

# Teoria da Computação

---

**Decidibilidade**

**Guilherme Meira**

# Agenda

1. Algoritmos

2. Indecidibilidade

# Algoritmos

O que é um algoritmo?

# Algoritmos

O que é um algoritmo?

- Informalmente: é uma coleção de instruções simples para realizar alguma tarefa
  - **Exemplo:** uma receita
- Até o século XX não havia uma definição precisa do que é um algoritmo

# Algoritmos

## *Problemas de Hilbert*

- Em 1900, o matemático David Hilbert apresentou 23 problemas matemáticos que seriam desafios para o novo século



# Algoritmos

## *Problemas de Hilbert*

- **Polinômios:**

- Um polinômio é uma soma de termos, onde cada termo é o produto de algumas variáveis e uma constante chamada de **coeficiente**

- **Exemplo:**

$$6 \cdot x \cdot x \cdot x \cdot y \cdot z \cdot z = 6x^3yz^2$$

é um termo com coeficiente 6 e

$$6x^3yz^2 + 3xy^2 - x^3 - 10$$

é um polinômio de 4 termos nas variáveis  $x$ ,  $y$  e  $z$

- Vamos considerar apenas coeficientes inteiros

# Algoritmos

## *Problemas de Hilbert*

- **Polinômios:**

- A **raiz** de um polinômio é um conjunto de valores de suas variáveis que faz com que o valor do polinômio seja zero
- **Exemplo:** uma raiz de

$$6x^3yz^2 + 3xy^2 - x^3 - 10$$

é  $x = 5, y = 3$  e  $z = 0$

- Essa raiz é **inteira** porque todas as variáveis recebem valores inteiros
- Alguns polinômios possuem raízes inteiras e outros não

# Algoritmos

## *Problemas de Hilbert*

- O décimo problema de Hilbert envolvia descrever um algoritmo que determinasse se um polinômio tem ou não uma raiz inteira
  - Na época, ele não usou o termo “algoritmo”, mas “um processo de acordo com o qual possa ser determinado por um número finito de operações”
- Hilbert aparentemente assumiu que o algoritmo deveria existir e que só precisava ser descoberto
- Hoje sabemos que não existe um algoritmo que resolva esse problema
- Chegar a essa conclusão exigiu a criação de uma definição precisa de um algoritmo



# Algoritmos

## *A Tese de Church-Turing*

- Em 1936 foram publicados os artigos de Alonzo Church e Alan Turing, com suas definições de algoritmo
- Church propôs um sistema chamado Cálculo Lambda e Turing propôs as Máquinas de Turing
- Posteriormente, foi provado que as duas definições eram equivalentes
- A **Tese de Church-Turing** conecta a noção informal de algoritmos e a definição precisa por meio de Máquinas de Turing

# Algoritmos

## *A Tese de Church-Turing*

- Vamos descrever o décimo problema de Hilbert na nossa terminologia. Seja:

$$D = \{p \mid p \text{ é um polinômio com raízes inteiras}\}$$

- O décimo problema de Hilbert que saber se  $D$  é decidível (não é)
- $D$  é Turing-reconhecível?

# Algoritmos

## *A Tese de Church-Turing*

- Vamos considerar uma variante mais simples do problema: polinômios com somente uma variável, como  $4x^3 - 2x^2 + x - 7$

$$D_1 = \{p \mid p \text{ é um polinômio em } x \text{ com uma raiz real}\}$$

- A Máquina de Turing  $M_1$  que reconhece  $D_1$  é:  
 $M_1 =$ “Para uma entrada  $\langle p \rangle$  onde  $p$  é um polinômio em  $X$ :
  1. Calcule o valor de  $p$  para os valores  $0, 1, -1, 2, -2, 3, -3, \dots$ . Se em algum ponto, o valor do polinômio for zero, **aceite**”

# Algoritmos

## *A Tese de Church-Turing*

- Se  $p$  tem uma raiz inteira,  $M_1$  eventualmente a encontrará e aceitará
- Se  $p$  não tem uma raiz inteira,  $M_1$  executará para sempre
- Para o caso de múltiplas variáveis, podemos utilizar uma máquina semelhante, que testa todas as combinações de variáveis
- Ambas as máquinas reconhecem, mas não decidem  $D$
- No caso de  $D_1$ , podemos converter  $M_1$  em um decisor porque é possível calcular os limites máximos até onde a raiz pode estar. Para o caso de múltiplas variáveis, é impossível

