ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 15

O Problema da Parada

Profa. Ariane Machado Lima
ariane.machado@usp.br

Na última aula

- Tese de Church-Turing
- Problemas computacionais descritos como linguagens
  - Ex: verificar se um grafo G1 é subgrafo de um grafo G2
- Linguagens (problemas) decidíveis
- Linguagens regulares e linguagens livres de contexto são decidíveis

Limites da computação

- Existem problemas que são Turing-reconhecíveis mas NÃO Turing-decidíveis
- Existem problemas que NÃO são Turing-reconhecíveis? (Insolúveis)

Limites da computação

- Existem problemas que são Turing-reconhecíveis mas NÃO Turing-decidíveis
- Existem problemas que NÃO são Turing-reconhecíveis? (Insolúveis)

SIM!

Ex: verificação de software

Limites da computação

- Existem problemas que são Turing-reconhecíveis mas NÃO Turing-decidíveis
- Existem problemas que NÃO são Turing-reconhecíveis? (Insolúveis)

SIM!

Ex: verificação de software

- Para provar, precisamos primeiro de alguns resultados da Matemática

Determinação de tamanho de conjuntos

- Comparar o tamanho de conjuntos finitos é fácil
- E para conjuntos infinitos?
- Georg Cantor: dois conjuntos infinitos têm o mesmo tamanho que seus elementos puderem ser **emparelhados**
  - $f : A \rightarrow B$, onde f é uma função bijetora (uma correspondência)

DEFINIÇÃO 4.12

Suponha que tenhamos os conjuntos A e B e uma função f de A para B. Digamos que f é ***um-para-um*** se ela nunca mapeia dois elementos diferentes para um mesmo lugar — ou seja, se $f(a) \neq f(b)$ sempre que $a \neq b$. Digamos que f é ***sobrejetora*** se ela atinge todo elemento de B — ou seja, se para todo $b \in B$ existe um $a \in A$ tal que $f(a) = b$. Digamos que A e B são de ***mesmo tamanho*** se existe uma função um-para-um e sobrejetora $f \colon A \longrightarrow B$. Uma função que é tanto um-para-um quanto sobrejetora é denominada uma ***correspondência***. Em uma correspondência, todo elemento de A mapeia para um único elemento de B e cada elemento de B tem um único elemento de A mapeando para ele. Uma correspondência é simplesmente uma maneira de emparelhar os elementos de A com os elementos de B.

Exemplo – N (naturais) e os naturais pares

Exemplo – N (naturais) e os naturais pares

- f(n) = 2n

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

- Têm o mesmo tamanho!

