5 Gramáticas Geradoras

5.6 Exercícios

5.6.1. Determine gramáticas independentes do contexto geradoras das linguagens a seguir definidas, argumentando de modo semi-formal sobre a correcção das soluções encontradas:

(a) a^*;

Solução. $S \to aS \mid \varepsilon$.

(b) a^+;

Solução. $S \to aS \mid a$.

(c) $\{a^n b^n : n \geq 0\}$;

Solução. $S \to aSb \mid \varepsilon$.

(d) $\{a^n b^n : n > 0\}$;

Solução. $S \to aSb \mid a$.

(e) $\{a^n b^{n+1} : n \geq 0\}$;

Solução. $S \to aSb \mid b$.

(f) $\{a^m b^n : 0 \leq m \leq n\}$;

Resolução. Uma vez que $m \leq n$, $n - m \geq 0$. Logo, $a^m b^n = a^m b^{n-m} b^m$. Assim, uma gramática geradora é definida pelas produções:

$$S \to aSb \mid X \qquad\qquad X \to bX \mid \varepsilon.$$

De forma equivalente podemos escrever que $a^m b^n = a^m b^m b^{n-m}$, com $m \geq 0$ e $n - m \geq 0$. Assim, uma gramática geradora é definida pelas produções:

$$S \to XY \qquad X \to aXb \mid \varepsilon \qquad Y \to bY \mid \varepsilon.$$

(g) $\{a^m b^n : m \geq n \geq 0\}$;

Resolução. Uma vez que $m \geq n$, $m - n \geq 0$. Logo, $a^m b^n = a^{m-n} a^n b^n$. Assim, uma gramática geradora é definida pelas produções:

$$S \to XY \qquad X \to aX \mid \varepsilon \qquad Y \to aYb \mid \varepsilon.$$

De forma equivalente podemos escrever que $a^m b^n = a^n b^{m-n} b^n$, com $m \geq 0$ e $m - n \geq 0$. Assim, uma outra gramática geradora é definida pelas produções:

$$S \to aSb \mid X \qquad X \to aX \mid \varepsilon.$$

(h) $\{a^m b^n : 0 \leq m \leq n + 3,\ n \geq 0\}$;

Resolução. Nesta linguagem as palavras têm no máximo mais três a´s do que b´s. Assim, esta linguagem pode ser vista como a união de quatro linguagens cujas palavras são de uma das seguintes formas:

(I) $a^m b^n$ com $0 \leq m \leq n$;
(II) $a^{n+1} b^n$ com $n \geq 0$;
(III) $a^{n+2} b^n$ com $n \geq 0$;
(IV) $a^{n+3} b^n$ com $n \geq 0$.

É simples especificar uma GIC para cada uma delas e, em seguida, realizar a união e simplificar. Por exemplo:

$$S \to aSb \mid A \mid B \qquad B \to bB \mid \varepsilon \qquad A \to a \mid aa \mid aaa.$$

(i) $\{a^n b^{2n} : n \geq 0\}$;

Solução. $S \to aSbb \mid \varepsilon$.

(j) $\{a^m b^n : m \leq n \leq 2m,\ m \geq 0\}$;

Solução. $S \to aSb \mid aSbb \mid \varepsilon$.

(k) $\{a^m b^n : m < n < 2m,\ n \geq 0\}$;

Solução. $S \to aSb \mid aSbb \mid aabbb$.

(l) $\{a^m b^n : 2n \leq m \leq 3n,\ n \geq 0\}$;

Solução. $S \to aaSb \mid aaaSb \mid \varepsilon$.

(m) $\{a^m b^n : m \neq n,\ m, n \geq 0\}$;

Resolução. Esta linguagem é a união das linguagens $L_1 = \{a^m b^n : m < n,\ m, n \geq 0\}$ e $L_2 = \{a^m b^n : m > n,\ m, n \geq 0\}$.

