ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 21
Cap 4.2
Linguagens que NÃO são Turing-reconhecíveis
O Problema da Parada

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

Aula anterior...

Cap 4 – Decidibilidade

Cap. 4.1 – Linguagens Decidíveis

Motivação

Saber se um problema é indecidível é importante para podermos fazer uma simplificação do mesmo a fim de propormos uma solução algorítmica.

Como provar que um problema é decidível?

1) Escreva-o na forma de uma linguagem

2) Mostre uma MT que a DECIDE:

a) Descreva a máquina
b) Mostre que

- para toda sequência da linguagem a MT PÁRA e *aceita*
- para toda sequência que NÃO É da linguagem a MT PÁRA e *rejeita*

Problemas decidíveis concernentes a linguagens regulares

A_{AFD} = {<B,w> | B é um AFD que aceita a cadeia de entrada w}

$$A_{\text{AFN}} = \{\langle B, w\rangle |\ B \text{ é um AFN que aceita a cadeia de entrada } w\}.$$

$$A_{\text{EXR}} = \{\langle R, w\rangle |\ R \text{ é uma expressão regular que gera a cadeia } w\}.$$

A_{GR} = {<G,w> | G é uma gramática regular que gera a cadeia w}

$$V_{\text{AFD}} = \{\langle A\rangle |\ A \text{ é um AFD e } L(A) = \emptyset\}.$$

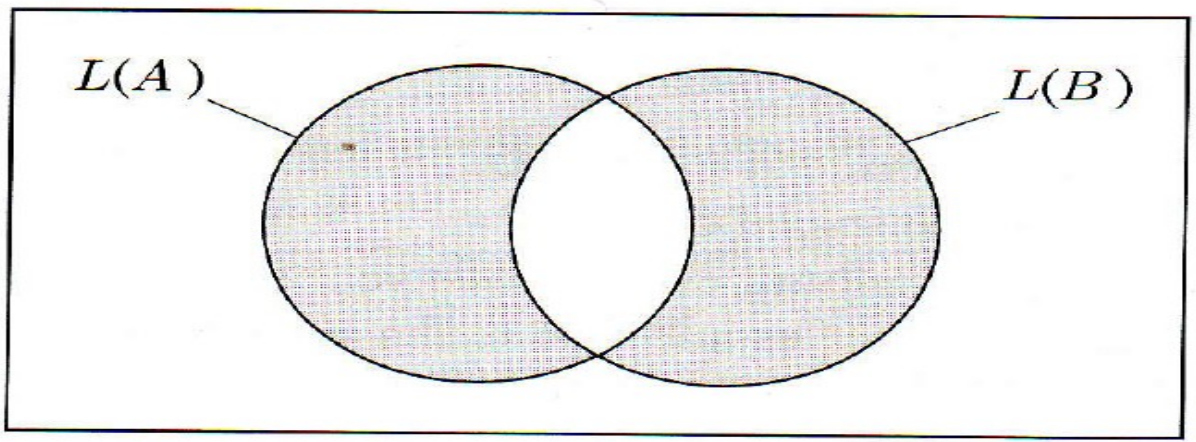
$$EQ_{\text{AFD}} = \{\langle A, B\rangle |\ A \text{ e } B \text{ são AFDs e } L(A) = L(B)\}.$$

Equivalência de dois AFDs

da vacuidade

PROVA Para provar esse teorema, usamos o Teorema 4.4. Construímos um novo AFD C a partir de A e B, tal que C aceita somente aquelas cadeias que são aceitas ou por A ou por B, mas não por ambos. Conseqüentemente, se A e B reconhecem a mesma linguagem, C não aceitará nada. A linguagem de C é

$$L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right).$$



Problemas decidíveis concernentes a linguagens livres de contexto

$$A_{\mathrm{GLC}} = \{\langle G, w\rangle \mid G \text{ é uma GLC que gera a cadeia } w\}$$

Linguagens livres de contexto (LLCs) são decidíveis

$$V_{\mathrm{GLC}} = \{\langle G\rangle \mid G \text{ é uma GLC e } L(G) = \emptyset\}.$$

Equivalência de GLCs

$$EQ_{\text{GLC}} = \{\langle G, H\rangle \mid G \text{ e } H \text{ são GLCs e } L(G) = L(H)\}.$$

- Técnica semelhante a mostrar se dois AFDs são equivalentes?
- Problema: LLCs não são fechadas com relação às operações de complementação e intersecção!
- Na verdade, EQ_{GLC} é indecidível!

