Linguagens Formais e Autômatos

Aula 25 - A linguagem da diagonalização

Prof. Dr. Daniel Lucrédio
Departamento de Computação / UFSCar
Última revisão: ago/2015

Referências bibliográficas

- **Introdução à teoria dos autômatos, linguagens e computação / John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman** ; tradução da 2.ed. original de Vandenberg D. de Souza. - Rio de Janeiro : Elsevier, 2002 (Tradução de: Introduction to automata theory, languages, and computation - ISBN 85-352-1072-5)
  - Capítulo 9 - Seção 9.1
- **Introdução à teoria da computação / Michael Sipser** ; tradução técnica Ruy José Guerra Barretto de Queiroz ; revisão técnica Newton José Vieira. -- São Paulo : Thomson Learning, 2007 (Título original : Introduction to the theory of computation. "Tradução da segunda edição norte-americana" - ISBN 978-85-221-0499-4)
  - Capítulo 4 - Seção 4.2

Relembrando a hierarquia de Chomsky

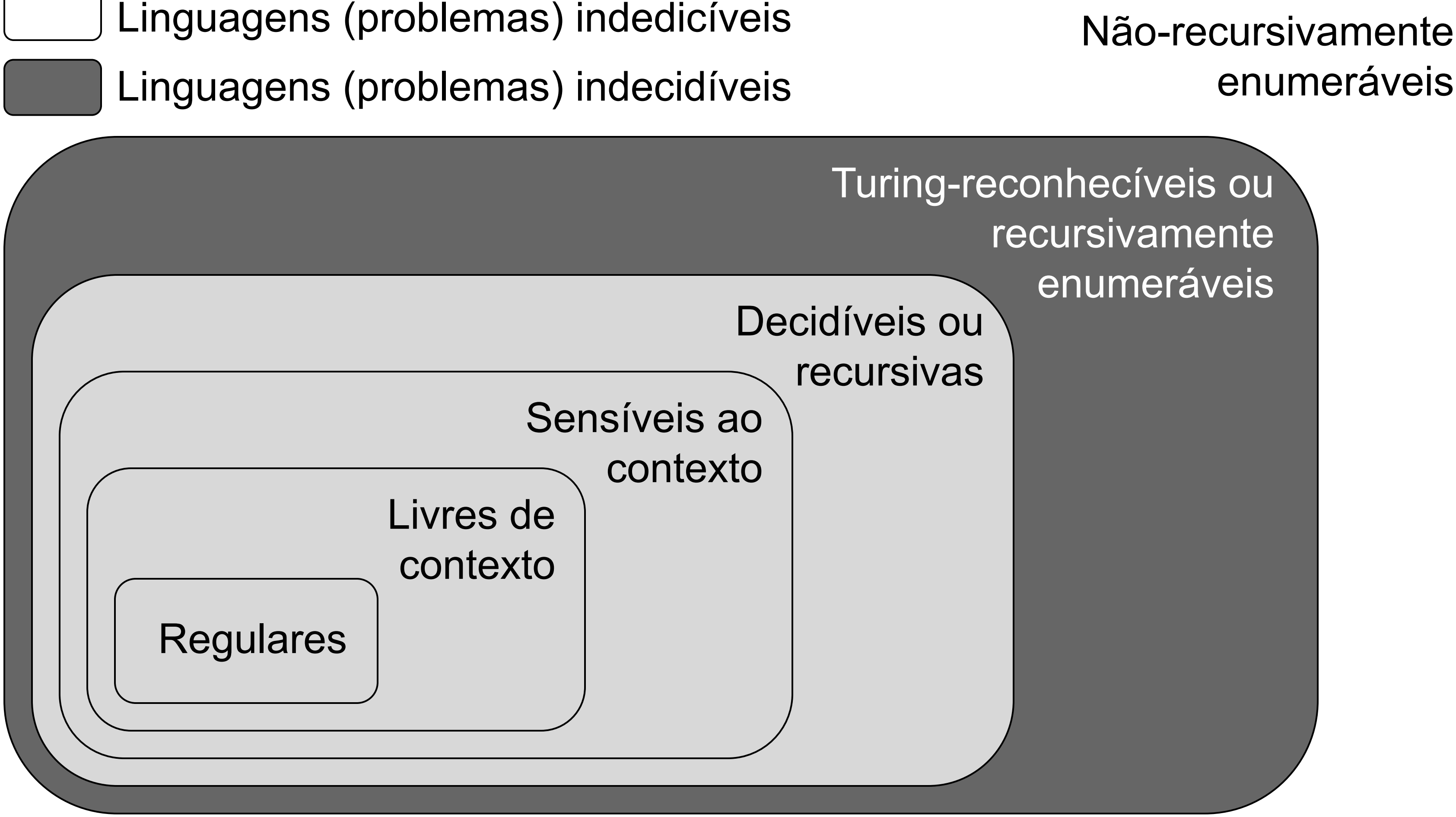
Hierarquia	Gramáticas	Linguagens	Autômato mínimo
Tipo-0	Recursivamente Enumeráveis ou irrestritas	Recursivamente Enumeráveis	Máquinas de Turing
Tipo-1	Sensíveis ao contexto	Sensíveis ao contexto	MT com fita limitada
Tipo-2	Livres de contexto	Livres de contexto	Autômatos de pilha
Tipo-3	Regulares (Expressões regulares)	Regulares	Autômatos finitos

