  $i \rightarrow w_j$  esta en  $L(M_i)$
  - $(i, j) = 0$  si la cadena  $j$  no es aceptada por la maquina que se codifica en binario como  $i \rightarrow w_j$  esta en  $L(M_i)$



# Lenguaje de Diagonalizacion II

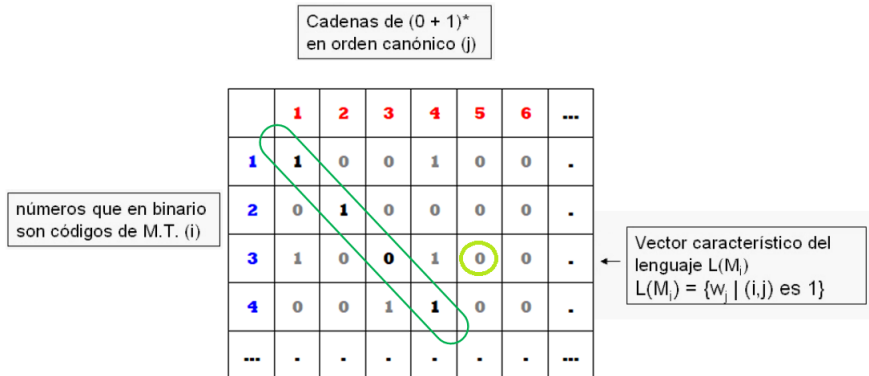


Figura: Matriz de Diagonalizacion



# Lenguaje de Diagonalizacion III

Consideremos la diagonal de la tabla anterior y consideremos

$$L_d = \{w_i : w_i \notin L(M_i)\}$$

- Si  $i(i, i) = 0, w_i \in L_d$
- Si  $i(i, i) = 1, w_i \notin L_d$

**$L_d$  no es recursivamente enumerable**

- Supongamos que si lo es  $\rightarrow \exists M : L(M) = L_d$
- En las filas está todas las M.T.  $\rightarrow \exists j : M_j = M$
- Entonces:
  - Si  $x_j \in L_d = L(M_j) \rightarrow (i, j) = 0 \rightarrow x_j \notin L(M_j) \text{ ó } x_j \notin L_d$
  - Si  $x_j \notin L_d = L(M_j) \rightarrow (i, j) = 1 \rightarrow x_j \in L(M_j) \text{ ó } x_j \in L_d$

Luego no existe una MT capaz de reconocer  $L_d$  y por tanto  $L_d$  no es recursivamente enumerable. Luego la consecuencia de  $L_d$  es:

- No recursivamente enumerable
- Recursivamente enumerable no recursivo



# Un Lenguaje Indecidable

Considerando el Teorema inicial [2] que dice :

## Theorem

$A_{TM}$  es indecible

( ver ecuación 2)

Asumimos

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ es una TM y } M \text{ acepta } w \} \quad (2)$$

## Demostración.

Asumimos que  $H$  es un decisor para  $A_{TM}$ . Con la entrada  $\langle M, w \rangle$  donde  $M$  es una Máquina de Turing y  $w$  es un string. ( ver ecuación 3 ) ☐

$$H(A_{TM}) = \begin{cases} \text{aceptado} & \text{si } M \text{ acepta } w \\ \text{rechazado} & \text{si } M \text{ no acepta } w. \end{cases}$$



# Probando que un Problema de Parada (H) es Indecidable

Construir una Máquina de Turing (MT)  $D$  que usa a  $H$  como subrutina: Llama a  $H$  para determinar como  $M$  se comporta en la entrada  $\langle M \rangle$  y sale el opuesto.

$D =$  “En el string de entrada  $\langle M \rangle$ , donde  $M$  es una MT:

1. Ejecutar  $H$  sobre la entrada  $\langle M, \langle M \rangle \rangle$
2. Salida como el opuesto de  $H$ , por ejemplo si  $H$  acepta, rechaza; si  $H$  rechaza, acepta”.