# Algoritmos

## *Descrição de algoritmos*

- De agora para frente, vamos nos afastar das definições de baixo nível da Máquina de Turing e focarmos apenas nos algoritmos
- Descreveremos algoritmos em alto nível, não nos preocupando com a fita ou a cabeça da Máquina de Turing

# Algoritmos

## *Descrição de algoritmos*

Considere a linguagem a seguir:

$$A = \{ \langle G \rangle \mid G \text{ é um grafo conectado não-direcionado} \}$$

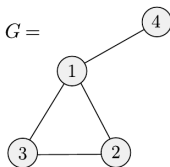
# Algoritmos

## Descrição de algoritmos

Considere a linguagem a seguir:

$$A = \{\langle G \rangle \mid G \text{ é um grafo conectado não-direcionado}\}$$

- Utilizaremos esse símbolo para indicar **uma codificação de  $G$** , isto é, uma forma de representar  $G$  em uma string
- **Exemplo:**  $\langle G \rangle = (1, 2, 3, 4)((1, 2), (2, 3), (3, 1), (1, 4))$



# Algoritmos

## *Descrição de algoritmos*

Considere a linguagem a seguir:

$$A = \{ \langle G \rangle \mid G \text{ é um grafo conectado não-direcionado} \}$$

- Um **grafo** é uma estrutura composta de um conjunto de **nós** conectados por um conjunto de **arestas**



# Algoritmos

## *Descrição de algoritmos*

Considere a linguagem a seguir:

$$A = \{ \langle G \rangle \mid G \text{ é um grafo conectado não-direcionado} \}$$

- O grafo é **não-direcionado** quando as arestas podem ser percorridas nos dois sentidos

# Algoritmos

## *Descrição de algoritmos*

Considere a linguagem a seguir:

$$A = \{ \langle G \rangle \mid G \text{ é um grafo conectado não-direcionado} \}$$

- O grafo é **conectado** se é possível chegar a qualquer nó a partir de qualquer outro nó

# Algoritmos

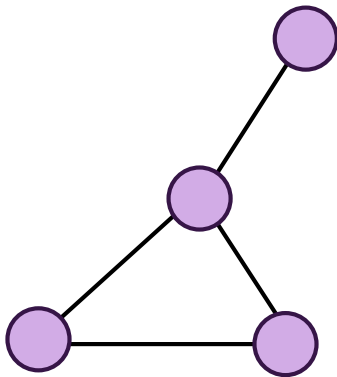
## *Descrição de algoritmos*

$M =$  “Para uma entrada  $\langle G \rangle$ , que é a codificação de um grafo  $G$ :

1. Selecione o primeiro nó de  $G$  e marque-o
2. Repita até que nenhum novo nó seja marcado:
  - Para cada nó de  $G$ , marque-o se existe uma aresta entre ele e um nó já marcado
3. Percorra todos os nós de  $G$  para determinar se eles estão todos marcados. Se sim, aceite, caso contrário, rejeite”

