

# Lógica e Computabilidade 2025-1

Hugo Nobrega

## Lista de Exercícios 3

As entregas podem ser feitas em duplas, mas lembre-se que não poderá haver repetição de duplas em listas diferentes!

Entregue todas as questões marcadas com \* até

**14 de julho às 20:00.**

Em <https://www.hugonobrega.com/logica/turing.py> você encontra um simulador de Máquinas de Turing em Python. Nesse programa, representamos símbolos por strings de comprimento 1, palavras por strings de comprimento qualquer e estados por strings não-vazias. Use-o chamando a função `executa(entrada, (estado_inicial, estados_terminais, transição))`, onde

- `entrada` é a palavra de entrada
- `estado_inicial` é o estado inicial
- `estados_terminais` é o conjunto de estados terminais
- `transição` é um dicionário representando a função de transição da máquina; a chave

(`símbolo_lido`, `estado_atual`)

estar associada ao valor

(`símbolo_escrito`, `novo_estado`, `movimento`)

significa que ao ler `símbolo_lido` no estado `estado_atual`, a máquina deve escrever `símbolo_escrito`, passar ao estado `novo_estado` e mover-se de acordo com `movimento`. As strings ‘←’ e ‘e’ indicam movimento para a esquerda; as strings ‘→’ e ‘d’ indicam movimento para a direita; qualquer outra string indica que a máquina deve ficar parada.

Você **pode**, mas **não é obrigado a** usar esse programa sempre que desejar nas questões dessa lista. Em questões onde você deve escrever uma Máquina de Turing, é aceitável (e desejável, mas não obrigatório) que você submeta o código em Python que represente a máquina desejada.

**Questão 1.** Em cada item abaixo, escreva uma Máquina de Turing que aceite a linguagem dada.

a.  $\{w \in \{x, y\}^* ; \text{as quantidades de } x\text{'s e de } y\text{'s em } w \text{ são iguais}\}$ .

b.  $\{w \in \{x, y, z\}^* ; \exists n \in \mathbb{N} . w = x^n y^n z^n\}$

\* c.  $\{w \in \Sigma^* ; w \text{ é um palíndromo}\}$  (um *palíndromo* é uma palavra que é igual ao seu próprio reverso, i.e., é uma palavra que é igual lida de frente pra trás ou de trás pra frente, como “socorram-me, subi no ônibus em Marrocos” se você ignorar acentos, pontuação, espaços e se as letras são maiúsculas/minúsculas).

\***Questão 2.** Considere a Máquina de Turing que tem estado inicial  $q_0$ , estado terminal  $q_f$ , e a seguinte função de transição:

ao ler	no estado	escreva	vá para o estado	ande para
0	$q_0$	X	$q_0$	$\rightarrow$
1	$q_0$	X	$q_1$	$\rightarrow$
0	$q_1$	X	$q_2$	$\rightarrow$
1	$q_1$	X	$q_1$	$\rightarrow$
0	$q_2$	X	$q_2$	$\rightarrow$
1	$q_2$	X	$q_3$	$\rightarrow$
B	$q_2$	X	$q_f$	$\rightarrow$
0	$q_3$	X	$q_0$	$\rightarrow$
1	$q_3$	X	$q_3$	$\rightarrow$
B	$q_3$	X	$q_f$	$\rightarrow$

Descreva de maneira clara e sucinta a linguagem  $L \subseteq \{0, 1\}^*$  que essa Máquina de Turing aceita.

**Questão 3.** Considere a seguinte noção levemente alterada de Máquina de Turing:

**Definição.** Um *enumerador* é uma Máquina de Turing com 2 fitas, uma chamada *fita de trabalho* e a outra chamada *fita de enumeração*, e um estado especial chamado *estado de enumeração* (que não é terminal).

A *execução de enumeração* de um enumerador  $E$  começa com ambas as fitas vazias. Durante essa execução, sempre que  $E$  entra no estado de enumeração, se as células não-vazias da fita de enumeração formam um bloco contíguo, consideramos que a palavra formada por essas células foi *enumerada*.

Note que a execução de enumeração de  $E$  pode enumerar diversas, talvez infinitas, palavras (a execução pode, ou não, terminar).

Prove que uma linguagem  $L \subseteq \{0, 1\}^*$  é recursivamente enumerável (i.e., aceita por alguma Máquina de Turing comum) se, e somente se, existe um enumerador  $E$  tal que  $L$  é exatamente o conjunto das palavras enumeradas por  $E$ .

**Questão 4.** Diga se cada uma das linguagens abaixo é decidível (i.e., reconhecida por alguma Máquina de Turing que sempre para) ou não, e prove sua resposta.

