Lógica e Computabilidade 2025-2

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
8/12 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.

Uma Máquina de Turing M é representada por uma "tupla nomeada" (*named tuple*), no seguinte formato:

$$\begin{aligned} M = \texttt{MT(inicial} &= I\texttt{,} \\ \texttt{aceitação} &= A\texttt{,} \\ \texttt{rejeição} &= R\texttt{,} \\ \texttt{transição} &= T\texttt{)} \end{aligned}$$

ou por tuplas simples, no formato

$$M = (I,\ A,\ R,\ T),$$

onde

- I é uma string: o nome do estado inicial de M
- A é um conjunto de strings: nomes dos estados terminais de aceitação de M
- R é um conjunto de strings: nomes dos estados terminais de rejeição de M
- T é um dicionário que representa a função de transição de M; 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 '←' ou '`e`' indicam movimento para a esquerda; as strings '→' ou '`d`' indicam movimento para a direita; qualquer outra string indica que a máquina deve ficar parada. Alternativamente, T pode ser dada como uma função que *recebe* como argumento

(`símbolo_lido`, `estado_atual`)

e *retorna*

`(símbolo_escrito, novo_estado, movimento)`

Uma execução da máquina é feita chamando a função

`executa(entrada, máquina)`,

onde

- `entrada` é a palavra de entrada
- `máquina` é a Máquina de Turing representada como acima

O arquivo `turing.py` contém exemplos. 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, prove que existe uma máquina de Turing que **decide** a linguagem dada. Suponha que Σ seja um conjunto finito e fixo, digamos $\Sigma = \{\sigma_0, \sigma_1, \ldots, \sigma_n\}$.

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).

d. $\{w_0, w_1, w_2, \ldots, w_k\}$, onde $w_0, w_1, w_2, \ldots, w_k \in \Sigma^*$ são palavras arbitrárias dadas.

***Questão 2.** Considere a MT que tem estado inicial q_0, estado terminal de aceitação 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 sucinta ("clara para um ser humano") a linguagem $L \subseteq \{0,1\}^*$ que essa MT aceita.

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

Definição. Um *enumerador* é uma MT 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 (e que a execução pode, ou não, terminar).

Prove que uma linguagem $L \subseteq \{0,1\}^*$ é recursivamente enumerável (i.e., aceita por alguma MT 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 $:= \{c \in \{0,1\}^* \mid c$ codifica alguma máquina M, alguma entrada x para M e algum $n \in \mathbb{N}$, tais que M para em no máximo n passos quando executada com entrada $x\}$

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

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

Questão 5. Seja $f : (\{0,1\}^*)^n \to \{0,1\}^*$ uma função n-ária parcial (i.e., uma função cujas saídas são sempre elementos de $\{0,1\}^*$ e cujos argumentos de entrada são n elementos de $\{0,1\}^*$, para algum $n > 0$, mas não necessariamente *todas* as n-uplas desse tipo são entradas aceitas). Dizemos que uma máquina 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 $(c_0, \ldots, c_{n-1}) \in (\{0,1\}^*)^n$ é uma entrada possível para f, então M chega a um estado terminal quando executada com $c_0\#c_1\#\cdots\#c_{n-1}$ inicialmente na fita de entrada;
- quando $(c_0, \ldots, c_{n-1}) \in (\{0,1\}^*)^n$ é uma entrada possível para f, e M é executada com $c_0\#c_1\#\cdots\#c_{n-1}$ inicialmente na fita de entrada, então quando M chega ao seu estado terminal, o conteúdo da fita de saída é exatamente $f(c_0, \ldots, c_{n-1})$.

Note: quando $(c_0, \ldots, c_{n-1}) \in (\{0,1\}^*)^n$ **não** é uma entrada possível para f e M é executada com $(c_0, \ldots, c_{n-1})$ 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. "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.

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

*** c.** "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.

d. "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.

e. "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 $f : (\{0,1\}^*)^n \to \{0,1\}^*$ 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: 1 estado terminal de aceitação e n outros estados, que não são terminais (de aceitação nem de rejeição);
- M tem apenas 1 fita;
- M tem alfabeto de entrada $\Sigma = \varnothing$ (vazio) e de fita $\Gamma = \{1, B\}$;
- 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 $\mathsf{BB}(n)$, essa quantidade (algumas fontes usam a notação $\Sigma(n)$ para o que estamos denotando por $\mathsf{BB}(n)$). Até hoje só se conhecem os valores exatos de $\mathsf{BB}(n)$ para $n \leqslant 5$. Por exemplo, $\mathsf{BB}(3) = 6$, $\mathsf{BB}(4) = 13$, $\mathsf{BB}(5) = 4098$ (esse resultado foi provado em 2025!), e $\mathsf{BB}(6)$ certamente não caberia nessa página: ele é maior do que o número

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

com 15 números 10 aparecendo na expressão!

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

*** b.** Prove que $\mathsf{BB}(2) \geqslant 4$.

c (Desafio!)**.** Prove que BB não é uma função computável.