Teoria da Computação

Prof. Maicon R. Zatelli

Aula 4 - Tese de Church-Turing

Universidade Federal de Santa Catarina
Florianópolis - Brasil

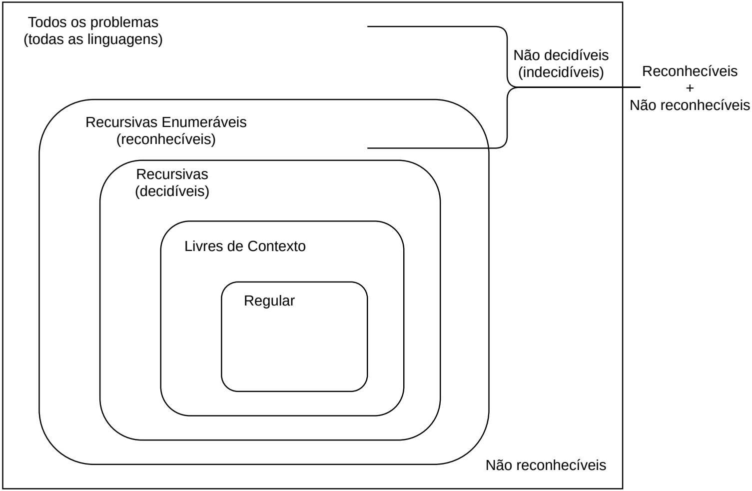
Introdução

- Máquina de Turing
- Variantes da máquina de Turing
- Definição de algoritmo

Material de apoio

- Livro Sipser, Capítulo 3
- Livro Hopcroft, Capítulo 8

Hierarquia de Chomsky



Indecidíveis - são todas as linguagens Turing-reconhecíveis mas não decidíveis e também as linguagens não Turing-reconhecíveis

Tese de Church-Turing

Alan Turing (1936) - criou a Máquina de Turing (MT)

- Motivação: capturar a noção de computação/computabilidade
- Quais todos os problemas computacionais podem ser resolvidos?
- O que "computável"significa?

Tudo que pode ser feito por um computador moderno pode ser feito por uma Máquina de Turing e vice-versa

Qualquer computação que pode ser executada por meios mecânicos pode ser executada por uma Máquina de Turing

Hipótese de Church

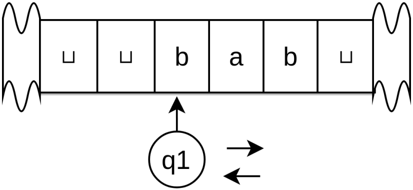
- A capacidade de computação representada pela Máquina de Turing é o limite máximo que pode ser atingido por qualquer dispositivo de computação

Tese de Church-Turing

Outros formalismos para descrever algoritmos/computabilidade:

- Cálculo Lambda ($\lambda - \mathit{Calculus}$) - Alonzo Church
- Máquinas de Post - Emil Post
- Funções recursivas - Kurt Gödel
- Algoritmos de Markov - Andrey Markov
- ...

Máquina de Turing



- Fina infinita é dividida em células
- Cabeçote lê e escreve símbolos na fita
- Cabeçote move para a esquerda e para a direita exatamente uma posição
- Cabeçote armazena um estado
- Células da fita contém exatamente um símbolo
- Células não inicialmente preenchidas possuem um símbolo de vazio

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	0	1	0	1	1	2	0	1	0	1	1	$\sqcup$
	$\uparrow$											

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	1	0	1	1	2	0	1	0	1	1	$\sqcup$
		$\uparrow$										
		$\rightarrow$										

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	1	0	1	1	2	0	1	0	1	1	$\sqcup$
							$\uparrow$					

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	1	0	1	1	2	x	1	0	1	1	$\sqcup$
						$\uparrow$						
						$\leftarrow$						

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	1	0	1	1	2	x	1	0	1	1	$\sqcup$
	$\uparrow$											
	$\rightarrow$											

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	1	0	1	1	2	x	1	0	1	1	$\sqcup$
		$\uparrow$										

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0, 1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	x	0	1	1	2	x	1	0	1	1	$\sqcup$
			$\uparrow$									
			$\rightarrow$									

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	x	0	1	1	2	x	1	0	1	1	$\sqcup$
								$\uparrow$				

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	x	0	1	1	2	x	x	0	1	1	$\sqcup$
							$\uparrow$					
							$\leftarrow$					

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

⊔	x	x	0	1	1	2	x	x	0	1	1	⊔
							↑					
		...	...	...	...	...	... ←					

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de ⊔, rejeita. Se leu símbolos 0 ou 1 e depois ⊔ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois ⊔, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	x	x	x	x	2	x	x	x	x	1	$\sqcup$
											$\uparrow$	

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	x	x	x	x	2	x	x	x	x	x	$\sqcup$
										$\uparrow$		
										$\leftarrow$		

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	x	x	x	x	2	x	x	x	x	x	$\sqcup$
					$\uparrow$							

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	x	x	x	x	2	x	x	x	x	x	$\sqcup$

$\uparrow$

$\rightarrow$

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	x	x	x	x	2	x	x	x	x	x	$\sqcup$
												$\uparrow$

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	x	x	x	x	x	2	x	x	x	x	x	$\sqcup$

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$M = (Q, \Sigma, \Gamma, \delta, q0, q_{acc}, q_{rej})$

- Q é um conjunto finito de estados
- Σ é um conjunto finito de símbolos de entrada (alfabeto de entrada)
- Γ é um conjunto finito de símbolos da fita (alfabeto da fita), onde $\Sigma \subset \Gamma$ e $\sqcup \in \Gamma$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ é uma função de transição
  - δ(estado atual,símbolo lido fita) $\rightarrow$ $\{$(novo estado,novo símbolo da fita,direção movimento cabeçote)$\}$
- $q0 \in Q$ é um estado inicial
- $q_{acc} \in Q$ é um estado de aceitação
- $q_{rej} \in Q$ é um estado de rejeição, onde $q_{acc} \neq q_{rej}$

Máquina de Turing

$M = (Q, \Sigma, \Gamma, \delta, q0, q_{acc}, q_{rej})$

- Q é um conjunto finito de estados
- Σ é um conjunto finito de símbolos de entrada (alfabeto de entrada)
- Γ é um conjunto finito de símbolos da fita (alfabeto da fita), onde $\Sigma \subset \Gamma$ e $\sqcup \in \Gamma$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ é uma função de transição
  - δ(estado atual,símbolo lido fita) $\to$ $\{$(novo estado,novo símbolo da fita,direção movimento cabeçote)$\}$
- $q0 \in Q$ é um estado inicial
- $q_{acc} \in Q$ é um estado de aceitação
- $q_{rej} \in Q$ é um estado de rejeição, onde $q_{acc} \neq q_{rej}$

Uma transição $\delta(q, a) \to (p, b, R)$ indica que a MT, ao estar no estado q e ler o símbolo a da fita, irá para o estado p, escreverá b no lugar de a e moverá o cabeçote para a direita (R)

Máquina de Turing

Ao iniciar, a Máquina de Turing (MT) contém em sua fita os símbolos da palavra de entrada, sendo todas as demais posições da fita preenchidas com $\sqcup$

O estado da MT será o estado inicial e o cabeçote estará posicionado no primeiro símbolo da entrada

Enquanto a MT computa, ocorrem mudanças em sua configuração

- no estado atual
- no conteúdo da fita
- na posição do cabeçote

Formalmente, a **configuração de uma MT** é dada por uqv onde uv é o conteúdo da fita, q é o estado atual e o cabeçote aponta para o primeiro símbolo de v

Máquina de Turing

O **histórico de computação de uma** MT é a sequência de configurações geradas durante a computação

- $\underline{\mathbf{q0}}01101 \rightarrow x\underline{\mathbf{q1}}1101 \rightarrow xx\underline{\mathbf{q1}}101$

Os estados de aceite e rejeite são **estados de parada**, ou seja, uma vez a MT alcançar um deles, ela irá parar, aceitando ou rejeitando a palavra de entrada

Uma MT que sempre para, tanto para aceitar como para rejeitar a sua entrada, é chamada de **MT decisora**

Uma MT que sempre para apenas quando a entrada pertence à linguagem, mas pode não parar quando a entrada não pertence à linguagem é chamada de **MT reconhecedora** (ou **MT aceitadora**)

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	a	a	a	b	b	b	c	c	c	⊔
	↑									

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	a	a	b	b	b	c	c	c	⊔
		$\uparrow$								
		$\rightarrow$								

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

$\sqcup$	x	a	a	b	b	b	c	c	c	$\sqcup$
				$\uparrow$						

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	a	a	x	b	b	c	c	c	⊔

$$\uparrow$$
$$\rightarrow$$

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	a	a	x	b	b	c	c	c	⊔
							↑			

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	a	a	x	b	b	x	c	c	⊔
						$\uparrow$				
						$\leftarrow$				

