

O Problema da Parada

Teoria da Computação – Ciência da Computação



Prof. Daniel Saad Nogueira
Nunes

IFB – Instituto Federal de Brasília,
Campus Taguatinga



Sumário

- 1 Introdução
- 2 Diagonalização
- 3 O problema da parada



Sumário

1 Introdução



Decidibilidade

- Agora temos uma noção precisa de algoritmos e sabemos que uma série de modelos relevantes de computação são equivalentes.
- Podemos falar em termos de algoritmos.
- Várias linguagens (problemas) mostraram ser decidíveis \Leftrightarrow existência de um algoritmo.
- Investigaremos agora, linguagens **indecidíveis**.



Indecidibilidade

- Nós provaremos agora um dos teoremas mais importantes em Teoria da Computação.
- Existem problemas que são insolúveis do ponto de vista algorítmico.
- Computadores aparentam ser cada vez mais poderosos, o que dá a impressão de que podemos resolver qualquer coisa com eles.
- Este teorema irá apresentar que computadores estão limitados de uma certa forma.



Indecibilidade

- Que tipo de problemas não podem ser resolvidos?
- São problemas obscuros?
- Problemas que existem só na cabeça dos teóricos?



Indecibilidade

- Que tipo de problemas não podem ser resolvidos?
- São problemas obscuros?
- Problemas que existem só na cabeça dos teóricos?
- **Não!**



Indecidibilidade

- Existem muitos problemas que são interessantíssimos para a prática mas que não possuem uma solução algorítmica.



Indecibilidade

Exemplo

- Dado uma especificação formal do que o programa está suposto a fazer e um programa de computador, verificar se o programa cumpre o prometido.
- Como o programa e a especificação são objetos matemáticos precisos, é natural pensar que podemos automatizar o processo de verificar se um programa está correto, isto é, de acordo com a especificação.
- Mas **não existe** um algoritmo para este problema.



Indecibilidade

- Vamos mostrar um problema fundamental indecidível, que servirá para ganharmos a intuição de que tipo de problemas são indecidíveis.
- Mas para isso, precisamos examinar o método de diagonalização de Cantor.



Sumário

2 Diagonalização



O método da diagonalização

- Método de diagonalização: descoberto por Georg Cantor em 1873.
- Cantor estava preocupado com o problema de mensurar conjuntos de cardinalidade infinita.
- Sim, ele queria saber se um infinito era maior do que o outro.



O método da diagonalização

- Por exemplo, tome o conjunto \mathbb{Z} e o conjunto de todas as *strings* binárias.
- Os dois conjuntos são infinitos, mas qual é o maior?
- Como podemos comparar o tamanho de duas coisas infinitas?



O método da diagonalização

- Cantor propôs uma solução bem simples para este problema.
- Ele observou que dois conjuntos possuem o mesmo elemento, se existe um **pareamento** de elementos do primeiro no segundo.
- Podemos comparar os tamanhos sem precisar contar os conjuntos.
- Vamos definir esta noção mais precisamente?



O método da diagonalização

Definição (função bijetora)

Uma função $f : A \rightarrow B$ é bijetora, quando é injetora e sobrejetora. Ela nunca mapeia dois elementos distintos no mesmo elemento, isto é, f é injetora:

$$x \neq y \Rightarrow f(x) \neq f(y)$$

Além disso, todo elemento de B é mapeado por algum elemento de A através de f , ou seja, f é sobrejetora:

$$\forall y \in B, \exists x \in A \text{ tal que } f(x) = y$$



O método da diagonalização

- Se encontramos uma bijeção $f : A \rightarrow B$ é uma maneira de argumentar que A tem o mesmo tamanho de B .
- Temos um pareamento entre elementos de A e B .
- Noção informal de pareamento = bijeção.



Pareando elementos

Exemplo

- Tome \mathbb{N} e $P = \{x \in \mathbb{N} \mid x \text{ é par}\}$.
- Ambos são infinitos.
- Será que eles possuem o mesmo tamanho?
- Intuitivamente parece que P tem a metade de elementos de \mathbb{N} .
- Mas na verdade eles possuem a mesma quantidade.