No confundir con la noción de ejecutar una Máquina con su propia descripción, en resumen,

$$D(\langle M \rangle) = \begin{cases} \text{aceptado} & \text{si } M \text{ no acepta } \langle M \rangle \\ \text{rechazado} & \text{si } M \text{ acepta } \langle M \rangle. \end{cases} \quad (4)$$



# Probando que un Problema de Parada (H) es Indecidable

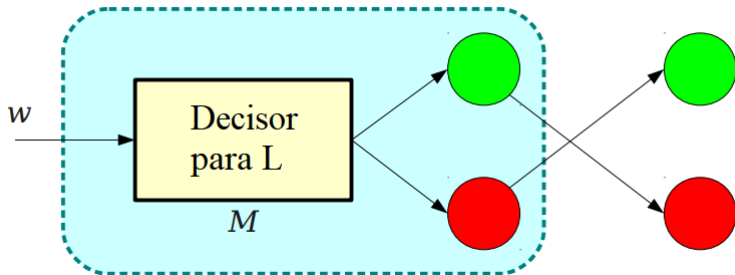


Figura: Ejemplo de salida opuesta a la entrada para un lenguaje  $L$



# Probando que un Problema de Parada (H) es Indecidable

¿Que ocurriría si ejecutamos  $D$  en  $\langle D \rangle$  como entrada?

$$D(\langle D \rangle) = \begin{cases} \text{aceptado} & \text{si } D \text{ no acepta } \langle D \rangle \\ \text{rechazado} & \text{si } D \text{ acepta } \langle D \rangle. \end{cases} \quad (5)$$

Lo que pasaria es que  $D$ , esta forzado a hacer lo opuesto, por lo tanto ni la MT de  $D$  ni la  $H$  pueden existir.





# ¿Dónde esta la Diagonalización?

El uso de la diagonalización puede ser visto si no construimos una tabla para toda Maquina de Turing  $M_0, M_1, \dots, M_n$  (filas) codificando las MT  $\langle M_0 \rangle, \langle M_1 \rangle \dots \langle M_n \rangle$  (columnas) como entradas.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...
$M_1$	<i>acepta</i>		<i>acepta</i>		
$M_2$	<i>acepta</i>	<i>acepta</i>	<i>acepta</i>	<i>acepta</i>	
$M_3$					...
$M_4$	<i>acepta</i>	<i>acepta</i>			
$\vdots$		$\vdots$			

**Cuadro:** Entrada  $i, j$  es *aceptada* si  $M_i$  acepta  $\langle M_j \rangle$ .



# ¿Dónde esta la Diagonalización?

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...
$M_1$	<i>acepta</i>	<i>rechaza</i>	<i>acepta</i>	<i>rechaza</i>	
$M_2$	<i>acepta</i>	<i>acepta</i>	<i>acepta</i>	<i>acepta</i>	
$M_3$	<i>rechaza</i>	<i>rechaza</i>	<i>rechaza</i>	<i>rechaza</i>	...
$M_4$	<i>acepta</i>	<i>acepta</i>	<i>rechaza</i>	<i>rechaza</i>	
$\vdots$		$\vdots$			

**Cuadro:** Entrada  $i, j$  es le valor de  $H$  con entrada  $\langle M_i, \langle M_j \rangle \rangle$ .



# ¿Dónde esta la Diagonalización?

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$	...
$M_1$	<u>acepta</u>	rechaza	acepta	rechaza		acepta	
$M_2$	acepta	<u>acepta</u>	acepta	acepta		acepta	
$M_3$	rechaza	rechaza	<u>rechaza</u>	rechaza	...	rechaza	...
$M_4$	acepta	acepta	rechaza	<u>rechaza</u>		acepta	
$\vdots$		$\vdots$			$\ddots$		
$D$	rechaza	rechaza	acepta	acepta		<u>?</u>	
$\vdots$		$\vdots$					$\ddots$

**Cuadro:** Si  $D$  es una contradicción que ocurre con “?” .

Las entradas  $(i,j)$  son aceptadas si  $M_i$  acepta  $M_j$ , sino lo rechaza:  $H(\langle M_i, \langle M_j \rangle)$



# Un Lenguaje Turing No-Reconocible

## Definition

Ahora exhibimos un lenguaje que no es aun Turing-reconocible.  $A_{TM}$  no es suficiente para este propósito debido a que mostramos que  $A_{TM}$  es Turing-reconocible. Decimos que un lenguaje es *co-Turing-reconocible* si es el complemento de un lenguaje Turing-reconocible.

## Theorem

*Un lenguaje es decidible si y solo si es Turing-reconocible y co-Turing-reconocible.*

## Demostración.

Son dos direcciones para probar, Primero si  $A$  es decidible, podemos decir que  $A$  y su complemento  $\bar{A}$  son Turing-reconocibles. Algun lenguaje decidible es Turing-reconocible y el complemento de un lenguaje decidible también es decidible. □

# Un Lenguaje Turing No-Reconocible

Tenemos  $M_1$  ser reconocedor para  $A$  y  $M_2$  ser reconocedor para  $\bar{A}$ . La siguiente MT  $M$  es un decisor para  $A$

$M =$  “ En la entrada  $w$

1. Ejecutar  $M_1$  y  $M_2$  con entrada  $w$  en paralelo.
2. Si  $M_1$  acepta, *acepta*; si  $M_2$  acepta, *rechaza*”.