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	a	a	x	b	b	x	c	c	⊔

↑

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	a	a	x	b	b	x	c	c	⊔
	$\uparrow$									
	$\rightarrow$									

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

$\sqcup$	x	a	a	x	b	b	x	c	c	$\sqcup$
		$\uparrow$								

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	a	x	b	b	x	c	c	⊔
			↑							
			→							

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	a	x	b	b	x	c	c	⊔
					↑					

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w|w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	a	x	x	b	x	c	c	⊔
						$\uparrow$				
						$\rightarrow$				

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	a	x	x	b	x	c	c	⊔
								↑		

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

$\sqcup$	x	x	a	x	x	b	x	x	c	$\sqcup$
							$\uparrow$			
							$\leftarrow$			

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

$\sqcup$	x	x	a	x	x	b	x	x	c	$\sqcup$

$\uparrow$

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	a	x	x	b	x	x	c	⊔
	$\uparrow$									
	$\rightarrow$									

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	a	x	x	b	x	x	c	⊔
			$\uparrow$							

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	x	x	x	b	x	x	c	⊔
				$\uparrow$						
				$\rightarrow$						

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	x	x	x	b	x	x	c	⊔
						↑				

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	x	x	x	x	x	x	c	⊔
							↑			
							→			

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	x	x	x	x	x	x	c	⊔
									↑	

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	x	x	x	x	x	x	x	⊔

$\uparrow$

$\leftarrow$

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

$\sqcup$	x	x	x	x	x	x	x	x	x	$\sqcup$
$\uparrow$										

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

⊔	x	x	x	x	x	x	x	x	x	⊔
	$\uparrow$									
	$\rightarrow$									

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

$\sqcup$	x	x	x	x	x	x	x	x	x	$\sqcup$
										$\uparrow$

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada *aaabbbccc*

$\sqcup$	x	x	x	x	x	x	x	x	x	$\sqcup$
										$\uparrow$
										$\checkmark$

- Move em zig-zag. A cada *a* encontrado devemos encontrar um *b* e depois um *c*.
- Se encontrar *a* e depois não encontrar *b*, rejeita. Se encontrar *a* e *b* e depois não encontrar *c*, rejeita.
- Ao não encontrar mais *a*, *b* ou *c*, aceita.

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Definição do alfabeto de entrada e da fita

- $\Sigma = \{a, b, c\}$
- $\Gamma = \{a, b, c, x, \sqcup\}$

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Definição do alfabeto de entrada e da fita

- $\Sigma = \{a, b, c\}$
- $\Gamma = \{a, b, c, x, \sqcup\}$

Definição dos estados

- qa = procura e marca o primeiro a
- qb = procura e marca o primeiro b
- qc = procura e marca o primeiro c
- qv = volta até encontrar o início da entrada ($\sqcup$)
- $qacc$ = estado de aceitação
- $qrej$ = estado de rejeição

Máquina de Turing

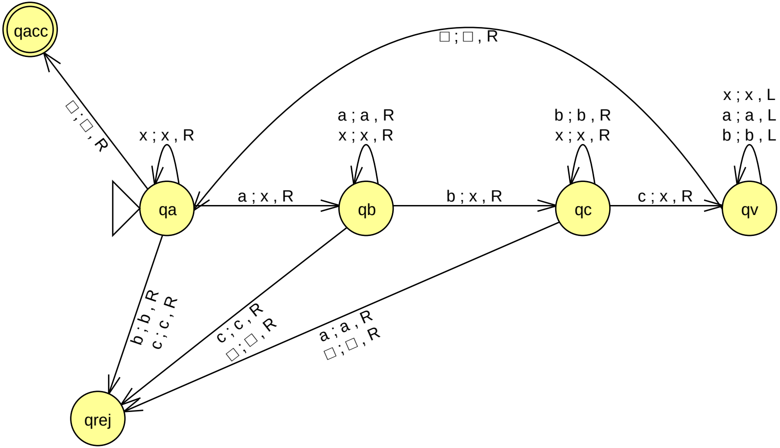
$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Definição das transições

- $\delta(qa, a) \rightarrow (qb, x, R)$
- $\delta(qa, b) \rightarrow (qrej, b, R)$
- $\delta(qa, c) \rightarrow (qrej, c, R)$
- $\delta(qa, x) \rightarrow (qa, x, R)$
- $\delta(qa, \sqcup) \rightarrow (qacc, \sqcup, R)$
- $\delta(qb, a) \rightarrow (qb, a, R)$
- $\delta(qb, b) \rightarrow (qc, x, R)$
- $\delta(qb, c) \rightarrow (qrej, c, R)$
- $\delta(qb, x) \rightarrow (qb, x, R)$
- $\delta(qb, \sqcup) \rightarrow (qrej, \sqcup, R)$
- ...

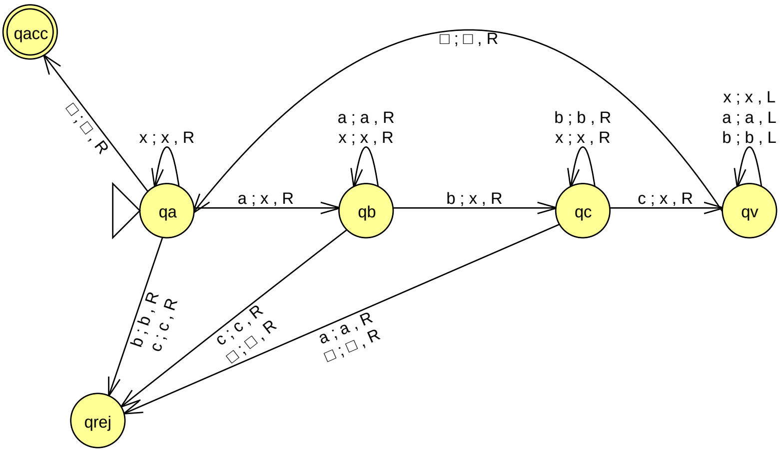
Máquina de Turing

$$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$$



Máquina de Turing

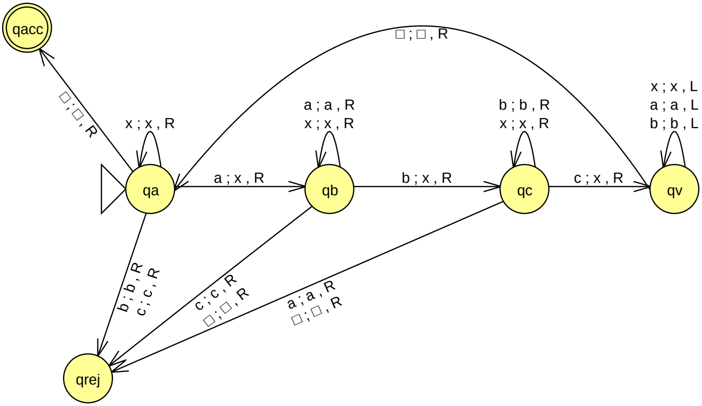
$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$



Será que funciona? Tente com a entrada: *aaabcbbcc*

Máquina de Turing

$$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$$



⊔	a	a	a	b	c	b	b	c	c	⊔

↑

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

É necessário garantir que depois de ler o primeiro c não hajam a e b após ele. É necessário analisar até o final da palavra!

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

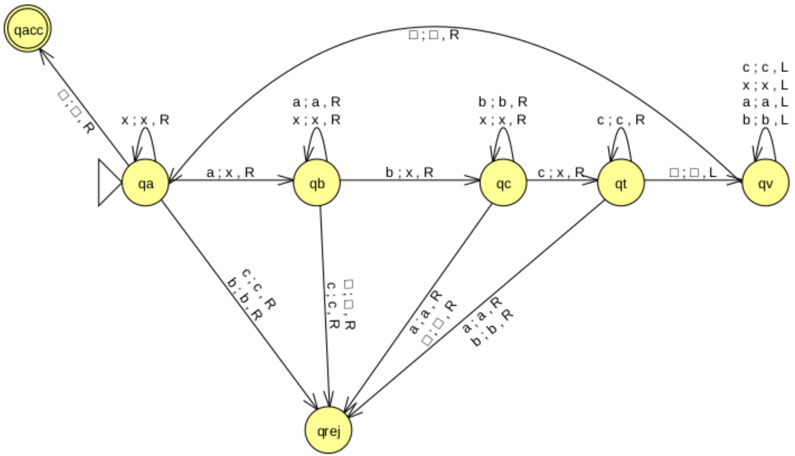
É necessário garantir que depois de ler o primeiro c não hajam a e b após ele. É necessário analisar até o final da palavra!