Pareando elementos

Exemplo

- Só precisamos achar uma bijeção.
- Tome

$$f : \mathbb{N} \rightarrow P$$

Tal que

$$f(x) \mapsto 2x$$



Pareando elementos

Exemplo

x	$f(x)$
1	2
2	4
\vdots	\vdots
n	$2n$
\vdots	\vdots

- Todos os elementos de \mathbb{N} foram pareados com elementos de P .
- Eles possuem o mesmo tamanho!



Pareando elementos

- Vamos formalizar a noção de conjuntos contáveis agora.
- Estes conjuntos possuem uma relação próxima com os objetos de computação.



Conjuntos contáveis

Definição (Conjuntos contáveis)

Um conjunto A é dito contável se é finito ou possui a mesma cardinalidade de \mathbb{N} .



Conjuntos contáveis

- Vamos pegar um exemplo interessante.
- Será que \mathbb{Q}^+ é contável?

$$\mathbb{Q}^+ = \left\{ \frac{m}{n} \mid m, n \in \mathbb{N} \right\}$$

- Sabemos que \mathbb{Q} é infinito, pois $\mathbb{N} \subseteq \mathbb{Q}^+$.
- Se quisermos mostrar que ambos possuem o mesmo tamanho, temos que achar uma bijeção

$$f : \mathbb{N} \rightarrow \mathbb{Q}^+$$



Conjuntos contáveis

- Uma maneira de fazer isso é listar todos os elementos de \mathbb{Q}^+ e parear o primeiro elemento de \mathbb{N} com o primeiro da lista, o segundo elemento de \mathbb{N} com o segundo da lista e assim sucessivamente.
- Temos que nos certificar que todo elemento de \mathbb{Q}^+ aparece uma única vez nesta lista.

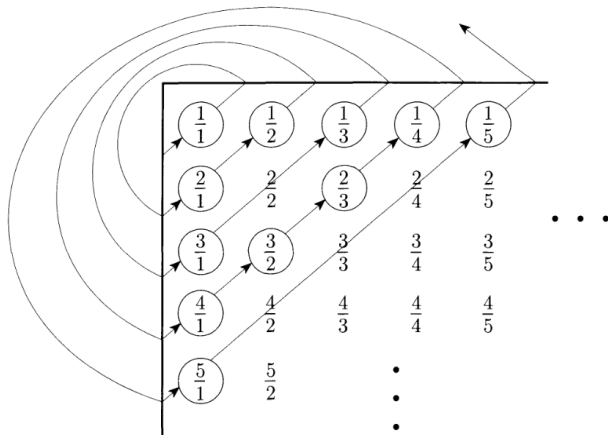


Conjuntos contáveis

- Construiremos esta lista através de uma matriz infinita.
- A i -ésima linha possui todos os números tendo i como numerador.
- A j -ésima linha possui todos os números contendo j como denominador.
- Assim, a célula $[i, j]$ contém exatamente o número $\frac{i}{j}$.



Conjuntos contáveis





Conjuntos contáveis

- Como podemos achar uma correspondência de \mathbb{N} com os elementos desta matriz?
- Primeira abordagem: construir a lista utilizando os elementos da primeira linha inicialmente.
- Como a primeira linha é infinita, nunca chegaremos na segunda linha.
- Nossa lista não terá todos os elementos de \mathbb{Q}^+ .