Hierarquia de linguagens

Linguagens (problemas) dedicíveis

Linguagens (problemas) indedicíveis

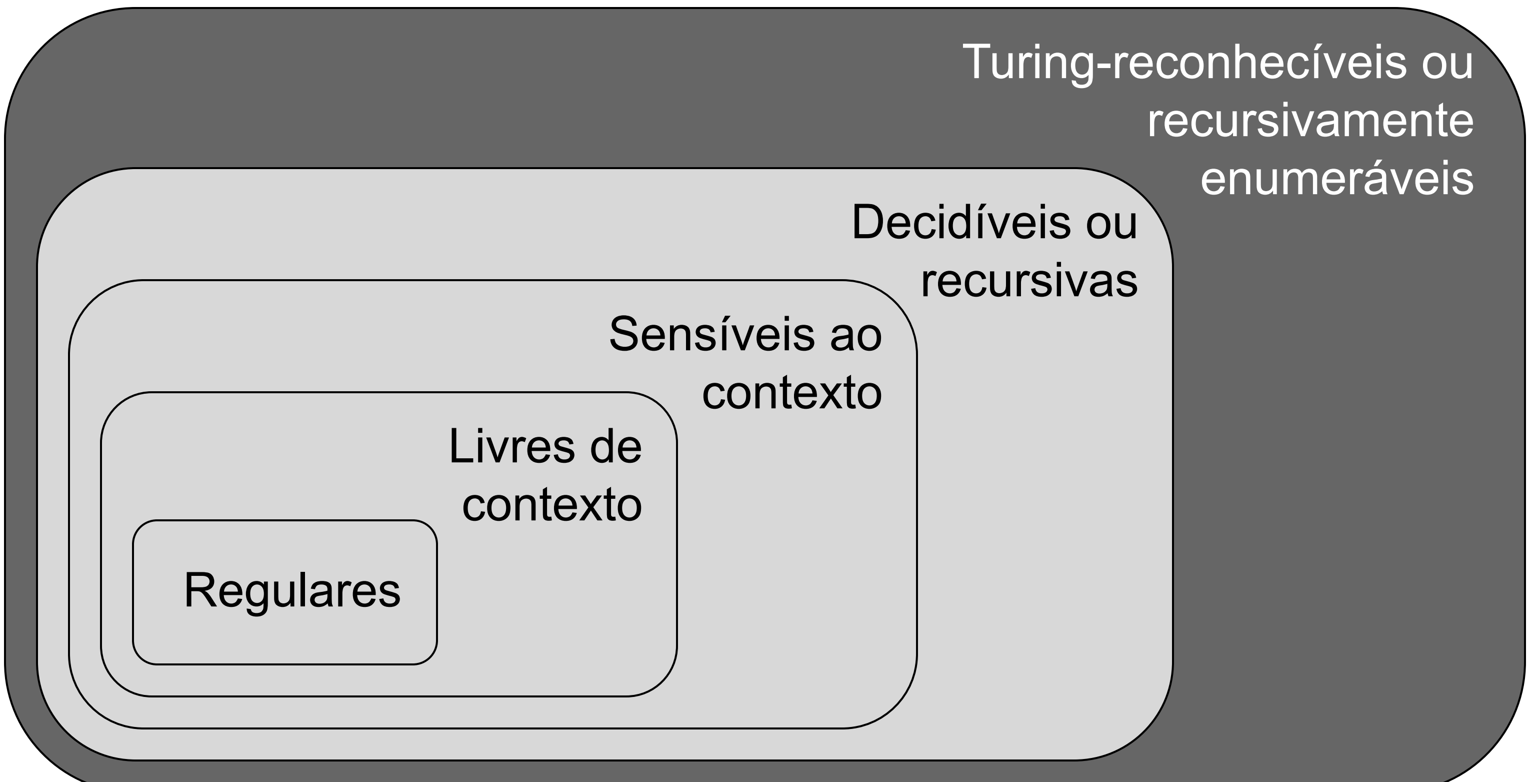
Linguagens (problemas) indecidíveis



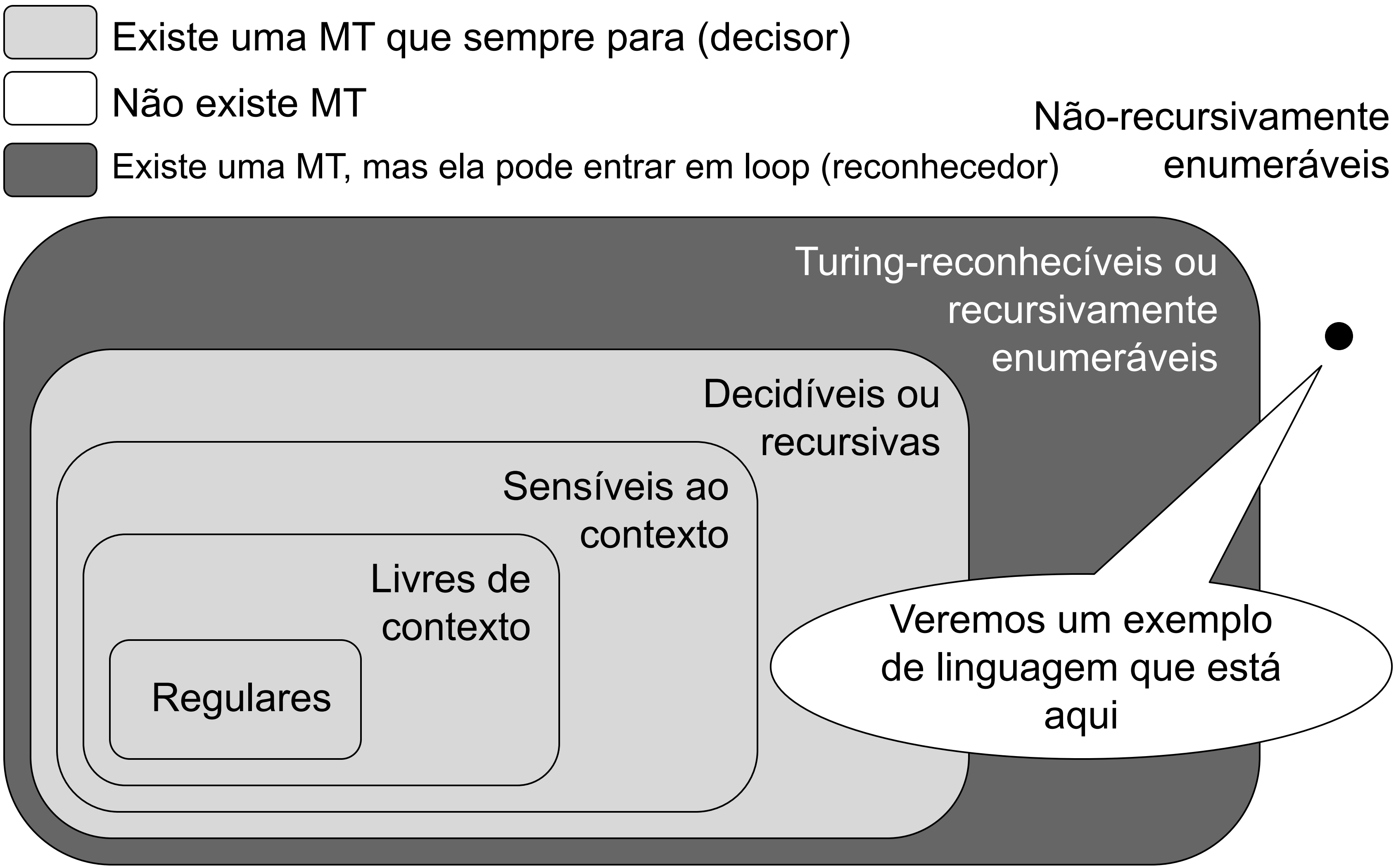
Hierarquia de linguagens

- Existe uma MT que sempre para (decisor)
- Não existe MT
- Existe uma MT, mas ela pode entrar em loop (reconhecedor)

Não-recursivamente enumeráveis



Hierarquia de linguagens



Mas antes, alguns conceitos

- Enumeração de strings binários
  - Veremos como codificar algumas coisas como strings binários
    - A exemplo do que fizemos na codificação das listas do PCP
  - Será útil atribuir inteiros a todos os strings binários possíveis
    - Cada string irá corresponder a um único inteiro
    - E cada inteiro irá corresponder a um único string

1	ε
2	0
3	1
4	00
5	01
6	11
...	...
i	wi
...	...

Mas antes, alguns conceitos

- Enumeração de strings binários
  - Veremos como codificar algumas coisas como strings binários
    - A exemplo do que fizemos na codificação das listas do PCP
  - Será útil atribuir inteiros a todos os strings binários possí