Definição dos estados

- qa = procura e marca o primeiro a
- qb = procura e marca o primeiro b
- qc = procura e marca o primeiro c
- qt = vai até o final da entrada verificando se há a ou b após o primeiro c
- qv = volta até encontrar o início da entrada ($\sqcup$)
- $qacc$ = estado de aceitação
- $qrej$ = estado de rejeição

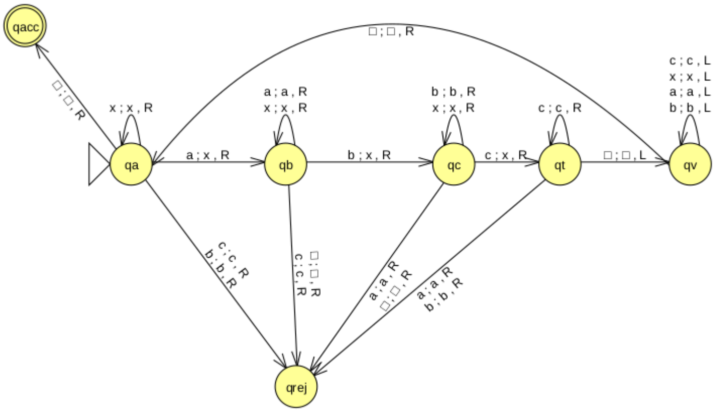
Máquina de Turing

$$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$$



Máquina de Turing

$$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$$

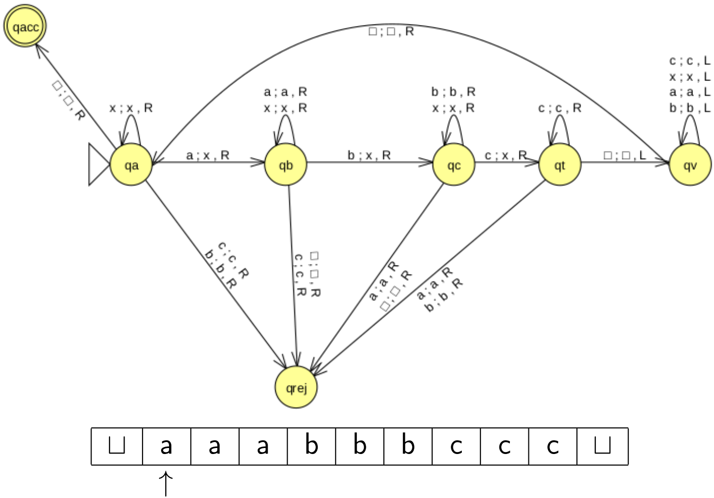


⊔	a	a	a	b	c	b	b	c	c	⊔

↑

Máquina de Turing

$$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$$



Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Representação em forma de tabela

	a	b	c	$\sqcup$	x
$\rightarrow qa$	(qb, x, R)	$(qrej, b, R)$	$(qrej, c, R)$	$(qacc, \sqcup, R)$	(qa, x, R)
qb	(qb, a, R)	(qc, x, R)	$(qrej, c, R)$	$(qrej, \sqcup, R)$	(qb, x, R)
qc	$(qrej, a, R)$	(qc, b, R)	(qt, x, R)	$(qrej, \sqcup, R)$	(qc, x, R)
qt	$(qrej, a, R)$	$(qrej, b, R)$	(qt, c, R)	$(qv, \sqcup, L)$	$(qrej, x, R)$
qv	(qv, a, L)	(qv, b, L)	(qv, c, L)	$(qa, \sqcup, R)$	(qv, x, R)
$* \ qacc$					
$\dagger \ qrej$					

Máquina de Turing

$L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^n c^n \text{ e } n \geq 0\}$

Representação em forma de tabela

	a	b	c	$\sqcup$	x
$\rightarrow qa$	(qb, x, R)	$(qrej, b, R)$	$(qrej, c, R)$	$(qacc, \sqcup, R)$	(qa, x, R)
qb	(qb, a, R)	(qc, x, R)	$(qrej, c, R)$	$(qrej, \sqcup, R)$	(qb, x, R)
qc	$(qrej, a, R)$	(qc, b, R)	(qt, x, R)	$(qrej, \sqcup, R)$	(qc, x, R)
qt	$(qrej, a, R)$	$(qrej, b, R)$	(qt, c, R)	$(qv, \sqcup, L)$	$(qrej, x, R)$
qv	(qv, a, L)	(qv, b, L)	(qv, c, L)	$(qa, \sqcup, R)$	(qv, x, R)
$* \ qacc$					
$\dagger \ qrej$					

Tente criar Máquinas de Turing para as linguagens abaixo:

- $L = \{w | w \in \{a, b, c\}^* \text{ e } w = a^n b^{2n} c^n \text{ e } n \geq 0\}$
- $L = \{w | w \in \{a, b, c\}^*$ e w possui o mesmo número de a, b e c, independente da ordem em que eles aparecem $\}$

Máquina de Turing

$L = \{w|w \in \{0\}^* \text{ e } w = 0^{2^n} \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada 00000000

⊔	0	0	0	0	0	0	0	0	⊔

$\uparrow$

- Note que para ter uma palavra cujo tamanho é uma potência de 2 temos que poder dividir o tamanho dela pela metade até chegarmos no valor 1
  - Ex: $8/2 = 4 \rightarrow 4/2 = 2 \rightarrow 2/2 = 1$
- Observe que todos os valores gerados no processo de divisões sucessivas sempre é par, exceto pelo 1 gerado na última divisão
- Assim, podemos perceber que se durante o processo de divisões sucessivas algum valor resultante for ímpar, então a palavra certamente não pertence à linguagem
- Caso chegarmos ao valor 1, então devemos aceitar

Máquina de Turing

$L = \{w|w \in \{0\}^* \text{ e } w = 0^{2^n} \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada 00000000

⊔	**0**	0	0	0	0	0	0	0	⊔

- Note que para ter uma palavra cujo tamanho é uma potência de 2 temos que poder dividir o tamanho dela pela metade até chegarmos no valor 1
  - Ex: $8/2 = 4 \rightarrow 4/2 = 2 \rightarrow 2/2 = 1$
- Observe que todos os valores gerados no processo de divisões sucessivas sempre é par, exceto pelo 1 gerado na última divisão
- Assim, podemos perceber que se durante o processo de divisões sucessivas algum valor resultante for ímpar, então a palavra certamente não pertence à linguagem
- Caso chegarmos ao valor 1, então devemos aceitar

Máquina de Turing

$L = \{w | w \in \{0\}^* \text{ e } w = 0^{2^n} \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada 00000000

$\sqcup$	**0**	0	0	0	0	0	0	0	$\sqcup$
$\sqcup$	**0**	x	0	x	0	x	0	x	$\sqcup$

- Note que para ter uma palavra cujo tamanho é uma potência de 2 temos que poder dividir o tamanho dela pela metade até chegarmos no valor 1
  - Ex: $8/2 = 4 \rightarrow 4/2 = 2 \rightarrow 2/2 = 1$
- Observe que todos os valores gerados no processo de divisões sucessivas sempre é par, exceto pelo 1 gerado na última divisão
- Assim, podemos perceber que se durante o processo de divisões sucessivas algum valor resultante for ímpar, então a palavra certamente não pertence à linguagem
- Caso chegarmos ao valor 1, então devemos aceitar

Máquina de Turing

$L = \{w | w \in \{0\}^* \text{ e } w = 0^{2^n} \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada 00000000

⊔	**0**	x	0	x	0	x	0	x	⊔
⊔	**0**	x	x	x	0	x	x	x	⊔

- Note que para ter uma palavra cujo tamanho é uma potência de 2 temos que poder dividir o tamanho dela pela metade até chegarmos no valor 1
  - Ex: $8/2 = 4 \rightarrow 4/2 = 2 \rightarrow 2/2 = 1$
- Observe que todos os valores gerados no processo de divisões sucessivas sempre é par, exceto pelo 1 gerado na última divisão
- Assim, podemos perceber que se durante o processo de divisões sucessivas algum valor resultante for ímpar, então a palavra certamente não pertence à linguagem
- Caso chegarmos ao valor 1, então devemos aceitar

Máquina de Turing

$L = \{w | w \in \{0\}^* \text{ e } w = 0^{2^n} \text{ e } n \geq 0\}$

Intuição de funcionamento com a palavra de entrada 00000000

⊔	**0**	x	x	x	0	x	x	x	⊔
⊔	**0**	x	x	x	x	x	x	x	⊔
									✓

- Note que para ter uma palavra cujo tamanho é uma potência de 2 temos que poder dividir o tamanho dela pela metade até chegarmos no valor 1
  - Ex: $8/2 = 4 \rightarrow 4/2 = 2 \rightarrow 2/2 = 1$
- Observe que todos os valores gerados no processo de divisões sucessivas sempre é par, exceto pelo 1 gerado na última divisão
- Assim, podemos perceber que se durante o processo de divisões sucessivas algum valor resultante for ímpar, então a palavra certamente não pertence à linguagem
- Caso chegarmos ao valor 1, então devemos aceitar