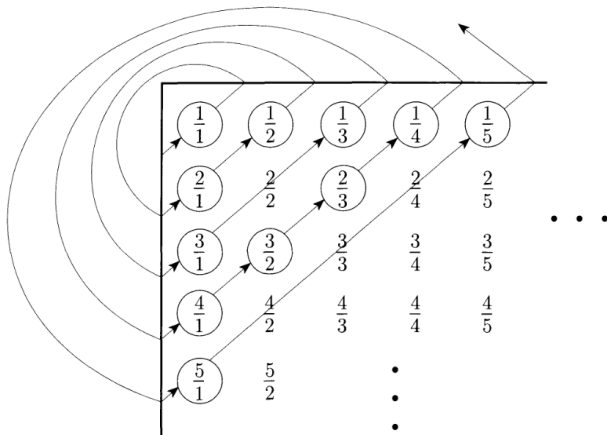


Método da diagonalização de Cantor

- Podemos gerar a lista percorrendo a matriz diagonalmente.
- **Método de diagonalização de Cantor.**



Conjuntos contáveis





Conjuntos contáveis

- Único cuidado: deixar elementos repetidos de fora.
- Percorrendo a matriz desta forma, todos os elementos da matriz estarão na nossa lista.
- O percurso nessa matriz nos dá a bijeção esperada!
- Cada elemento de \mathbb{N} está pareado com um elemento da lista.
- $|\mathbb{N}| = |\mathbb{Q}^+|$.



Conjuntos contáveis

- Depois de ver a demonstração que $\mathbb{N} = \mathbb{Q}^+$ podemos pensar que podemos seguir a mesma abordagem para demonstrar que quaisquer dois conjuntos infinitos possuem a mesma cardinalidade.
- No entanto, para alguns conjuntos, não existe esta bijeção dos \mathbb{N} .
- Um exemplo de conjunto infinito maior que \mathbb{N} é \mathbb{R} .
- Vamos mostrar que na verdade \mathbb{R} é incontável.



Conjuntos incontáveis

Teorema

\mathbb{R} é incontável.



Conjuntos incontáveis

Demonstração

- A Demonstração é por contradição.
- Suponha que \mathbb{R} é contável, isto é, existe uma bijeção f de \mathbb{N} em \mathbb{R}
- Vamos mostrar que f não é possível, chegando em um absurdo.
- A ideia é mostrar que existe um $x \in \mathbb{R}$ que não está mapeado.
- Vamos construir este x .



Conjuntos incontáveis

Demonstração

- Supondo que a bijeção f exista.
- Sem perda de generalidade, tome $f(1) = 3.14159\dots$, $f(2) = 5.555\dots$, $f(3) = \dots$ e assim em diante.
- Temos um pareamento hipotético entre \mathbb{N} e \mathbb{R} .



Conjuntos incontáveis

Demonstração

- A construção de um x não mapeado por f finalizaria a Demonstração, uma vez que teríamos um absurdo, e logo \mathbb{R} é incontável.
- Para construir este x devemos certificar que ele é diferente de $f(n)$ para qualquer $n \in \mathbb{N}$



Conjuntos incontáveis

Demonstração

- Pegamos um $0 < x < 1$.
- Colocamos no primeiro dígito depois da vírgula de x , um algarismo diferente do primeiro dígito depois da vírgula de $f(1)$.
- Colocamos no segundo dígito depois da vírgula de x , um algarismo diferente do segundo dígito depois da vírgula de $f(2)$.
- E assim em diante.
- Obs: só evitamos de atribuir para x os algarismos 0 ou 9, para evitar situações do tipo $0.999\dots = 1$.



Conjuntos incontáveis

Demonstração

n	$f(n)$
1	3. 1 4159...
2	5.5 5 5...
3	0.12 3 45...
4	0.500 0 0...
\vdots	\vdots

- $x = 0.3281 \dots$



Cojuntos incontáveis

Demonstração.

- Como construímos x de modo que ele difere de $f(1)$ considerando o primeiro algarismo depois da vírgula, difere de $f(2)$ considerando o segundo algarismo depois da vírgula, e assim em diante. . .
- Concluimos que x não está mapeado por nenhum elemento de $f(n)$.
- Assim f não pode ser bijetora e \mathbb{R} não pode ser contável.





Conjuntos incontáveis

- O teorema anterior tem uma aplicação profunda para nós.
- Ele mostra que algumas linguagens são incontáveis e sequer podem ser **reconhecidas** por MTs.