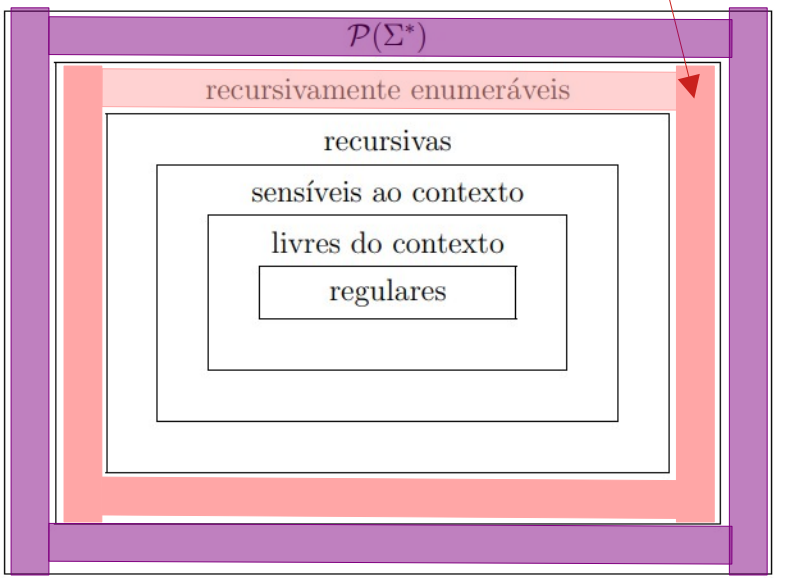
Aula de hoje

Cap 4.2

Linguagens que NÃO são Turing-reconhecíveis

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!

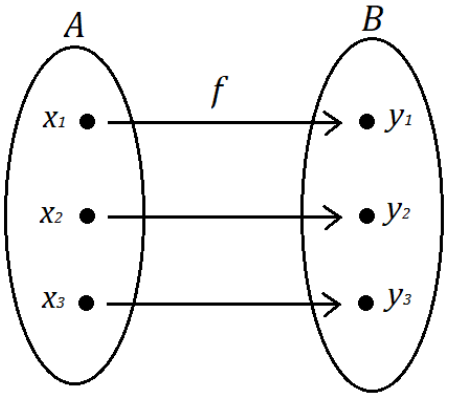
Ex: verificação de software é insolúvel (dado um programa e sua especificação, verificar se o programa está correto)

Limites da computação

- 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 se seus elementos puderem ser **emparelhados**
  - f : A → B, onde f é uma função bijetora (uma correspondência)



https://www.infoescola.com/matematica/funcao-bijetora/

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 $f : A \longrightarrow B$. 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
⋮	⋮

- Têm o mesmo tamanho!