Máquina de Turing

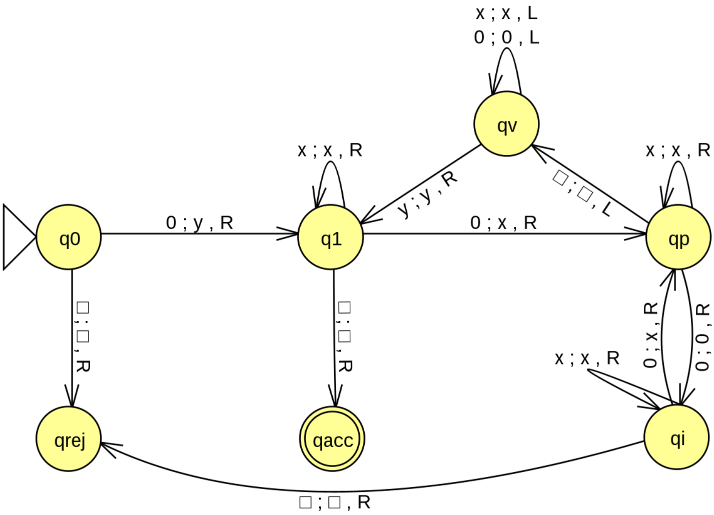
$L = \{w | w \in \{0\}^* \text{ e } w = 0^{2^n} \text{ e } n \geq 0\}$

Definição dos estados

- $q0$ = verifica se a palavra é vazia ou encontra ao menos um 0
- $q1$ = resta ao menos um 0
- qp = quantidade de 0's é par na passagem atual
- qi = quantidade de 0's é ímpar na passagem atual
- qv = volta até encontrar o primeiro 0
- $qacc$ = estado de aceitação
- $qrej$ = estado de rejeição

Máquina de Turing

$$L = \{w | w \in \{0\}^* \text{ e } w = 0^{2^n} \text{ e } n \geq 0\}$$



Máquina de Turing

$L = \{w2w | w \in \{0,1\}^*\}$

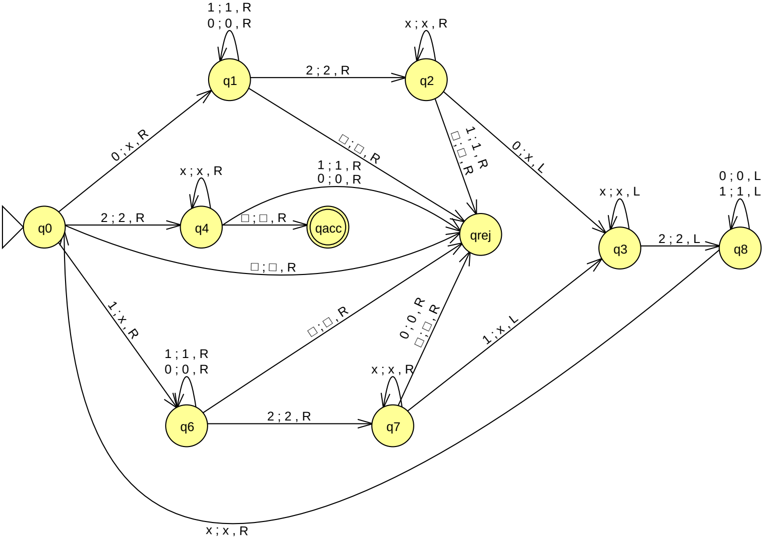
Intuição de funcionamento com a palavra de entrada 01011201011

$\sqcup$	0	1	0	1	1	2	0	1	0	1	1	$\sqcup$
	$\uparrow$											

- Move em zig-zag e marca símbolos buscando pelo mesmo à direita do símbolo 2
- Se o símbolo à direita de 2 é diferente, rejeita. Se leu 2 e depois encontrou símbolos diferentes de $\sqcup$, rejeita. Se leu símbolos 0 ou 1 e depois $\sqcup$ à direita do 2, rejeita. Se não encontrou 2, rejeita.
- Se leu 2 e depois $\sqcup$, aceita.

Máquina de Turing

$$L = \{w2w | w \in \{0,1\}^*\}$$



Máquina de Turing

Tente criar Máquinas de Turing para as linguagens abaixo:

- $L = \{ww | w \in \{0, 1\}^*\}$
- $L = \{ww^R | w \in \{0, 1\}^*\}$

Máquina de Turing Não Determinística

A principal diferença está na função de transição que passa a ser

- $\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R\})$

Variantes da Máquina de Turing

Máquina de Turing Não Determinística

A principal diferença está na função de transição que passa a ser

- $\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R\})$

- Se alguma escolha aceita, então a MT irá parar e aceitar
- Se todas as escolhas rejeitam e param, então a MT irá parar e rejeitar
- Se a computação continuar para sempre e nenhuma aceitação é encontrada, então a MT irá ficar em loop

Variantes da Máquina de Turing

Máquina de Turing Não Determinística

A principal diferença está na função de transição que passa a ser

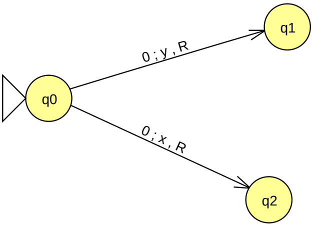
- $\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R\})$

- Se alguma escolha aceita, então a MT irá parar e aceitar
- Se todas as escolhas rejeitam e param, então a MT irá parar e rejeitar
- Se a computação continuar para sempre e nenhuma aceitação é encontrada, então a MT irá ficar em loop

Em uma MT não determinística a partir de uma configuração pode-se chegar a mais de uma configuração diferente em um único passo lendo o mesmo símbolo da fita

Variantes da Máquina de Turing

Máquina de Turing Não Determinística



Transição

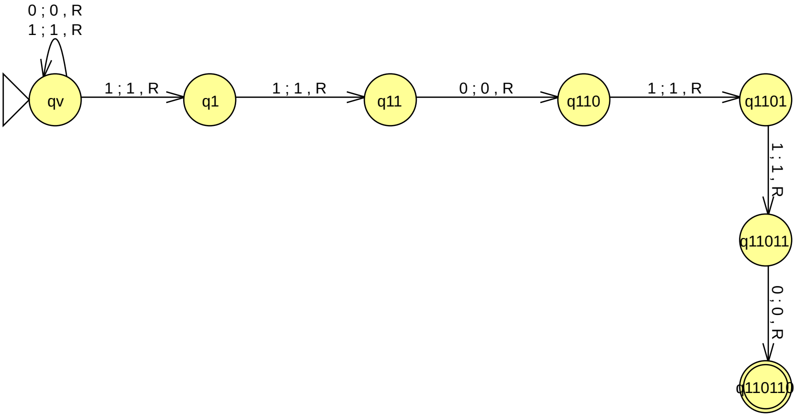
- $\delta(q0, 0) \rightarrow \{(q1, y, R), (q2, x, R)\}$

Árvore de configurações

- $\underline{\mathbf{q0}}01001 \rightarrow \{y\underline{\mathbf{q1}}1001, x\underline{\mathbf{q2}}1001\}$

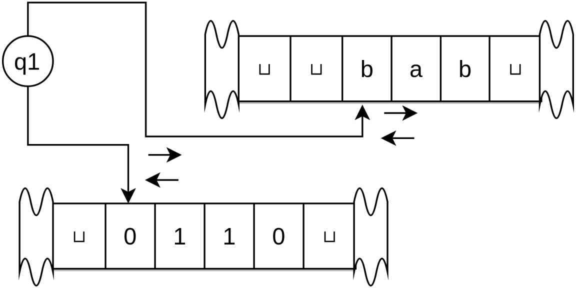
Variantes da Máquina de Turing

$L = \{w | w \in \{0,1\}^* \text{e } w \text{ contém } 110110\}$



Variantes da Máquina de Turing

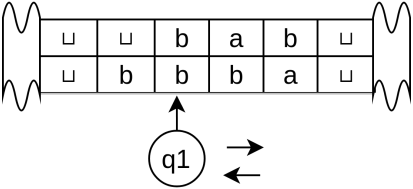
Máquina de Turing com Múltiplas Fitas



Função de transição

- $\delta : Q \times \Gamma^k \rightarrow P(Q \times \Gamma^k \times \{L, R\}^k)$
- $\delta(q, a, b) \rightarrow (p, x, y, L, R)$

Máquina de Turing com Múltiplas Trilhas



Função de transição

- $\delta : Q \times \Gamma^k \rightarrow P(Q \times \Gamma^k \times \{L, R\})$
- $\delta(q, a, b) \rightarrow (p, x, y, L)$

Variantes da Máquina de Turing

Há diversas outras variantes da Máquina de Turing, sendo todas elas com o mesmo poder computacional

- Máquina de Turing com Opção de Parada
  - Cabeçote pode mover-se para esquerda, direita ou ficar parado
- Máquina de Turing com Fita Semi-infinita
  - Limita os movimentos do cabeçote em um dos extremos da fita, ou seja, ou o lado direito ou o lado esquerdo é infinito
- Máquina de Turing com Restrições de Símbolos da Fita
  - Limita quais símbolos podem ser usados na fita
- Máquina de Turing Multidimensional
  - A fita é substituída por uma estrutura n-dimensional, infinita em todas as direções

Teorema: toda MT com múltiplas fitas possui uma MT com única fita equivalente.