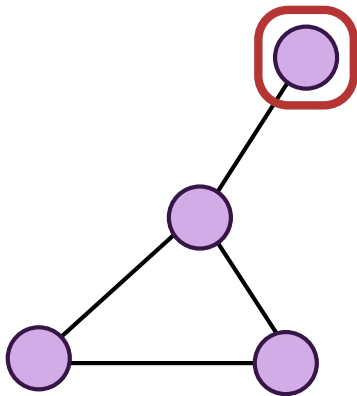
# Algoritmos

*Descrição de algoritmos*



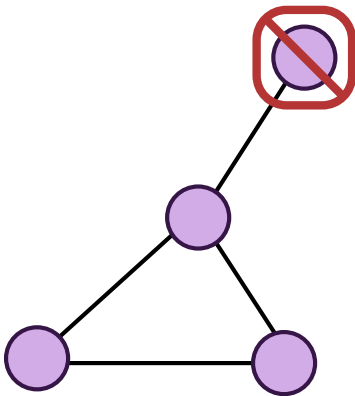
# Algoritmos

*Descrição de algoritmos*



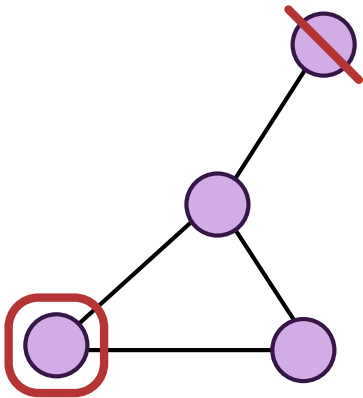
# Algoritmos

*Descrição de algoritmos*



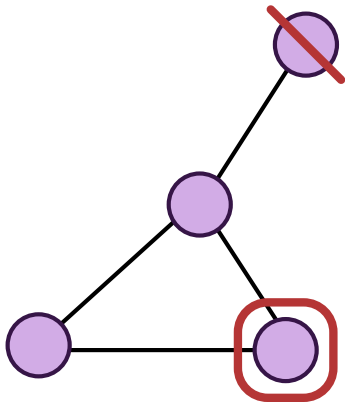
# Algoritmos

*Descrição de algoritmos*



# Algoritmos

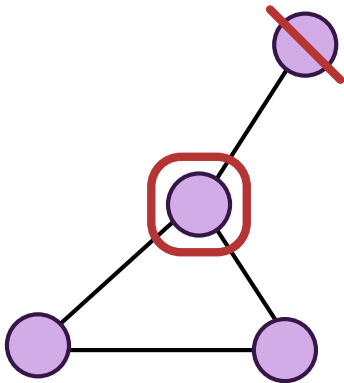
*Descrição de algoritmos*





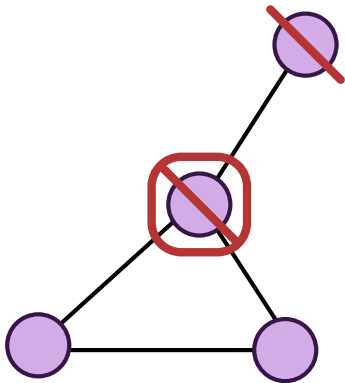
# Algoritmos

*Descrição de algoritmos*



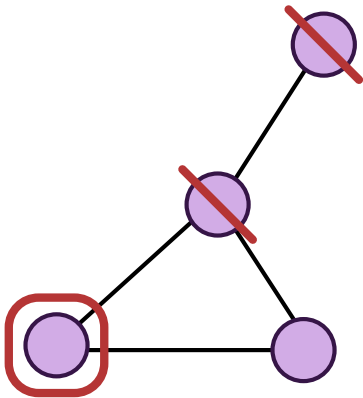
# Algoritmos

*Descrição de algoritmos*



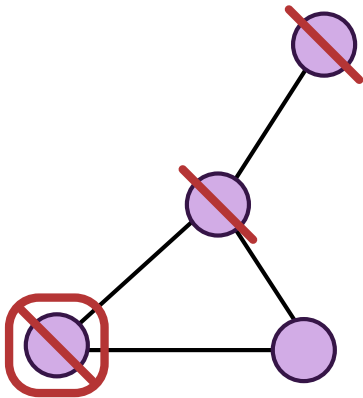
# Algoritmos

*Descrição de algoritmos*



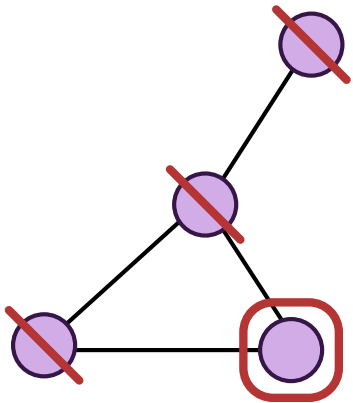
# Algoritmos

*Descrição de algoritmos*



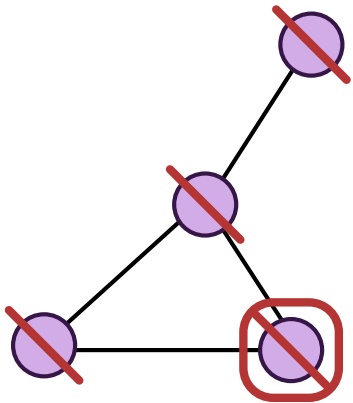
# Algoritmos

*Descrição de algoritmos*



# Algoritmos

*Descrição de algoritmos*



# Agenda

1. Algoritmos

2. Indecidibilidade

# Indecidibilidade

- Computadores são tão poderosos que parecem ser capazes de resolver qualquer problema
- Existem alguns problemas que não podem ser resolvidos



# Indecidibilidade

Considere a linguagem:

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ é uma Máquina de Turing e } M \text{ aceita } w \}$$

# Indecidibilidade

Considere a linguagem:

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ é uma Máquina de Turing e } M \text{ aceita } w \}$$

**Teorema**

$A_{TM}$  é indecidível.

# Indecidibilidade

Primeiramente,  $A_{TM}$  é Turing-reconhecível?

# Indecidibilidade

Primeiramente,  $A_{TM}$  é Turing-reconhecível?

$U$  = “Para uma entrada  $\langle M, w \rangle$ , onde  $M$  é uma Máquina de Turing e  $w$  é uma string:

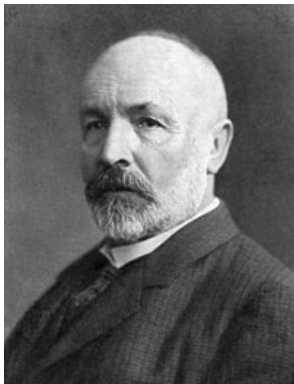
1. Simule  $M$  com uma entrada  $w$