    - Cada string irá corresponder a um único inteiro
    - E cada inteiro irá corresponder a um único string

segundo string

1	ε
2	0
3	1
4	00
5	01
6	11
	...
i	wi
...	...

quinto string

i-ésimo string

Códigos para máquinas de Turing

- Vamos criar um código binário para máquinas de Turing
- Ou seja, representaremos uma máquina de Turing M em uma longa string de 0s e 1s

$$M = (Q, \Sigma, \Gamma, \delta, q0, B, F)$$

1	0	1	1	1	0	0	1	0	1	1	0	1	1	0	1	1	0	1	0	0	1	1	0	1	0	1	1	1	0	0	1	1	1	0	0	0

Atenção! Exemplo meramente ilustrativo. Essa não é uma codificação válida de uma MT!

Códigos para máquinas de Turing

- Para que codificar uma MT em binário?
- Acabamos de enumerar todos os strings binários
- Poderemos, portanto, enumerar todas as MTs possíveis!
- Essa enumeração será útil na prova da indecidibilidade

Índice	**MT**	**MT codificada em binário**
1	M1	ε
2	M2	0
3	M3	1
4	M4	00
...	...	...
293924	M293924	10110101000100110101010010010101001001110 1...
293925	M293925	10110101000100110101010010010101001001111 0...
...	...	...
i	Mi	wi
...	...	...

Códigos para máquinas de Turing