Prova: por construção

Variantes da Máquina de Turing

Teorema: toda MT com múltiplas fitas possui uma MT com única fita equivalente.

Prova: por construção

- Convertemos uma MT com múltiplas fitas em uma MT com única fita

Variantes da Máquina de Turing

Teorema: toda MT com múltiplas fitas possui uma MT com única fita equivalente.

Prova: por construção

- Convertemos uma MT com múltiplas fitas em uma MT com única fita
- Seja M uma MT com k fitas

Variantes da Máquina de Turing

Teorema: toda MT com múltiplas fitas possui uma MT com única fita equivalente.

Prova: por construção

- Convertemos uma MT com múltiplas fitas em uma MT com única fita
- Seja M uma MT com k fitas
- Criamos uma MT S com única fita e que simula o efeito das k fitas de M

Teorema: toda MT com múltiplas fitas possui uma MT com única fita equivalente.

Prova: por construção

- Convertemos uma MT com múltiplas fitas em uma MT com única fita
- Seja M uma MT com k fitas
- Criamos uma MT S com única fita e que simula o efeito das k fitas de M
- A única fita de S armazena em uma única fita o conteúdo das k fitas de M

Variantes da Máquina de Turing

Teorema: toda MT com múltiplas fitas possui uma MT com única fita equivalente.

Prova: por construção

- Convertemos uma MT com múltiplas fitas em uma MT com única fita
- Seja M uma MT com k fitas
- Criamos uma MT S com única fita e que simula o efeito das k fitas de M
- A única fita de S armazena em uma única fita o conteúdo das k fitas de M
- Para isso, S usa um separador de conteúdo

Teorema: toda MT com múltiplas fitas possui uma MT com única fita equivalente.

Prova: por construção

- Convertemos uma MT com múltiplas fitas em uma MT com única fita
- Seja M uma MT com k fitas
- Criamos uma MT S com única fita e que simula o efeito das k fitas de M
- A única fita de S armazena em uma única fita o conteúdo das k fitas de M
- Para isso, S usa um separador de conteúdo
- S marca a posição do cabeçote de M em cada fita anotando um sinal sobre cada símbolo da fita

Variantes da Máquina de Turing

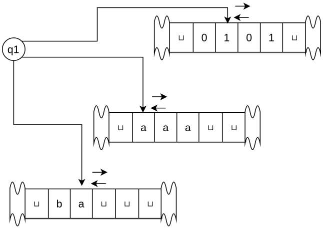
Teorema: toda MT com múltiplas fitas possui uma MT com única fita equivalente.

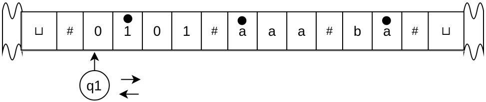
Prova: por construção

- Convertemos uma MT com múltiplas fitas em uma MT com única fita
- Seja M uma MT com k fitas
- Criamos uma MT S com única fita e que simula o efeito das k fitas de M
- A única fita de S armazena em uma única fita o conteúdo das k fitas de M
- Para isso, S usa um separador de conteúdo
- S marca a posição do cabeçote de M em cada fita anotando um sinal sobre cada símbolo da fita
- O símbolo com o sinal e o símbolo sem o sinal são exatamente o mesmo, sendo o sinal acima do símbolo apenas utilizado pela MT S saber em que posição M estaria apontando

Teorema: toda MT com múltiplas fitas possui uma MT com única fita equivalente.

Prova: por construção





Corolário: uma linguagem L é Turing-reconhecível se e somente se alguma MT com múltiplas fitas a reconhece.

Prova:

Corolário: uma linguagem L é Turing-reconhecível se e somente se alguma MT com múltiplas fitas a reconhece.

Prova:

1. Se uma linguagem L é Turing-reconhecível, então alguma MT com múltiplas fitas a reconhece

Corolário: uma linguagem L é Turing-reconhecível se e somente se alguma MT com múltiplas fitas a reconhece.

Prova:

1. Se uma linguagem L é Turing-reconhecível, então alguma MT com múltiplas fitas a reconhece
   - Se uma linguagem L é Turing-reconhecível, então existe uma MT M com uma única fita que a reconhece

Corolário: uma linguagem L é Turing-reconhecível se e somente se alguma MT com múltiplas fitas a reconhece.

Prova:

1. Se uma linguagem L é Turing-reconhecível, então alguma MT com múltiplas fitas a reconhece
   - Se uma linguagem L é Turing-reconhecível, então existe uma MT M com uma única fita que a reconhece
   - Note que uma MT com única fita é apenas um caso especial de uma MT com múltiplas fitas

Corolário: uma linguagem L é Turing-reconhecível se e somente se alguma MT com múltiplas fitas a reconhece.

Prova:

1. Se uma linguagem L é Turing-reconhecível, então alguma MT com múltiplas fitas a reconhece
   - Se uma linguagem L é Turing-reconhecível, então existe uma MT M com uma única fita que a reconhece
   - Note que uma MT com única fita é apenas um caso especial de uma MT com múltiplas fitas
2. Se uma linguagem L é reconhecida por alguma MT com múltiplas fitas, então L é Turing-reconhecível

Corolário: uma linguagem L é Turing-reconhecível se e somente se alguma MT com múltiplas fitas a reconhece.

Prova:

1. Se uma linguagem L é Turing-reconhecível, então alguma MT com múltiplas fitas a reconhece
   - Se uma linguagem L é Turing-reconhecível, então existe uma MT M com uma única fita que a reconhece
   - Note que uma MT com única fita é apenas um caso especial de uma MT com múltiplas fitas
2. Se uma linguagem L é reconhecida por alguma MT com múltiplas fitas, então L é Turing-reconhecível
   - A prova do teorema anterior é a prova para este item

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

- Podemos novamente simular qualquer MT não determinística com uma MT determinística

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

- Podemos novamente simular qualquer MT não determinística com uma MT determinística
- A ideia é tentar todas as escolhas da MT não determinística

Variantes da Máquina de Turing

Teorema: toda MT não determinística possui uma MT determinística equivalente.

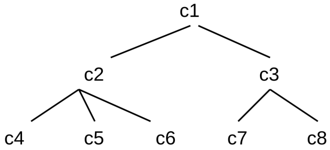
Prova: por construção

- Podemos novamente simular qualquer MT não determinística com uma MT determinística
- A ideia é tentar todas as escolhas da MT não determinística
- Se a MT determinística encontrar um estado de aceitação, ela irá parar e aceitar, caso contrário ela não irá terminar a menos que todas as escolhas levarem a um estado de rejeição

Variantes da Máquina de Turing

Teorema: toda MT não determinística possui uma MT determinística equivalente.

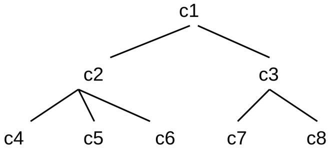
O nó raiz da árvore de configurações é a configuração inicial e a MT determinística busca uma configuração de aceitação



Como encontrar uma configuração de aceitação?

Teorema: toda MT não determinística possui uma MT determinística equivalente.

O nó raiz da árvore de configurações é a configuração inicial e a MT determinística busca uma configuração de aceitação



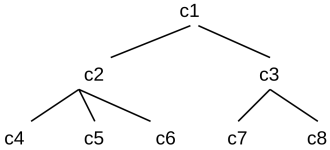
Como encontrar uma configuração de aceitação?

- Busca em profundidade (DFS) é uma má escolha! Poderia levar a uma busca infinitamente em determinado ramo da árvore

Variantes da Máquina de Turing

Teorema: toda MT não determinística possui uma MT determinística equivalente.

O nó raiz da árvore de configurações é a configuração inicial e a MT determinística busca uma configuração de aceitação



Como encontrar uma configuração de aceitação?