Ejecutar dos máquinas en paralelo significa que  $M$  tiene dos cintas, tomando la simulacion en un paso para cada máquina y continua hasta que acepta. Cada string  $w$  esta en  $A$  o  $\bar{A}$ . Por lo tanto ya sea  $M_1$  o  $M_2$  debe aceptar  $w$ . Porque  $M$  se detiene cuando  $M_1$  y  $M_2$  acepta, por lo tanto si acepta todas los strings en  $A$  y rechaza todos los string no en  $A$ ;  $M$  es decisor para  $A$  y por lo tanto  $A$  es decidable.



# Un Lenguaje Turing No-Reconocible

## Corollary

$\overline{A_{TM}}$  no es Turing-reconocible.

## Demostración.

Sabemos que  $A_{TM}$  es Turing-reconocible, Si  $\overline{A_{TM}}$  también es Turing-reconocible.  $\overline{A_{TM}}$  debería ser decidible, pero por el teorema inicial nos dice que  $A_{TM}$  no es decidible, entonces  $\overline{A_{TM}}$  no debería ser Turing-reconocible. □



# Décimo problema de Hilbert

Hay una gran cantidad de problemas interesantes que resultan ser indecidibles, algunos de los cuales aparentemente no tienen ninguna relación con las Máquinas Turing

Dado un polinomio multivariado con coeficientes enteros [3] [4], ¿Evaluar a 0 en algún punto entero? Por Ejemplo:

$$f(x, y, z) = 5x^2y + 3xyz - 7xy + z^3 + 2 \quad (6)$$

Tiene  $f(1, -3, 1) = 0$  Esto es conocido como el Décimo Problema de Hilbert



# Problema de la Post Correspondencia

Se nos da una colección  $S$  de dominó [5], cada uno con dos cadenas de algún alfabeto  $\Sigma$  (Uno en la mitad superior del dominó, uno en la baja). Por ejemplo:

$$S = \left\{ \left[ \begin{array}{c} a \\ ab \end{array} \right], \left[ \begin{array}{c} b \\ a \end{array} \right], \left[ \begin{array}{c} abc \\ c \end{array} \right] \right\} \quad (7)$$

El problema es determinar si, alineando dominós de  $S$  (con repeticiones permitidas) podemos hacer que las cadenas concatenadas en la parte superior de las fichas de dominó sean igual a las cuerdas concatenados a las de abajo, por ejemplo.

$$S = \left[ \begin{array}{c} a \\ ab \end{array} \right] \left[ \begin{array}{c} b \\ a \end{array} \right] \left[ \begin{array}{c} a \\ ab \end{array} \right] \left[ \begin{array}{c} abc \\ c \end{array} \right] \quad (8)$$

Seria una solución válida





# Dominó o Baldosas Wang

Es un cuadrado unitario con bordes de colores. Dado un conjunto  $S$  de mosaicos de Wang [6] cuyo problema es determinar si los mosaicos escogidos de  $S$  (sin rotaciones ni reflejos) se puede juntar borde a borde para enlosar el plano, de modo que los bordes contiguos de los dominós o baldosas tengan el mismo color. Por ejemplo, el siguiente conjunto  $S$  satisface esta propiedad:

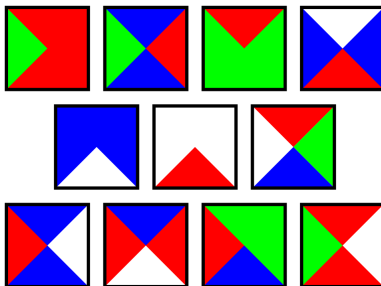


Figura: 11 dominós



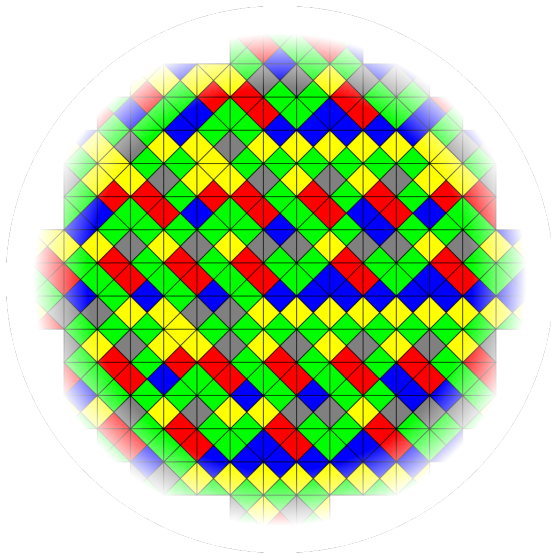


Figura: 13 dominós

# Contenido

## 1 Variantes de las Máquinas de Turing

- Máquinas de Turing Stay-Put
- Máquinas de Turing con cintas infinitas por ambos lados
- Máquinas de Turing Multicabecal
- Máquinas de Turing Multidimensional
- Máquinas de Turing con Múltiples Tracks
- Máquinas de Turing Multicinta
- Máquinas de Turing no Deterministas
- Enumeradores