DEFINIÇÃO 4.14

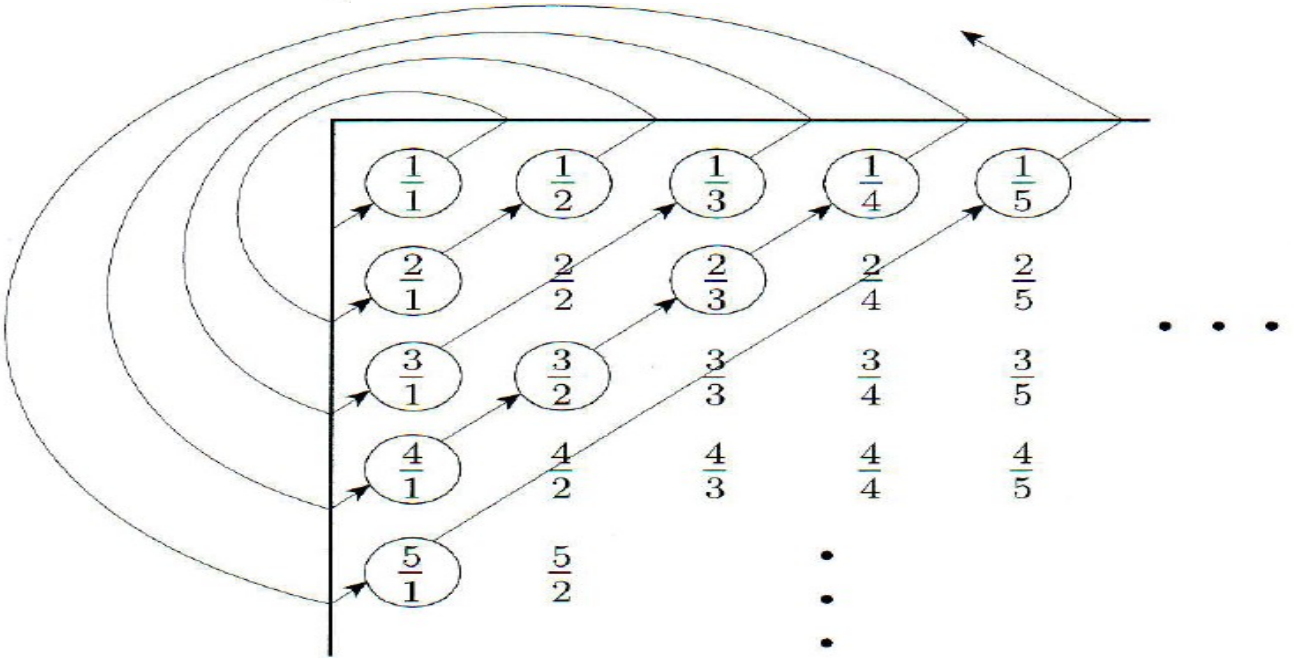
Um conjunto A é ***contável*** se é finito ou tem o mesmo tamanho que $\mathcal{N}$.

Exemplo - Q (racionais) é contável?

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

Exemplo - Q (racionais) é contável?

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



- Existe uma função bijetora de N para Q! Logo, eles têm o mesmo tamanho!
- Logo Q é contável!

Exemplo – R (reais) é contável?

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

Exemplo – R (reais) é contável?

- 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.

- Vamos mostrar que não existe uma correpondência f entre N e R
- Prova por contradição: vamos assumir que f existe e contruir um número real 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...
⋮	⋮

- Vamos contruir um número real x que seja diferente de cada real f(i) emparelhado com o natural i

R é incontável – Prova

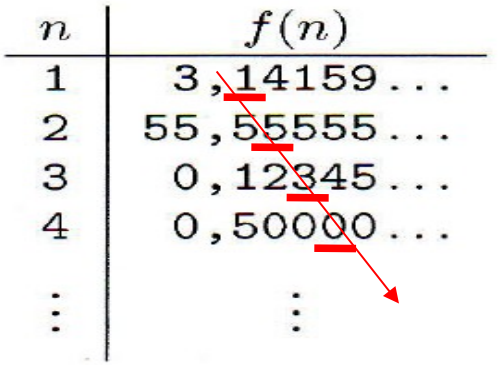
- Vamos assumir que f existe
- Por exemplo:

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

- Vamos contruir um número real x que seja diferente de cada real f(i) emparelhado com o natural i
- Construimos 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 (para evitar problemas como 0,1999.... ser igual a 0,200....)

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...
⋮	⋮

- Vamos contruir um número real x que seja diferente de cada real f(i) emparelhado com o natural i
- Construimos 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 (para evitar problemas como 0,1999.... ser igual a 0,200....)

O que isso tem a ver com Teoria da Computação ?