DEFINIÇÃO 4.14

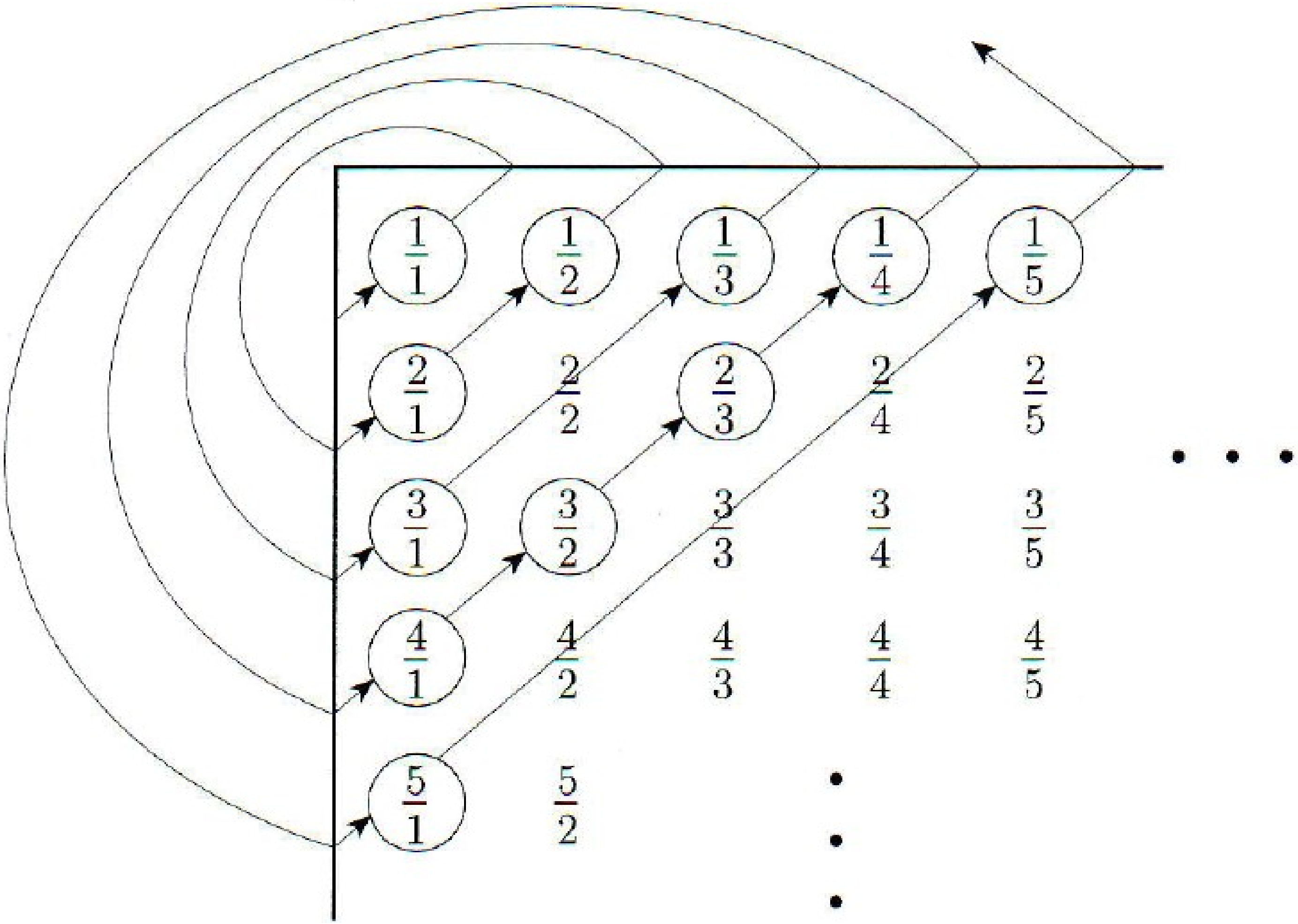
Um conjunto A é ***contável*** se é finito ou tem o mesmo tamanho que $\mathcal{N}$.

Exemplo - Q (racionais)

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

Exemplo - Q (racionais)

$Q = \{\frac{m}{n} | m, n \in N\}$



- Têm o mesmo tamanho!
- Q é contável!

Exemplo – R (reais)

- Qualquer número com representação decimal
- Inclui números como $\pi = 3{,}1415926...$, $\sqrt{2} = 1{,}4142135...$
- Como seria f?

Exemplo – R (reais)

- Qualquer número com representação decimal
- Inclui números como $\pi = 3,1415926...$, $\sqrt{2} = 1,4142135...$
- Como seria f?
- Não há!

R é incontável

TEOREMA 4.17 ....

$\mathcal{R}$ é incontável.

R é incontável

TEOREMA 4.17 ....

$\mathcal{R}$ é incontável.

- Prova por contradição:

R é incontável

TEOREMA 4.17

$\mathcal{R}$ é incontável.

- Prova por contradição: vamos assumir que f existe e contruir um x que esteja fora da correspondência

R é incontável - Prova

- Vamos assumir que f existe
- Por exemplo:

n	$f(n)$
1	3,14159...
2	55,55555...
3	0,12345...
4	0,50000...
$\vdots$	$\vdots$

- Construímos um x entre 0 e 1 cujo i-ésimo dígito após a vírgula seja diferente do i-ésimo dígito de f(i)
- Logo, x não é igual a nenhum f(i)
- Obs.: escolhemos dígitos diferentes de 0 e 9

R é incontável – Prova Diagonalização

- Vamos assumir que f existe
- Por exemplo:

n	$f(n)$
1	3,14159...
2	55,55555...
3	0,12345...
4	0,50000...
$\vdots$	$\vdots$

- Construímos um x entre 0 e 1 cujo i-ésimo dígito após a vírgula seja diferente do i-ésimo dígito de f(i)
- Logo, x não é igual a nenhum f(i)
- Obs.: escolhemos dígitos diferentes de 0 e 9

O que isso tem a ver com Teoria da Computação ?

