Teoria da Computação

Vicente Duarte

1 de Abril de 2025

Teste 1

Recurso (2022/2023)

a) (1.5 valores) Considere o alfabeto $\Sigma = \{0, 1\}$ e as linguagens $L_1, L_2 \subseteq \Sigma^*$ tais que:

L_1: conjunto das palavras da forma $0^{n+1}1^{2n+1}$ com $n \in \mathbb{N}_0$

L_2: conjunto das palavras de L_1 cujo comprimento é menor que dez

Por exemplo, a palavra 00001111111 pertence a L_1, nomeadamente tomando $n = 3$, mas não pertence a L_2, pois tem comprimento onze.

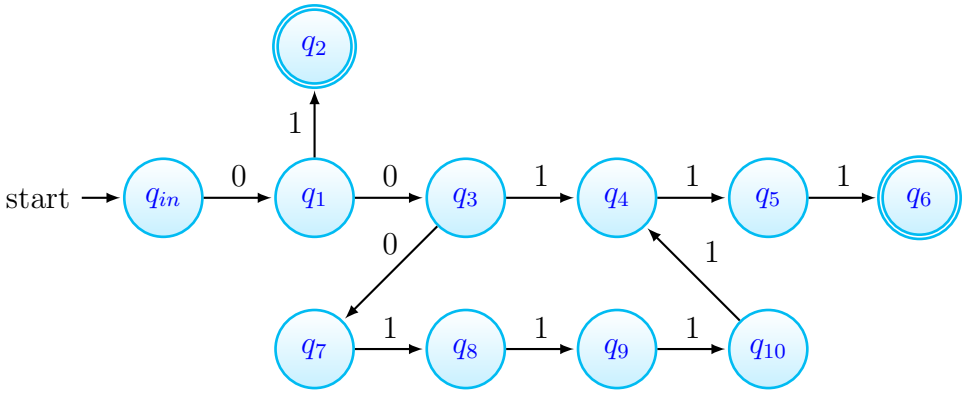
Indique qual das linguagens é regular e mostre-o (construindo um AFD).

b) (1.5 valores) Mostre que a linguagem restante não é regular.

c) (1.0 valores) Mostre (construindo um AP) que a linguagem não regular é independente do contexto.

Resolução

a) Claramente a linguagem L_2 é regular. Assim, podemos construir o AFD seguinte, que aceita precisamente as palavras de L_2:

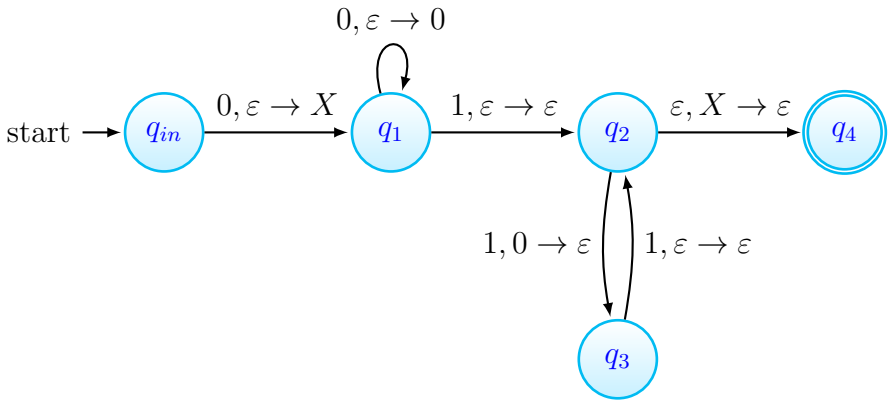


b) Se L_1 fosse regular, pelo lema da bombagem, existiria $k \in \mathbb{N}$ tal que se $w \in L_2$ com $|w| \geq k$ então $w = w_1w_2w_3$ com $|w_1w_2| \leq k$, $w_2 \neq \epsilon$ e $w_1w_2^iw_3 \in L_1$ para todo $i \in \mathbb{N}_0$.

No entanto, dado k e, tomando $w = 0^{k+1}1^{2k+1} \in L_1$ (pois $|w| = 3k+2 \geq k$) ter-se-ia $w_1 = 0^j$, $w_2 = 0^l$ com $l \neq 0$ e $w_3 = 0^{k+1-l-j}1^{2k+1}$.

Logo, com $n = 2$, tem-se $w_1w_2^2w_3 = 0^j0^l0^l0^{k+1-l-j}1^{2k+1} = 0^{k+1+l}1^{2k+1} \notin L_1$ pois $k + 1 + l \neq k + 1$ (pois $l \neq 0$). Assim L_1 não é regular.

c) Basta considerar o AP seguinte, que aceita precisamente as palavras de L_1, com alfabeto auxiliar $\Gamma = \{0, 1, X\}$:



Recurso (2022/2023)

a) (1.5 valores) Considere o alfabeto $\Sigma = \{0, 1, 2\}$ e as linguagens $A, B \subseteq \Sigma^*$ tais que:

A: conjunto das palavras onde nunca ocorrem três símbolos consecutivos iguais

B: conjunto das palavras em que o número de 0s é o dobro do número de 1s

Por exemplo, a palavra 02210001 não pertence a A, pois ocorre a subpalavra 000 com três símbolos consecutivos iguais, mas pertence a B, pois ocorrem quatro 0s e dois 1s.

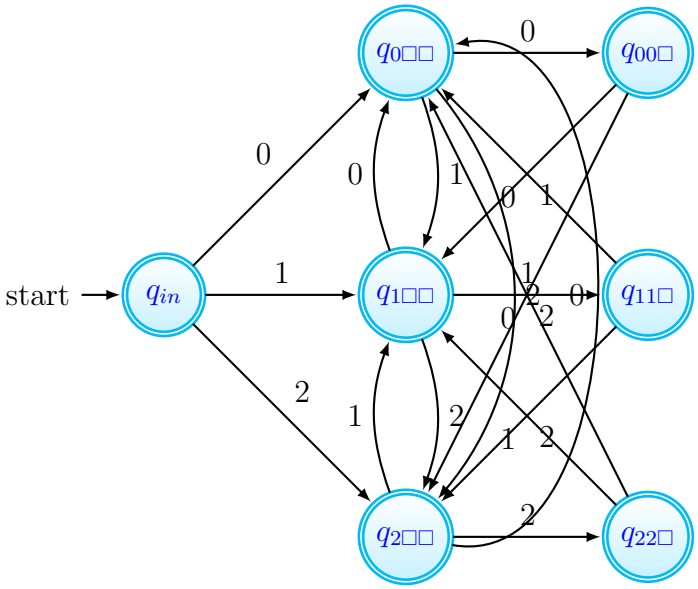
Mostre (construindo um AFD) que A é uma linguagem regular.

b) (1.5 valores) Mostre que B não é uma linguagem regular.

c) (1.0 valores) Mostre (construindo um AP) que B é independente do contexto.

Resolução

a) Basta considerar o AFD seguinte, que aceita precisamente as palavras de A:

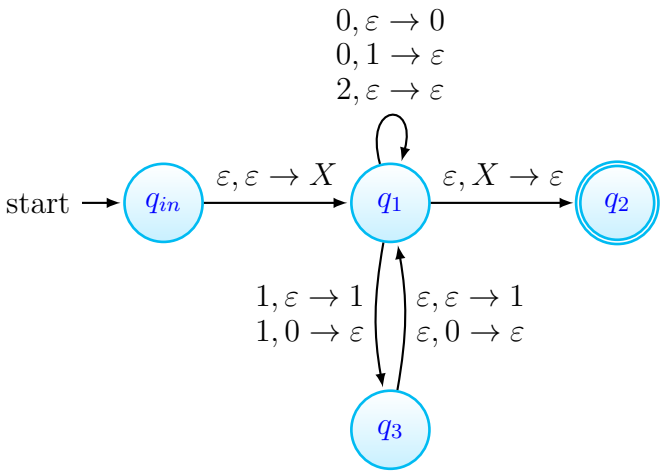


b) Se B fosse uma linguagem regular então, pelo lema da bombagem, existiria $k \in \mathbb{N}$ tal que se $w \in B$ com $|w| \geq k$ então $w = w_1w_2w_3$ com $|w_1w_2| \leq k$, $w_2 \neq \varepsilon$ e $w_1w_2^iw_3 \in B$ para todo $i \in \mathbb{N}_0$.

No entanto, dado k e, tomando $w = 0^{2k}1^k \in B$ (pois $|w| = 3k$) ter-se-ia $w_1 = 0^j$, $w_2 = 0^l$ com $l \neq 0$ e $w_3 = 0^{2k-l-j}1^k$.

Por contradição, dado $i = 2$, onde $w = w_1w_2w_2w_3 = 0^j0^l0^l0^{2k-l-j}1^k = 0^{2k+l}1^k \notin B$ pois $2k + l \neq 2k$ (pois $l \neq 0$).

c) Basta considerar o AP seguinte, que aceita precisamente as palavras de B, com alfabeto auxiliar $\Gamma = \{0, 1, 2, X\}$:



Teste 2

Recurso (2022/2023)

a) (2.5 valores) Mostre (construindo uma máquina de Turing determinista, possivelmente bidireccional, multifita e com movimentos-S) que é computável a função que para cada par de números em notação binária da forma $x\$y$ com $x, y \in \{0,1\}^*$, devolve como resultado a sua soma (também em notação binária).

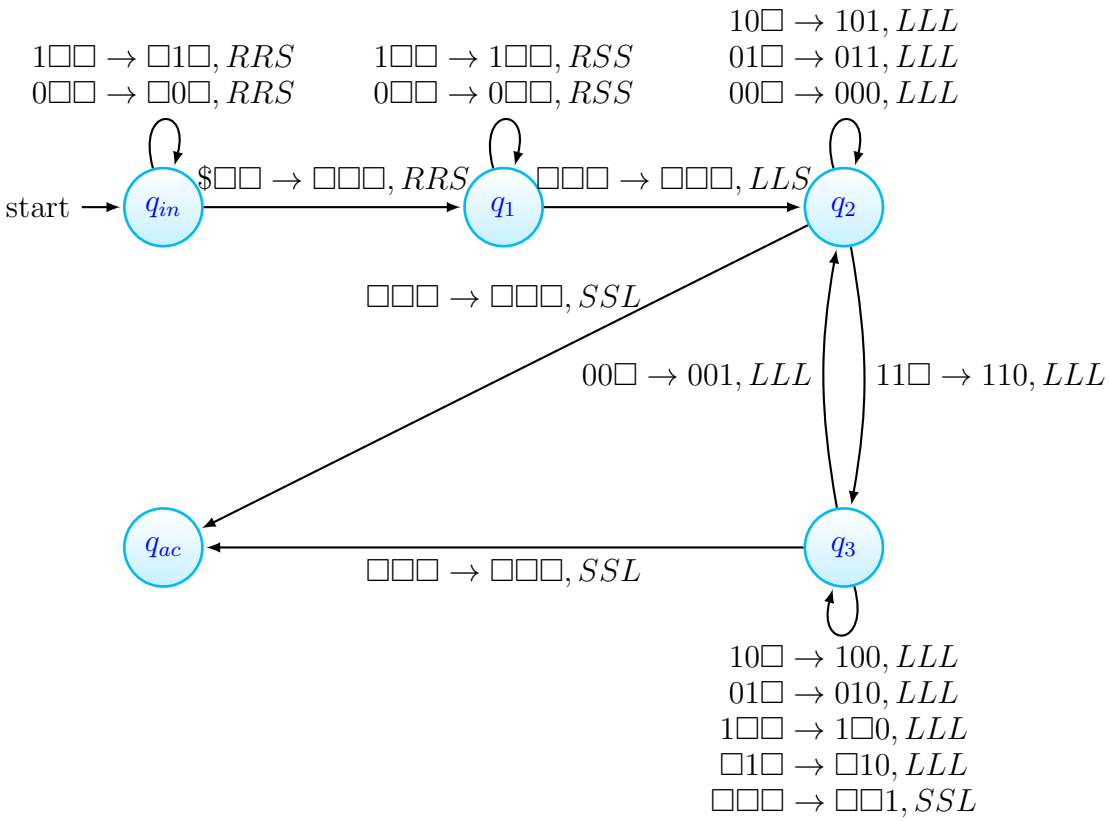
Por exemplo, para a palavra de input 111\$1001 o output deverá ser 10000 (i.e., $7 + 9 = 16$).