Linguagens não Turing-reconhecíveis

- Mostraremos que algumas linguagens não são Turing-reconhecíveis.
- Ponto chave: existem mais linguagens do que possíveis máquinas de Turing.



Linguagens não Turing-reconhecíveis

Teorema

Algumas linguagens não são Turing-reconhecíveis.



Linguagens não Turing-reconhecíveis

Demonstração

- A primeira observação a ser feita é que o conjunto Σ^* é contável para qualquer alfabeto finito Σ .
- Σ^* : conjunto de todas as strings finitas

$$\{\epsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$$

- Como temos finitas *strings* de tamanho 0,1,2, podemos construir uma lista que pode ser pareada com os elementos de \mathbb{N} .



Linguagens não Turing-reconhecíveis

Demonstração

1	\mapsto	ϵ
2	\mapsto	0
3	\mapsto	1
4	\mapsto	00
5	\mapsto	01
	\vdots	

- O i -ésimo natural com valor $\lfloor \log_2(k) \rfloor$ corresponde a i -ésima *string* com k bits.
- Para um alfabeto maior que 2, este raciocínio pode ser generalizado.



Linguagens não Turing-reconhecíveis

Demonstração

- O conjunto de todas as máquinas de Turing é contável.
- Toda máquina M tem uma codificação $\langle M \rangle$ em Σ^* .
- Se omitirmos as *strings* que não possuem uma codificação válida de MT, o que resta são descrições válidas de MT.
- Para mostrar que algumas linguagens não são Turing-decidíveis, vamos mostrar que o conjunto de todas as linguagens é incontável!
- Para isso, usaremos uma linguagem especial inicialmente.



Linguagens não Turing-reconhecíveis

Demonstração

- Tome a linguagem B como sendo a linguagem das strings binárias infinitas.
- Podemos mostrar que B é incontável utilizando um argumento por diagonalização parecido com o anterior.



Linguagens não Turing-reconhecíveis

s_1	=	0	0	0	0	0	0	0	0	0	0	0	0	...
s_2	=	1	1	1	1	1	1	1	1	1	1	1	1	...
s_3	=	0	1	0	1	0	1	0	1	0	1	0	1	...
s_4	=	1	0	1	0	1	0	1	0	1	0	1	...	
s_5	=	1	1	0	1	0	1	1	0	1	0	1	...	
s_6	=	0	0	1	1	0	1	1	0	1	1	0	...	
s_7	=	1	0	0	0	1	0	0	0	1	0	0	...	
s_8	=	0	0	1	1	0	0	1	1	0	0	1	...	
s_9	=	1	1	0	0	1	1	0	0	1	1	0	...	
s_{10}	=	1	1	0	1	1	1	0	0	1	0	1	...	
s_{11}	=	1	1	0	1	0	1	0	0	1	0	0	...	
\vdots		\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	

s	=	1	0	1	1	1	0	1	0	0	1	1	...
-----	---	---	---	---	---	---	---	---	---	---	---	---	-----



Linguagens não Turing-reconhecíveis

Demonstração

- Seja \mathcal{L} o conjunto de todas as possíveis linguagens.
- Mostrarmos que existe um pareamento entre B e \mathcal{L} , isto é, ambos são incontáveis e do mesmo tamanho.
- Queremos mostrar que $f : \mathcal{L} \rightarrow B$ é bijetora.
- Cada linguagem $A \in \mathcal{L}$ tem um par que corresponde a uma string binária infinita em B .
- Seja $\Sigma^* = \{s_1, s_2, \dots, s_i, \dots\}$.
- O i -ésimo de $f(A)$ é 1 se e somente se $s_i \in A$.
- Chamamos essa sequência de bits de sequência característica de A (χ_A).



Linguagens não Turing-reconhecíveis

Demonstração

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

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

$$\chi_A = 010110011\dots$$



Linguagens não Turing-reconhecíveis

Demonstração.