\* **a.** PASSOS :=  $\{c \in \{0, 1\}^* \mid c \text{ codifica alguma máquina } M, \text{ alguma entrada } x \text{ para } M \text{ e algum } n \in \mathbb{N}, \text{ tais que } M \text{ para em no máximo } n \text{ passos quando executada com entrada } x\}$

\* **b.** FINITO :=  $\{c \in \{0, 1\}^* \mid c \text{ codifica máquina } M \text{ tal que } M \text{ aceita apenas uma quantidade finita de entradas}\}$

**c.** EQUIVALENTES :=  $\{c \in \{0, 1\}^* \mid c \text{ codifica máquinas } M \text{ e } N \text{ que aceitam exatamente as mesmas entradas}\}$

**Questão 5.** Seja  $f : (\{0, 1\}^*)^n \rightarrow \{0, 1\}^*$  uma função  $n$ -ária parcial (i.e., uma função cujas saídas são sempre elementos de  $\{0, 1\}^*$  e cujos argumentos de entrada são (algumas, talvez todas as)  $n$ -uplas de elementos de  $\{0, 1\}^*$ ). Dizemos que uma Máquina de Turing  $M$  computa  $f$  se:

- $M$  tem como alfabeto de entrada  $\{0, 1, \#\}$ ;
- $M$  tem pelo menos 2 fitas: uma fita *de entrada* e uma fita *de saída*, mais alguma quantidade finita (talvez zero) de fitas *de rascunho*;
- no início de qualquer execução de  $M$ , todas as fitas exceto a de entrada estão vazias;
- quando  $(c_0, \dots, c_{n-1}) \in (\{0, 1\}^*)^n$  é uma entrada possível para  $f$ , então  $M$  chega a um estado terminal quando executada com  $c_0\#c_1\#\dots\#c_{n-1}$  inicialmente na fita de entrada;
- quando  $(c_0, \dots, c_{n-1}) \in (\{0, 1\}^*)^n$  é uma entrada possível para  $f$ , e  $M$  é executada com  $c_0\#c_1\#\dots\#c_{n-1}$  inicialmente na fita de entrada, então quando  $M$  chega ao seu estado terminal, o conteúdo da fita de saída é exatamente  $f(c_0, \dots, c_{n-1})$ .

Note: quando  $(c_0, \dots, c_{n-1}) \in (\{0, 1\}^*)^n$  **não** é uma entrada possível para  $f$  e  $M$  é executada com  $(c_0, \dots, c_{n-1})$  inicialmente na fita de entrada, **nada afirmamos** sobre o comportamento de  $M$  (não importa o que acontece nesse caso)! Além disso, a máquina pode usar no seu alfabeto de fita quaisquer outros símbolos além de  $0, 1, \#, B$  (“blank”, símbolo representando o vazio).

Prove que as funções a seguir são computáveis:

**a.** “Sucessor em base unária”:  $f(1^x) = 1^{x+1}$ . Aqui,  $f$  só tem como entradas pares de palavras sem ocorrências de  $0$ .

\* **b.** “Soma em base unária”:  $f(1^x, 1^y) = 1^{x+y}$ . Aqui,  $f$  só tem como entradas pares de palavras sem ocorrências de  $0$ .

**c.** “Dobro em base unária”:  $f(1^x) = 1^{2x}$ . Aqui,  $f$  só tem como entradas palavras sem ocorrências de  $0$ .

\* **d.** “Sucessor em base binária”:  $f(c)$  é o binário que representa o sucessor do número natural cuja representação binária é  $c$ . Aqui,  $f$  tem como entrada qualquer  $c$  não-vazia.

**e.** “Dobro em base binária”:  $f(c)$  é o binário que representa o dobro do número natural cuja representação binária é  $c$ . Aqui,  $f$  tem como entrada qualquer  $c$  não-vazia.

\* **f.** “Fatorial em base unária”:  $f(1^n) = 1^{n!}$ , sendo  $n!$  o *fatorial* de  $n$ . Aqui,  $f$  só tem como entradas palavras sem ocorrências de 0.

\***Questão 6.** Dê exemplo de função parcial  $f : (\{0, 1\}^*)^n \rightarrow \{0, 1\}^*$  que não seja computável, para algum  $n \in \mathbb{N}$  (e prove que ela não é computável).

**Questão 7** (Busy Beaver, o “castor ocupado”). É fato (e você não precisa provar!) que para cada  $n \in \mathbb{N}$ , existe uma quantidade finita de Máquinas de Turing  $M$  satisfazendo:

- $M$  tem exatamente  $n + 1$  estados, sendo 1 deles o estado terminal;
- $M$  tem apenas 1 fita;
- $M$  tem alfabetos de entrada  $\{1\}$  e de fita  $\{1, B\}$ ;
- quando executada com sua fita vazia (i.e., com  $B$  em todas as células),

$M$  **sempre chega ao estado terminal** após um número finito de passos.

Assim, para cada  $n \in \mathbb{N}$ , há uma máquina com essas propriedades que deixa uma quantidade *máxima* de 1s escrita na fita ao parar (não necessariamente todos juntos); chamamos de *número de Busy Beaver de  $n$* , denotado  $\text{BB}(n)$ , essa quantidade (algumas fontes usam a notação  $\Sigma(n)$  para o que estamos denotando por  $\text{BB}(n)$ ). Até hoje só se conhecem os valores exatos de  $\text{BB}(n)$  para  $n \leq 4$ . Por exemplo,  $\text{BB}(3) = 6$ ,  $\text{BB}(4) = 13$ ,  $\text{BB}(5) \geq 4098$ , e  $\text{BB}(6)$  certamente não caberia nessa página: ele é maior do que o número

$$10 \uparrow\uparrow 15 := 10 \text{ elevado a } (10 \text{ elevado a } (10 \text{ elevado a } (\dots \text{ elevado a } 10)))$$

com 15 números 10 aparecendo na expressão!

\* **a.** Encontre (e prove que estão corretos) os valores de  $\text{BB}(0)$  e  $\text{BB}(1)$ .

\* **b.** Prove que  $\text{BB}(2) \geq 4$ .