b) (2.5 valores) Sejam $L_1, L_2 \subseteq \Sigma^*$ linguagens reconhecíveis. Mostre (directamente, sem recorrer a outras propriedades de fecho estudadas), justificando, que também é reconhecível a linguagem

$$L = \{uvw \in \Sigma^* : (u \in L_1 \text{ e } w \in L_2) \text{ ou } (u \in L_2 \text{ e } w \in L_1)\}.$$

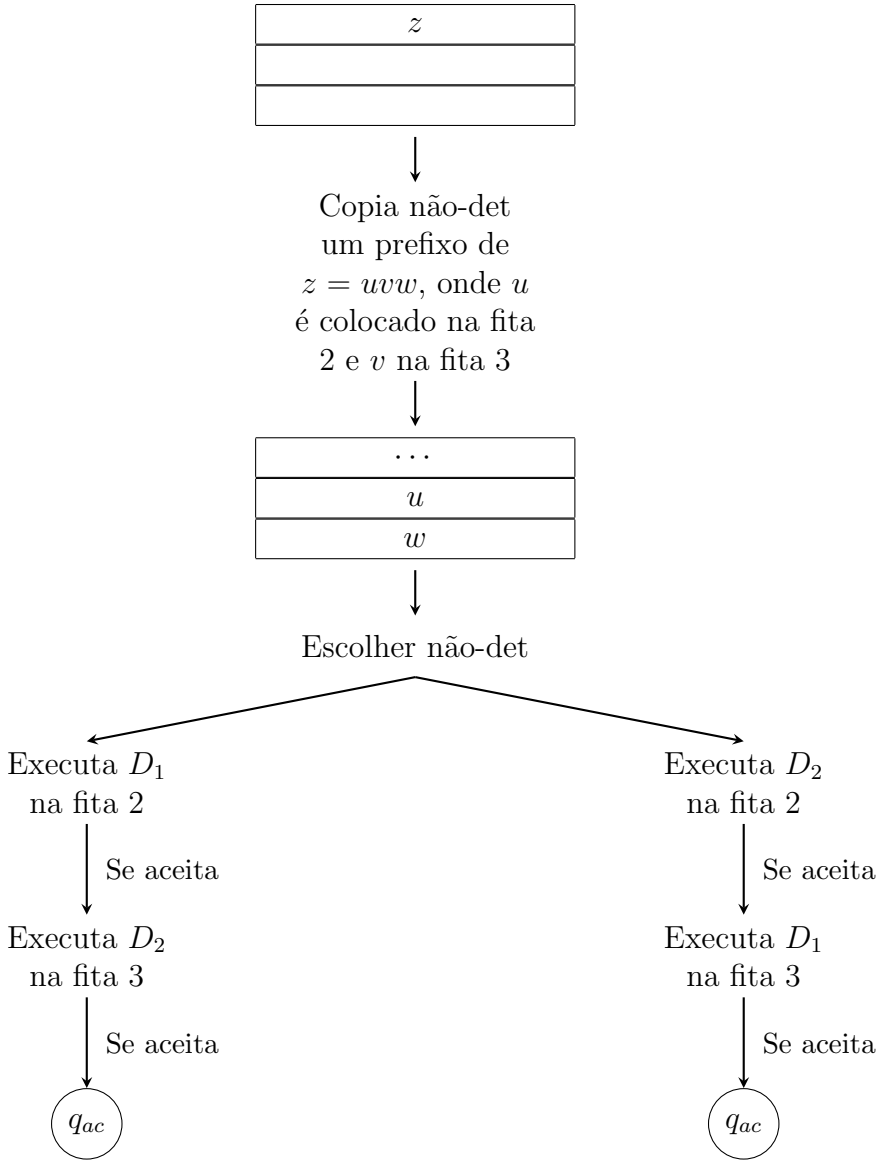
Resolução

a) Basta considerar a MT de duas fitas seguinte, que calcula a função pretendida:



b) Seja D_1, D_2 máquinas classificadoras de uma fita tal que $L_{ac}(D_1) = L_1$ e $L_{rj}(D_1) = \overline{L_1}$ e $L_{ac}(D_2) = L_2$ e $L_{rj}(D_2) = \overline{L_2}$.

Considere-se a MT não-determinista M de 3 fitas seguinte:



M é classificadora pois tem árvores de computação finitas, já que D_1, D_2 são classificadoras e a palavra z se decompõe em $z = uvw$ de um número finito de maneiras.

M aceita a palavra z se, e só se:

- $z = uvw$, D_1 aceita u e D_2 aceita w, ou D_2 aceita u e D_1 aceita w.
- $z = uvw$, $u \in L_1$ e $w \in L_2$ ou $u \in L_2$ e $w \in L_1$
- $L = \{uvw \in \Sigma^* : (u \in L_1 \text{ e } w \in L_2) \text{ ou } (u \in L_2 \text{ e } w \in L_1)\}$.

Recurso (2023/2024)

a) (2.5 valores) Mostre (construindo uma máquina de Turing determinista, possivelmente bidireccional, multifita e com movimentos-S) que é decidível a linguagem $L \subseteq \{1, \$\}^*$ formada pelas listas ordenadas (por ordem crescente, não estrita) de números em notação unária da forma $1^{n_1}\$1^{n_2}\$ \ldots \$1^{n_k}$ com $k, n_1, \ldots, n_k \in \mathbb{N}_0$ e $n_1 \leq n_2 \leq \cdots \leq n_k$.

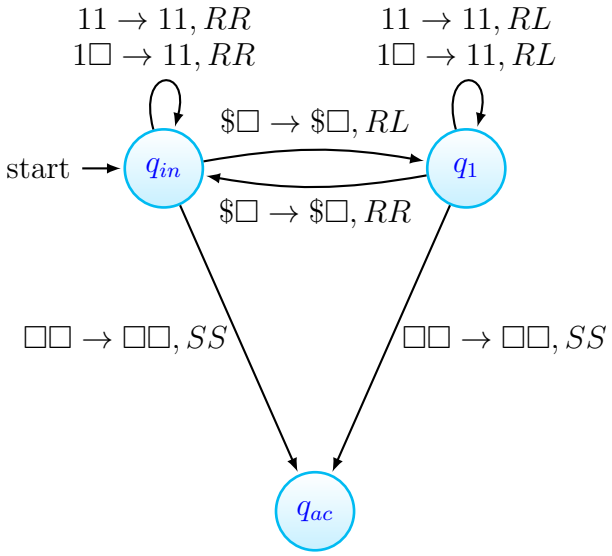
Por exemplo, $11\$11\$111 \in L$ e $11\$111\$11 \notin L$.

b) (2.5 valores) Seja $L \subseteq \Sigma^*$ uma linguagem decidível. Mostre (directamente, sem recorrer a outras propriedades de fecho estudadas), justificando, que também é decidível a linguagem

$$\{uvw : u, v, w \in \Sigma^* \text{ e } v \in L\} \cap \{uvw : u, v, w \in \Sigma^* \text{ e } v \notin L\}.$$

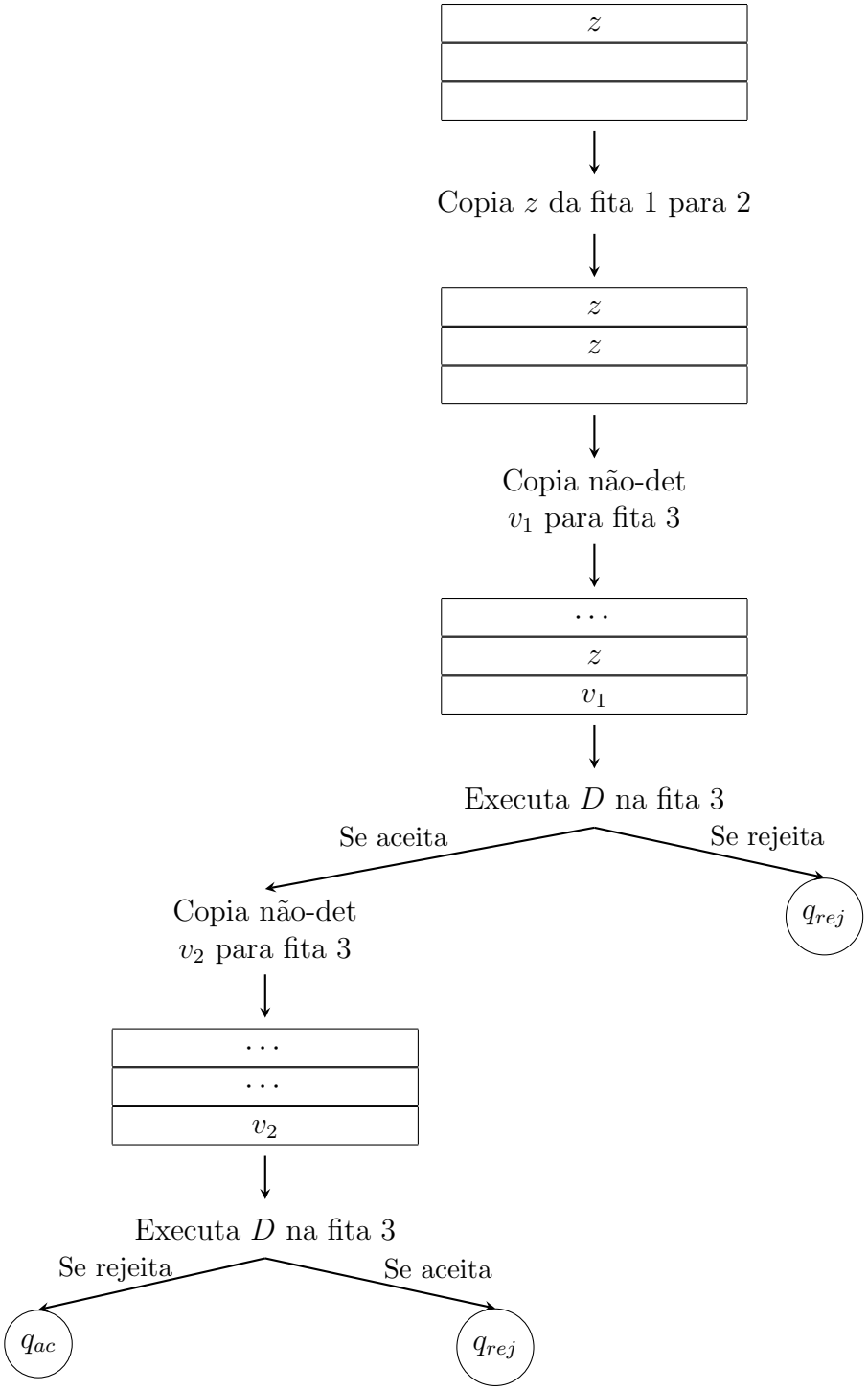
Resolução

a) Basta considerar a MT de duas fitas seguinte, que decide precisamente a linguagem desejada.



b) Seja D uma máquina classificadora de uma fita tal que $L_{ac}(D) = L$ e $L_{rj}(D) = \overline{L}$.

Considere-se a MT não-determinista M de 3 fitas seguinte:



M é classificadora pois tem árvores de computação finitas, já que D é classificadora e a palavra z se decompõe em segmentos v_1 e v_2 de um

número finito de maneiras.

M aceita z se, e só se:

- Existe uma decomposição $z = u_1v_1w_1$ e uma decomposição $z = u_2v_2w_2$ tal que D aceita v_1 e D rejeita v_2.
- Existe um segmento v_1 de z tal que $v_1 \in L$ e existe um segmento v_2 de z tal que $v_2 \notin L$.
- $z \in \{uvw : u, v, w \in \Sigma^* \text{ e } v \in L\}$ e $z \in \{uvw : u, v, w \in \Sigma^* \text{ e } v \notin L\}$.
- $z \in \{uvw : v \in L\} \cap \{uvw : v \notin L\}$.

