

# Máquinas de Turing

## Computabilidade

**Prof. Edson Alves**

Faculdade UnB Gama

2020

# Sumário

1. Computabilidade e Tese de Turing
2. Incomputabilidade

## Especificação para uma função de $k$ argumentos

- (a) Os argumentos  $m_1, m_2, \dots, m_k$  são apresentados em notação monádica por  $k$  blocos com  $m_i$  traços cada; os blocos são separados por um único espaço em branco e a fita, de resto, está em branco
- (b) O computador começa examinando o traço mais à esquerda do bloco  $m_1$ ; (a) e (b) caracterizam a **configuração inicial** da máquina
- (c) Se  $f(m_1, m_2, \dots, m_k) = n$ , a máquina para no traço mais à esquerda de um bloco contendo  $n$  traços; de resto, a fita está em branco. Esta é a **configuração (posição) final padrão**
- (d) Se a função  $f$  não está definida para os argumentos dados, ou a máquina não irá parar, ou irá parar em uma configuração final que não é a padrão

## Exemplo de máquina que segue a especificação

- ▶ Considere a máquina

$$q_1 S_1 S_1 q_2,$$

que representa uma função de um único argumento  $m$

- ▶ Ela examina o primeiro traço do bloco de  $m$  traços, escreve 1 (o que equivale a não fazer nada) e segue para o estado 2
- ▶ A máquina para no estado 2: neste momento, ela está sobre o quadrado mais à esquerda de um bloco de  $m$  traços; de resto, a fita está vazia
- ▶ Logo a máquina para na configuração final padrão, e  $f(m) = m$  para todo inteiro positivo  $m$
- ▶ Assim,  $f(x) = \text{id}(x)$

# Computabilidade por Máquina de Turing

## Definição

Uma função numérica de  $k$  argumentos é **computável por Máquina de Turing**, ou **Turing computável** se existe uma máquina que atenda as especificações apresentadas e que compute  $f(x)$  para todos os elementos  $x$  no domínio de  $f$ .

# A Tese de Turing

## Tese de Turing

Toda função efetivamente computável é Turing computável.

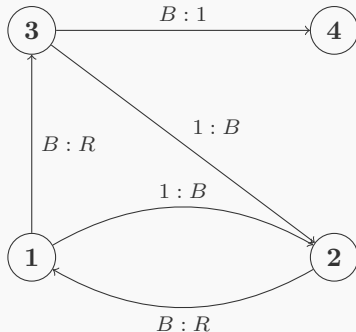
**Observações:** naturalmente, toda função Turing computável é efetivamente computável. Note também que, uma vez que a noção de computabilidade não é rigorosamente definida, não é possível demonstrar formalmente a Tese de Turing, de modo que ela é, de fato, uma conjectura.

# Existência de funções incomputáveis

- ▶ É possível demonstrar que o conjunto de todas as funções de inteiros positivos em inteiros positivos não é enumerável
- ▶ Por outro lado, o conjunto de todas as máquinas de Turing é enumerável: cada máquina pode ser especificado por uma sequência de quádruplas, que equivale a uma cadeia finita de símbolos de um alfabeto finito, e o conjunto de tais cadeias é enumerável
- ▶ Deste modo, existem funções que não são Turing computáveis
- ▶ Especificar exemplos de tais funções, contudo, não é tarefa trivial

# Enumeração das máquinas de Turing

Para ilustrar o processo de enumeração das máquinas de Turing, considere a máquina abaixo, que atribui o valor 1 para qualquer  $k$ -upla:





## Enumeração das máquinas de Turing

- ▶ A máquina apresentada pode ser representada pela seguinte lista de quádruplas

$$q_1 S_0 R q_3, \quad q_1 S_1 S_0 q_2, \quad q_2 S_0 R q_1, \quad q_3 S_0 S_1 q_4, \quad q_3 S_1 S_0 q_2$$

- ▶ Tal especificação, embora correta, não permite a enumeração de todas as máquinas de Turing
- ▶ Para tal fim, da mesma forma que foi feito para as máquina de Turing, é preciso especificar precisamente a representação de uma máquina por meio de uma lista de quádruplas