2. Se  $M$  aceitar, aceite. Se  $M$  rejeitar, rejeite”

Se  $M$  entrar em loop,  $U$  também entra em loop (não é um decisor).

# Indecidibilidade

## *O método da diagonalização*

- Para provar a indecidibilidade de  $A_{TM}$ , utilizamos uma técnica chamada **diagonalização**
- Esse método foi descoberto por Georg Cantor, em 1873



# Indecidibilidade

## *O método da diagonalização*

- Considere dois conjuntos,  $A = \{1, 3, 5, 7, 9\}$  e  $B = \{2, 3, 4\}$
- Qual dos dois é maior?

# Indecidibilidade

## *O método da diagonalização*

- Considere dois conjuntos,  $A = \{1, 3, 5, 7, 9\}$  e  $B = \{2, 3, 4\}$
- Qual dos dois é maior?
- $A$  tem 5 elementos, enquanto  $B$  tem apenas 3.  $A$  é maior

# Indecidibilidade

## *O método da diagonalização*

- Considere dois conjuntos,  $A = \{1, 3, 5, 7, 9\}$  e  $B = \{2, 3, 4\}$
- Qual dos dois é maior?
- $A$  tem 5 elementos, enquanto  $B$  tem apenas 3.  $A$  é maior
- E se  $A$  e  $B$  forem infinitos?



# Indecidibilidade

## *O método da diagonalização*

- No caso de conjuntos infinitos, não podemos utilizar contagem
- Cantor considera que dois conjuntos tem o mesmo tamanho se pudermos **emparelhar todos os elementos de um conjunto nos elementos do outro**

# Indecidibilidade

## *O método da diagonalização*

Seja  $\mathcal{N}$  o conjunto dos números naturais  $\{1, 2, 3, \dots\}$  e  $\mathcal{E}$  o conjunto dos números naturais pares  $\{2, 4, 6, \dots\}$ . Qual conjunto é maior?

# Indecidibilidade

## *O método da diagonalização*

Seja  $\mathcal{N}$  o conjunto dos números naturais  $\{1, 2, 3, \dots\}$  e  $\mathcal{E}$  o conjunto dos números naturais pares  $\{2, 4, 6, \dots\}$ . Qual conjunto é maior?

A **correspondência**  $f$  que mapeia  $\mathcal{N}$  para  $\mathcal{E}$  é  $f(n) = 2n$ .