## 2 Indecibilidad

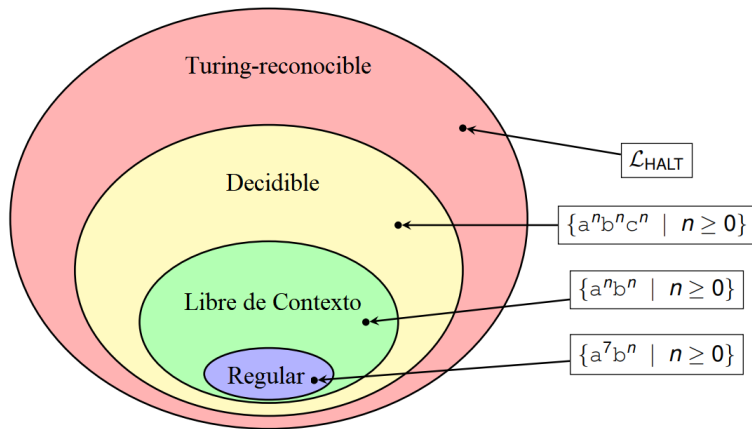
- Metodo de la Diagonalización
- Un Lenguaje Indecidable
- Un Lenguaje Turing No-Reconocible
- Otros Problemas Indecidibles

## 3 Conclusiones

## 4 Referencias



# Conclusiones - Jerarquía de los lenguajes

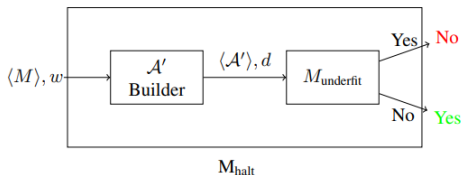


**Figura:** L es una maquina de Turing-reconocible, ver que la jerarquía esta desde un lenguaje regular, lo que esta fuera es considerado un lenguaje indecible



# Conclusiones - Underfit es indecidible

$L_{underfit} = \{\langle A \rangle, d \mid A \in S, A \text{ underfits } d \text{ en todas la iteraciones}\}$  es indecidible



**Figura:** Teorema de investigación que basado en el problema de Parada [7] define que un algoritmo de aprendizaje entraría en un estado underfit con ciertos dataset, pero esta investigación requiere ser reforzada con prueba de probabilidad para dar una validez



# Contenido

## 1 Variantes de las Máquinas de Turing

- Máquinas de Turing Stay-Put
- Máquinas de Turing con cintas infinitas por ambos lados
- Máquinas de Turing Multicabecal
- Máquinas de Turing Multidimensional
- Máquinas de Turing con Múltiples Tracks
- Máquinas de Turing Multicinta
- Máquinas de Turing no Deterministas
- Enumeradores

## 2 Indecibilidad







- Metodo de la Diagonalización
- Un Lenguaje Indecidable
- Un Lenguaje Turing No-Reconocible
- Otros Problemas Indecidibles

## 3 Conclusiones

## 4 Referencias



# References I

-  S. H. Rodger, “JFLAP.” <https://www.jflap.org/>, 1993.  
[Java Formal Languages and Automata Package].
-  M. Sipser, *Introduction to the Theory of Computation*.  
Boston, MA: Course Technology, third ed., 2013.
-  M. Davis, “Arithmetical problems and recursively enumerable predicates,” *J. Symb. Log.*, vol. 18, no. 1, pp. 33–41, 1953.
-  Y. V. Matiyasevich, “Existential arithmetization of diophantine equations,” *Ann. Pure Appl. Log.*, vol. 157, no. 2-3, pp. 225–233, 2009.
-  M. Davis, “Unsolvable problems,” in *Handbook of Mathematical Logic* (J. Barwise, ed.), pp. 567–594, Amsterdam: North-Holland, 1977.
-  K. C. II, “An aperiodic set of 13 wang tiles,” *Discret. Math.*, vol. 160, no. 1-3, pp. 245–251, 1996.



# References II



S. Sehra, D. Flores, and G. D. Montañez, “Undecidability of underfitting in learning algorithms,” *ArXiv*, vol. abs/2102.02850, 2021.