- Busca em profundidade (DFS) é uma má escolha! Poderia levar a uma busca infinitamente em determinado ramo da árvore
- Busca em largura (BFS) é melhor! Exploramos todos os ramos até uma mesma profundidade antes de seguir para um nível mais profundo

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

- Seja M uma MT não determinística

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

- Seja M uma MT não determinística
- Construímos uma MT determinística D que simule M

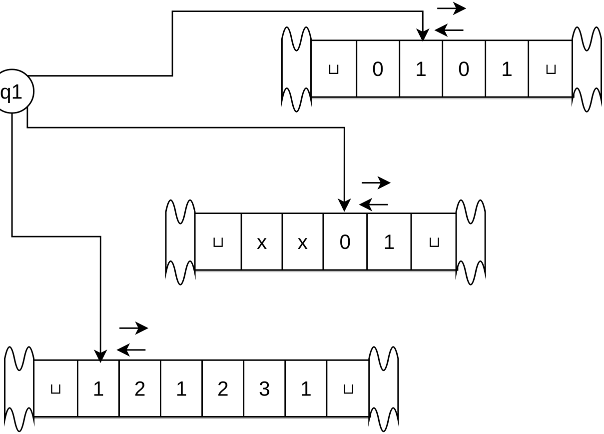
Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

- Seja M uma MT não determinística
- Construímos uma MT determinística D que simule M
- D possui 3 fitas
  - Fita 1 - entrada (nunca muda)
  - Fita 2 - simulação (cópia do que ocorre sobre a entrada numa das sequências de escolhas de M)
  - Fita 3 - mantém a sequência de escolhas a serem feitas considerando a árvore de configurações de M

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção



Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Representação da fita 3.

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Representação da fita 3.

- Note que cada nó da árvore tem no máximo b filhos, sendo b o número máximo de escolhas dada pela função de transição

Teorema: toda MT não determinística possui uma MT determinística equivalente.

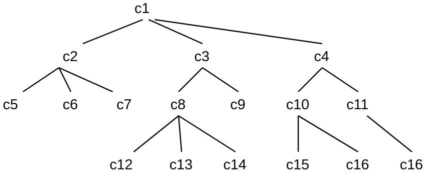
Prova: por construção. Representação da fita 3.

- Note que cada nó da árvore tem no máximo b filhos, sendo b o número máximo de escolhas dada pela função de transição
- Cada nó da árvore terá uma palavra equivalente sobre o alfabeto $\Sigma = \{1, 2, 3, ..., b\}$

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Representação da fita 3.

- Note que cada nó da árvore tem no máximo b filhos, sendo b o número máximo de escolhas dada pela função de transição
- Cada nó da árvore terá uma palavra equivalente sobre o alfabeto $\Sigma = \{1, 2, 3, ..., b\}$
- Esta palavra será como o endereço do nó

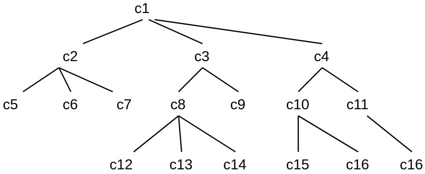


Variantes da Máquina de Turing

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Representação da fita 3.

- Note que cada nó da árvore tem no máximo b filhos, sendo b o número máximo de escolhas dada pela função de transição
- Cada nó da árvore terá uma palavra equivalente sobre o alfabeto $\Sigma = \{1, 2, 3, ..., b\}$
- Esta palavra será como o endereço do nó
  - Ex: 211 é o nó resultante da escolha 2 a partir da raiz, depois escolha 1 e depois escolha 1 novamente



Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

- Cada símbolo da palavra indica qual escolha deve ser feita na sequência da computação na máquina não determinística

Variantes da Máquina de Turing

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

- Cada símbolo da palavra indica qual escolha deve ser feita na sequência da computação na máquina não determinística
  - Algum símbolo pode não corresponder a nenhuma escolha válida, se existem poucas escolhas a partir de um dado nó

Teorema: toda MT não determinística possui uma MT determinística equivalente.

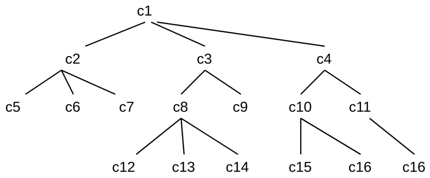
Prova: por construção

- Cada símbolo da palavra indica qual escolha deve ser feita na sequência da computação na máquina não determinística
  - Algum símbolo pode não corresponder a nenhuma escolha válida, se existem poucas escolhas a partir de um dado nó
- O conteúdo da fita 3 é atualizado com uma nova palavra na sequência

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

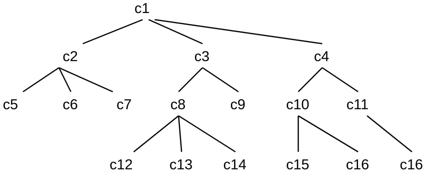
- Cada símbolo da palavra indica qual escolha deve ser feita na sequência da computação na máquina não determinística
    - Algum símbolo pode não corresponder a nenhuma escolha válida, se existem poucas escolhas a partir de um dado nó
- O conteúdo da fita 3 é atualizado com uma nova palavra na sequência
    - A palavra vazia indica a raiz da árvore de configurações



Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção

- Cada símbolo da palavra indica qual escolha deve ser feita na sequência da computação na máquina não determinística
  - Algum símbolo pode não corresponder a nenhuma escolha válida, se existem poucas escolhas a partir de um dado nó
- O conteúdo da fita 3 é atualizado com uma nova palavra na sequência
  - A palavra vazia indica a raiz da árvore de configurações
  - Na sequência viriam as palavras 1, 2, 3, 11, 12, 13, 21, 23, ...



Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Funcionamento da máquina D.

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Funcionamento da máquina D.

- Inicialmente, a fita 1 contém a palavra w e as fitas 2 e 3 estão vazias

Variantes da Máquina de Turing

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Funcionamento da máquina D.

- Inicialmente, a fita 1 contém a palavra w e as fitas 2 e 3 estão vazias
- Passo 1: Inicialize a fita 3 com ε

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Funcionamento da máquina D.

- Inicialmente, a fita 1 contém a palavra w e as fitas 2 e 3 estão vazias
- Passo 1: Inicialize a fita 3 com ε
- Passo 2: Copie o conteúdo da fita 1 para a fita 2

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Funcionamento da máquina D.

- Inicialmente, a fita 1 contém a palavra w e as fitas 2 e 3 estão vazias
- Passo 1: Inicialize a fita 3 com ε
- Passo 2: Copie o conteúdo da fita 1 para a fita 2
- Passo 3: Use a fita 2 para simular a MT não determinística M com a entrada w em uma das sequências de escolhas. Para isso consulte a fita 3 para decidir qual escolha fazer. Esta informação estará em cada símbolo da fita 3

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Funcionamento da máquina D.

- Inicialmente, a fita 1 contém a palavra w e as fitas 2 e 3 estão vazias
- Passo 1: Inicialize a fita 3 com ε
- Passo 2: Copie o conteúdo da fita 1 para a fita 2
- Passo 3: Use a fita 2 para simular a MT não determinística M com a entrada w em uma das sequências de escolhas. Para isso consulte a fita 3 para decidir qual escolha fazer. Esta informação estará em cada símbolo da fita 3
    - Se não houver mais símbolos a serem lidos na fita 3 ou a configuração for inválida (escolhas inválidas), aborte e vá para o passo 4

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Funcionamento da máquina D.

- Inicialmente, a fita 1 contém a palavra w e as fitas 2 e 3 estão vazias
- Passo 1: Inicialize a fita 3 com ε
- Passo 2: Copie o conteúdo da fita 1 para a fita 2
- Passo 3: Use a fita 2 para simular a MT não determinística M com a entrada w em uma das sequências de escolhas. Para isso consulte a fita 3 para decidir qual escolha fazer. Esta informação estará em cada símbolo da fita 3
  - Se não houver mais símbolos a serem lidos na fita 3 ou a configuração for inválida (escolhas inválidas), aborte e vá para o passo 4
  - Se uma configuração de rejeição é alcançada, vá para o passo 4

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Funcionamento da máquina D.

- Inicialmente, a fita 1 contém a palavra w e as fitas 2 e 3 estão vazias
- Passo 1: Inicialize a fita 3 com ε
- Passo 2: Copie o conteúdo da fita 1 para a fita 2
- Passo 3: Use a fita 2 para simular a MT não determinística M com a entrada w em uma das sequências de escolhas. Para isso consulte a fita 3 para decidir qual escolha fazer. Esta informação estará em cada símbolo da fita 3
  - Se não houver mais símbolos a serem lidos na fita 3 ou a configuração for inválida (escolhas inválidas), aborte e vá para o passo 4
  - Se uma configuração de rejeição é alcançada, vá para o passo 4
  - Se uma configuração de aceitação é alcançada, aceite a entrada

Teorema: toda MT não determinística possui uma MT determinística equivalente.

Prova: por construção. Funcionamento da máquina D.