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

# Indecidibilidade

*O método da diagonalização*

## Definição

Um conjunto  $A$  é **contável** se ele for finito ou se tiver o mesmo tamanho que  $\mathcal{N}$ .

# Indecidibilidade

## *O método da diagonalização*

Considere o conjunto dos números racionais positivos:

$$\mathcal{Q} = \left\{ \frac{m}{n} \mid m, n \in \mathcal{N} \right\}$$

$\mathcal{Q}$  é maior que  $\mathcal{N}$ ?

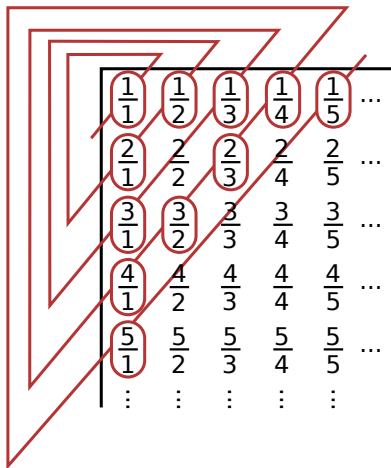
# Indecidibilidade

*O método da diagonalização*

$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	...
$\frac{2}{1}$	$\frac{2}{2}$	$\frac{2}{3}$	$\frac{2}{4}$	$\frac{2}{5}$	...
$\frac{3}{1}$	$\frac{3}{2}$	$\frac{3}{3}$	$\frac{3}{4}$	$\frac{3}{5}$	...
$\frac{4}{1}$	$\frac{4}{2}$	$\frac{4}{3}$	$\frac{4}{4}$	$\frac{4}{5}$	...
$\frac{5}{1}$	$\frac{5}{2}$	$\frac{5}{3}$	$\frac{5}{4}$	$\frac{5}{5}$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

# Indecidibilidade

O método da diagonalização



$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	...
$\frac{2}{1}$	$\frac{2}{2}$	$\frac{2}{3}$	$\frac{2}{4}$	$\frac{2}{5}$	...
$\frac{3}{1}$	$\frac{3}{2}$	$\frac{3}{3}$	$\frac{3}{4}$	$\frac{3}{5}$	...
$\frac{4}{1}$	$\frac{4}{2}$	$\frac{4}{3}$	$\frac{4}{4}$	$\frac{4}{5}$	...
$\frac{5}{1}$	$\frac{5}{2}$	$\frac{5}{3}$	$\frac{5}{4}$	$\frac{5}{5}$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

# Indecidibilidade

## *O método da diagonalização*

Considere, agora, o conjunto  $\mathcal{R}$  dos números reais. São números reais:

- $\pi = 3.1415926\dots$
- $\sqrt{2} = 1.4142135\dots$

O conjunto dos reais é contável?

### Teorema

$\mathcal{R}$  é incontável.



# Indecidibilidade

## *O método da diagonalização*

Vamos tentar mapear todos os reais para os naturais:

<b>n</b>	<b>f(n)</b>
1	3.14159...
2	55.55555...
3	0.12345...
4	0.50000...
$\vdots$	$\vdots$

Todos os números reais estão listados aqui?

# Indecidibilidade

*O método da diagonalização*

n	f(n)
1	3. <b>1</b> 4159...
2	55.5 <b>5</b> 555...
3	0.12 <b>3</b> 45...
4	0.500 <b>0</b> 0...
⋮	⋮

O número  $x = 0.4641\dots$  não está na tabela.

# Indecidibilidade

*O método da diagonalização*

O conjunto de Máquinas de Turing é contável?

# Indecidibilidade

## *O método da diagonalização*

O conjunto de Máquinas de Turing é contável?

- Toda Máquina de Turing pode ser representada por uma string  $\langle M \rangle$
- Se representamos cada Máquina de Turing com uma string de um alfabeto  $\Sigma$ , podemos listar todas as strings utilizando esse alfabeto
- Começamos pelas strings de comprimento 0, depois 1, depois 2 e assim sucessivamente

## Indecidibilidade

*O método da diagonalização*

**Exemplo:** suponha que  $\Sigma = \{0, 1\}$ .

n	f(n)
1	$\epsilon$
2	0
3	1
4	00
5	01
6	10
7	11
8	100
$\vdots$	$\vdots$

# Indecidibilidade

*O método da diagonalização*

Portanto, o conjunto das Máquinas de Turing é **contável**.