Uma gramática geradora de L_1 é

$$S \to AY \qquad A \to aA \mid a \qquad Y \to aYb \mid \varepsilon.$$

Uma gramática geradora de L_2 é

$$S \to YB \qquad B \to bB \mid b \qquad Y \to aYb \mid \varepsilon.$$

Assim, uma gramática geradora da linguagem união é

$$S \to AY|YB \qquad A \to aA \mid a \qquad B \to bB \mid b \qquad Y \to aYb \mid \varepsilon.$$

(n) $\{a^m b^n c^p \colon m = n \text{ ou } n = p,\ m, n, p \geq 0\}$;

Resolução. De forma semelhante à alínea anterior, olhando a linguagem como união de duas linguagens, obtemos a gramática:

$$\begin{array}{lll} S \to XC|AW & X \to aXb \mid \varepsilon & C \to cC \mid \varepsilon \\ A \to aA \mid \varepsilon & W \to bWc \mid \varepsilon. & \end{array}$$

(o) $\{a^m b^n c^p \colon m \neq n \text{ ou } n \neq p,\ m, n, p \geq 0\}$;

Resolução. Uma vez mais esta linguagem é a união das duas linguagens cujas palavras são da forma: (i) $a^m b^n c^p$ com $m \neq n$, $m, n, p \geq 0$; (ii) $a^m b^n c^p$ com $n \neq p$, $m, n, p \geq 0$. Assim, basta aplicar a cada uma destas linguagens o método utilizado na alínea anterior.

(p) $\{a^m b^n a^n b^m \colon m, n \geq 0\}$;

Resolução. Começamos por observar que os símbolos das palavras desta linguagem se podem agrupar na forma $a^m \boxed{b^n a^n} b^m$. Assim, basta começar por gerar a parte exterior, finalizando com a parte interior:

$$S \to aSb \mid X \qquad X \to bXa \mid \varepsilon.$$

(q) $\{a^m b^m a^n b^n \colon m, n \geq 0\}$;

Resolução. Podemos agrupar os símbolos das palavras desta linguagem na forma $\boxed{a^m b^m}\boxed{a^n b^n}$. Basta então gerar as duas partes e, em seguida, concatená-las:

$$S \to XY \qquad X \to aXb \mid \varepsilon \qquad Y \to aYb \mid \varepsilon.$$

Uma vez que as variáveis X e Y são independentes e geram a mesma linguagem, podemos simplificar a gramática anterior:

$$S \to XX \qquad X \to aXb \mid \varepsilon,$$

ou ainda,

$$S \to aSb \mid aSbaSb \mid \varepsilon.$$

(r) $\{a^m b^n c^p \colon m = n \text{ ou } n \leq p,\ m, n, p \geq 0\}$;

Resolução. A linguagem L é a união das linguagens $L_1 = \{a^m b^m c^p \colon m, p \geq 0\}$ e $L_2 = \{a^m b^n c^p \colon m, n, p \geq 0 \wedge n \leq p\}$.

A linguagem L_1 é a concatenação das linguagens $L_3 = \{a^m b^m \colon m \geq 0\}$ e $L_4 = \{c^p \colon p \geq 0\}$.

A linguagem L_2 é a concatenação das linguagens $L_5 = \{a^m \colon m \geq 0\}$ e $L_6 = \{b^n c^p \colon n, p \geq 0 \wedge n \leq p\}$.

Para obter uma gramática geardora de L, basta construir gramáticas para estas 6 linguagens e, em seguida, realizar as respectivas operações de concatenação e união.

(s) $\{a^m b^n c^p \colon |m - n| = p,\ m, n, p \geq 0\}$;

Resolução. Notando que $|m - n| = p$ se e só se $m = p + n$ ou $n = p + m$, basta construir gramáticas para as linguagens $L_1 = \{a^p a^n b^n c^p \colon n, p \geq 0\}$ e $L_2 = \{a^m b^m b^p c^p \colon m, p \geq 0\}$ e, em seguida, realizar a sua união.