- Passo 4: Substitua a palavra da fita 3 pela próxima palavra na sequência de escolhas a serem feitas pela MT não determinística M. Depois, vá para o passo 2 para simular a MT não determinística M com a nova sequência de escolhas.

Corolário: uma linguagem é Turing-reconhecível se e somente se alguma MT não determinística a reconhece.

Prova: por construção

Corolário: uma linguagem é Turing-reconhecível se e somente se alguma MT não determinística a reconhece.

Prova: por construção

1. Se uma linguagem é Turing-reconhecível então alguma MT não determinística a reconhece

Corolário: uma linguagem é Turing-reconhecível se e somente se alguma MT não determinística a reconhece.

Prova: por construção

1. Se uma linguagem é Turing-reconhecível então alguma MT não determinística a reconhece
    - Prova: Se uma linguagem L é Turing-reconhecível, existe uma MT determinística que a reconhece, e uma MT determinística é apenas um caso especial de MT não determinística

Corolário: uma linguagem é Turing-reconhecível se e somente se alguma MT não determinística a reconhece.

Prova: por construção

1. Se uma linguagem é Turing-reconhecível então alguma MT não determinística a reconhece
   - Prova: Se uma linguagem L é Turing-reconhecível, existe uma MT determinística que a reconhece, e uma MT determinística é apenas um caso especial de MT não determinística
2. Se uma linguagem é reconhecida por uma MT não determinística, então a linguagem é Turing-reconhecível

Corolário: uma linguagem é Turing-reconhecível se e somente se alguma MT não determinística a reconhece.

Prova: por construção

1. Se uma linguagem é Turing-reconhecível então alguma MT não determinística a reconhece
   - Prova: Se uma linguagem L é Turing-reconhecível, existe uma MT determinística que a reconhece, e uma MT determinística é apenas um caso especial de MT não determinística
2. Se uma linguagem é reconhecida por uma MT não determinística, então a linguagem é Turing-reconhecível
   - Prova: É a prova do teorema anterior

Note que se a MT não determinística sempre para, para todas as entradas, então a MT determinística equivalente a ela também sempre para. Assim, dizemos que a MT não determinística é uma MT decisora.

- **Corolário**: Uma linguagem é decidível se e somente se alguma MT não determinística a decide

Enumeradores

Uma linguagem é recursivamente enumerável se ela é Turing-reconhecível

Um **enumerador** é uma Máquina de Turing com uma "impressora"

- A fita é vazia inicialmente (não tem entrada)
- Palavras pertencentes à linguagem são geradas e impressas
  - O enumerador não aceita/rejeita palavras
- Um enumerador pode parar ou não (loop)
  - Ele não para quando a linguagem é infinita
- As palavras podem ser impressas em qualquer ordem e também com repetições

Enumeradores

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 1: se há um enumerador E que enumerada uma linguagem A, então A é Turing-reconhecível

- Seja E o enumerador para a linguagem A
- Podemos construir uma MT M que reconhece A

Enumeradores

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 1: se há um enumerador E que enumerada uma linguagem A, então A é Turing-reconhecível

- Seja E o enumerador para a linguagem A
- Podemos construir uma MT M que reconhece A

$M =$ com a entrada w faz

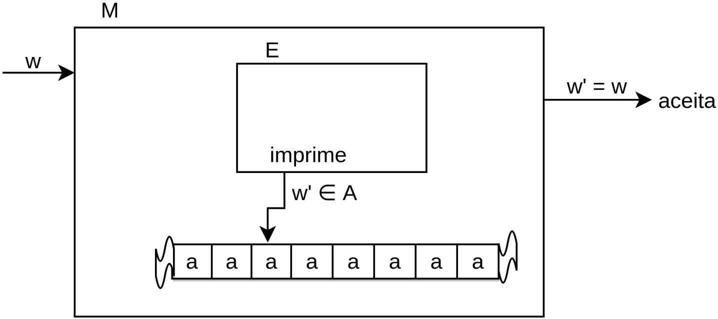
- Roda o enumerador E
- Toda vez que E imprimir uma palavra, compare com w
- Se a palavra impressa é igual a w, então aceita

Enumeradores

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 1: se há um enumerador E que enumerada uma linguagem A, então A é Turing-reconhecível

- Seja E o enumerador para a linguagem A
- Podemos construir uma MT M que reconhece A



Enumeradores

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 2: se A é Turing-reconhecível, então há um enumerador E que a enumera

- Seja M uma MT que reconhece A
- Podemos construir um enumerador E que a enumera

Enumeradores

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 2: se A é Turing-reconhecível, então há um enumerador E que a enumera

- Seja M uma MT que reconhece A
- Podemos construir um enumerador E que a enumera

Ideia: rodar a MT em todas as possíveis entradas e toda vez que M aceitar uma palavra, imprime a palavra

Enumeradores

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 2: se A é Turing-reconhecível, então há um enumerador E que a enumera

- Seja M uma MT que reconhece A
- Podemos construir um enumerador E que a enumera

Ideia: rodar a MT em todas as possíveis entradas e toda vez que M aceitar uma palavra, imprime a palavra

Problema: M pode não parar para alguma entrada

Enumeradores

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 2: se A é Turing-reconhecível, então há um enumerador E que a enumera

- Seja M uma MT que reconhece A
- Podemos construir um enumerador E que a enumera

Ideia: rodar a MT em todas as possíveis entradas e toda vez que M aceitar uma palavra, imprime a palavra

Problema: M pode não parar para alguma entrada

Solução: rodar a MT em todas as entradas, mas em paralelo

Enumeradores

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 2: se A é Turing-reconhecível, então há um enumerador E que a enumera

- Seja M uma MT que reconhece A
- Podemos construir um enumerador E que a enumera

Ideia: rodar a MT em todas as possíveis entradas e toda vez que M aceitar uma palavra, imprime a palavra

Problema: M pode não parar para alguma entrada

Solução: rodar a MT em todas as entradas, mas em paralelo

- Executar um pouco de cada palavra (intercaladamente)

Enumeradores

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 2: se A é Turing-reconhecível, então há um enumerador E que a enumera

- Seja M uma MT que reconhece A
- Podemos construir um enumerador E que a enumera

Ideia: rodar a MT em todas as possíveis entradas e toda vez que M aceitar uma palavra, imprime a palavra

Problema: M pode não parar para alguma entrada

Solução: rodar a MT em todas as entradas, mas em paralelo

- Executar um pouco de cada palavra (intercaladamente)
- Executamos i passos para cada palavra até a palavra i, para $i = 1..\infty$

Enumeradores

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 2: se A é Turing-reconhecível, então há um enumerador E que a enumera

- Seja M uma MT que reconhece A
- Podemos construir um enumerador E que a enumera

Ideia: rodar a MT em todas as possíveis entradas e toda vez que M aceitar uma palavra, imprime a palavra

Problema: M pode não parar para alguma entrada

Solução: rodar a MT em todas as entradas, mas em paralelo

- Executar um pouco de cada palavra (intercaladamente)
- Executamos i passos para cada palavra até a palavra i, para $i = 1..\infty$
- Se M aceitar, imprime a palavra

Enumeradores

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 2: se A é Turing-reconhecível, então há um enumerador E que a enumera

- Seja M uma MT que reconhece A
- Podemos construir um enumerador E que a enumera

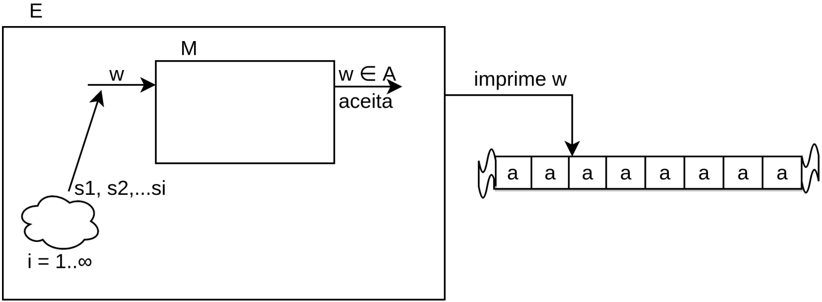
E = ignora a entrada

- Repita para $i = 1..\infty$
  - Rode M por i passos para cada palavra $s1, s2, ..., si$
  - Se M aceitar alguma palavra, imprima ela

Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 2: se A é Turing-reconhecível, então há um enumerador E que a enumera

- Seja M uma MT que reconhece A
- Podemos construir um enumerador E que a enumera



Teorema: Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera

Prova: Parte 2: se A é Turing-reconhecível, então há um enumerador E que a enumera

- Seja M uma MT que reconhece A
- Podemos construir um enumerador E que a enumera

Funcionamento do enumerador E

- Para $i = 1$, apenas verifica a palavra $s1$ com 1 passo. Aceita e imprime $s1$, se for o caso.
- Para $i = 2$, apenas verifica as palavras $s1$ e $s2$ com 2 passos cada. Aceita e imprime $s1$, $s2$ se for o caso.
- Para $i = 3$, apenas verifica as palavras $s1$, $s2$, $s3$ com 3 passos cada. Aceita e imprime $s1$, $s2$, $s3$ se for o caso.
- ...