Linguagens não Turing-reconhecíveis

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova:
  - Provar que o conjunto de todas as MTs é contável, mas o conjunto de todas as linguagens possíveis é incontável
  - Para isso, mostrar que o conjunto de todas as linguagens têm o mesmo tamanho que o conjunto de todas as sequências binárias infinitas, que é incontável
  - Para isso, usar diagonalização

Conjunto de MTs é contável

- Cada Máquina de Turing M tem uma codificação em uma cadeia <M>
  - Descartando aquelas cadeias que não são MT legítimas, podemos listar cadeias que representem MTs
- Σ^* é contável
  - Basta listar suas cadeias por ordem crescente de tamanho e ordem lexicográfica (e associar um natural a cada uma)
  - Ex: $\Sigma = \{0,1\}$, lista = 0, 1, 00, 01, 10, 11, 000, ...

Linguagens não Turing-reconhecíveis

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova:
  - Provar que o conjunto de todas as MTs é contável, mas o conjunto de todas as linguagens possíveis é incontável
  - Para isso, mostrar que o conjunto de todas as linguagens têm o mesmo tamanho que o conjunto de todas as sequências binárias infinitas, que é incontável
  - Para isso, usar diagonalização

Linguagens não Turing-reconhecíveis

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova:
  - Provar que o conjunto de todas as MTs é contável, mas o conjunto de todas as linguagens possíveis é incontável
  - Para isso, mostrar que o conjunto de todas as linguagens têm o mesmo tamanho que o conjunto de todas as sequências binárias infinitas, que é incontável
  - Para isso, usar diagonalização

O conjunto de todas as linguagens e o conjunto de todas as strings binárias infinitas possuem o mesmo tamanho

- Cada linguagem L_k pode ser representada por uma string binária infinita b_k
  - Ordene as cadeias de Σ^* (s_1, s_2, ...)
  - A posição i da string binária b_k possui valor 1 se a cadeia s_i pertencer à linguagem L_k, e valor 0 caso contrário
  - Ex: A = {cadeias binárias começando com 0}

$$
\begin{array}{lcccccccccccc}
\Sigma^* & = & \{ & \varepsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & \cdots \}; \\
A & = & \{ & & 0, & & 00, & 01, & & & 000, & 001, & \cdots \}; \\
\chi_A & = & & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \cdots .
\end{array}
$$

O conjunto de todas as strings binárias infinitas é incontável

- Usar técnica de diagonalização (usada para prova que R é incontável)

Linguagens não Turing-reconhecíveis

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova:
  - Provar que o conjunto de todas as MTs é contável, mas o conjunto de todas as linguagens possíveis é incontável
  - Para isso, mostrar que o conjunto de todas as linguagens têm o mesmo tamanho que o conjunto de todas as sequências binárias infinitas, que é incontável
  - Para isso, usar diagonalização

Linguagens não Turing-reconhecíveis

COROLÁRIO 4.18 ....

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova:
  - Provar que o conjunto de todas as MTs é contável, mas o conjunto de todas as linguagens possíveis é incontável
  - Logo, existem linguagens que não podem ser reconhecidas por nenhuma máquina de Turing

Voltando às linguagens Turing-reconhecíveis...

- Vimos que linguagens regulares e livres de contexto são Turing-decidíveis
- O problema de determinar se uma MT aceita uma cadeia w é decidível?
  - Primeiro: como escrevemos esse problema em termos de linguagem?

Voltando às linguagens Turing-reconhecíveis...

- Vimos que linguagens regulares e livres de contexto são Turing-decidíveis
- O problema de determinar se uma MT aceita uma cadeia w é decidível?
  - Primeiro: como escrevemos esse problema em termos de linguagem?

$$A_{\mathsf{MT}} = \{\langle M, w\rangle \mid M \text{ é uma MT e } M \text{ aceita } w\}$$

Voltando às linguagens Turing-reconhecíveis...

- Vimos que linguagens regulares e livres de contexto são Turing-decidíveis
- O problema de determinar se uma MT aceita uma cadeia w é decidível?
  - Segundo: como poderia ser uma MT para esse problema?

Voltando às linguagens Turing-reconhecíveis...

- Vimos que linguagens regulares e livres de contexto são Turing-decidíveis
- O problema de determinar se uma MT aceita uma cadeia w é decidível?
  - Segundo: como poderia ser uma MT para esse problema?