## Especificação para a lista de quádruplas

- (a) O estado de menor número (1) é o **estado inicial**
- (b) O estado de maior número ( $n + 1$ ) será o **estado de parada**: para este estado, não há instruções nem quádruplas
- (c) Para cada estado, exceto para o estado de parada, existe uma quádrupla iniciando com  $q_i S_j$ , com  $i = 1, 2, \dots, n, j = 0, 1$
- (d) De acordo com (c), se as quádruplas forem listadas em ordem crescente de  $i$  e de  $j$ , os dois primeiros símbolos de cada quádrupla são previsíveis, e poderão ser omitidos
- (e) Os estados  $q_i$  devem ser representados pelo inteiro  $i$ , os símbolos  $S_j$  por  $j + 1$  (para evitar o zero) e as instruções  $L$  e  $R$  pelos inteiros 3 e 4, respectivamente

**Observação:** uma máquina de Turing descrita por uma lista de quádruplas que atende a especificação acima corresponde a um inteiro positivo, de acordo com a codificação baseada no Teorema Fundamental da Aritmética.

## Exemplo de codificação de uma máquina de Turing

- ▶ A lista de quádruplas da máquina que retorna zero para qualquer  $k$ -upla dada abaixo

$$q_1 S_0 R q_3, \quad q_1 S_1 S_0 q_2, \quad q_2 S_0 R q_1, \quad q_3 S_0 S_1 q_4, \quad q_3 S_1 S_0 q_2$$

não atende à especificação

- ▶ Observe que as duas primeiras especificações são atendidas: (**1**) é o estado final e (**n + 1 = 4**) é o estado final
- ▶ Contudo, não existe uma quádrupla iniciando com  $q_2 S_1$ , violando (**c**): para corrigir isto, basta adicionar uma nova quádrupla, que mantém o símbolo e vai para o estado final:

$$q_1 S_0 R q_3, \quad q_1 S_1 S_0 q_2, \quad q_2 S_0 R q_1, \quad q_2 S_1 S_1 q_4, \quad q_3 S_0 S_1 q_4, \quad q_3 S_1 S_0 q_2$$

## Exemplo de codificação de uma máquina de Turing

- ▶ Aplicando o critério **(d)**, a lista se reduz à

$$Rq_3, S_0q_2, Rq_1, S_1q_4, S_1q_4, S_0q_2$$

- ▶ Usando o critério **(e)** obtêm-se:

$$4, 3, 1, 2, 4, 1, 2, 4, 2, 4, 1, 2$$

- ▶ Codificando esta máquina o resultado é o inteiro positivo:

$$2^4 \cdot 3^3 \cdot 5 \cdot 7^2 \cdot 11^4 \cdot 13 \cdot 17^2 \cdot 19^4 \cdot 23^2 \cdot 29^4 \cdot 31 \cdot 37^2$$

# Enumerabilidade das máquinas de Turing

- ▶ A codificação acima permite enumerar as máquinas de Turing  $M_1, M_2, M_3, \dots$
- ▶ Observe que nem todo inteiro positivo corresponde à uma máquina de Turing: isto depende de sua decomposição em fatores primos
- ▶ Além disso, nem toda sequência  $a_k$  formada pelos números de 1 a 4 corresponde a uma máquina de Turing
- ▶ Para que tal sequência represente uma máquina de Turing, ela precisa atender três critérios:
  - i.  $|a_k| = 4n$ , para algum inteiro positivo  $n$
  - ii.  $a_i \in [1, 4]$ , se  $i$  é ímpar (uma das quatro instruções possíveis)
  - iii.  $a_j \in [1, n + 1]$ , se  $j$  é par (um dos  $n + 1$  estados possíveis)
- ▶ Assim, a codificação apresentada é uma função parcial dos inteiros positivos que enumera as máquinas de Turing

## Exemplos de máquinas de Turing

- ▶ Considere a máquina

$$(1, 1, 1, 1) = 2 \cdot 3 \cdot 5 \cdot 7 = 210$$

- ▶ O fluxograma correspondente seria



- ▶ A partir da configuração inicial, esta máquina apaga o traço que está no bloco e retorna para o estado (1)
- ▶ A partir daí ela não faz mais nada, jamais atingindo a configuração final (2)