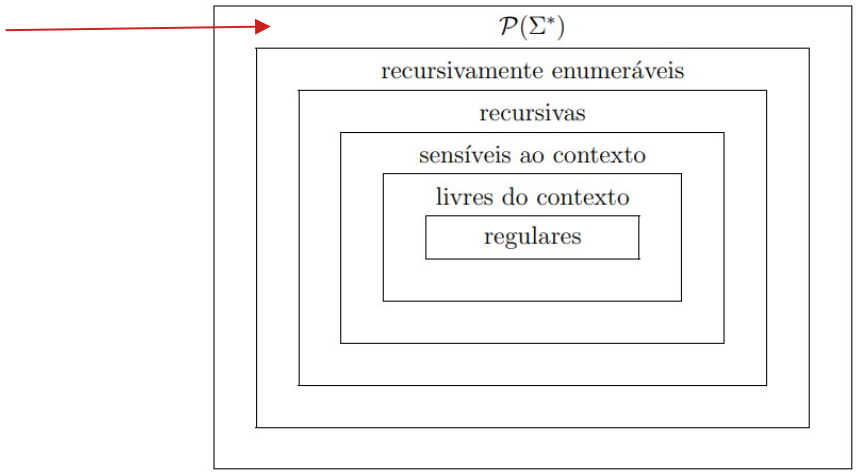
O que isso tem a ver com Teoria da Computação ?

TEOREMA 4.17

$\mathcal{R}$ é incontável.

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.



O que isso tem a ver com Teoria da Computação ?

TEOREMA 4.17

$\mathcal{R}$ é incontável.

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova: vamos mostrar que:
  - Há um conjunto contável de Máquinas de Turing
  - Há um conjunto incontável de linguagens
  - Cada MT reconhece apenas uma linguagem
  - Logo, há linguagens que não são reconhecidas por nenhuma MT

O que isso tem a ver com Teoria da Computação ?

TEOREMA 4.17

$\mathcal{R}$ é incontável.

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova: vamos mostrar que:
  - Há um conjunto contável de Máquinas de Turing
  - Há um conjunto incontável de linguagens
  - Cada MT reconhece apenas uma linguagem
  - Logo, há linguagens que não são reconhecidas por nenhuma MT

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: Σ = {0,1}, lista = 0, 1, 00, 01, 10, 11, 000, …
  - MT escritas em português: Σ = {a .. z, A .. Z, 0 .. 9, #, %, …}
  - Logo, o conjunto de todas as MT (que é um subconjunto de Σ*) é contável

O que isso tem a ver com Teoria da Computação ?

TEOREMA 4.17

$\mathcal{R}$ é incontável.

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova: vamos mostrar que:
  - Há um conjunto contável de Máquinas de Turing
  - Há um conjunto incontável de linguagens
  - Cada MT reconhece apenas uma linguagem
  - Logo, há linguagens que não são reconhecidas por nenhuma MT

O que isso tem a ver com Teoria da Computação ?

TEOREMA 4.17

$\mathcal{R}$ é incontável.

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova: vamos mostrar que:
  - Há um conjunto contável de Máquinas de Turing
  - Há um conjunto incontável de linguagens
  - Cada MT reconhece apenas uma linguagem
  - Logo, há linguagens que não são reconhecidas por nenhuma MT

- Para provar que o conjunto de todas as linguagens possíveis é incontável:
  - mostrar que o conjunto de todas as linguagens têm o mesmo tamanho que o conjunto de todas as sequências binárias infinitas
  - Provar, por diagonalização, que o conjunto de todas as sequências binárias infinitas é incontável
  - Logo, o conjunto de todas as linguagens possíveis é incontável

O conjunto de todas as strings binárias infinitas é incontável

O conjunto de todas as strings binárias infinitas é incontável

Seja f(i) uma correspondência entre N e B

N = conjunto dos números naturais

B = conjunto de todas as strings binárias infinitas

- Vamos construir uma string binária infinita fora dessa correspondência (usando diagonalização)
  - O i-ésimo dígito é diferente de f(i)

CONTRADIÇÃO!