E o conjunto das linguagens?

# Indecidibilidade

## *O método da diagonalização*

Vamos começar olhando o conjunto  $\mathcal{B}$  de todas as sequências binárias infinitas.  $\mathcal{B}$  é contável ou incontável?

# Indecidibilidade

## *O método da diagonalização*

Vamos começar olhando o conjunto  $\mathcal{B}$  de todas as sequências binárias infinitas.  $\mathcal{B}$  é contável ou incontável?

n	f(n)
1	100101010...
2	010110101...
3	010110001...
4	100101110...
$\vdots$	$\vdots$

$x = 0010\dots$  não está na tabela.  $\mathcal{B}$  é incontável.



# Indecidibilidade

## *O método da diagonalização*

- Considere um alfabeto  $\Sigma$ . O conjunto de todas as palavras formadas por esse alfabeto é  $\Sigma^* = \{s_1, s_2, s_3, \dots\}$ .
- Considere o conjunto  $\mathcal{L}$  de todas as linguagens no alfabeto  $\Sigma$
- Cada linguagem  $A \in \mathcal{L}$  tem uma sequência equivalente em  $\mathcal{B}$ : o  $i$ -ésimo bit da sequência é 1 se  $s_i \in A$  e 0 se  $s_i \notin A$

# Indecidibilidade

## *O método da diagonalização*

**Exemplo:**  $\Sigma = \{0, 1\}$  e  $A$  é a linguagem de todas as palavras que começam com zero

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$$

$$A = \{0, 00, 01, 000, 001, \dots\}$$

$$\chi_A = 010110011\dots$$

# Indecidibilidade

## *O método da diagonalização*

**Exemplo:**  $\Sigma = \{0, 1\}$  e  $A$  é a linguagem de todas as palavras que começam com zero

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$$

$$A = \{0, 00, 01, 000, 001, \dots\}$$

$$\chi_A = 010110011\dots$$

Podemos mapear cada elemento de  $\mathcal{L}$  em um elemento de  $\mathcal{B}$ .  
Vimos que  $\mathcal{B}$  é incontável, logo,  $\mathcal{L}$  também é **incontável**.

# Indecidibilidade

## *O método da diagonalização*

O que sabemos até agora:

- O conjunto de todas as Máquinas de Turing é **contável**
- Uma Máquina de Turing reconhece uma única linguagem, logo, o conjunto das linguagens reconhecíveis é **contável**
- O conjunto de todas as linguagens é **incontável**

Logo...

# Indecidibilidade

## *O método da diagonalização*

O que sabemos até agora:

- O conjunto de todas as Máquinas de Turing é **contável**
- Uma Máquina de Turing reconhece uma única linguagem, logo, o conjunto das linguagens reconhecíveis é **contável**
- O conjunto de todas as linguagens é **incontável**

Logo...

### Corolário

Algumas linguagens não são Turing-reconhecíveis.

# Indecidibilidade

*Uma linguagem indecidível*

Agora, podemos provar que a linguagem  $A_{TM}$  é indecidível:

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ é uma Máquina de Turing e } M \text{ aceita } w \}$$

# Indecidibilidade

## *Uma linguagem indecidível*

Vamos utilizar uma **prova por contradição**. Supomos que algo existe, e mostramos que isso gera contradição.

- Suponha que  $H$  é um decisor para  $A_{TM}$
- Para uma entrada  $\langle M, w \rangle$ , onde  $M$  é uma Máquina de Turing e  $w$  é uma string,  $H$  para e aceita se  $M$  aceita  $w$ . Além disso,  $H$  para e rejeita se  $M$  não aceitar  $w$  ( $M$  rejeita ou entra em loop).  
Em outras palavras:

$$H(\langle M, w \rangle) = \begin{cases} \text{aceita se } M \text{ aceita} \\ \text{rejeita se } M \text{ não aceita} \end{cases}$$