## Exemplos de máquinas de Turing

- ▶ Considere a máquina

$$(2, 1, 1, 1) = 2^2 \cdot 3 \cdot 5 \cdot 7 = 420$$

- ▶ O fluxograma correspondente seria



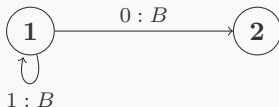
- ▶ A partir da configuração inicial, esta máquina apaga o traço que está no bloco e retorna para o estado (1)
- ▶ Em seguida, ele reescreve o traço e permanece em (1), reiniciando o ciclo, sem jamais parar

# Exemplos de máquinas de Turing

- ▶ Seja a máquina

$$(1, 2, 1, 1) = 2 \cdot 3^2 \cdot 5 \cdot 7 = 630$$

- ▶ O fluxograma correspondente seria



- ▶ A partir da configuração inicial, esta máquina apaga o traço que está no bloco e, ao o reexaminar, segue para o estado (2)
- ▶ Como (2) é o estado final, e o quadrado está em branco, a máquina parou em um configuração final que não é a padrão
- ▶ As três máquinas exemplificadas correspondem aos menores inteiros positivos associados à uma máquina de Turing (isto é,  $M_1, M_2, M_3$ ), e todas elas computam a função vazia



# Função diagonal

## Definição

Seja  $f_i$  a função computada pela  $i$ -ésima máquina de Turing. A **função diagonal**  $d$  é definida por

$$d(n) = \begin{cases} 2, & \text{se } f_n(n) \text{ é definida e } f_n(n) = 1, \\ 1, & \text{caso contrário} \end{cases}$$

# Incomputabilidade da função diagonal

## Teorema

A função diagonal não é Turing computável.

## Demonstração

Suponha, por contradição, que a função diagonal seja Turing computável. Assim, para algum  $m$  positivo,  $d(n) = f_m(n)$ , para qualquer  $n$  positivo.

No caso em que  $n = m$ , porém, surge uma contradição: se  $f_m(n) = 1$  então, por definição,  $d(m) = 2$ ; caso contrário, se  $f_m(n) \neq 1$  ou se  $f_m(n)$  não for definida,  $d(m) = 1$ . Em todos os casos,  $d(m) \neq f_m(m)$ , o que contradiz a hipótese de  $d$  ser Turing computável.

Portanto, a função diagonal  $d$  não é Turing computável.

## Problema da parada

- ▶ Se a Tese de Turing estiver correta, a função diagonal não seria efetivamente computável
- ▶ Porém, a princípio, parece ser possível computar a função  $d$  para qualquer argumento  $n$
- ▶ Por exemplo, para as três primeiras máquinas de Turing, apresentadas anteriormente,  $f_1, f_2$  e  $f_3$  são iguais a função vazia, de modo que  $d(1) = d(2) = d(3) = 1$
- ▶ Se  $f_n(m)$  está definida para  $m$ , o valor de  $d(m)$  será 1, se  $f_n(m) = 1$ , ou  $d(m) = 2$ , se  $f_n(m) \neq 1$
- ▶ Se  $f_n(m)$  parar em uma configuração final diferente da padrão,  $d_n(m) = 1$
- ▶ A situação difícil de identificar e computar acontece quando  $f_n(m)$  não para
- ▶ Não existe, até o presente momento, um procedimento mecânico uniforme que permita, para qualquer Máquina  $M_i$ , decidir se a função  $f_i$  para ou não para o argumento  $m$
- ▶ Este é o problema da parada

# Função de parada

## Definição

A **função de parada** de dois argumentos  $h(m, n)$  é definida por

$$h(m, n) = \begin{cases} 1, & \text{se } f_m(n) \text{ para em alguma configuração,} \\ 2, & \text{caso contrário} \end{cases}$$

## Teorema

A função de parada não é Turing computável.

# Referências

1. **BOOLOS**, George S.; **BURGESS**, John P.; **JEFFREY**, Richard C. *Computabilidade e Lógica*, Editora Unesp, 2012.