MAP30–2A.2 (2024/2025)

a) (2.5 valores) Mostre (construindo uma máquina de Turing determinista, possivelmente bidireccional, multifita e com movimentos-S) que é reconhecível a linguagem sobre o alfabeto $\{1, \$, \#\}$ que consiste das listas de pares de números em notação unária da forma

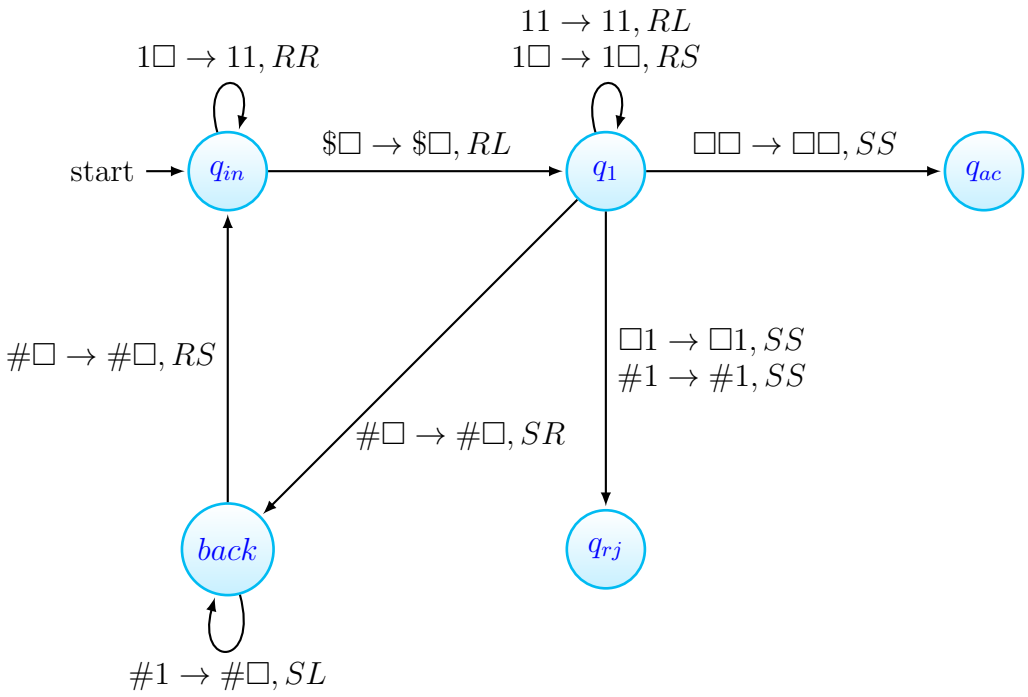
$$x_1\$y_1\#x_2\$y_2\#\ldots\#x_n\$y_n$$

com $x_1, y_1, x_2, y_2, \ldots, x_n, y_n \in \{1\}^*$, para as quais cada $x_i \leq y_i$. Por exemplo, a palavra 11\$11#1\$111 deverá ser aceite pela máquina, mas não a palavra 111\$1.

b) (2.5 valores) Seja $L \subseteq \Sigma^*$ uma linguagem decidível. Mostre (diretamente, sem recorrer a outras propriedades de fecho estudadas) que também é decidível a linguagem $L \cap (L \cdot \overline{L})$. Justifique.

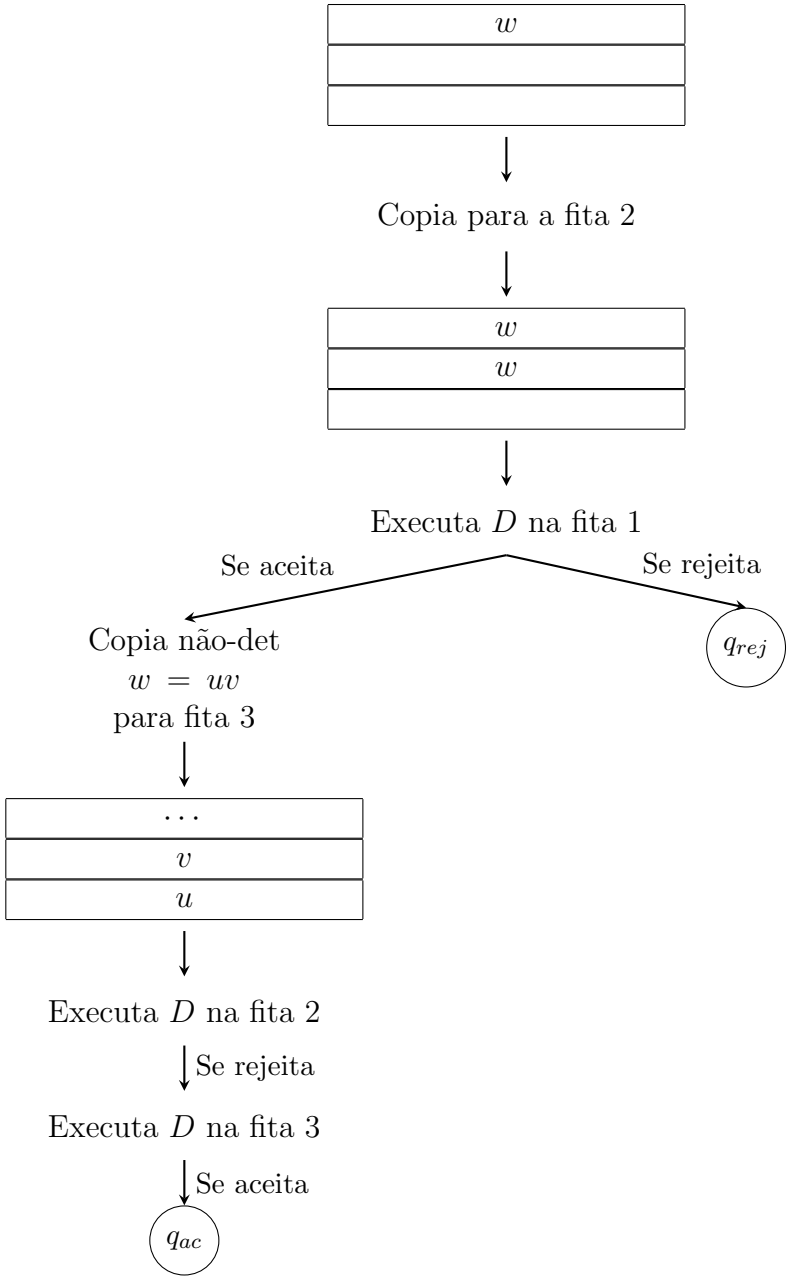
Resolução

a) Basta considerar a MT de duas fitas seguinte, que aceita a linguagem desejada.



b) Seja D uma máquina classificadora de uma fita tal que $L_{ac}(D) = L$ e $L_{rj}(D) = \overline{L}$.

Considere-se a MT não-determinista M de 3 fitas seguinte:



M é classificadora pois tem árvores de computação finitas já que D é classificadora e w se decompõe em $w = uv$ de um número finito de maneiras.

M aceita w se, e só se:

- $w = uv$, D rejeita v e D aceita u, e D aceita w.

- $w = uv$, $v \notin L$ e $u \in L$, e $w \in L$.
- $w \in (L \cdot \overline{L})$ e $w \in L$.
- $w \in L \cap (L \cdot \overline{L})$.

MAP30–2B.1 (2024/2025)

a) (2.5 valores) Mostre (construindo uma máquina de Turing determinista, possivelmente bidireccional, multifita e com movimentos-S) que é computável a função que para cada lista de pares de números em notação unária da forma

$$x_1\$y_1\#x_2\$y_2\#\ldots\#x_n\$y_n$$

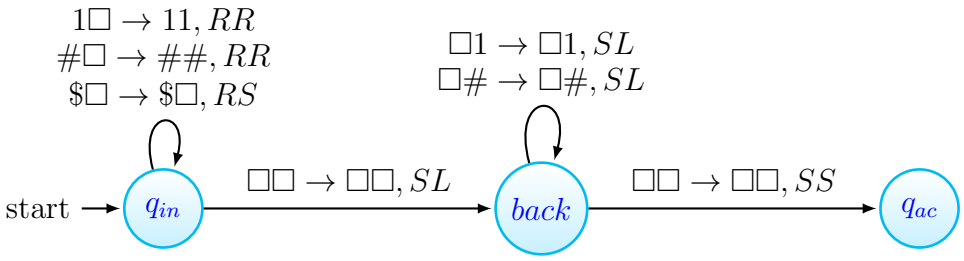
com $x_1, y_1, x_2, y_2, \ldots, x_n, y_n \in \{1\}^*$, devolve como resultado a lista $z_1\#\ldots\#z_n$ de números também em notação unária em que cada $z_i = x_i + y_1$.

Por exemplo, para a palavra de input 111\$11#1\$11 o output deverá ser 11111#111.

b) (2.5 valores) Sejam $L_1, L_2 \subseteq \Sigma^*$ linguagens decidíveis. Mostre (directamente, sem recorrer a outras propriedades de fecho estudadas) que também é decidível a linguagem definida por $\{uvw \in L_1 : v \notin L_2\}$. Justifique.

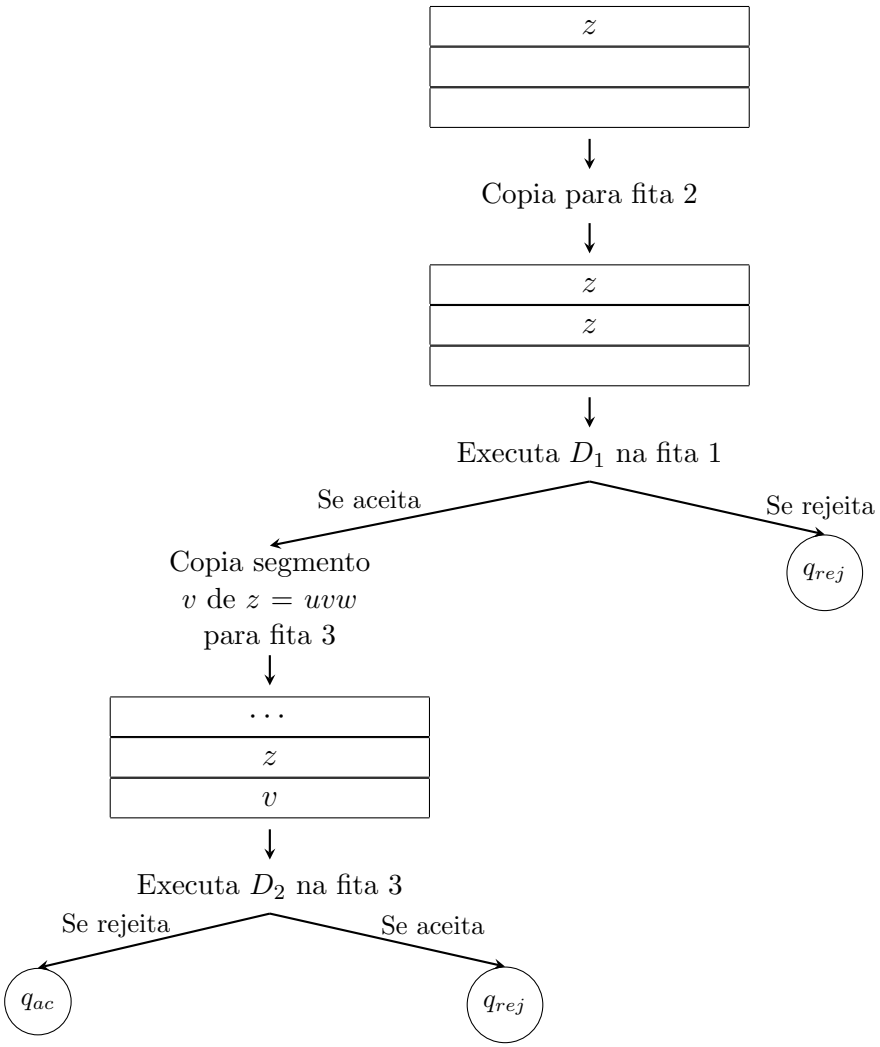
Resolução

a) Basta considerar a MT de duas fitas seguinte, que calcula a função pretendida.



b) Seja D_1 e D_2 máquinas classificadoras de uma fita tal que $L_{ac}(D_1) = L_1$ e $L_{rj}(D_1) = \overline{L_1}$ e $L_{ac}(D_2) = L_2$ e $L_{rj}(D_2) = \overline{L_2}$.

Considere-se a MT não-determinista M de 3 fitas seguinte:



M é classificadora pois tem árvores de computação finitas já que D_1 e D_2 são classificadoras e z se decompõe em $z = uvw$ de um número finito de maneiras.

M aceita z se, e só se:

- $z = uvw$, D_1 aceita z e D_2 rejeita v;
- $z \in L_1$ e $v \notin L_2$;

- $z \in \{uvw \in L_1 : v \notin L_2\}$.