# Indecidibilidade

## *Uma linguagem indecidível*

- Agora, construa uma Máquina de Turing  $D$  que utilize  $H$  como uma subrotina
- $D$  chama  $H$  para determinar o que  $M$  faz quando recebe como entrada sua própria descrição  $\langle M \rangle$ . Quando  $D$  recebe uma resposta de  $H$ , ela faz o oposto, isto é, se  $H$  aceita,  $D$  rejeita e vice-versa

$D =$  “Para uma entrada  $\langle M \rangle$ , onde  $M$  é uma Máquina de Turing:

1. Rode  $H$  na entrada  $\langle M, \langle M \rangle \rangle$
2. Se  $H$  aceita, **rejeite**. Se  $H$  rejeita, **aceite**”



# Indecidibilidade

*Uma linguagem indecidível*

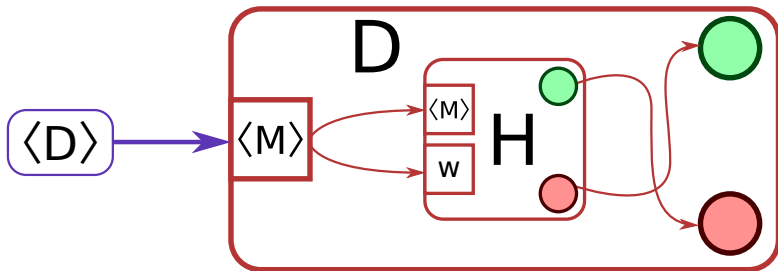
Em outras palavras:

$$D(\langle M \rangle) = \begin{cases} \text{aceita se } M \text{ não aceita } \langle M \rangle \\ \text{rejeita se } M \text{ aceita } \langle M \rangle \end{cases}$$

O que acontece quando passamos a descrição  $\langle D \rangle$  como entrada para  $D$ ?

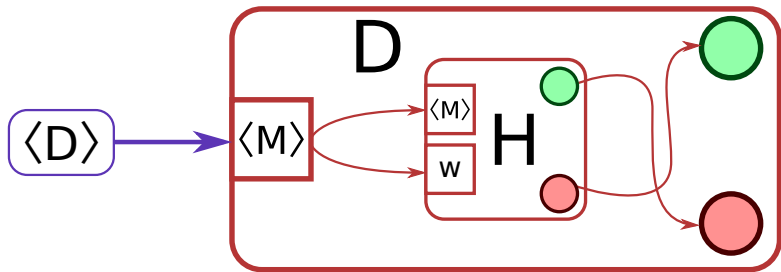
# Indecidibilidade

*Uma linguagem indecidível*



## Indecidibilidade

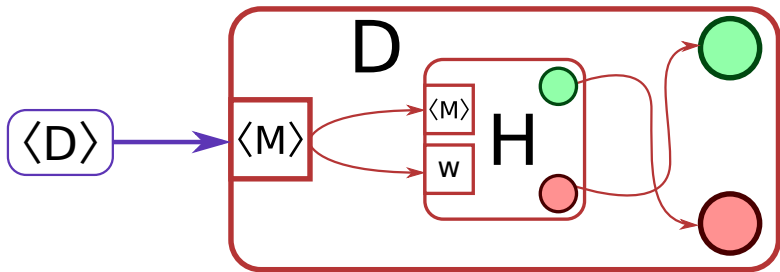
*Uma linguagem indecidível*



Contradição! Portanto,  $H$  e  $D$  não podem existir.

## Indecidibilidade

*Uma linguagem indecidível*



Contradição! Portanto,  $H$  e  $D$  não podem existir.

<https://www.youtube.com/watch?v=92WHN-pAFCs>

# Indecidibilidade

*Uma linguagem indecidível*

Onde entra a diagonalização nessa prova?

# Indecidibilidade

## *Uma linguagem indecidível*

Onde entra a diagonalização nessa prova?