- É fácil ver que f é bijetora, e portanto, \mathcal{L} tem o mesmo tamanho de B .
- \mathcal{L} é incontável.
- Temos mais linguagens do que máquinas de Turing.
- Algumas linguagens não podem sequer ser reconhecidas, quanto menos decididas.





Sumário

3 O problema da parada



O problema da parada

- Tome a linguagem

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

- Ou seja, a entrada para este problema é uma descrição de uma MT e a entrada.
- A palavra é aceita pela linguagem se a MT aceita a palavra.



O problema da parada

- Mostraremos primeiramente que A_{MT} é Turing-reconhecível.
- Para mostrar que A_{MT} é Turing-reconhecível, só precisamos de uma MT que reconheça A_{MT} , isto é, que aceite as palavras que estão em A_{MT} .



O problema da parada

Algorithm 1: Construção da Máquina U , que reconhece A_{MT} .

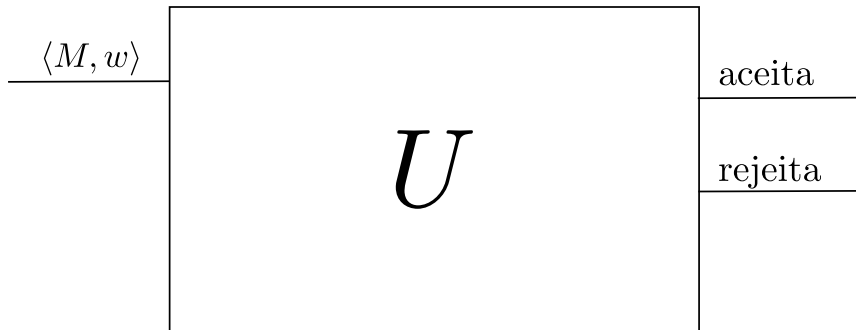
Input: $\langle M, w \rangle$

Output: Aceita, se M aceita w e rejeita se M entra no estado de rejeição sobre w .

- 1 Simule M na entrada w .
 - 2 **if**(M entra no estado de aceitação)
 - 3 | **return** Aceite
 - 4 **else if**(M entra no estado de rejeição)
 - 5 | **return** Rejeite
-



O problema da parada





O problema da parada

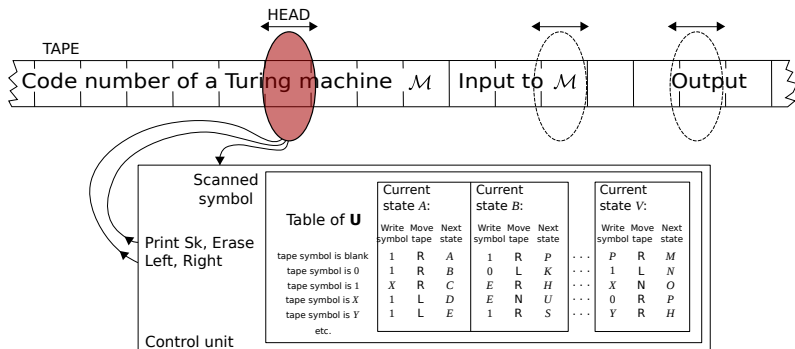


Figura: By Buckley - Own work, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=3097974>



O problema da parada

- Note que a máquina U entra em loop na entrada $\langle M, w \rangle$ se M entra em loop sobre w .
- Um algoritmo seria possível se existisse alguma forma de determinar que M não parava em w . Nesta condição poderíamos rejeitar.



O problema da parada

- A máquina U é interessante por si própria.
- É um exemplo de uma MT universal, primeiramente proposta por Turing.
- Ela é chamada universal, pois é capaz de simular qualquer outra máquina a partir de sua descrição.
- Teve um papel muito importante no estímulo de computadores de propósito geral, que armazenavam o programa a ser executado.



O problema da parada

- https://en.wikipedia.org/wiki/Electronic_delay_storage_automatic_calculator



O problema da parada

- Será que existe um algoritmo para A_{MT} ?