5.6.2. Considere a linguagem $L = \{a^j b^{i+j} c^i \colon i, j \in \mathbb{N}_0\}$.

(a) Use o lema da bombagem para mostrar que L não é uma linguagem regular.

Resolução. Admitamos que L é uma linguagem regular e seja $n > 0$ o inteiro especificado no lema da bombagem para linguagens regulares.

Fixando $i = j = n$ na definição de L, vamos considerar a palavra $w = a^n b^{2n} c^n$ com tamanho $|w| = 4n \geq n$. Pelo lema, existe uma partição $w = xyz$ que satisfaz: (i) $y \neq \varepsilon$; (ii) $|xy| \leq n$; (iii) $\forall k \geq 0,\ xy^k z \in L$.

Por (ii), x e y só podem ser sequências de a´s. Assim, $x = a^p$, $y = a^q$ e $z = a^r b^{2n} c^n$, com $p + q \leq n$, $q > 0$ (por (i)) e $p + q + r = n$ (para que xyz seja uma partição de w).

Por (iii), para $k = 0$, $xy^0 z$ pertence à linguagem L. Mas $xy^0 z = a^p a^r b^{2n} c^n = a^{p+r} b^{2n} c^n$. Uma vez que $p + r + n < 2n$, já que $p + r < p + q + r = n$, concluímos que a palavra $xy^0 z$ não pertence à linguagem, o que é uma contradição.

Concluímos que L não satisfaz o lema da bombagem, pelo que não é regular.

(b) Especifique uma gramática independente do contexto, G, geradora de L.

Resolução. A gramática $G = (\{S, X, Y\}, \{a, b, c\}, P, S)$, com produções P dadas por

$$S \to XY \qquad X \to aXb \mid \varepsilon \qquad Y \to bYc \mid \varepsilon$$

é geradora da linguagem L.

(c) Prove que $L = \mathcal{L}(G)$.

Resolução. Começamos por observar que $L = L_1L_2$ com $L_1 = \{a^jb^j \mid j \in \mathbb{N}_0\}$ e $L_2 = \{b^ic^i \colon i \in \mathbb{N}_0\}$.

Uma vez que a única derivação num só passo a partir do axioma S da gramática G é $S \Rightarrow XY$, para concluirmos que $L = \mathcal{L}(G)$, é suficiente provar que:

(a) para qualquer $w \in \{a, b, c\}^\star$, $X \overset{\star}{\Rightarrow} w$ se e só se $w \in L_1$;

(b) para qualquer $w \in \{a, b, c\}^\star$, $Y \overset{\star}{\Rightarrow} w$ se e só se $w \in L_2$.

Uma vez que as demonstrações são semelhantes, vamos apenas provar (i).

Seja então $w = a^jb^j \in L_1$ para algum $j \geq 0$. Se $j = 0$ então $w = \varepsilon$ e $X \Rightarrow w$ num único passo aplicando a produção $X \to \varepsilon$. Se $j > 0$ então $X \overset{\star}{\Rightarrow} a^jXb^j$, aplicando j vezes a produção $X \to aXb$ (a demonstração formal deste facto faz-se por indução sobre j). Em seguida, ao aplicar a produção $X \to \varepsilon$, obtemos $X \overset{\star}{\Rightarrow} a^jb^j = w$. Em qualquer dos casos, concluímos que $w \in \mathcal{L}(G)$, pelo que $L_1 \subseteq L(G)$.

Reciprocamente, a derivação num só passo de uma palavra $w \in \{a, b, c\}^\star$ a partir de X só pode ser obtida aplicando $X \to \varepsilon$, pelo que $X \Rightarrow \varepsilon \in L_1$. (De facto, ao aplicar a produção alternativa $X \to aXb$ obtemos a sentença aXb, a qual não é uma palavra.)