Vamos listar todas as Máquinas de Turing e todas as descrições dessas máquinas. Na tabela abaixo, vemos o resultado de passar a descrição de uma máquina como parâmetro para outra:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\dots$
$M_1$	<i>aceita</i>		<i>aceita</i>		
$M_2$	<i>aceita</i>	<i>aceita</i>	<i>aceita</i>	<i>aceita</i>	
$M_3$					
$M_4$	<i>aceita</i>	<i>aceita</i>			
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

## Indecidibilidade

*Uma linguagem indecidível*

O resultado de rodar  $H$  com as entradas da tabela anterior são representados abaixo:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\dots$
$M_1$	<i>aceita</i>	<i>rejeita</i>	<i>aceita</i>	<i>rejeita</i>	
$M_2$	<i>aceita</i>	<i>aceita</i>	<i>aceita</i>	<i>aceita</i>	
$M_3$	<i>rejeita</i>	<i>rejeita</i>	<i>rejeita</i>	<i>rejeita</i>	
$M_4$	<i>aceita</i>	<i>aceita</i>	<i>rejeita</i>	<i>rejeita</i>	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

## Indecidibilidade

### *Uma linguagem indecidível*

Como  $D$  é uma Máquina de Turing, ela deve aparecer nessa tabela

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\dots$	$\langle D \rangle$	$\dots$
$M_1$	<u>aceita</u>	rejeita	aceita	rejeita	$\dots$	aceita	
$M_2$	aceita	<u>aceita</u>	aceita	aceita	$\dots$	aceita	
$M_3$	rejeita	rejeita	<u>rejeita</u>	rejeita	$\dots$	rejeita	
$M_4$	aceita	aceita	rejeita	<u>rejeita</u>	$\dots$	aceita	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
$D$	rejeita	rejeita	aceita	aceita	$\dots$	?	

A contradição ocorre no “?”, onde a saída da máquina deve ser o oposto dela mesma.



# Indecidibilidade

## *Uma linguagem irreconhecível*

Até agora, vimos que existem linguagens que não podem ser decididas (como  $A_{TM}$ ). Existem linguagens que sequer podem ser reconhecidas. Antes de vermos uma dessas linguagens, precisamos conhecer alguns conceitos novos:

- O **complemento** de uma linguagem é a linguagem formada por todas as palavras que **não** pertencem a uma linguagem
- Uma linguagem é **co-Turing-reconhecível** se o seu complemento é Turing-reconhecível

# Indecidibilidade

## *Uma linguagem irreconhecível*

Até agora, vimos que existem linguagens que não podem ser decididas (como  $A_{TM}$ ). Existem linguagens que sequer podem ser reconhecidas. Antes de vermos uma dessas linguagens, precisamos conhecer alguns conceitos novos:

- O **complemento** de uma linguagem é a linguagem formada por todas as palavras que **não** pertencem a uma linguagem
- Uma linguagem é **co-Turing-reconhecível** se o seu complemento é Turing-reconhecível

## Teorema

Uma linguagem é decidível se, e somente se ela é Turing-reconhecível e co-Turing-reconhecível.

# Indecidibilidade

*Uma linguagem irreconhecível*

Precisamos provar duas direções desse teorema.

- 1) Se  $A$  é decidível, então  $A$  e  $\bar{A}$  são Turing-reconhecíveis

Toda linguagem decidível é Turing-reconhecível, e o complemento de uma linguagem decidível também é decidível.

# Indecidibilidade

## *Uma linguagem irreconhecível*

- 2) Se  $A$  e  $\bar{A}$  são Turing-reconhecíveis, então  $A$  é decidível

Seja  $M_1$  um reconhecedor de  $A$  e  $M_2$  um reconhecedor de  $\bar{A}$ . A Máquina de Turing a seguir é um decisor de  $A$ :  $M$  = “Para uma entrada  $w$ :

1. Execute as máquinas  $M_1$  e  $M_2$  em paralelo com a entrada  $w$
2. Se  $M_1$  aceitar, **aceite**. Se  $M_2$  aceitar, **rejeite**”

# Indecidibilidade

*Uma linguagem irreconhecível*

## Corolário

$\overline{A_{TM}}$  não é Turing-reconhecível.

Sabemos que  $A_{TM}$  é Turing-reconhecível. Se  $\overline{A_{TM}}$  também fosse Turing-reconhecível,  $A_{TM}$  seria decidível. Sabemos que  $A_{TM}$  não é decidível, portanto,  $\overline{A_{TM}}$  não pode ser Turing-reconhecível.