O problema da parada

- Vamos assumir que existe uma máquina H que decide A_{MT} .
- Chegaremos em um absurdo.
- Concluiremos que A_{MT} é indecidível.



O problema da parada

Teorema

O problema da parada é indecidível.



O problema da parada

Demonstração

- Suponha que A_{MT} seja decidível.
- Então existe uma MT H que decide a linguagem

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



O problema da parada

Demonstração

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



O problema da parada

Demonstração

- Construiremos uma máquina D que usa H como sub-rotina.
- Essa nova MT usa H para determinar o que uma máquina M faz quando recebe $\langle M \rangle$.
- Após obter a resposta de H , D faz o oposto do que H faz.



O problema da parada

Demonstração

Algorithm 2: Construção da máquina D .

Input: $\langle M \rangle$

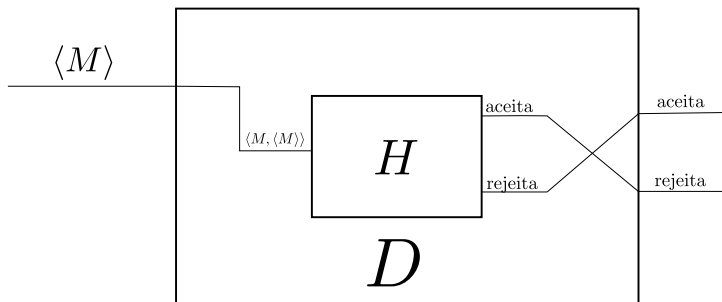
Output: Aceita se M não aceita a sua descrição, rejeita caso M aceita sua própria descrição

- 1 Rode H sobre a entrada $\langle M, \langle M \rangle \rangle$
 - 2 **if**(H aceita)
 - 3 | **return** *rejeita*
 - 4 **else**
 - 5 | **return** *aceita*
-



O problema da parada

Demonstração





O problema da parada

Demonstração

- Não se confuda com o fato da Máquina rodar sobre a própria descrição dela.
- Isso é como se um programa rodasse passando ele próprio como entrada.
- Um compilador de C pode ser escrito em C . Você pode compilar o próprio código do compilador, não pode?



O problema da parada

Demonstração

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



O problema da parada

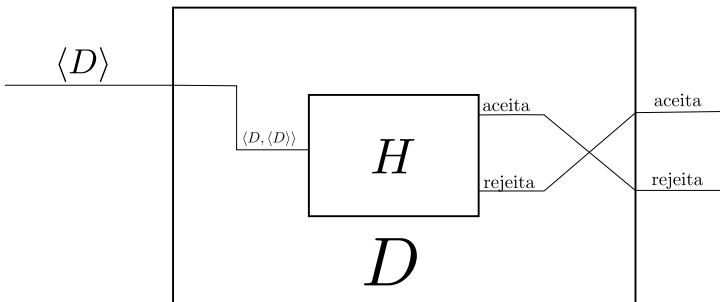
Demonstração

- O que acontece quando D tem como entrada a sua própria descrição?



O problema da parada

Demonstração





O problema da parada

Demonstração

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

- Não importa o que D faça, ele é forçado a fazer o oposto.
- Contradição.
- Nem D e nem H podem existir.
- Não temos uma máquina que decide A_{MT} .



O problema da parada

- Uma maneira via diagonalização pode ser utilizada para mostrar que o problema da parada é indecidível.
- Tome uma tabela em que as linhas são as máquinas, as colunas são descrições de máquinas e a célula i, j corresponde ao resultado da simulação de M_i sobre $\langle M_j \rangle$.
- A célula contém aceita, se M_i aceita $\langle M_j \rangle$, e branco caso não aceita (rejeita ou entra em loop).



O problema da parada

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



O problema da parada

- Assumindo que H decida A_{MT} , onde existia vazio, teremos rejeição.