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}

poderia ser a,b por ex

$$\Sigma^* = \{ \ \epsilon \ , \ 0 \ , \ 1 \ , \ 00 \ , \ 01 \ , \ 10 \ , \ 11 \ , 000 \ , 001 \ , \ \cdots \ \} \ ;$$
$$A = \{ \quad 0 \ , \quad 00 \ , \ 01 \ , \quad 000 \ , 001 \ , \ \cdots \ \} \ ;$$
$$\chi_A = \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad \cdots \quad .$$

String binária

O que isso tem a ver com Teoria da Computação ?

TEOREMA 4.17

$\mathcal{R}$ é incontável.

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova: vamos mostrar que:
  - Há um conjunto contável de Máquinas de Turing
  - Há um conjunto incontável de linguagens
  - Cada MT reconhece apenas uma linguagem
  - Logo, há linguagens que não são reconhecidas por nenhuma MT

O que isso tem a ver com Teoria da Computação ?

TEOREMA 4.17

$\mathcal{R}$ é incontável.

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova: vamos mostrar que:
  - Há um conjunto contável de Máquinas de Turing
  - Há um conjunto incontável de linguagens
  - Cada MT reconhece apenas uma linguagem
  - Logo, há linguagens que não são reconhecidas por nenhuma MT

O que isso tem a ver com Teoria da Computação ?

TEOREMA 4.17

$\mathcal{R}$ é incontável.

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.

- Ideia da Prova: vamos mostrar que:
  - Há um conjunto contável de Máquinas de Turing
  - Há um conjunto incontável de linguagens
  - Cada MT reconhece apenas uma linguagem
  - Logo, há linguagens que não são reconhecidas por nenhuma MT

Problema da Parada

Voltando às linguagens Turing-reconhecíveis...

- Vimos que linguagens regulares e livres de contexto são Turing-decidíveis, ou seja:
  - o problema de determinar se um AFD aceita uma cadeia w é decidível
  - o problema de determinar se uma GLC gera uma cadeia w é decidível

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?

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

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...

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 inicialmente proposta por Turing – executa qualquer outra MT

→ Estímulo ao desenvolvimento dos computadores que executam programas armazenados

Voltando às linguagens Turing-reconhecíveis...

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*."

Há algum problema aqui?

Voltando às linguagens Turing-reconhecíveis...

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*."

Há algum problema aqui?

M PODE ENTRAR EM LOOP!!!!

Voltando às linguagens Turing-reconhecíveis...

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*."

- M só entra em loop se w não pertencer à linguagem

Voltando às linguagens Turing-reconhecíveis...

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*."

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

Voltando às linguagens Turing-reconhecíveis...

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*."

- 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 (**sem executá-la**), rejeita

Problema: dá para prever? (Problema da parada)

Voltando às linguagens Turing-reconhecíveis...

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*."

- 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 (**sem executá-la**), rejeita

Problema: dá para prever? (Problema da parada)

NÃO

O problema da parada (The Halting Problem)

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

Observações para quem for usar legendas em português:

- “stuck” é erroneamente traduzido para “pára”. O certo seria “trava” (ou “entra em loop”)

Logo, determinar se uma MT aceita uma cadeia w é INdecidível

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

TEOREMA 4.11

A_{MT} é indecidível.

Afinal, não há como saber se a MT vai travar ou não…
Mas a prova aqui é diretamente sobre A_{MT}, mas é semelhante à prova do vídeo (que prova que o problema da parada é indecidível)

O problema da aceitação de uma cadeia w por uma MT M é Indecidível (mas isso porque o 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.

A_{MT} é indecidível – Prova por contradição

- Supomos A_{MT} decidível e H uma MT decisora de A_{MT}:

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

A_{MT} é indecidível – Prova por contradição

- Supomos A_{MT} decidível e H uma MT decisora de A_{MT}:

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

Fará o papel da MT X do vídeo

- 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*."

Por exemplo, um compilador Java, escrito em Java, que é compilado por ele mesmo

O Problema da Parada é indecidível – Prova por contradição

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

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

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

- E se D tiver <D> como entrada?

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

O Problema da Parada é indecidível – Prova por contradição

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

- E se D tiver <D> como entrada?