MAP30–2B.1 (2024/2025)

a) (2.5 valores) Mostre (construindo uma máquina de Turing determinista, possivelmente bidireccional, multifita e com movimentos-S) que é computável a função que para cada lista de pares de números em notação unária da forma

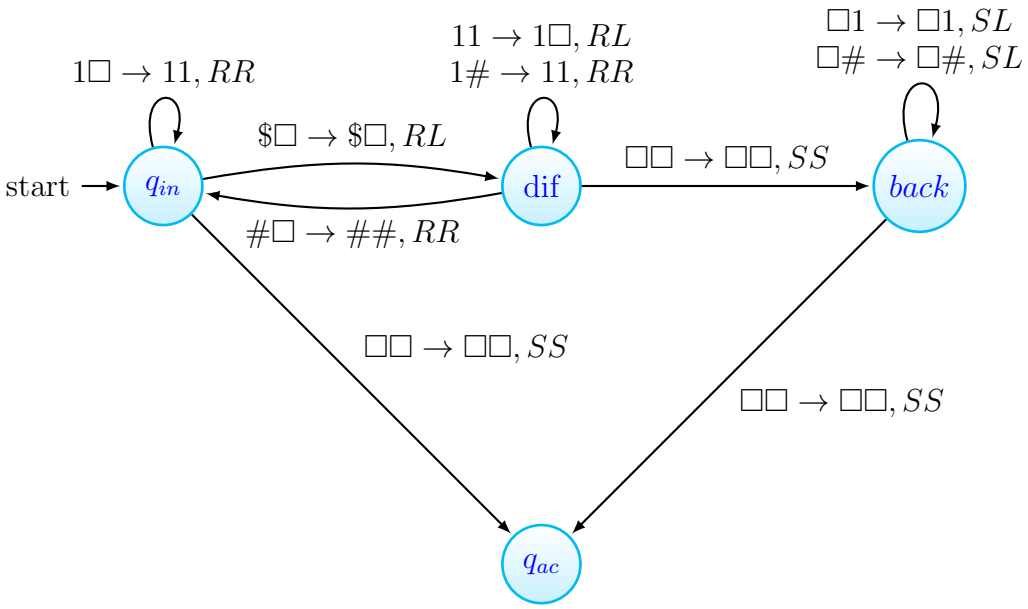
$$x_1\$y_1\#x_2\$y_2\#\ldots\#x_n\$y_n$$

com $x_1, y_1, x_2, y_2, \ldots, x_n, y_n \in \{1\}^*$, devolve como resultado a lista $z_1\#\ldots\#z_n$ de números também em notação unária em que cada $z_i = |x_i - y_i|$ (ou seja, z_i é o módulo da diferença entre x_i e y_i). Por exemplo, para a palavra de *input* 111\$11#1\$1#1\$111 o *output* deverá ser 1##11.

b) (2.5 valores) Sejam $L_1, L_2 \subseteq \Sigma^*$ linguagens decidíveis. Mostre (directamente, sem recorrer a outras propriedades de fecho estudadas) que também é decidível a linguagem $L_1 \cdot \overline{L_1 \cap L_2}$. Justifique.

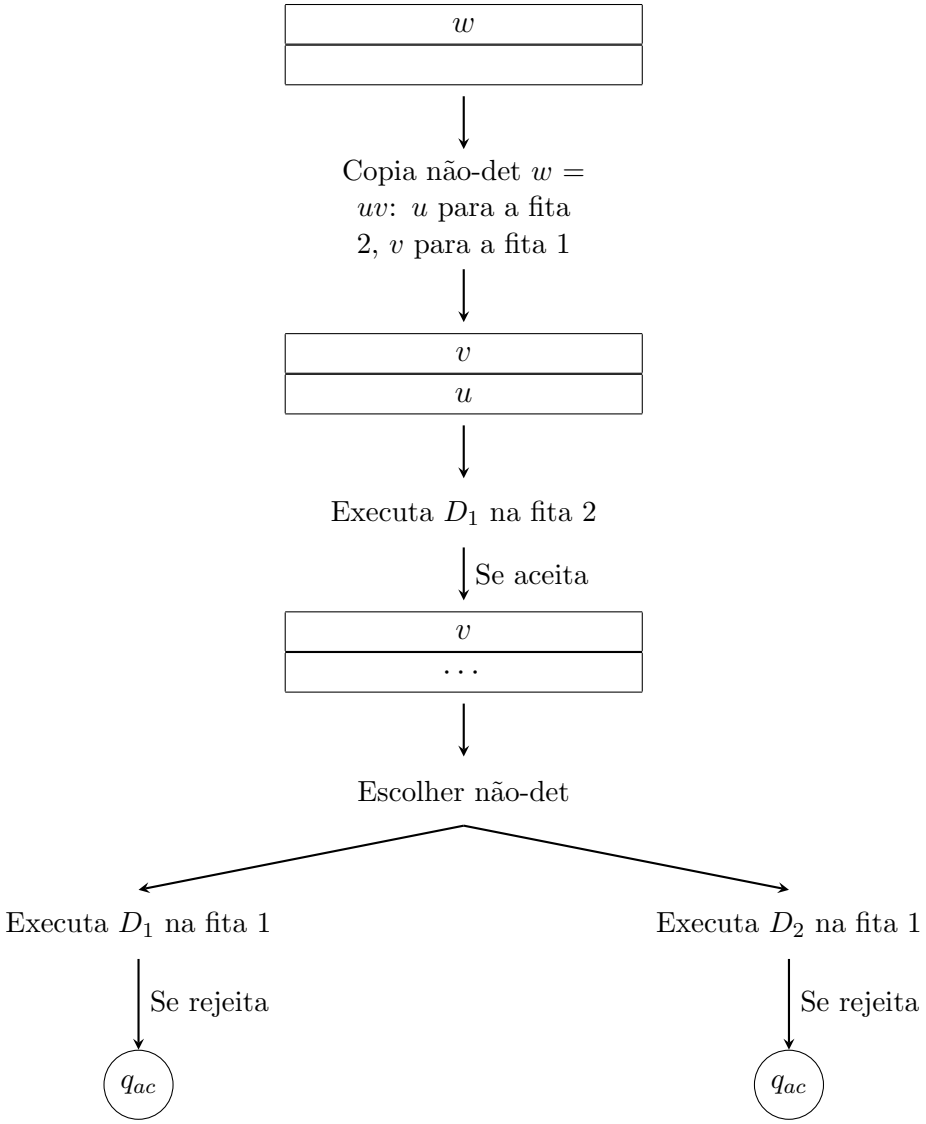
Resolução

a) Basta considerar a MT de duas fitas seguinte, que calcula a função pretendida.



b) Sejam D_1 e D_2 máquinas classificadoras de uma fita tal que $L_{ac}(D_1) = L_1$, $L_{rj}(D_1) = \overline{L_1}$, $L_{ac}(D_2) = L_2$ e $L_{rj}(D_2) = \overline{L_2}$.

Considere-se a MT não-determinista M de 3 fitas seguinte:



M é classificadora pois tem árvores de computação finitas já que D_1 e D_2 são classificadoras e w se decompõe em $w = uv$ de um número finito de maneiras.

M aceita w se, e só se existe uma decomposição $w = uv$ tal que:

- D_1 aceita u **e** (D_1 rejeita v **ou** D_2 rejeita v)
- $u \in L_1$ **e** ($v \notin L_1$ **ou** $v \notin L_2$)
- $u \in L_1$ **e** $v \in (\overline{L_1} \cup \overline{L_2})$

- $u \in L_1$ **e** $v \in \overline{L_1 \cap L_2}$ (pela lei de De Morgan)
- $w \in L_1 \cdot \overline{L_1 \cap L_2}$

Teste 3

MAP30–3A.2 (2024/2025)

a) (1.0 valores) Considere o alfabeto $\Sigma = \{0, 1\}$ e as linguagens L_1, L_2, L_3 seguintes, sabendo que uma das linguagens é decidível e as outras duas linguagens são indecidíveis.

$$L_1 = \{M \in \mathcal{M}^\Sigma : L_{ac}(M) = \overline{L_3}\}$$
$$L_2 = \{M\$w : M \in L_1, w \in L_{ac}(M)\}$$
$$L_3 = \{w \in \Sigma^* : |w| \text{ é múltiplo de } 100\}$$

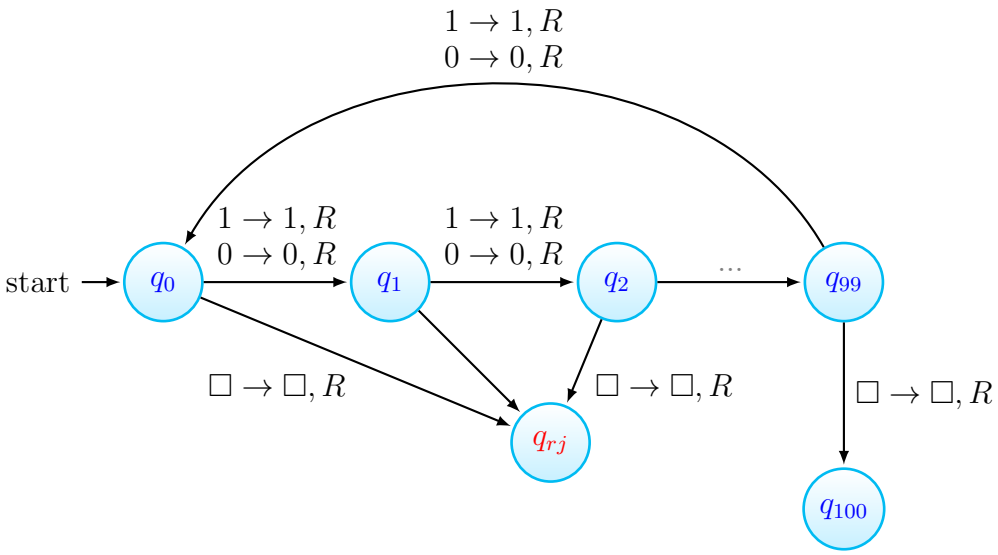
Das linguagens dadas, identifique a que é decidível e mostre que está na classe **TIME**(n). Justifique.

b) (2.0 valores) Use o teorema de Rice para demonstrar a indecidibilidade de uma das outras duas linguagens (escolhida criteriosamente).

c) (2.0 valores) Demonstre a indecidibilidade da linguagem restante, por redução da linguagem indecidível da alínea anterior.

Resolução

a) A linguagem L_3 é decidível. Basta considerar a MT seguinte:



Facilmente, para esta máquina M, tem-se $\text{time}_M = \mathcal{O}(n)$, onde n é o tamanho do input, e portanto conclui-se que $L_3 \in \mathbf{TIME}(n)$

b) Usamos o *Teorema de Rice* para demonstrar a indecidibilidade de $L_1 \subseteq \mathcal{M}^\Sigma$, por definição de L_1. Verificamos cada uma das condições.

1) $\underline{L_1 \neq \emptyset}$
Considere-se a máquina M' que aceita todas as palavras que não têm comprimento múltiplo de 100 (trocando os estados de q_{rj}/q_{ac} na máquina M que decide L_3, obtendo-se M' que decide precisamente $\overline{L_3}$). De facto, $M' \in L_1$, portanto $L_1 \neq \emptyset$.

2) $\underline{L_1 \neq \mathcal{M}^\Sigma}$
A máquina M definida na alínea a) é tal que $L_{ac}(M) = L_3$, logo $M \notin L_1$. Portanto, $L_1 \neq \mathcal{M}^\Sigma$.

3) Considere-se duas máquinas M_1, M_2 equivalentes, isto é, $L_{ac}(M_1) = L_{ac}(M_2)$. Se $M_1 \in L_1$, então $L_{ac}(M_1) = \overline{L_3}$, logo $L_{ac}(M_2) = \overline{L_3}$, e portanto $M_2 \in L_1$.

Conclui-se pelo Teorema de Rice que L_1 é indecidível.

c) Para demonstrar a indecidibilidade de L_2 vamos mostrar que $L_1 \leq L_2$.

Considere-se a função $f : \{0,1\} \to \{0,1,\$\}$ definida por $f(x) = x\$010$. Obviamente, f é total e computável. Além disso:

- Se $x \in L_1$ e portanto $L_{ac}(x) = \overline{L_3}$, então $f(x) = x\$010 \in L_2$, pois $010 \in L_{ac}(x)$ (não tem comprimento múltiplo de 100)
- Se $x \notin L_1$, então $f(x) = x\$010 \notin L_2$ e nada podemos concluir sobre $010 \in L_{ac}(x)$.

Como sabemos da alínea b) que L_1 é indecidível, então L_2 também é indecidível.