Por outro lado, uma derivação de uma palavra em $j + 1$ passos, com $j \geq 1$, só pode ser obtida pela aplicação sequencial da produção $X \to aXb$, j vezes, seguida da aplicação da produção $X \to \varepsilon$. (De facto, a aplicação antecipada da produção $X \to \varepsilon$ não permite continuar a derivação pois a variável X desaparece da sentença que se está a derivar).

Assim, ao aplicar j vezes a produção $X \to aXb$ obtemos a derivação $X \overset{\star}{\Rightarrow} a^jXb^j$. Em seguida, aplicando a produção $X \to \varepsilon$ obtemos a derivação $X \overset{\star}{\Rightarrow} a^jb^j \in L_1$. Concluímos que qualquer que seja o número de passos na derivação de uma palavra usando a gramática G essa palavra pertence a L_1, pelo que $\mathcal{L}(G) \subseteq L_1$.

5.6.3. Considere a gramática ambígua

$$G = (\{S\}, \{a, b\}, \{S \to aS \mid aSbS \mid \varepsilon\}, S)\,.$$

(a) Para a palavra aab determine:

(a) duas árvores de derivação distintas;

(b) duas derivações à esquerda distintas;

(c) duas derivações à direita distintas.

Resolução.

(a)

(b) Uma vez que a derivação da palavra aab tem duas árvores de derivação distintas, esta palavra tem também duas derivações à esquerda distintas:

- $S \Rightarrow aS \Rightarrow aaSbS \Rightarrow aabS \Rightarrow aab$
- $S \Rightarrow aSbS \Rightarrow aaSbS \Rightarrow aabS \Rightarrow aab$

(c) Pela mesma razão, tem também duas derivações à direita distintas:

- $S \Rightarrow aS \Rightarrow aaSbS \Rightarrow aaSb \Rightarrow aab$
- $S \Rightarrow aSbS \Rightarrow aSb \Rightarrow aaSb \Rightarrow aab$

(b) Construa uma gramática não ambígua equivalente.

Resolução. Por factorização à esquerda da gramática obtemos, neste caso, uma gramática equivalente não ambígua com produções:

$$S \to aSX \mid \varepsilon \qquad\qquad X \to bS \mid \varepsilon$$

(Para mostrar que esta gramática não é ambígua é preciso argumentar que as possíveis derivações associadas a cada palavra gerada pela gramática correspondem a uma única árvore.)

5.6.4. Construa uma gramática independente do contexto para gerar todas as expressões regulares sobre o alfabeto $\{a, b\}$.

Resolução. Denotamos por e a expressão regular ε, para não a confundir com a palavra vazia nas produções. Uma gramática que gera todas as expressões regulares é:

$$\begin{aligned} &S \to \emptyset \mid e \\ &S \to a, \text{ para } a \in \Sigma \\ &S \to (S + S) \mid (SS) \mid (S^*). \end{aligned}$$

5.6.5. Determine se cada uma das linguagens seguintes é regular ou se é não regular mas independente do contexto. Justifique as respostas.

(a) $L_1 = \{(11)^m(00)^n \colon m, n \geq 0\}$;

Resolução. É regular. Uma expressão regular para L_1 é $(11)^*(00)^*$.

(b) $L_2 = \{(11)^n(00)^n \colon n \geq 0\}$;

Resolução. Não é regular. Consideremos o homomorfismo $h \colon \{a, b\}^\star \to \{0, 1\}^\star$ definido por $h(a) = 11$, $h(b) = 00$. Uma vez que $h^{-1}(L_2) = \{a^n b^n \colon n \geq 0\}$ é não regular, L_2 também não pode ser regular, já que a imagem inversa de uma linguagem regular por intermédio de um homomorfismo é uma linguagem regular.

É independente do contexto. Uma GIC geradora de L_2 é definida pelas produções $S \to 11S00 \mid \varepsilon$.