- Para que codificar uma MT em binário?
- Acabamos de enumerar todos os strings binários
- Poderemos, portanto, enumerar todas as MT[illegible] possíveis!
- Essa enumera[illegible] será [illegible]

Índice	MT	MT codificada em binário
1	M1	ε
2	M2	0
3	M3	1
4	M4	00
...	...	...
293924	M293924	1011010100010011010101 00100101010010011101...
293925	M293925	1011010100010011010101 00100101010010011110...
...	...	...
i	Mi	wi
...	...	...

Obviamente, as primeiras posições não são códigos "válidos" para MTs

Códigos para máquinas de Turing

- **M = (Q,Σ,Γ,δ,q0,B,F)**
  - Primeiro: Σ = {0,1}
  - Ou seja, assumimos que a entrada é codificada em binário
- Em seguida, precisamos codificar estados, símbolos de fita e sentidos E e D
  - Q = {q1,q2,...,qr}
    - chamaremos os estados de q1,q2,...,qr, para algum r
    - q1 será o estado inicial (q0)
    - q2 será o único estado de aceitação (F) (é sempre possível converter uma MT com mais de um estado de aceitação para uma que tenha apenas um estado de aceitação)
    - Usaremos um código simples para representar cada qi
    - Ex: q1 = 0, q2 = 00, q3 = 000, q4 = 0000, ...