O problema da parada

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



O problema da parada

- Note que D computa o oposto da diagonal da tabela.



O problema da parada

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	\dots	$\langle D \rangle$	\dots
M_1	<u>accept</u>	reject	accept	reject		accept	
M_2	accept	<u>accept</u>	accept	accept	\dots	accept	\dots
M_3	reject	reject	<u>reject</u>	reject		reject	
M_4	accept	accept	reject	<u>reject</u>		accept	
\vdots			\vdots		\ddots		
D	reject	reject	accept	accept		<u>?</u>	
\vdots			\vdots				\ddots



O problema da parada

- Na célula $[D, \langle D \rangle]$ temos uma contradição.



O problema da parada

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



O problema da parada

- Mostramos que A_{MT} é indecidível.
- Não temos um **algoritmo** que resolve o problema.
- No entanto A_{MT} é reconhecível.
- Mostraremos agora que uma linguagem é decidível se e somente se ela e seu complemento são reconhecíveis.



Linguagens decidíveis e reconhecíveis

Teorema

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



Linguagens decidíveis e reconhecíveis

Demonstração

- Temos duas direções de Demonstração.
- Provaremos a ida: Se uma linguagem é decidível implica que ela é Turing-reconhecível e co-Turing-reconhecível.
- Suponha que A seja uma linguagem decidível.
- Definitivamente \bar{A} é decidível.
- Qualquer linguagem decidível é reconhecível e o complemento de uma linguagem decidível também é decidível, e portanto, reconhecível.



Linguagens decidíveis e reconhecíveis

Demonstração

- Agora provaremos a volta.
- Se A e \bar{A} são Turing-reconhecíveis, então A é decidível.
- Seja M_1 a reconhecedora de A e M_2 a de \bar{A} .
- Podemos construir uma máquina M que decide A .



Linguagens decidíveis e reconhecíveis

Demonstração

Algorithm 3: Simulando M_1 e M_2 para decidir A .

Input: $w \in \Sigma^*$

Output: aceita, se $w \in A$ e rejeita caso contrário

- 1 Rode M_1 e M_2 sobre w “em paralelo”
 - 2 **if**(M_1 aceita)
 - 3 | **return** aceita
 - 4 **else if**(M_2 aceita)
 - 5 | **return** rejeite
-



Linguagens decidíveis e reconhecíveis

Demonstração.

- Por paralelo, queremos dizer que M tem duas fitas, uma para simular M_1 e uma para simular M_2 .
- M simula as máquinas um passo de cada vez de maneira alternada.
- Eventualmente, uma vai parar.
- M decide A .





Uma linguagem que não é Turing-reconhecível

Corolário

\bar{A}_{MT} não é Turing-reconhecível.



Uma linguagem que não é Turing-reconhecível

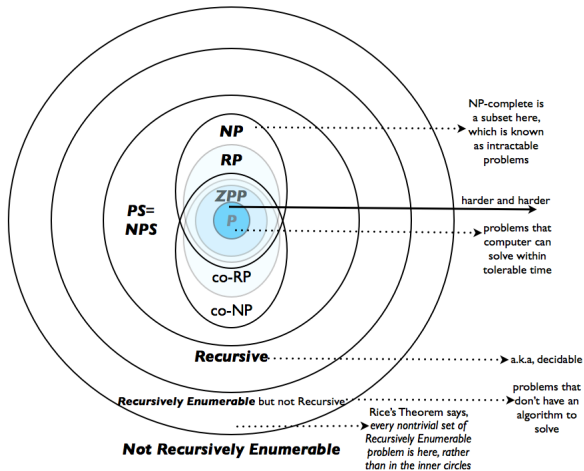
Demonstração.

- Sabemos que A_{MT} é reconhecível.
- Se \bar{A}_{MT} fosse reconhecível, A_{MT} seria decidível.
- O que é impossível.
- \bar{A}_{MT} não é sequer reconhecível.





Linguagens





Linguagens

Linguagens não recursivamente
enumeráveis