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

- Contradição! H não pode 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 $\langle M_j \rangle$.

Entradas em branco: Mi rejeita <Mj> ou entra em loop.

Usando H para decidir:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
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 $\langle M_i, \langle M_j \rangle \rangle$.

D tem que responder o contrário do que ela responde...(!?!)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
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 "?".

Exercícios do cap 4

- Sobre linguagens decidíveis: 4.1 a 4.4, 4.9 a 4.16
- Sobre aula de hoje até aqui: 4.5 a 4.8

Linguagens co-Turing reconhecíveis

- Uma linguagem é **co-Turing-reconhecível** se seu complemento for Turing-reconhecível
- O que acontece se uma linguagem e seu complemento forem AMBAS reconhecíveis?

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.

- (=>)

Prova

TEOREMA 4.22

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

- (=>) fácil (direto das definições)

Se L é decidível sei se uma cadeia w pertence a L ou se não pertence a L (ou seja, se pertence ao seu complemento), logo L e o complemento de L são Turing-reconhecíveis

Prova

TEOREMA 4.22

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

- (=>) fácil (direto das definições)
- (<=)

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, há uma MT M1 reconhecedora de A, e posso usá-la para aceitar todas as cadeias (que pertencem a A) que são aceitas por M1. O problema são as cadeias do complemento de A... Mas se o complemento também é reconhecível há uma MT M2 do complemento, e posso usá-la (M2) para rejeitar tudo o que M2 aceita…

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, há uma MT M1 reconhecedora de A, e posso usá-la para aceitar todas as cadeias (que pertencem a A) que são aceitas por M1. O problema são as cadeias do complemento de A... Mas se o complemento também é reconhecível há uma MT M2 do complemento, e posso usá-la (M2) para rejeitar tudo o que M2 aceita…

Vamos escrevê-la?

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, há uma MT M1 reconhecedora de A, e posso usá-la para aceitar todas as cadeias (que pertencem a A) que são aceitas por M1. O problema são as cadeias do complemento de A... Mas se o complemento também é reconhecível há uma MT M2 do complemento, e posso usá-la (M2) para rejeitar tudo o que M2 aceita…

Vamos escrevê-la? **Mas se uma delas travar?**

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 usar uma MT multifita e simular cada MT em uma fita diferente, rodando um passo de cada uma de cada vez e alternadamente, até que uma delas aceite

Retomando...

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

TEOREMA 4.11

A_{MT} é indecidível.

- Uma linguagem é Turing-decidível se ela E seu complemento forem ambas Turing-reconhecíveis
- A_{MT} é indecidível porém Turing-reconhecível
- Então o complemento de A_{MT} é o quê?

COROLÁRIO 4.23

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

COROLÁRIO 4.23

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

Ou seja, dadas uma MT M e uma cadeia w, saber se M NÃO aceita w NÃO é Turing-reconhecível

COROLÁRIO 4.23

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

Ou seja, dadas uma MT M e uma cadeia w, saber se M NÃO aceita w NÃO é Turing-reconhecível

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

Decidibilidade e reconhecibilidade

TEOREMA 4.22

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

Isso nos ajuda a provar que uma linguagem NÃO é RECONHECÍVEL (se eu sei que seu complemento é reconhecível porém INDECIDÍVEL)

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

Decidibilidade e reconhecibilidade

TEOREMA 4.22

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

Isso nos ajuda a provar que uma linguagem é DECIDÍVEL

(alternativamente à mostrar uma máquina de Turing decisora)

Mas tenho que mostrar a máquina reconhecedora para ela e para seu complemento.

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

Decidibilidade e reconhecibilidade

TEOREMA 4.22

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

Isso nos ajuda a provar que uma linguagem é INDECIDÍVEL

Quando?

Decidibilidade e reconhecibilidade

TEOREMA 4.22

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

Isso nos ajuda a provar que uma linguagem é INDECIDÍVEL
Quando ela é reconhecível porém não co-reconhecível
ou vice-versa (é não-reconhecível mas co-reconhecível)

Exercícios do cap 4

- Já podem fazer todos