Códigos para máquinas de Turing

- **M = (Q,Σ,Γ,δ,q0,B,F)**
    - Γ = {X1,X2,...,Xs}
        - chamaremos os símbolos de fita de X1,X2,...,Xs, para algum s
        - X1 será o símbolo 0
        - X2 será o símbolo 1
        - X3 será B, o branco
        - Outros símbolos podem ser atribuídos aos inteiros restantes (4,5,6, ...)
        - Usaremos o código anterior para representar cada Xi
        - Ex: X1 = 0, X2 = 00, X3 = 000, X4 = 0000, ...

Códigos para máquinas de Turing

- **M = (Q,Σ,Γ,δ,q0,B,F)**
  - Sentido E e D:
    - D1 = esquerda
    - D2 = direita
    - Usaremos o código anterior para representar cada Di
    - Ex: D1 = 0, D2 = 00

Códigos para máquinas de Turing

- **M = (Q,Σ,Γ,δ,q0,B,F)**
  - δ:
    - Uma regra de transição no formato:
    - $\delta(q_i, X_j) = (q_k, X_l, D_m)$
    - É codificada como:
    - $0^i10^j10^k10^l10^m$
    - Nesse código, i,j,k,l e m são no mínimo 1, ou seja, não existirá nenhuma ocorrência de dois ou mais 1s consecutivos dentro de uma única transição

Códigos para máquinas de Turing

- **M = (Q,Σ,Γ,δ,q0,B,F)**
  - M:
  - O código para a máquina M será:

$$C_1\ 11\ C_2\ 11\ \dots\ C_{n-1}\ 11\ C_n$$

  - Onde cada um dos C's é o código para uma transição de M
- Obs: nenhum código válido para uma MT possui três 1s em sequência
  - Isso será útil depois

Códigos para máquinas de Turing

- Exemplo: para a seguinte MT:
- M = ({q1,q2,q3},{0,1},{0,1,B},δ,q1,B,{q2})
- Onde δ é definido pelas regras:
  - $\delta(q_1, 1) = (q_3, 0, D)$
  - $\delta(q_3, 0) = (q_1, 1, D)$
  - $\delta(q_3, 1) = (q_2, 0, D)$
  - $\delta(q_3, B) = (q_3, 1, E)$
- Os códigos para as regras são:
  - 0 1 00 1 000 1 0 1 00
  - 000 1 0 1 0 1 00 1 00
  - 000 1 00 1 00 1 0 1 00
  - 000 1 000 1 000 1 00 1 0
- O código para M é
  - 0100100010100 11 0001010100100 11 00010010010100 11 0001000100010010

Códigos para máquinas de Turing

- Existem muitos códigos "inválidos"
  - 1, 0, 11, 111111111111, etc
- Nesses casos, assumiremos que a MT possui apenas um estado e nenhuma transição
  - Ou seja, a MT para imediatamente sobre qualquer entrada, sem aceitar
  - Ou seja, a linguagem dessas máquinas é vazia (Ø)

Índice	**MT**	**MT codificada em binário**	**Linguagem**
1	M1	ε	Ø
2	M2	0	Ø
3	M3	1	Ø
4	M4	00	Ø
...	...	...	...
293924	M293924	0100100010100110 0010101001...	L293924 = {11,101,...}
293925	M293925	10110110010010 1010010011110...	Ø
...	...	...	...
i	Mi	wi	Li
...	...	...	...

Códigos para máquinas de Turing

- Existem muitos códigos “inválidos”
  - 1, 0, 11, 111111111111, etc
- Nesses casos, assumiremos que a MT possui apenas um estado e nenhuma transição
  - Ou seja, a MT para imediatamente sobre qualquer entrada, sem aceitar
  - Ou seja, a linguagem dessas máquinas é vazia (Ø)

Índice	MT	MT codificada em binário	Linguagem
1	M1	ε	Ø
[illegible]	[illegible]	0	Ø
3	M3	1	Ø
[illegible]	[illegible]	00	Ø
...	...	...	...
[illegible]	[illegible]	0100100010100110 0010101001...	L293924 = {11,101,...}
293925	M293925	10110110010010 1010010011110...	Ø
...	[illegible]	...	...
[illegible]	[illegible]	wi	Li
...	...	...	...