Recurso (2023/2024)

a) (1.5 valores) Seja $\Sigma = \{0,1\}$. Considere as linguagens $L_1, L_2, L_3 \subseteq \Sigma^*$ seguintes, sabendo que uma das linguagens é decidível e as outras são indecidíveis.

$$L_1 = \{M \in \mathcal{M}^\Sigma : \mathcal{M}^\Sigma \subseteq L_{ac}(M)\}$$
$$L_2 = \{M \in \mathcal{M}^\Sigma : M \text{ aceita } M \text{ em não mais de cem passos}\}$$
$$L_3 = \{M \in \mathcal{M}^\Sigma : M \in L_{ac}(M)\}$$

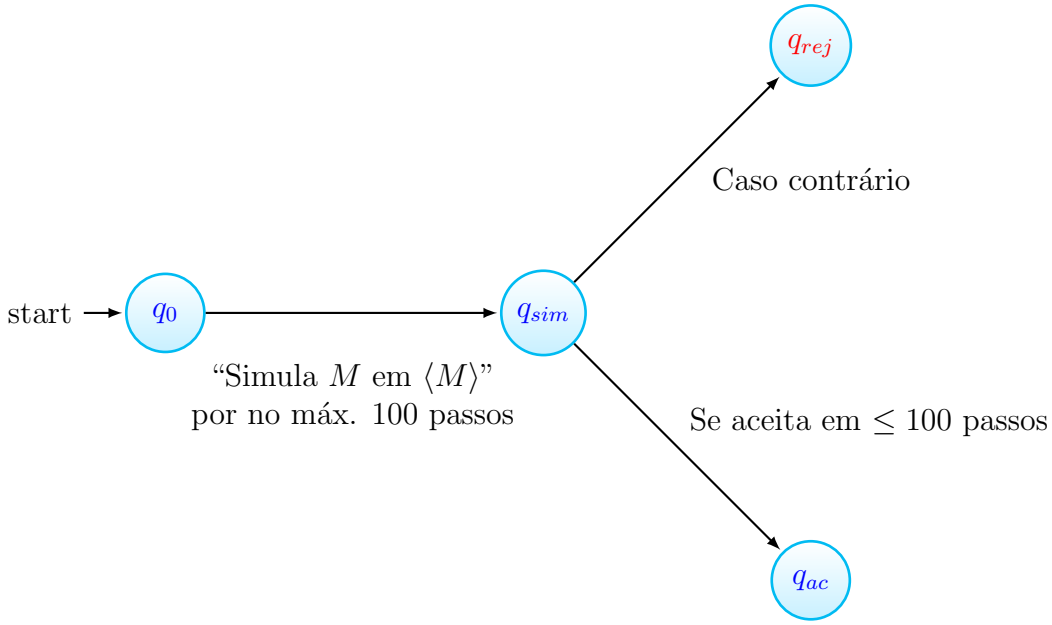
Das linguagens dadas, identifique a que é decidível e mostre que está na classe **P**. Justifique.

b) (2.0 valores) Use o teorema de Rice para demonstrar a indecidibilidade de uma das outras duas linguagens (escolhida criteriosamente).

c) (1.5 valores) Demonstre a indecidibilidade da linguagem restante (se necessário recorrendo aos resultados sobre indecidibilidade estudados).

Resolução

a) A linguagem L_2 é decidível. Basta considerar a MT seguinte:



Para esta máquina M, como a simulação é limitada por um número constante de passos, e cada passo pode ser simulado em tempo polinomial no tamanho da codificação de M, conclui-se que $\text{time}_M = \mathcal{O}(n^k)$ para algum k, onde $n = |\langle M \rangle|$. Logo, $L_2 \in \mathbf{P}$.

Justificação

A linguagem L_2 é decidível e pertence a $\mathbf{P}$ porque podemos construir uma máquina S que, para uma entrada $\langle M \rangle$, simula a máquina M com a entrada $\langle M \rangle$ por, no máximo, 100 passos.

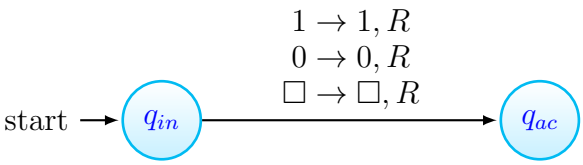
A simulação de um passo de M envolve consultar a função de transição de M (que está em $\langle M \rangle$) e atualizar o estado, a fita e a posição da cabeça simulados. Como o tamanho de $\langle M \rangle$ é n, encontrar a transição e atualizar a configuração pode ser feito em tempo polinomial em n, digamos $\mathcal{O}(n^k)$.

Como a simulação total é limitada a um número **constante** de passos (100), o tempo total gasto por S é $100 \times \mathcal{O}(n^k)$, que ainda é $\mathcal{O}(n^k)$, ou seja, polinomial. A simulação sempre termina após no máximo 100 passos, aceitando se M aceitou $\langle M \rangle$ dentro desse limite, e rejeitando caso contrário. Este limite constante é essencial, visto que, sem ele

(como em L_3), a simulação poderia não terminar ou levar tempo não polinomial.

b) Usamos o *Teorema de Rice* para demonstrar a indecidibilidade de $L_1 \subseteq \mathcal{M}^\Sigma$, por definição de L_1. Verificamos cada uma das condições.

1) $\underline{L_1 \neq \emptyset}$
Considere-se a máquina M_{ALL} que aceita todas os inputs.



Claramente, $L_{ac}(M_{ALL}) = \Sigma^*$, logo $M_{ALL} \in L_1$. Assim, $L_1 \neq \emptyset$.

2) $\underline{L_1 \neq \mathcal{M}^\Sigma}$
Considere-se uma máquina $M_\emptyset$ sem transições que aborta para todos os inputs. Obviamente, $L_{ac}(M_\emptyset) = \emptyset$, portanto, neste caso, temos que $\Sigma^* \nsubseteq L_{ac}(M_\emptyset)$, logo $M_\emptyset \notin L_1$, e portanto $L_1 \neq \mathcal{M}^\Sigma$.

3) Seja M_1, M_2 duas máquinas equivalentes, tais que $L_{ac}(M_1) = L_{ac}(M_2)$. Se $M_1 \in L_1$, então $\Sigma^* \subseteq L_{ac}(M_1)$, e como $L_{ac}(M_2) = L_{ac}(M_1)$, também $M_2 \in L_1$.

Conclui-se pelo Teorema de Rice que L_1 é indecidível.

c) Para demonstrar a indecidibilidade de L_3 vamos mostrar que $L_1 \leq L_3$. Considere-se a função $f : \mathcal{M}^\Sigma \to \mathcal{M}^\Sigma$ definida por

$$f(M) = N,$$

onde a máquina N é construída da seguinte forma:

- N ignora o seu input e, internamente, simula M em todas as possíveis entradas.
- Se M aceita todas as palavras (isto é, se $L_{ac}(M) = \Sigma^*$, isto é, se $M \in L_1$), então N aceita qualquer input (portanto, $L_{ac}(N) = \Sigma^*$).
- Caso contrário, se $M \notin L_1$, então N rejeita qualquer input (isto é, $L_{ac}(N) = \emptyset$).

Note-se que essa construção garante que:

- Se $M \in L_1$, então $L_{ac}(N) = \Sigma^*$ e, em particular, N aceita a sua própria descrição (isto é, $N \in L_3$).
- Se $M \notin L_1$, então $L_{ac}(N) = \emptyset$, de modo que N não aceita a sua própria descrição (isto é, $N \notin L_3$).

Como sabemos (pela aplicação do Teorema de Rice) que L_1 é indecidível, conclui-se que L_3 também é indecidível.

Teste 4

Recurso (2023/2024)

a) (3 valores) Um número $n \in \mathbb{N}_0$ diz-se *semi-primo* se existem primos p e q (possivelmente iguais) tais que $n = p \times q$. Portanto, se $n > 1$ não for semi-primo então ou n é primo, ou n é o produto de três números maiores que 1.

Considere as linguagens $L_{sp} = \{1^n \mid n \text{ é semi-primo}\}$ e $L_p = \{1^n \mid n \text{ é primo}\}$. Sabendo que $L_p \in \mathbb{P}$ mostre que:

(i) $L_{sp} \in \mathbb{NP}$,

(ii) $\overline{L_{sp}} \in \mathbb{NP}$.

b) (2 valores) Demonstre que $\mathbf{SPACE}(\log n) \subseteq \mathbf{P}$.

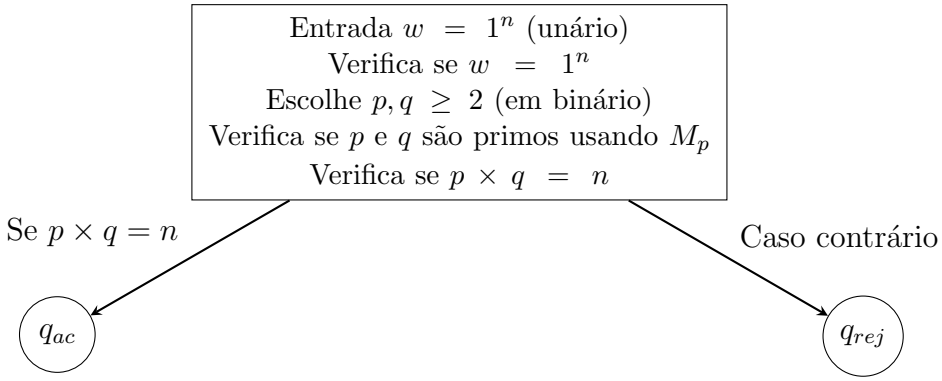
c) (1 valor) Demonstre que se $f : \mathbb{N}_0 \to \mathbb{N}_0$ é uma função construtível no tempo então também é construtível no espaço.

Resolução

a) (i) $L_{sp} \in \mathbb{NP}$

Como $L_p \in \mathbb{P}$, existe uma máquina de Turing determinística M_p que decide se n é primo em tempo polinomial em relação ao tamanho da entrada.

Seja M_{sp} uma máquina de Turing não-determinística que decide L_{sp}:



A complexidade temporal de M_{sp} é $O(n)$, pois:

- Verificação da entrada: $O(n)$.
- Verificação da primalidade: $O((\log n)^k)$ usando M_p.
- Verificação do produto: $O((\log n)^2)$.

Portanto, $L_{sp} \in \mathbb{NP}$.

(ii) $\overline{L_{sp}} \in \mathbb{NP}$

Para demonstrar que $\overline{L_{sp}} \in \mathbb{NP}$, basta mostrar um certificado verificável em tempo polinomial. Uma palavra $w = 1^n$ está em $\overline{L_{sp}}$ se:

- Caso 1: n é primo (verificável em tempo polinomial usando M_p), ou
- Caso 2: n é o produto de pelo menos três fatores maiores que 1.

A máquina de Turing não-determinística $M_{\overline{sp}}$ para $\overline{L_{sp}}$ adivinha qual dos casos se aplica e verifica em tempo polinomial. Para o caso 2, adivinha três fatores $a, b, c \geq 2$ e verifica se $a \times b \times c = n$. Como a verificação ocorre em tempo polinomial (dominada por $O(n)$ para leitura da entrada), concluímos que $\overline{L_{sp}} \in \mathbb{NP}$.

b) Seja $L \in \mathbf{SPACE}(\log n)$ e D uma máquina classificadora determinista tal que $L_{ac}(D) = L$ e $\text{space}_D(n) = O(\log n)$.

Como qualquer computação de D termina, não é possível que a mesma configuração ocorra duas vezes na mesma computação.

Logo, o comprimento máximo de qualquer computação de D é limitado pelo número de configurações possíveis (em espaço $space_D(n)$), ou seja:

$$time_D(n) \leq |T|^{space_D(n)} \times |Q| \times space_D(n)$$

Uma configuração de uma máquina de Turing é determinada por:

- **Estado atual** $q \in Q$: $|Q|$ possibilidades.
- **Conteúdo da fita**: $|T|^{space_D(n)}$ combinações, onde T é o alfabeto.
- **Posição da cabeça**: $space_D(n)$ posições possíveis.

$$\begin{aligned} time_D(n) &\leq |T|^{space_D(n)} \times |Q| \times space_D(n) \\ &\leq |T|^{O(\log n)} \times |Q| \times O(\log n) \\ &= n^{O(\log |T|)} \times |Q| \times O(\log n) \\ &= n^{O(1)} \times O(\log n) \\ &= O(n^c \log n) \end{aligned}$$

Conclui-se que $L \in \mathbf{TIME}(n^{O(1)})$, ou seja, $L \in \mathbf{P}$. Portanto, $\mathbf{SPACE}(\log n) \subseteq \mathbf{P}$.

c) Se f é construtível no tempo, então existe uma máquina de Turing M tal que, para toda entrada de tamanho n, M produz uma string de comprimento exatamente $f(n)$ em $O(f(n))$ passos de computação.