Algoritmo

Em 1900 , David Hilbert identificou 23 problemas matemáticos para serem resolvidos no próximo século

Um deles era sobre a raiz de polinômios. O problema era verificar se um dado polinômio possui uma raiz inteira por meio da apresentação de um algoritmo.

- Polinômio é uma soma de termos, onde cada termo é o produto de certas variáveis por uma constante, chamada coeficiente. Ex: $6x^3yz^2$
- A raiz do polinômio é uma atribuição de valores para as variáveis tal que o valor do polinômio é 0. A raiz é dita inteira se todos os valores são inteiros

Hilbert não usou exatamente o termo "algoritmo", mas "um processo que pode ser determinado por um número finito de operações"

Algoritmo

A definição de algoritmo surge em 1936, com os trabalhos de Church e Turing, com λ-calculus e a Máquina de Turing. Ambos são equivalentes e definem a noção de algoritmo

Em 1970, foi provado que não há um algoritmo para verificar se um polinômio possui uma raiz inteira

O problema do polinômio de raiz inteira é um problema de decisão, onde deve-se dar um resposta sim ou não

Qualquer problema de decisão pode ser transformado em uma linguagem

- Tem-se assim, problemas decidíveis e problemas reconhecíveis

Algoritmo

O problema do polinômio de raiz inteira pode ser mapeado para uma linguagem da seguinte forma

- $D = \{p|p \text{ é um polinômio com uma raiz inteira }\}$

O problema de Hilbert pede se D é decidível. Sabe-se que não é! Mas ainda assim ele é reconhecível.

Prova: por construção

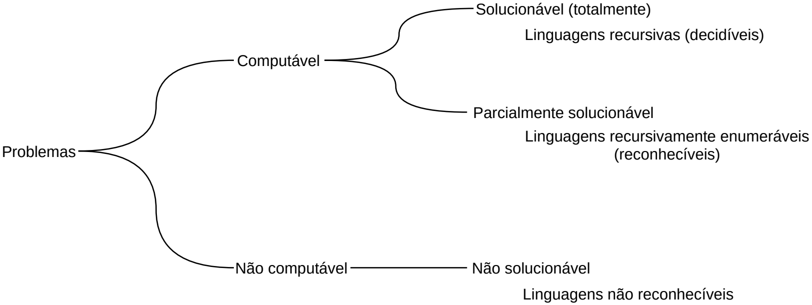
- Vamos construir uma MT M que reconheça D

$M =$ com a entrada p faz

- Para cada variável em p atribua valores na forma 0, -1, 1, 2, -2, ... de maneira intercalada
- Se algum conjunto de valores for a raiz do polinômio, aceite p

Note que, se houver uma raiz inteira, a MT M irá aceitar e parar, caso contrário ela irá ficar em loop

Algoritmo


Problemas
Computável
Solucionável (totalmente)
Linguagens recursivas (decidíveis)
Parcialmente solucionável
Linguagens recursivamente enumeráveis (reconhecíveis)
Não computável
Não solucionável
Linguagens não reconhecíveis

Algoritmo

Se um problema pode ser representado por uma linguagem recursiva, então ele é decidível, caso contrário é indecidível

- Pode ser ainda reconhecível ou não

Um problema é decidível se sua solução é encontrada num tempo finito

- Decidibilidade - se tempo é finito
- Complexidade - quanto tempo

Um problema é decidível se existe um algoritmo que resolva o problema, para qualquer entrada, informando uma aceitação (sim) ou rejeição (não)

Algoritmo

Problema: verificar se um dado grafo G é conexo

Algoritmo

Problema: verificar se um dado grafo G é conexo

- Vamos mostrar que este problema é decidível

Algoritmo

Problema: verificar se um dado grafo G é conexo

- Vamos mostrar que este problema é decidível

Linguagem: $L_G = \{< G > \mid < G >$ é um grafo conexo $\}$

Algoritmo

Problema: verificar se um dado grafo G é conexo

- Vamos mostrar que este problema é decidível

Linguagem: $L_G = \{< G > | < G >$ é um grafo conexo $\}$

Prova: por construção

- Vamos construir uma MT M que decide L_G

$M =$ com a entrada $< G >$ faz

Algoritmo

Problema: verificar se um dado grafo G é conexo

- Vamos mostrar que este problema é decidível

Linguagem: $L_G = \{< G > \mid < G >$ é um grafo conexo $\}$

Prova: por construção

- Vamos construir uma MT M que decide L_G

$M =$ com a entrada $< G >$ faz

- Passo 1: Selecione e marque o primeiro vértice

Algoritmo

Problema: verificar se um dado grafo G é conexo

- Vamos mostrar que este problema é decidível

Linguagem: $L_G = \{< G > \mid < G >$ é um grafo conexo $\}$

Prova: por construção

- Vamos construir uma MT M que decide L_G

$M =$ com a entrada $< G >$ faz

- Passo 1: Selecione e marque o primeiro vértice
- Passo 2: Repita

Algoritmo

Problema: verificar se um dado grafo G é conexo

- Vamos mostrar que este problema é decidível

Linguagem: $L_G = \{< G > \mid < G >$ é um grafo conexo $\}$

Prova: por construção

- Vamos construir uma MT M que decide L_G

$M =$ com a entrada $< G >$ faz

- Passo 1: Selecione e marque o primeiro vértice
- Passo 2: Repita
  - Passo 2.1: Para cada vértice em G, marque-o se há alguma aresta que o conecte com outro vértice já marcado

Algoritmo

Problema: verificar se um dado grafo G é conexo

- Vamos mostrar que este problema é decidível

Linguagem: $L_G = \{< G > \mid < G >$ é um grafo conexo $\}$

Prova: por construção

- Vamos construir uma MT M que decide L_G

$M =$ com a entrada $< G >$ faz

- Passo 1: Selecione e marque o primeiro vértice
- Passo 2: Repita
  - Passo 2.1: Para cada vértice em G, marque-o se há alguma aresta que o conecte com outro vértice já marcado
  - Passo 2.2: Se nenhum vértice novo for marcado no passo 2.1, então vá para o passo 3

Algoritmo

Problema: verificar se um dado grafo G é conexo

- Vamos mostrar que este problema é decidível

Linguagem: $L_G = \{< G > \mid < G > \text{ é um grafo conexo } \}$

Prova: por construção

- Vamos construir uma MT M que decide L_G

$M =$ com a entrada $< G >$ faz

- Passo 1: Selecione e marque o primeiro vértice
- Passo 2: Repita
  - Passo 2.1: Para cada vértice em G, marque-o se há alguma aresta que o conecte com outro vértice já marcado
  - Passo 2.2: Se nenhum vértice novo for marcado no passo 2.1, então vá para o passo 3
- Passo 3: Verifique se todos os vértices estão marcados

Algoritmo

Problema: verificar se um dado grafo G é conexo

- Vamos mostrar que este problema é decidível

Linguagem: $L_G = \{< G > \mid < G >$ é um grafo conexo $\}$

Prova: por construção

- Vamos construir uma MT M que decide L_G

$M =$ com a entrada $< G >$ faz

- Passo 1: Selecione e marque o primeiro vértice
- Passo 2: Repita
  - Passo 2.1: Para cada vértice em G, marque-o se há alguma aresta que o conecte com outro vértice já marcado
  - Passo 2.2: Se nenhum vértice novo for marcado no passo 2.1, então vá para o passo 3
- Passo 3: Verifique se todos os vértices estão marcados
  - Se sim, aceite $< G >$

Algoritmo

Problema: verificar se um dado grafo G é conexo

- Vamos mostrar que este problema é decidível

Linguagem: $L_G = \{< G > \mid < G >$ é um grafo conexo $\}$

Prova: por construção

- Vamos construir uma MT M que decide L_G

$M =$ com a entrada $< G >$ faz

- Passo 1: Selecione e marque o primeiro vértice
- Passo 2: Repita
  - Passo 2.1: Para cada vértice em G, marque-o se há alguma aresta que o conecte com outro vértice já marcado
  - Passo 2.2: Se nenhum vértice novo for marcado no passo 2.1, então vá para o passo 3
- Passo 3: Verifique se todos os vértices estão marcados
  - Se sim, aceite $< G >$
  - Senão, rejeite $< G >$

Conclusão

- Máquina de Turing
- Variantes da máquina de Turing
- Definição de algoritmo

Material de apoio

- Livro Sipser, Capítulo 3
- Livro Hopcroft, Capítulos 8