Código “inválido”

Código “inválido”

Código “válido”

Código “inválido”

Uma definição vital

Uma definição vital

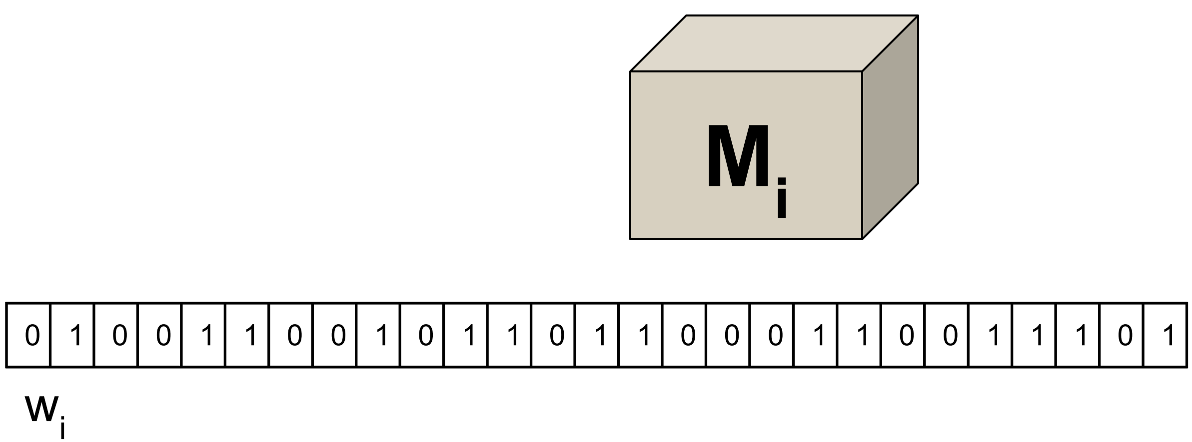
- A linguagem L_d, a *linguagem da diagonalização*, é o conjunto de strings w_i, tais que w_i não está em $L(M_i)$
- Vamos destrinchar essa definição:
  - w_i é uma string:

0	1	0	0	1	1	0	0	1	0	1	1	0	1	1	0	0	0	1	1	0	0	1	1	1	0	1

w_i

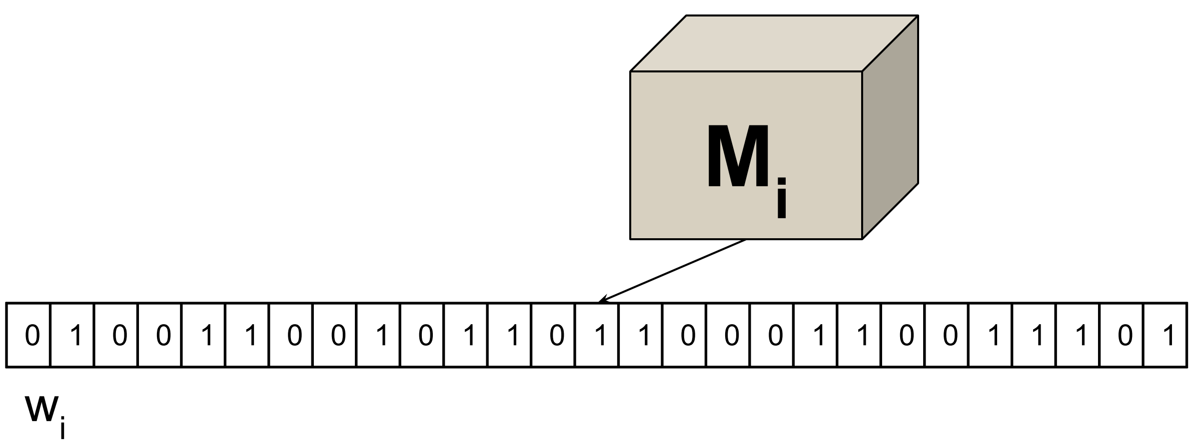
Uma definição vital

- A linguagem L_d, a *linguagem da diagonalização*, é o conjunto de strings w_i, tais que w_i não está em $L(M_i)$
- Vamos destrinchar essa definição:
  - w_i é uma string:
    - w_i codifica uma MT ("válida" ou "inválida") chamada M_i