Como sabemos, o número de células da fita que uma máquina de Turing pode visitar é limitado pelo número de passos de computação executados. Em outras palavras, o espaço usado nunca pode exceder o tempo gasto.

Portanto, se M computa f em tempo $O(f(n))$, então M também usa no máximo $O(f(n))$ células da fita. Isso significa que M demonstra que f é construtível no espaço, pois consegue produzir uma string de comprimento $f(n)$ usando $O(f(n))$ espaço.

Conclui-se que toda função construtível no tempo também é construtível no espaço.

MAP30–4A.2 (2023/2024)

a) (3 valores) Seja um alfabeto $\Sigma \notin \$$, e considere linguagens $A, B, C, L \subseteq \Sigma^*$ tais que:

- $A \leq_P B \cap C$,
- $B \leq_P L$,
- $C \leq_P \overline{L}$.

Mostre, justificando, que $A \leq_P \{w_1\$w_2 : w_1 \in L \text{ e } w_2 \in \Sigma^* \setminus L\}$.

b) (1 valor) Seja $p(n)$ um polinómio. Demonstre (diretamente, sem recorrer a outros resultados estudados) que $\mathbf{SPACE}(p(n)) \subseteq \mathbf{EXPTIME}$.

c) (2 valores) Considere a classe $\mathbf{duNP} = \{L : \overline{L} \in \mathbf{NP}\}$. Demonstre, justificando, que se tem $\mathbf{P} \subseteq \mathbf{duNP}$ e $\mathbf{duNP} \subseteq \mathbf{EXPTIME}$ (pode invocar outros resultados estudados acerca de classes de complexidade e suas propriedades).

Resolução

a)
- Se $A \leq_P B \cap C$, então existe $f : \Sigma^* \to \Sigma^*$ total e computável por uma máquina de Turing F com $time_F(n) = O(n^a)$ tal que $x \in A$ sse $f(x) \in B \cap C$.

- Se $B \leq_P L$, então existe $g : \Sigma^* \to \Sigma^*$ total e computável por uma máquina de Turing G com $time_G(n) = O(n^b)$ tal que $x \in B$ sse $g(x) \in L$.

- Se $C \leq_P \overline{L}$, então existe $h : \Sigma^* \to \Sigma^*$ total e computável por uma máquina de Turing H com $time_H(n) = O(n^c)$ tal que $x \in C$ sse $h(x) \in \overline{L}$.

Então, a função $k : \Sigma^* \to (\Sigma^* \cup \{\$\})$ dada por $k(x) = g(f(x))\$h(f(x))$ é total e computável pois g, f e h são polinomiais e a concatenação com \$ é feita em tempo polinomial. Uma certa máquina de Turing K computa k:

$$\begin{aligned} time_k(n) &= O(time_F(n) + time_G(n + time_F(n)) + time_H(n + time_F(n))) \\ &= O(n^a + (n + n^a)^b + (n + n^a)^c) \text{ que é um polinómio.} \end{aligned}$$

Além disso,

$$\begin{aligned} x \in A &\iff f(x) \in B \cap C \\ &\iff g(f(x)) \in L \text{ e } h(f(x)) \in \overline{L} \\ &\iff k(x) = g(f(x))\$h(f(x)) \in L\$\overline{L} \\ &\iff k(x) = g(f(x))\$h(f(x)) \in \{w_1\$w_2 : w_1 \in L \text{ e } w_2 \in \Sigma^* \setminus L\}. \end{aligned}$$

Conclui-se que $A \leq_P \{w_1\$w_2 : w_1 \in L \text{ e } w_2 \in \Sigma^* \setminus L\}$.

b) Seja $L \in \mathbf{SPACE}(p(n))$ e D uma máquina classificadora determinista tal que $L_{ac}(D) = L$ e $\text{space}_D(n) = O(p(n))$.

Como qualquer computação de D termina, não é possível que a mesma configuração ocorra duas vezes na mesma computação.

Logo, o comprimento máximo de qualquer computação de D é limitado pelo número de configurações possíveis (em espaço $space_D(n)$), ou seja:

$$time_D(n) \leq |T|^{space_D(n)} \times |Q| \times space_D(n)$$

Uma configuração de uma máquina de Turing é determinada por:

- **Estado atual** $q \in Q$: $|Q|$ possibilidades.
- **Conteúdo da fita**: $O(p(n))$ combinações, onde T é o alfabeto.
- **Posição da cabeça**: $O(p(n))$ posições possíveis.

$$\begin{aligned} time_D(n) &\leq |T|^{space_D(n)} \times |Q| \times space_D(n) \\ &\leq 2^{\log_2 T \times space_D(n)} \times |Q| \times space_D(n) \\ &\leq 2^{space_D(n)} \times O(space_D(n)) \\ &= 2^{space_D(n)} \end{aligned}$$

Conclui-se que $L \in \mathbf{TIME}(2^{O(p(n))})$, ou seja, $L \in \mathbf{EXPTIME}$. Portanto, $\mathbf{SPACE}(p(n)) \subseteq \mathbf{EXPTIME}$.

c) Se $L \in \mathbf{P}$, sabemos que $\overline{L} \in \mathbf{P}$ (trocando q_{ac}/q_{rj} na máquina de Turing que decide L, obtendo precisamente a máquina que decide $\overline{L}$ na mesma eficiencia temporal).

Sabemos também que $\mathbf{P} \subseteq \mathbf{NP}$ (pois uma máquina determinista é um caso particular de uma máquina não determinista). Logo $\overline{L} \in \mathbf{NP}$ e, portanto, $L \in \mathbf{duNP}$. Conclui-se que $\mathbf{P} \subseteq \mathbf{duNP}$ (1).

Se $L \in \mathbf{duNP}$ então $\overline{L} \in \mathbf{NP}$. Sabemos que $\mathbf{NP} \subseteq \mathbf{EXPTIME}$ (pois uma máquina não determinista pode ser simulada por uma máquina determinista em tempo exponencial). Logo, $\overline{L} \in \mathbf{EXPTIME}$ e, portanto, $L \in \mathbf{EXPTIME}$. Conclui-se que $\mathbf{duNP} \subseteq \mathbf{EXPTIME}$ (2).

MAP30–4A.2 (2022/2023)

a) (3.0 valores) Sabendo que $L_1 \in \mathbf{NSPACE}(n)$ e que $L_2 \leq_p \overline{L_1}$, pode garantir que:

i) $\overline{L_1} \in \mathbf{SPACE}(n^2)$?

ii) $L_2 \in \mathbf{SPACE}(n^2)$?

iii) $L_1 \setminus L_2 \in \mathbf{PSPACE}$?

Justifique cuidadosamente cada uma das respostas.

Resolução

a) i) **SIM**

Notar que $L_1 \in \mathbf{NSPACE}(n)$ e visto que este é fechado por complemento, então $\overline{L_1} \in \mathbf{NSPACE}(n)$.

Em seguida, aplicando o *Teorema de Savitch*, temos que $\mathbf{NSPACE}(n) \subseteq \mathbf{SPACE}(n^2)$, logo $\overline{L_1} \in \mathbf{SPACE}(n^2)$.

ii) **NÃO**

Seja $f : \Sigma^* \to \Sigma^*$ uma função total e computável por uma máquina de Turing F tal que $time_F(n) = O(n^k)$ e $x \in L_2$ sse $f(x) \in \overline{L_1}$.

Sabe-se que $L_1 \in \mathbf{NSPACE}(n)$, e por conseguinte $\overline{L_1} \in \mathbf{NSPACE}(n)$, temos que $\overline{L_1}$ pode ser reconhecida por uma máquina de Turing não deterministica em espaço $\mathcal{O}(n)$. Como a função f computa a redução em tempo polinomial e tem complexidade temporal $\mathcal{O}(n^k)$, o tamanho da palavra $f(x)$ pode ser, no pior caso, $\mathcal{O}(n^k)$, onde n é o tamanho da entrada x.

Tira-se que $L_2 \subseteq \mathbf{NSPACE}(n^k)$, conclui-se novamente pela aplicação do *Teorema de Savitch* que $L_2 \in \mathbf{SPACE}(n^{2k})$, não garantindo que $k = 1$.

iii) **SIM**

Como $L_1 \in \mathbf{NSPACE}(n)$ então $L_1 \in \mathbf{PSPACE}(n)$ (pode-se simular uma máquina de Turing não determinística numa máquina de Turing determinística equivalente no mesmo espaço, embora com complexidade temporal mais elevada).

De seguida, da relação polinomial, $L_2 \leq_p \overline{L_1}$, segue que $L_2 \in \mathbf{PSPACE}(n)$ (podemos decidir L_2 em espaço polinomial, aplicando a redução em espaço polinomial e depois decidir $\overline{L_1} \in \mathbf{PSPACE}$).

Assim, e uma vez que **PSPACE** é fechado por diferença, temos que $L_1 \setminus L_2 \in \mathbf{PSPACE}$.

MAP30–4A.2 (2024/2025)

a) (3.0 valores) Seja Σ um alfabeto e $L_1, L_2 \subseteq \Sigma^*$. Sabendo que $L_1 \in$ **NP-difícil** e que $L_1 \leq L_2$ com redução $f : \Sigma^* \to \Sigma^*$ total e computável em tempo quadrático, pode garantir que:

(i) $L_2 \in \mathbf{NTIME}(n^2)$?

(ii) $L_2 \in$ **NP-difícil**?

(iii) $L_2 \in \mathbf{NP}$?

Justifique cuidadosamente cada uma das respostas.

b) (3.0 valores) Dado um número natural $k \in \mathbb{N}_0$, seja k_{un} a sua representação em unário, e k_{bin} a sua representação em binário.

Seja $h : \mathbb{N}_0 \to \mathbb{N}_0$ uma função, e suponha que dispõe de uma máquina de Turing H_1 tal que $\text{space}_{H_1}(n) = O(n)$, cuja execução sobre cada input k_{un} devolve como output $h(k)_{\text{bin}}$.

Mostre que existe uma máquina de Turing H_2 cuja execução sobre cada input k_{bin} devolve como output $h(k)_{\text{un}}$ tal que $\text{space}_{H_2}(n) = O(2^{2^n})$.

Resolução

a) i) **NÃO**

Embora a função de redução f seja computável em tempo quadrático, isso apenas garante que, para toda entrada x, o tamanho de $f(x)$ é polinomialmente relacionado com $|x|$ (no caso, no máximo $\mathcal{O}(|x|^2)$). Contudo, não temos nenhuma garantia de que exista uma máquina de Turing não-determinística que decida L_2 em tempo $\mathcal{O}(n^2)$, pois o tempo de decisão dependerá tanto do tempo de computação da redução quanto da complexidade de decidir L_1 a partir da instância transformada. Noutras palavras, a composição da máquina que decide L_1 (que é **NP-difícil** e pode estar fora de $\mathbf{NTIME}(n^2)$) com a redução de tempo quadrático pode resultar numa complexidade superior a n^2. Portanto, não podemos concluir que $L_2 \in \mathbf{NTIME}(n^2)$.

ii) **SIM**

Sabemos que L_1 é **NP-difícil** e, sendo f uma redução polinomial (no caso, com tempo quadrático) de L_1 para L_2, segue pela propriedade de fechamento da NP-dificuldade sob reduções polinomiais que L_2 também é **NP-difícil**.

iii) **NÃO**

O facto de L_2 ser **NP-difícil** não implica necessariamente que ele pertença a **NP**. Um problema **NP-difícil** pode, em princípio, estar fora de **NP** (por exemplo, ser **NP-completo** ou até ser um problema que é decidível mas que não pertence a **NP** se ele requer mais que tempo não-determinístico polinomial para ser decidido). Assim, sem informações adicionais que indiquem que L_2 tem uma

máquina não-determinística de tempo polinomial, não podemos garantir que $L_2 \in \mathbf{NP}$.

b) Recorde-se que é possível construir uma máquina de Turing T que traduz números em binário para unário (e vice-versa), i.e., a execução de T sobre k_{bin} devolve k_{un}, tal que:

$$\text{space}_T(n) = O(2^n)$$

pois em geral $|k_{\text{un}}| = O(2^n)$ para $|k_{\text{bin}}| = n$. Considere-se agora a máquina H_2 definida por:

Recebe input x
Executa T sobre x, obtendo k_{un}
Executa H_1 sobre k_{un}, obtendo $h(k)_{\text{bin}}$
Executa T sobre $h(k)_{\text{bin}}$, obtendo $h(k)_{\text{un}}$

A máquina H_2 está correta pois T e H_1 terminam sempre, e:

$$H_2(k_{\text{bin}}) = T(H_1(T(k_{\text{bin}}))) = T(h(k)_{\text{bin}}) = h(k)_{\text{un}}$$

Quanto à complexidade espacial:

$$\text{space}_{H_2}(n) = \text{space}_T(n) + \text{space}_{H_1}(2^n) + \text{space}_T(\log h(k))$$

Sabendo que:

- $\text{space}_T(n) = O(2^n)$
- $\text{space}_{H_1}(2^n) = O(2^n)$ (pois H_1 é linear)
- $\text{space}_T(\log h(k)) = O(h(k)) = O(2^{2^n})$

Logo,

$$\text{space}_{H_2}(n) = O(2^n) + O(2^n) + O(2^{2^n}) = O(2^{2^n})$$

MAP30–4B.1 (2024/2025)

a) (3.0 valores) Seja Σ um alfabeto e $L_1, L_2 \subseteq \Sigma^*$. Sabendo que $L_2 \in \mathbf{TIME}(g(n))$, com $n \leq g(n)$, e que $L_1 \leq L_2$ com redução $f : \Sigma^* \to \Sigma^*$ total e computável em tempo linear, pode garantir que:

(i) $L_1 \in \mathbf{TIME}(n)$?