U = "Sobre a entrada $\langle M, w \rangle$, onde M é uma MT e w é uma cadeia:

1. Simule M sobre a entrada w.
2. Se M em algum momento entra no seu estado de aceitação, *aceite*; se M em algum momento entra em seu estado de rejeição, *rejeite*."

Voltando às linguagens Turing-reconhecíveis...

- Vimos que linguagens regulares e livres de contexto são Turing-decidíveis
- O problema de determinar se uma MT aceita uma cadeia w é decidível?
  - Segundo: como poderia ser uma MT para esse problema?

U = "Sobre a entrada $\langle M, w \rangle$, onde M é uma MT e w é uma cadeia:

1. Simule M sobre a entrada w.
2. Se M em algum momento entra no seu estado de aceitação, *aceite*; se M em algum momento entra em seu estado de rejeição, *rejeite*."

Máquina de Turing universal – estímulo ao programa armazenado

Determinar se uma MT aceita uma cadeia w é decidível?

- M só entra em loop se w não pertencer à linguagem
- Como U poderia usar isso para decidir A_{MT}?

Determinar se uma MT aceita uma cadeia w é decidível?

- M só entra em loop se w não pertencer à linguagem
- Como U poderia usar isso para decidir A_{MT}?
  - Se puder prever que M entrará em loop, rejeita

Determinar se uma MT aceita uma cadeia w é decidível?

- M só entra em loop se w não pertencer à linguagem
- Como U poderia usar isso para decidir A_{MT}?
  - Se puder prever que M entrará em loop, rejeita
- Problema: dá para prever? (Problema da parada)

Determinar se uma MT aceita uma cadeia w é decidível?

- M só entra em loop se w não pertencer à linguagem
- Como U poderia usar isso para decidir A_{MT}?
  - Se puder prever que M entrará em loop, rejeita
- Problema: dá para prever? (Problema da parada) NÃO

Determinar se uma MT aceita uma cadeia w é INdecidível

$$A_{\mathsf{MT}} = \{\langle M, w\rangle \mid M \text{ é uma MT e } M \text{ aceita } w\}.$$

TEOREMA 4.11

A_{MT} é indecidível.

O Problema da Parada é INdecidível

$$A_{\mathsf{MT}} = \{\langle M, w\rangle \mid M \text{ é uma MT e } M \text{ aceita } w\}.$$

TEOREMA 4.11

A_{MT} é indecidível.

O Problema da Parada é indecidível – Prova por contradição

- Supomos A_{MT} decidível e H um decisor:

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

- D outra MT, que usa H para determinar o que M faz com <M>, e faz o oposto:

$D =$ "Sobre a entrada $\langle M\rangle$, onde M é uma MT:

1. Rode H sobre a entrada $\langle M, \langle M\rangle\rangle$.
2. Dê como saída o oposto do que H dá como saída; ou seja, se H aceita, *rejeite* e se H rejeita, *aceite*."

O Problema da Parada é indecidível – Prova por contradição

$$D(\langle M \rangle) = \begin{cases} aceite & \text{se } M \text{ não aceita } \langle M \rangle \\ rejeite & \text{se } M \text{ aceita } \langle M \rangle. \end{cases}$$

O Problema da Parada é indecidível – Prova por contradição

$$D(\langle M \rangle) = \begin{cases} aceite & \text{se } M \text{ não aceita } \langle M \rangle \\ rejeite & \text{se } M \text{ aceita } \langle M \rangle. \end{cases}$$

- E se D tiver <D> como entrada?

O Problema da Parada é indecidível – Prova por contradição

$$D(\langle M \rangle) = \begin{cases} \textit{aceite} & \text{se } M \text{ não aceita } \langle M \rangle \\ \textit{rejeite} & \text{se } M \text{ aceita } \langle M \rangle. \end{cases}$$

- E se D tiver <D> como entrada?

$$D(\langle D \rangle) = \begin{cases} \textit{aceite} & \text{se } D \text{ não aceita } \langle D \rangle \\ \textit{rejeite} & \text{se } D \text{ aceita } \langle D \rangle. \end{cases}$$

O Problema da Parada é indecidível – Prova por contradição

$$D(\langle M \rangle) = \begin{cases} \mathit{aceite} & \text{se } M \text{ não aceita } \langle M \rangle \\ \mathit{rejeite} & \text{se } M \text{ aceita } \langle M \rangle. \end{cases}$$

- E se D tiver <D> como entrada?

$$D(\langle D \rangle) = \begin{cases} \mathit{aceite} & \text{se } D \text{ não aceita } \langle D \rangle \\ \mathit{rejeite} & \text{se } D \text{ aceita } \langle D \rangle. \end{cases}$$

- Contradição! H e D não podem existir!