(c) $L_3 = \{(11)^m(00)^n \colon m \neq n,\ m, n \geq 0\}$;

Resolução. Não é regular. Temos que $L_2 = L_1 \setminus L_3 = L_1 \cap \overline{L_3}$. Se L_3 fosse regular então $\overline{L_3}$ seria também regular e como L_1 é regular, também L_2 seria regular, o que é falso.

É independente do contexto. Uma gramática geradora de L_3 é

$$S \to 00S11 \mid X \mid Y \qquad X \to 00X \mid 00 \qquad Y \to 11Y \mid 11.$$

(d) $L_4 = \{(11)^n(00)^n \colon 0 \leq n \leq 100\}$;

Resolução. É regular. A linguagem é finita pelo que é regular.

(e) $L_5 = \{(11)^m(00)^m(11)^n(00)^n \colon m, n \geq 0\}$;

Resolução. Não é regular. Temos que $L_2 = L_1 \cap L_5$. Se L_5 fosse regular então L_2 também seria regular (uma vez que L_1 é regular). Mas L_2 não é regular!

É independente do contexto. Uma gramática geradora de L_5 é

$$S \to XX \qquad\qquad X \to 11X00 \mid \varepsilon.$$

(f) $L_6 = \{(11)^m(00)^n(11)^n(00)^m \colon m, n \geq 0\}$.

Resolução. Não é regular. Temos que $L_2 = L_1 \cap L_6$. Se L_6 fosse regular então L_2 também seria regular (uma vez que L_1 é regular). No entanto L_2 não é regular.

É independente do contexto. Uma gramática geradora de L_5 é

$$S \to 11S00 \mid X \mid \varepsilon \qquad\qquad X \to 00X11 \mid \varepsilon.$$

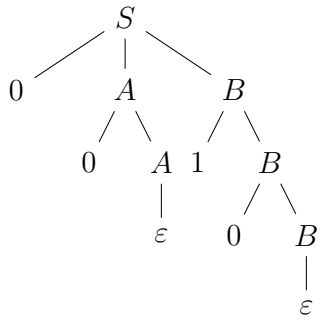
5.6.6. Considere a gramática $G = (\{S, A, B\}, \{0, 1\}, P, S)$ com as seguintes produções:

$$S \to 0AB \qquad A \to 0A \mid \varepsilon \qquad B \to 0B \mid 1B \mid \varepsilon.$$

(a) Construa derivações à esquerda e à direita para 0010 e as respectivas árvores de derivação

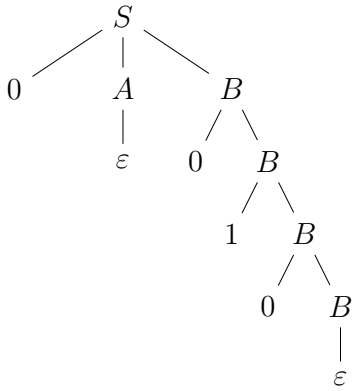
Resolução. Uma possível derivação à esquerda é:

$$\begin{aligned}
S &\underset{\text{esq}}{\Rightarrow} 0AB && (S \to 0AB)\\
&\underset{\text{esq}}{\Rightarrow} 00AB && (A \to 0A)\\
&\underset{\text{esq}}{\Rightarrow} 00B && (A \to \varepsilon)\\
&\underset{\text{esq}}{\Rightarrow} 001B && (B \to 1B)\\
&\underset{\text{esq}}{\Rightarrow} 0010B && (B \to 0B)\\
&\underset{\text{esq}}{\Rightarrow} 0010 && (B \to \varepsilon)
\end{aligned}$$



Uma possível derivação à direita é:

$$\begin{aligned}
S &\underset{\text{dir}}{\Rightarrow} 0AB && (S \to 0AB)\\
&\underset{\text{dir}}{\Rightarrow} 0A0B && (B \to 0B)\\
&\underset{\text{dir}}{\Rightarrow} 0A01B && (B \to 1B)\\
&\underset{\text{dir}}{\Rightarrow} 0A010B && (B \to 0B)\\
&\underset{\text{dir}}{\Rightarrow} 0A010 && (B \to \varepsilon)\\
&\underset{\text{dir}}{\Rightarrow} 0010 && (A \to \varepsilon)
\end{aligned}$$



(b) Será esta gramática ambígua?