(ii) $L_1 \in \mathbf{TIME}(g(n))$?

(iii) $L_1 \in \mathbf{PSPACE}$?

Justifique cuidadosamente cada uma das respostas.

Resolução

a) i) **NÃO.**

Sabemos que a função de redução f é computável em tempo linear, ou seja, para toda entrada x, temos

$$|f(x)| = O(|x|).$$

Entretanto, para concluir que $L_1 \in \mathbf{TIME}(n)$ seria necessário que a verificação da pertença de $f(x)$ em L_2 ocorresse em tempo linear em $|x|$. Mas como $L_2 \in \mathbf{TIME}(g(n))$ e, em geral, $g(n)$ pode ser maior que n (desde que $n \leq g(n)$), não se pode concluir que $L_1 \in \mathbf{TIME}(n)$.

ii) **SIM.**

Dado que f é uma redução total e computável em tempo linear, para cada $x \in \Sigma^*$ temos que:

$$x \in L_1 \iff f(x) \in L_2.$$

Como f é linear, existe uma constante $c > 0$ tal que $|f(x)| \leq c \cdot |x|$. Além disso, como $L_2 \in \mathbf{TIME}(g(n))$, existe uma máquina de Turing que decide L_2 em tempo $O(g(n))$. Ao compor essa decisão com a redução, obtemos uma máquina que decide L_1 em tempo

$$O(g(c \cdot |x|)) = O(g(|x|)),$$

pois g é, em particular, uma função que satisfaz $n \leq g(n)$. Assim, $L_1 \in \mathbf{TIME}(g(n))$.

iii) **SIM.**

Sabemos que $\mathbf{TIME}(g(n)) \subseteq \mathbf{PSPACE}$ para todo $g(n)$ (pois, por exemplo, uma máquina que roda em tempo polinomial também usa espaço polinomial, e de forma mais geral, qualquer linguagem decidida em tempo $g(n)$ pode ser decidida em espaço $O(g(n))$). Como, pelo item ii), já temos que $L_1 \in \mathbf{TIME}(g(n))$, conclui-se que $L_1 \in \mathbf{PSPACE}$.

MAP30–4D.2 (2022/2023)

a) (3.0 valores) Sabendo que $L_1 \in$ **NP-completa** e que $L_1 \leq_P L_2$, pode garantir que:

(i) $L_2 \in$ **NP-completa**?

(ii) $L_2 \in$ **NP-difícil**?

(iii) $L_2 \in \mathbf{P}$?

Justifique cuidadosamente cada uma das respostas.

Resolução

a) i) **NÃO**

Sabe-se que $L_1 \in$ **NP-completa**, ou seja, $L_1 \in \mathbf{NP}$ e $\forall A \in \mathbf{NP}$, $A \leq_p L_1$.

Além disso, é dado que $L_1 \leq_p L_2$. No entanto, não se conhece se $L_2 \in \mathbf{NP}$.

Como a completude em **NP** requer que a linguagem pertença a **NP** *e* seja **NP-difícil**, não se pode garantir que $L_2 \in \mathbf{NP}$, e logo não se pode garantir que $L_2 \in$ **NP-completa**.

ii) **SIM**

Sabemos que $L_1 \in$ **NP-completa**, portanto $L_1 \in \mathbf{NP}$ e $A \leq_p L_1$ para toda linguagem $A \in \mathbf{NP}$.

Como $L_1 \leq_p L_2$ por hipótese, e uma vez que as reduções polinomiais são transitivas, então temos:

$$\forall A \in \mathbf{NP}, \quad A \leq_p L_1 \leq_p L_2,$$

donde se conclui que $A \leq_p L_2$ para toda linguagem $A \in \mathbf{NP}$.

Logo, L_2 é **NP-difícil**, i.e., $L_2 \in$ **NP-difícil**.

iii) **NÃO**

Sabe-se que $L_1 \in$ **NP-completa** e $L_1 \leq_p L_2$, e pela alínea (ii), conclui-se que $L_2 \in$ **NP-difícil**.

Suponha, por absurdo, que $L_2 \in \mathbf{P}$. Então, existiria uma linguagem **NP-difícil** pertencente a **P**, o que implicaria:

$$\mathbf{NP} \subseteq \mathbf{P} \Rightarrow \mathbf{NP} = \mathbf{P}$$

(o que é um dos problemas em aberto sem solução conhecida).

Assim, não se pode garantir que $L_2 \in \mathbf{P}$, sob pena de assumir implicitamente que $\mathbf{P} = \mathbf{NP}$.

MAP30–4D.1 (2022/2023)

a) (3.0 valores) Sabendo que L_1 é **PSPACE**-completa e que $L_2 \leq_p \overline{L_1}$, pode garantir que:

(i) $\overline{L_1}$ é **PSPACE**-completa?

(ii) L_2 é **PSPACE**-completa?

(iii) $L_1 \setminus L_2 \in$ **PSPACE**?

Justifique cuidadosamente cada uma das respostas.

Resolução

a) (i) **SIM**

Sabemos que L_1 é **PSPACE**-completa e que a classe **PSPACE** é fechada por complemento (pelo Teorema de Savitch, por exemplo). Assim, $\overline{L_1} \in$ **PSPACE** e, além disso, toda linguagem em **PSPACE** se reduz a L_1, portanto também se reduz a $\overline{L_1}$. Concluímos, pois, que $\overline{L_1}$ é **PSPACE**-difícil e pertence a **PSPACE**, isto é, é **PSPACE**-completa.

(ii) **NÃO**

Temos que $L_2 \leq_p \overline{L_1}$, ou seja, existe uma redução polinomial que leva qualquer instância de L_2 a uma instância de $\overline{L_1}$. Entretanto, para afirmar que L_2 é **PSPACE**-completa seria necessário, adicionalmente, que $L_2 \in$ **PSPACE** e que toda linguagem em **PSPACE** se reduzisse a L_2. A redução dada (de L_2 para $\overline{L_1}$) apenas assegura que L_2 é, no máximo, "não mais difícil"que $\overline{L_1}$, ou seja, que L_2 é **PSPACE**-difícil se $\overline{L_1}$ fosse reduzida de forma inversa. Sem garantia de que cada linguagem em **PSPACE** se reduza a L_2, não se pode concluir que L_2 é **PSPACE**-completa.

(iii) **SIM**

Sabemos que $L_1 \in$ **PSPACE** (pois L_1 é **PSPACE**-completa) e que $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$.

Dada a redução $L_2 \leq_p \overline{L_1}$, temos que $L_2 \in$ **PSPACE**, pois uma linguagem redutível a outra em **PSPACE** também pertence a **PSPACE** (já que **PSPACE** é fechada sob reduções polinomiais). Como $\overline{L_1} \in$ **PSPACE** (pela alínea i), concluímos que $L_2 \in$ **PSPACE**.

Portanto, $\overline{L_2} \in$ **PSPACE** (pois **PSPACE** é fechada sob complementação). Como ambas as linguagens L_1 e $\overline{L_2}$ pertencem a **PSPACE**, e **PSPACE** é fechada sob interseção, concluímos que $L_1 \setminus L_2 = L_1 \cap \overline{L_2} \in$ **PSPACE**.

MAP30–4D.1 (2022/2023)

a) (3.0 valores) Sabendo que $L_1 \in$ **NP-completa** e que $\overline{L_2} \leq_P \overline{L_1}$, pode garantir que:

(i) $L_2 \in$ **NP**?

(ii) $L_2 \in$ **NP**-difícil?

(iii) $L_1 \setminus L_2 \in$ **EXPTIME**?

Justifique cuidadosamente cada uma das respostas.

Resolução

a) (i) **SIM.**

Sabemos que $L_1 \in$ **NP**-completa, ou seja, $L_1 \in$ **NP** e para todo $A \in$ **NP** temos $A \leq_P L_1$. Dada a hipótese $\overline{L_2} \leq_P \overline{L_1}$, pelo fechamento por complemento das reduções polinomiais, obtemos

$$L_2 \leq_P L_1.$$

Como $L_1 \in$ **NP**, podemos construir um verificador não-determinístico para L_2 da seguinte forma:

i. Para uma entrada x, calcular $f(x)$ em tempo polinomial, onde f é a função de redução
ii. Utilizar o verificador de L_1 (que funciona em tempo polinomial) para verificar $f(x)$

Como ambos os passos executam em tempo polinomial, o verificador completo para L_2 também executa em tempo polinomial não-determinístico. Portanto, $L_2 \in$ **NP**.

(ii) **NÃO.**

Para que uma linguagem seja **NP**-difícil, é necessário que toda a linguagem em **NP** se reduza a ela em tempo polinomial. A hipótese $\overline{L_2} \leq_P \overline{L_1}$ (equivalente a $L_2 \leq_P L_1$) apenas indica que L_2 é, no máximo, "mais fácil"que L_1. Não se tem, portanto, evidência de que toda linguagem em **NP** se reduz a L_2. Assim, não se pode concluir que L_2 é **NP**-difícil.

(iii) **SIM.**

Note que

$$L_1 \setminus L_2 = L_1 \cap \overline{L_2}.$$

Sabemos que $L_1 \in \mathbf{NP}$ (por definição de NP-completude) e que $L_2 \in \mathbf{NP}$ (como demonstrado no item (i)). Consequentemente, ambas as linguagens são decidíveis em tempo exponencial determinístico (isto é, $\mathbf{NP} \subseteq \mathbf{EXPTIME}$). Como a classe **EXPTIME** é fechada por complemento e interseção, temos que $\overline{L_2} \in \mathbf{EXPTIME}$ e $L_1 \cap \overline{L_2} \in \mathbf{EXPTIME}$. Portanto, conclui-se que

$$L_1 \setminus L_2 \in \mathbf{EXPTIME}.$$

MAP30–4C.1 (2022/2023)

a) (3.0 valores) Sabendo que L_1 é **PSPACE-completa** e que $\overline{L_2} \leq_P L_1$, pode garantir que:

(i) L_2 é **PSPACE**-completa?

(ii) $L_1 \cup L_2 \in \mathbf{PSPACE}$?

(iii) $L_2 \in \mathbf{PSPACE}$?

Justifique cuidadosamente cada uma das respostas.

Resolução

a) (i) **NÃO**

A redução dada é de $\overline{L_2}$ para L_1, ou seja, existe uma função f computável em tempo polinomial tal que para todo x:

$$x \in \overline{L_2} \iff f(x) \in L_1.$$

Como será justificado no item iii), podemos garantir que $L_2 \in \mathbf{PSPACE}$. No entanto, a redução $\overline{L_2} \leq_P L_1$ não garante que toda linguagem em **PSPACE** se reduza (em tempo polinomial) a L_2. Sem a garantia de que L_2 é **PSPACE**-difícil, não se pode garantir que L_2 seja **PSPACE**-completa.

(ii) **SIM**

Sabemos que $L_1 \in \mathbf{PSPACE}$ por hipótese. Como justificado no item (iii), a condição $\overline{L_2} \leq_P L_1$ implica que $L_2 \in \mathbf{PSPACE}$. Como a classe **PSPACE** é fechada sob a operação de união, e ambas as linguagens L_1 e L_2 pertencem a **PSPACE**, podemos garantir que $L_1 \cup L_2 \in \mathbf{PSPACE}$.