Uma definição vital

- A linguagem L_d, a *linguagem da diagonalização*, é o conjunto de strings w_i, tais que w_i não está em $L(M_i)$
- Vamos destrinchar essa definição:
  - w_i é uma string:
    - w_i codifica uma MT ("válida" ou "inválida") chamada M_i
    - Usaremos wi como entrada para Mi



Uma definição vital

- Se M_i aceitar w_i, w_i não faz parte de L_d
- Se M_i não aceitar w_i, w_i faz parte de L_d
- Ou seja, L_d consiste de todos os strings que codificam máquinas de Turing que não aceitam quando recebem a si mesmas como entrada

Diagonalização

- Considere a tabela à direita
  - Cada célula diz se uma Máquina de Turing M_i aceita o string de entrada w_j
    - 1 significa "sim, M_i aceita w_j"
    - 0 significa "não, M_i não aceita w_j"
- Cada linha i é chamada de vetor característico para a linguagem $L(M_i)$
  - Cada 1 nessa linha indica uma string que faz parte dessa linguagem
- Exs:
  - M_4 aceita w_2 e w_4
  - M_3 não aceita w_1 nem w_2
  - M_2 aceita w_1 e w_2

				j		
		1	2	3	4	...
i	1	0	1	1	0	...
	2	1	1	0	0	...
	3	0	0	1	1	...
	4	0	1	0	1	...
	...	...	...	...	...	...

Exemplo ilustrativo, pois as primeiras posições não são códigos "válidos" para MTs

Diagonalização

- Nessa tabela, a diagonal é interessante
  - Informam as MTs que aceitam a si próprias como entrada
  - Ou seja, o complemento de L_d
- Para construir L_d, basta complementar a diagonal
  - Nesse exemplo: 1000 ...
  - Ou seja:
    - w_1 pertence a L_d
    - w_2 não pertence a L_d
    - w_3 não pertence a L_d
    - w_4 não pertence a L_d
    - Etc...

		j				
		1	2	3	4	...
	1	0	1	1	0	...
	2	1	1	0	0	...
i	3	0	0	1	1	...
	4	0	1	0	1	...
	...	...	...	...	...	...

Diagonalização

- Suponha que L_d fosse L(M) para alguma MT M
  - Então existiria uma MT cujo vetor característico é o complemento da diagonal da tabela à direita
  - Ou seja, em alguma linha, por exemplo k, o complemento da diagonal (1000...) iria aparecer

		j						
		1	2	3	4	...	k	...
	1	0	1	1	0	...		...
	2	1	1	0	0	...		...
	3	0	0	1	1	...		...
i	4	0	1	0	1	...		...
	...	...	...	...	...	...		...
	k	1	0	0	0	...	?	...
	...	...	...	...	...	...		...

Diagonalização

- Observe a célula marcada com "?"
  - Ela fica no encontro entre a diagonal e a linha hipotética k
  - Que valor deveria ser colocado nessa célula?
- Precisamos olhar sob o ponto de vista da linha k e da diagonal
- Devemos analisar o processo de "transposição" da diagonal para a linha k

		j						
		1	2	3	4	...	k	...
	1	0	1	1	0	...		...
	2	1	1	0	0	...		...
	3	0	0	1	1	...		...
i	4	0	1	0	1	...		...
	...	...	...	...	...	...		...
	k	1	0	0	0	...	?	...
	...	...	...	...	...	...		...