Descrevendo a contradição por diagonalização

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*aceite*		*aceite*		
M_2	*aceite*	*aceite*	*aceite*	*aceite*	
M_3					$\cdots$
M_4	*aceite*	*aceite*			
$\vdots$		$\vdots$			

FIGURA 4.19

A entrada i, j é *aceite* se M_i aceita $\langle M_j \rangle$.

Descrevendo a contradição por diagonalização

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*aceite*	*rejeite*	*aceite*	*rejeite*	
M_2	*aceite*	*aceite*	*aceite*	*aceite*	$\cdots$
M_3	*rejeite*	*rejeite*	*rejeite*	*rejeite*	
M_4	*aceite*	*aceite*	*rejeite*	*rejeite*	
$\vdots$			$\vdots$		

FIGURA 4.20
A entrada i, j é o valor de H sobre a entrada $\langle M_i, \langle M_j \rangle \rangle$.

Descrevendo a contradição por diagonalização

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	*aceite*	*rejeite*	*aceite*	*rejeite*		*aceite*	
M_2	*aceite*	*aceite*	*aceite*	*aceite*	$\cdots$	*aceite*	$\cdots$
M_3	*rejeite*	*rejeite*	*rejeite*	*rejeite*		*rejeite*	
M_4	*aceite*	*aceite*	*rejeite*	*rejeite*		*aceite*	
$\vdots$			$\vdots$		$\ddots$		
D	*rejeite*	*rejeite*	*aceite*	*aceite*		?	
$\vdots$			$\vdots$				$\ddots$

FIGURA 4.21
Se D estiver na figura, uma contradição ocorre em "?".

Voltando às linguagens Turing-NÃO-Reconhecíveis

- Vimos que existem linguagens que NÃO são Turing-reconhecíveis
- Perguntas:
  - Quais são alguns exemplos delas?
  - O que isso tem a ver com linguagens Turing-decidíveis?

Decidibilidade e reconhecibilidade

- Uma linguagem é Turing-decidível se ela E seu complemento forem ambas Turing-reconhecíveis
- Uma linguagem é **co-Turing-reconhecível** se ela for o complemento de uma linguagem Turing-reconhecível (def. do livro)
- Uma linguagem é **co-Turing-reconhecível** se seu complemento for Turing-reconhecível (acho mais fácil de entender)

Decidibilidade e reconhecibilidade

TEOREMA 4.22

Uma linguagem é decidível sse ela é Turing-reconhecível e co-Turing-reconhecível.

Em outras palavras, uma linguagem é decidível exatamente quando ela e seu complemento são ambas Turing-reconhecíveis.

Prova

TEOREMA 4.22

Uma linguagem é decidível sse ela é Turing-reconhecível e co-Turing-reconhecível.

- (=>) fácil (direto das definições)
- (<=) Se uma linguagem A é Turing-reconhecível, consigo aceitar todas as cadeias de A. O problema são as cadeias do complemento de A... Mas se o complemento também é reconhecível, posso rejeitar tudo o que a MT do complemento aceita...

Prova

M = "Sobre a entrada w:

1. Rode ambas, M_1 e M_2, sobre a entrada w em paralelo.
2. Se M_1 aceita, *aceite*; se M_2 aceita, *rejeite*."

- Onde rodar em paralelo significa rodar cada MT em uma fita diferente, rodando um passo de cada uma de cada vez e alternadamente, até que uma delas aceita

COROLÁRIO 4.23

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

COROLÁRIO 4.23

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

PROVA Sabemos que A_{MT} é Turing-reconheível. Se $\overline{A_{\mathsf{MT}}}$ também fosse Turing-reconhecível, A_{MT} seria decidível. O Teorema 4.11 nos diz que A_{MT} não é decidível, portanto $\overline{A_{\mathsf{MT}}}$ não pode ser Turing-reconhecível.

Lista 4

- 4.12, 4.22 e 4.28
- Entrega: dia 22/10