Resolução. A gramática é ambígua pois, como vimos na alínea anterior, existem duas árvores de derivação diferentes para a mesma palavra.

Notamos que a derivação mais à esquerda para a segunda árvore apresentada na alínea anterior é

$$S \underset{\text{esq}}{\Rightarrow} 0AB \underset{\text{esq}}{\Rightarrow} 0B \underset{\text{esq}}{\Rightarrow} 00B \underset{\text{esq}}{\Rightarrow} 001B \underset{\text{esq}}{\Rightarrow} 0010B \underset{\text{esq}}{\Rightarrow} 0010.$$

Assim, a palavra 0010 tem duas derivações à esquerda diferentes.

(c) Verifique que a linguagem gerada por esta gramática é regular.

Resolução. As palavras derivadas a partir de A são da forma 0^*. As palavras derivadas a partir de B são da forma $(0+1)^*$. Assim, as palavras derivadas a partir de S são da forma $00^*(0+1)^* = 0(0+1)^*$, pelo que $\mathcal{L}(G)$é regular.

(d) Determine uma gramática regular equivalente.

Resolução. Atendendo a que $\mathcal{L}(G) = \mathcal{L}(0(0+1)^*)$, definimos a gramática regular e equivalente a G, $G' = (\{S, X\}, \{0, 1\}, P', S)$ com as produções P' a seguir indicadas:

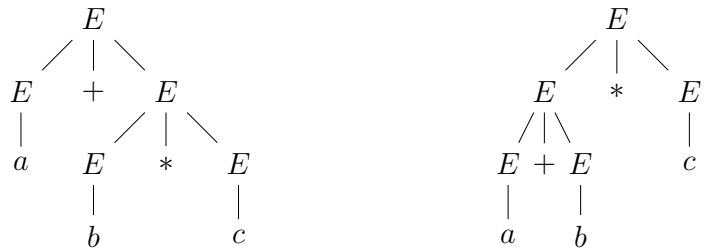
$$S \to 0X \qquad X \to 0X \mid 1X \mid \varepsilon.$$

5.6.7. Considere a seguinte versão simplificada de uma gramática ambígua de expressões aritméticas na notação *infix* $G = (\{E\}, \{a, b, c, +, *\}, P, E)$, com as produções:

$$E \to E + E \mid E * E \mid a \mid b \mid c.$$

(a) Verifique que existe ambiguidade no cálculo de $a + b * c$.

Resolução. A ambiguidade resulta da existência das duas árvores de derivação para $a + b * c$ a seguir indicadas:



(b) Construa uma gramática não ambígua equivalente.

Resolução. Obtemos uma gramática não ambígua eliminando o problema da precedência com a introdução da variável factor F e também da associatividade com a introdução da variável T :

$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to a \mid b \mid c$$

(c) Construa agora uma gramática para a linguagem em que as mesma operações são escritas na notação sufixa (também conhecida por notação Polaca ou *postfix*) e mostre que essa gramática não é ambígua.

Resolução. Uma gramática G' para a notação sufixa é dada pelas produções

$$E \to EE+ \mid EE* \mid a \mid b \mid c.$$

Vamos provar, por indução sobre o número de passos de uma derivação à direita, que para toda a palavra $w \in \mathcal{L}(G')$ existe uma e uma só derivação à direita $E \overset{\star}{\underset{\text{dir}}{\Rightarrow}} w$.

Caso base. Se w é gerada por uma derivação num único passo, então w só pode ser igual a a ou b ou c. Logo, existe uma única forma de derivar w, pelo que existe também uma só derivação à direita de w.

Passo indutivo. Consideremos como hipótese de indução que qualquer palavra da linguagem $\mathcal{L}(G')$, derivável em $n \geq 1$ passos, possui uma única derivação à direita.