(iii) **SIM**

Temos a informação de que $\overline{L_2} \leq_P L_1$ e que $L_1 \in \mathbf{PSPACE}$. A redução $\overline{L_2} \leq_P L_1$ significa que existe um algoritmo que decide $\overline{L_2}$ usando um oráculo para L_1, e este algoritmo corre em tempo polinomial. Mais especificamente, para decidir se $x \in \overline{L_2}$, calculamos $y = f(x)$ (onde f é a função de redução polinomial) e depois verificamos se $y \in L_1$. O cálculo de $f(x)$ leva tempo polinomial, logo espaço polinomial. A verificação de $y \in L_1$ leva espaço polinomial em $|y|$, que é polinomial em $|x|$. Portanto, o processo total para decidir $\overline{L_2}$ usa espaço polinomial. Assim, $\overline{L_2} \in \mathbf{PSPACE}$. Sabendo que a classe **PSPACE** é fechada por complemento, concluímos que $L_2 = \overline{\overline{L_2}} \in \mathbf{PSPACE}$.

Recurso (2022/2023)

a) (4.5 valores) Dada uma classe de linguagens $\mathcal{C}$ define-se a classe

$$r(\mathcal{C}) = \{L : \text{existe } A \in \mathcal{C} \text{ tal que } L \leq_P A\}.$$

Justifique cuidadosamente a resposta a cada uma das seguintes questões:

(i) Mostre que uma linguagem $\mathcal{C}$-completa é necessariamente $r(\mathcal{C})$-completa. Será que uma linguagem $r(\mathcal{C})$-completa é necessariamente $\mathcal{C}$-completa?

(ii) Mostre que se B é uma linguagem **NP**-completa então $r(\{B\}) = \mathbf{NP}$.

(iii) Será que $r(\mathbf{TIME}(n)) = \mathbf{P}$?

b) (1.5 valores) É um problema em aberto saber se $\mathbf{TIME}(n) = \mathbf{SPACE}(n)$. Demonstre, no entanto, que $\mathbf{TIME}(n) \subsetneq \mathbf{SPACE}(n^2)$.

Resolução

a) Dada uma classe de linguagens $\mathcal{C}$ define-se a classe

$$r(\mathcal{C}) = \{L \mid \exists A \in \mathcal{C} \text{ tal que } L \leq_P A\}.$$

(i) Se A é $\mathcal{C}$-completa, então, por definição, $A \in \mathcal{C}$ e para toda linguagem $B \in \mathcal{C}$ temos $B \leq_P A$. Seja $L \in r(\mathcal{C})$. Por definição, existe $C \in \mathcal{C}$ tal que $L \leq_P C$. Mas, como $C \leq_P A$ (pois A é $\mathcal{C}$-completa) e as reduções polinomiais são transitivas, segue que

$$L \leq_P C \leq_P A,$$

isto é, $L \leq_P A$. Portanto, toda linguagem em $r(\mathcal{C})$ se reduz a A, o que mostra que A é, além de pertencer a $\mathcal{C}$, $r(\mathcal{C})$-difícil. Como $A \in \mathcal{C} \subseteq r(\mathcal{C})$ (pois toda linguagem em $\mathcal{C}$ é, trivialmente, reduzível a si mesma), concluímos que A é $\mathcal{C}$-completa $\Rightarrow$ A é $r(\mathcal{C})$-completa.

Porém, se A é $r(\mathcal{C})$-completa, isto significa que $A \in r(\mathcal{C})$, isto é, existe alguma linguagem $C \in \mathcal{C}$ tal que $A \leq_P C$, e que para toda linguagem $B \in r(\mathcal{C})$, temos $B \leq_P A$. Contudo, não se garante, a partir desta hipótese, que $A \in \mathcal{C}$ (poderia ser, por exemplo, uma linguagem "mais difícil"que não pertença a $\mathcal{C}$, mas que ainda é capaz de "capturar"todas as linguagens de $\mathcal{C}$ via redução). Portanto, uma linguagem $r(\mathcal{C})$-completa não é, em geral, necessariamente $\mathcal{C}$-completa.

(ii) Observe que, por definição,

$$r(\{B\}) = \{L \mid L \leq_P B\}.$$

Como B é **NP**-completa, temos que:

- Todo problema $L \in \mathbf{NP}$ é polinomialmente redutível a B, isto é, $L \leq_P B$. Assim, $L \in r(\{B\})$.
- Reciprocamente, se $L \in r(\{B\})$, isto é, $L \leq_P B$, e sabendo que $B \in \mathbf{NP}$ e as reduções polinomiais preservam a decidibilidade em NP, segue que $L \in \mathbf{NP}$.

Portanto, temos a equivalência:

$$r(\{B\}) = \{L \mid L \leq_P B\} = \mathbf{NP}.$$

(iii) Por definição,

$$r(\mathbf{TIME}(n)) = \{L \mid \exists A \in \mathbf{TIME}(n) \text{ tal que } L \leq_P A\}.$$

Note que, para qualquer $A \in \mathbf{TIME}(n)$, temos $A \in \mathbf{P}$, já que $\mathbf{TIME}(n) \subseteq \mathbf{P}$. Se $L \in r(\mathbf{TIME}(n))$, então existe um $A \in \mathbf{TIME}(n)$ tal que $L \leq_P A$. Como as reduções polinomiais preservam a decidibilidade em tempo polinomial (isto é, se $A \in \mathbf{P}$, então $L \in \mathbf{P}$), conclui-se que

$$r(\mathbf{TIME}(n)) \subseteq \mathbf{P}.$$

Contudo, a inclusão inversa, isto é, se $L \in \mathbf{P}$ então $L \in r(\mathbf{TIME}(n))$, não é necessariamente verdadeira, pois uma linguagem em $\mathbf{P}$ pode não ser redutível, em tempo polinomial, a uma linguagem que pode ser decidida em tempo linear. Assim, em geral, não se tem $r(\mathbf{TIME}(n)) = \mathbf{P}$.

b) Para mostrar que $\mathbf{TIME}(n) \subsetneq \mathbf{SPACE}(n^2)$, vamos demonstrar ambas: inclusão e inclusão estrita.

Inclusão:

Se uma máquina de Turing roda em tempo linear, isto é, em $\mathbf{TIME}(n)$, então ela pode, no máximo, acessar $O(n)$ células na fita durante a sua execução. Portanto, temos

$$\mathbf{TIME}(n) \subseteq \mathbf{SPACE}(n).$$

Como é trivial que $\mathbf{SPACE}(n) \subseteq \mathbf{SPACE}(n^2)$, conclui-se que

$$\mathbf{TIME}(n) \subseteq \mathbf{SPACE}(n^2).$$

Inclusão estrita:

Para mostrar que a inclusão é estrita, aplicamos o **Teorema da Hierarquia de Tempo**, que estabelece que se $f(n)$ e $g(n)$ são funções de tempo tais que

$$f(n) \log f(n) = o(g(n)),$$

então

$$\mathbf{TIME}(f(n)) \subsetneq \mathbf{TIME}(g(n)).$$

Tomando $f(n) = n$ e $g(n) = n^2$, temos que

$$n \log n = o(n^2),$$

de modo que:

$$\mathbf{TIME}(n) \subsetneq \mathbf{TIME}(n^2).$$

Por outro lado, é sabido que

$$\mathbf{TIME}(n^2) \subseteq \mathbf{SPACE}(n^2),$$

pois uma máquina que decide uma linguagem em tempo n^2 utiliza, no máximo, n^2 células da fita.

Portanto, existe ao menos uma linguagem que pode ser decidida em tempo n^2 (e, consequentemente, em espaço n^2) que não pode ser decidida em tempo linear. Concluímos que a inclusão

$$\mathbf{TIME}(n) \subseteq \mathbf{SPACE}(n^2)$$

é, de fato, estrita, isto é,

$$\mathbf{TIME}(n) \subsetneq \mathbf{SPACE}(n^2).$$

Perguntas Extra

1) Para mostrar que $\mathbf{NTIME}(n) \subsetneq \mathbf{PSPACE}$, vamos demonstrar ambas: inclusão e inclusão estrita.

   Inclusão:

   Se uma máquina de Turing não determinística roda em tempo linear, isto é, em $\mathbf{NTIME}(n)$, então cada ramo computacional pode, no máximo, acessar $O(n)$ células na fita durante a sua execução. Portanto, temos

   $$\mathbf{NTIME}(n) \subseteq \mathbf{NSPACE}(n).$$

   Pelo Teorema de Savitch, sabemos que

   $$\mathbf{NSPACE}(n) \subseteq \mathbf{DSPACE}(n^2).$$

Como n^2 é um polinómio, segue-se que:

$$\mathbf{DSPACE}(n^2) \subseteq \mathbf{PSPACE}.$$

Assim, conclui-se que

$$\mathbf{NTIME}(n) \subseteq \mathbf{PSPACE}.$$

Inclusão estrita:

Para mostrar que a inclusão é estrita, utilizamos o **Teorema da Hierarquia de Espaço**, que estabelece que se $f(n)$ e $g(n)$ são funções construtíveis no espaço tais que $f(n) \in o(g(n))$, então

$$\mathbf{DSPACE}(f(n)) \subsetneq \mathbf{DSPACE}(g(n)).$$

Tomando $f(n) = n$ e $g(n) = n^2$, temos que

$$n = o(n^2),$$

de modo que:

$$\mathbf{DSPACE}(n) \subsetneq \mathbf{DSPACE}(n^2).$$

Por outro lado, sabemos que

$$\mathbf{NTIME}(n) \subseteq \mathbf{NSPACE}(n) \subseteq \mathbf{DSPACE}(n^2),$$

e também que

$$\mathbf{NTIME}(n) \subseteq \mathbf{DSPACE}(n) \subsetneq \mathbf{DSPACE}(n^2) \subseteq \mathbf{PSPACE}.$$

Portanto, existe ao menos uma linguagem que pode ser decidida em espaço n^2 (e, consequentemente, em **PSPACE**) que não pode ser decidida em tempo não determinístico linear. Concluímos que a inclusão

$$\mathbf{NTIME}(n) \subseteq \mathbf{PSPACE}$$

é, de fato, estrita, isto é,

$$\mathbf{NTIME}(n) \subsetneq \mathbf{PSPACE}.$$

2) Seja um alfabeto Σ sem o símbolo \$ e considere linguagens $A, B, C, L \subseteq \Sigma^*$ tais que:

- $A \leq_P B \cap C$,
- $B \leq_P L$,
- $C \leq_P \overline{L}$.

Mostre, justificando, que $A \leq_P \{w_1\$w_2 : w_1 \in L \text{ e } w_2 \in \Sigma^* \setminus L\}$.

Resolução:

- Como $A \leq_P B \cap C$, existe uma função $f : \Sigma^* \to \Sigma^*$ total e computável por uma máquina de Turing F com $time_F(n) = O(n^a)$ tal que $x \in A$ sse $f(x) \in B \cap C$.

- Como $B \leq_P L$, existe uma função $g : \Sigma^* \to \Sigma^*$ total e computável por uma máquina de Turing G com $time_G(n) = O(n^b)$ tal que $x \in B$ sse $g(x) \in L$.

- Como $C \leq_P \overline{L}$, existe uma função $h : \Sigma^* \to \Sigma^*$ total e computável por uma máquina de Turing H com $time_H(n) = O(n^c)$ tal que $x \in C$ sse $h(x) \in \overline{L}$.

Definimos a função $k : \Sigma^* \to \Sigma^*$ por $k(x) = g(f(x))\$h(f(x))$. Esta função é total e computável em tempo polinomial, pois f, g e h são computáveis em tempo polinomial e a concatenação com o símbolo \$ é feita em tempo linear. Uma máquina de Turing K computa k com tempo:

$$\begin{aligned} time_K(n) &= O(time_F(n) + time_G(n + time_F(n)) + time_H(n + time_F(n))) \\ &= O(n^a + (n + n^a)^b + (n + n^a)^c) \quad \text{que é um polinómio.} \end{aligned}$$

Além disso, para qualquer $x \in \Sigma^*$, temos:

$$\begin{aligned} x \in A &\iff f(x) \in B \cap C \\ &\iff f(x) \in B \text{ e } f(x) \in C \\ &\iff g(f(x)) \in L \text{ e } h(f(x)) \in \overline{L} \\ &\iff g(f(x)) \in L \text{ e } h(f(x)) \in \Sigma^* \setminus L \\ &\iff k(x) = g(f(x))\$h(f(x)) \in \{w_1\$w_2 : w_1 \in L \text{ e } w_2 \in \Sigma^* \setminus L\} \end{aligned}$$

Concluímos que $A \leq_P \{w_1\$w_2 : w_1 \in L \text{ e } w_2 \in \Sigma^* \setminus L\}$.