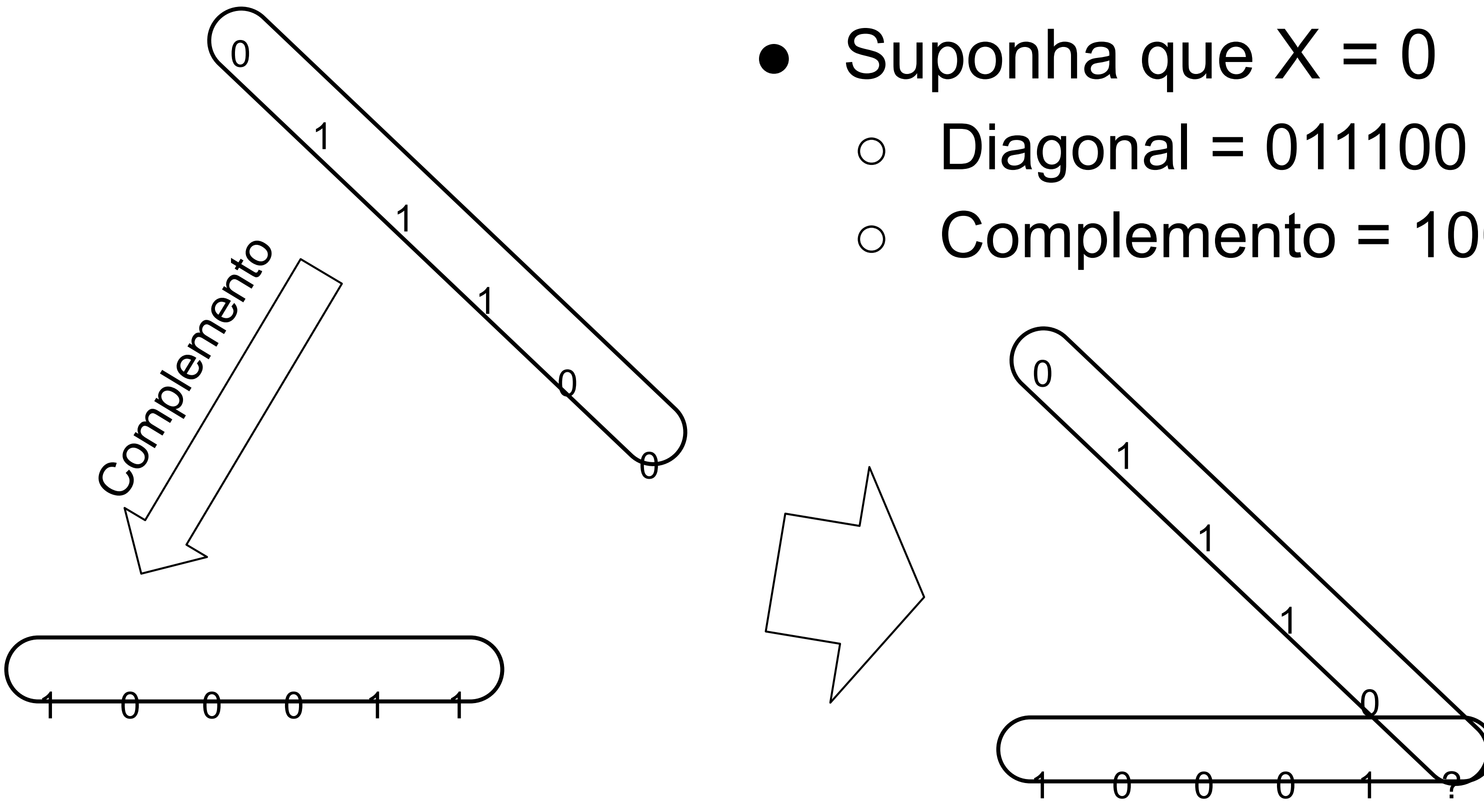
Diagonalização

		j						
		1	2	3	4	5	6	...
	1	0	1	1	0	1	1	...
	2	1	1	0	0	1	0	...
	3	0	0	1	1	1	0	...
i	4	0	1	0	1	1	0	...
	5	1	1	0	1	0	1	...
	6	1	0	0	0	1	X	...
	...	...	...	...	...	...	...	...

- Acompanhe no exemplo
- Suponha que k = 6
- Suponha que X = 0
  - Diagonal = 011100
  - Complemento = 100011

Diagonalização

- Acompanhe no exemplo
- Suponha que k = 6
- Suponha que X = 0
  - Diagonal = 011100
  - Complemento = 100011



Diagonalização

- O mesmo aconteceria para qualquer k, e para qualquer X
- Ou seja, há um paradoxo, uma contradição
- Nossa suposição de que M_k existe deve ser falsa
- Portanto, não existe uma máquina de Turing M_k
- Ou seja, a linguagem L_d não é reconhecível por uma máquina de Turing

Ou seja, L_d não é uma linguagem recursivamente enumerável

Uma linguagem não-RE

- O que significa isso?
- O “problema” Ld é indecidível
- Ou seja, dada uma máquina de Turing, é impossível decidir (determinar/resolver) se ela aceita a si mesma como entrada
- Ou seja, não existe um algoritmo que faça isso, para qualquer máquina de Turing

Fim

Aula 25 - Linguagem da diagonalização