Seja $w \in \mathcal{L}(G')$ derivável em $n+1 \geq 2$ passos. Então w é da forma $xy+$ ou $xy*$ com $x, y \in \mathcal{L}(G')$ e x e y são deriváveis em n ou menos passos. Vamos considerar apenas o caso $w = xy+$, já que o caso $w = xy*$ é tratado de forma semelhante. Pela hipótese de indução, as derivações à direita para x e y são únicas. Assim, também é única a derivação à direita de w, $E \Rightarrow EE+ \overset{\star}{\underset{\text{dir}}{\Rightarrow}} Ey+ \overset{\star}{\underset{\text{dir}}{\Rightarrow}} xy+$.

5.6.8. Mostre, usando um contra-exemplo, que é fundamental seguir a ordem indicada no Algoritmo 5.2.2 para eliminação de símbolos inúteis.

Resolução. Consideremos a gramática com produções

$$S \to AB \mid a \qquad A \to aA \qquad B \to C \qquad C \to c.$$

Se tentarmos eliminar em primeiro lugar os símbolos não atingíveis, vemos que todos são atingíveis. Se, em seguida, eliminarmos os símbolos não geradores, descobrimos que apenas A é não gerador, pelo que a gramática se reduz a

$$S \to a \qquad B \to C \qquad C \to c.$$

No entanto, esta gramática continua a ter símbolos não atingíveis e produções inúteis.

Assim, a ordem correcta é eliminar primeiro os símbolos não geradores e só depois os símbolos não atingíveis.

5.6.9. Considere uma gramática na forma normal de Chomsky.

Numa árvore de derivação de uma palavra cada nó folha é filho único do nó pai, pelo que existem tantas folhas como pais das folhas.

Ao remover as folhas, obtemos uma árvore com menos um nível do que a original e que é estritamente binária (cada nó tem 0 filhos ou 2 filhos). Para além disso, esta árvore tem tantas folhas como a original.

Demonstre as afirmações seguintes:

(a) Se uma árvore de derivação tem n níveis então o número de folhas (rotuladas por variáveis), f, satisfaz $n-1 \leq f \leq 2^{n-2}$.

Resolução. Se a árvore de derivação tem $n \geq 2$ níveis então a árvore associada, estritamente binária, tem $k = n-1$ níveis. Vamos mostrar por indução que numa árvore estritamente binária com $k \geq 1$ níveis, o número de folhas f satisfaz $k \leq f \leq 2^{k-1}$.

Caso base. Se a árvore estritamente binária tem um nível então é formada por um único nó raiz que também é folha. Assim, o número de folhas é $f = 1$, o que está de acordo com a fórmula.

Passo indutivo. Consideremos, como hipótese de indução, que o número de folhas em qualquer árvore estritamente binária com p níveis, $1 \leq p < k$ f, satisfaz $p \leq f \leq 2^{p-1}$.

Seja T uma árvore estritamente binária com $k \geq 2$ níveis. Esta árvore decompõe-se na raiz situada no nível 0 e em duas sub-árvores estritamente binárias, T_1 e T_2, com $p_1 \geq 1$ e $p_2 \geq 1$ níveis, respectivamente, de tal modo que $k = \max\{p_1, p_2\} + 1$.

Pela hipótese de indução, o número de folhas da sub-árvore T_1 satisfaz $p_1 \leq f \leq 2^{p_1-1}$ e o número de folhas da sub-árvore T_2 satisfaz $p_2 \leq f \leq 2^{p_2-1}$. Uma vez que as folhas de T são as folhas de T_1 e as de T_2, concluímos que

$$k \leq p_1 + p_2 \leq f_1 + f_2 \leq 2^{p_1-1} + 2^{p_2-1} \leq 2^{k-2} + 2^{k-2} = 2^{k-1}.$$

(b) Se uma árvore de derivação tem 2^m folhas então o número de nós num caminho mais longo da raiz até às folhas, $C_{\max}$, satisfaz $m+1 \leq C_{\max} \leq 2^m$.

Resolução. Substituindo f por 2^m na propriedade demonstrada na alínea anterior, concluímos que $2^m \leq 2^{n-1}$, pelo que $n \geq m+1$, e ainda que $n \leq 2^m + 1$. Resta observar que o número de níveis numa árvore é igual ao número de nós num caminho mais longo da raiz até às folhas.

5.6.10. Construa uma gramática independente do contexto para a linguagem

$$L = \{a^n w w^{-1} b^n \colon w \in \{a,b\}^\star, n \geq 1\}$$

e, em seguida, converta essa gramática à forma normal de Chomsky.

Resolução. As produções seguintes definem uma gramática geradora de L :

$$S \to aSb \mid aXb \mid ab \qquad X \to aXa \mid bXb \mid aa \mid bb.$$

Esta gramática não tem símbolos nem produções inúteis, nem produções ε, nem produções unitárias. Resta-nos, portanto, eliminar os símbolos não isolados nos

lados direitos das produções e, em seguida, reduzir a duas variáveis os lados direitos das produções que tenham mais do que duas variáveis.

Os símbolos a e b aparecem não isolados nos lados direitos das produções. Assim, uma gramática equivalente é:

$$S \to ASB \mid AXB \mid AB \qquad X \to AXA \mid BXB \mid AA \mid BB$$
$$A \to a \qquad B \to b.$$

Introduzindo três novas variáveis auxiliares, C, D e E, obtemos a seguinte gramática na forma normal de Chomsky, equivalente à gramática original:

$$S \to CB \mid DB \mid AB \qquad X \to DA \mid EB \mid AA \mid BB$$
$$A \to a \qquad B \to b \qquad C \to AS \qquad D \to AX \qquad E \to BX.$$

5.6.11. Considere a GIC $G = (\{S, X, Y, Z, W\}, \{a, b, c\}, P, S)$, com conjunto de produções P definido por

$$S \to XbS \mid YaS \mid ZcS \mid \varepsilon \qquad W \to Sa \mid a \mid b \mid c$$
$$X \to aS \mid bZ \qquad Y \to bS \mid aZ \qquad Z \to XYZ.$$

(a) Indique duas derivações distintas da palavra $abba$ com a gramática G.

Resolução. Uma possível derivação é

$$S \Rightarrow XbS \Rightarrow aSbS \Rightarrow abS \Rightarrow abYaS \Rightarrow abbSaS \Rightarrow abbaS \Rightarrow abba.$$

Obtemos uma derivação diferente quando trocamos a ordem pela qual se fazem as substituições, por exemplo

$$S \Rightarrow XbS \Rightarrow XbYaS \Rightarrow aSbYaS \Rightarrow abYaS \Rightarrow abbSaS \Rightarrow abbaS \Rightarrow abba.$$

(b) Comente a afirmação "Se uma palavra tiver duas derivações diferentes numa GIC então podemos concluir que é ambígua. Logo, G é ambígua."

Resolução. Uma gramática é ambígua quando existe pelo menos uma palavra com duas ou mais árvores de derivação diferentes. O facto de uma palavra ter duas derivações diferentes não significa que tenha necessariamente duas árvores de derivação diferentes, já que essas duas derivações podem estar associadas a árvores de derivação idênticas. Logo, a afirmação é falsa.

(c) Construa uma GIC na forma normal de Chomsky geradora de $\mathcal{L}(G) \setminus \{\varepsilon\}$.

Resolução. O conjunto de variáveis geradoras é $\{S, X, Y, W\}$. Assim, eliminamos as produções que envolvem a variável Z, obtendo a gramática equivalente:

$$S \to XbS \mid YaS \mid \varepsilon \qquad W \to Sa \mid a \mid b \mid c$$

$$X \to aS \qquad Y \to bS.$$

O conjunto das variáveis atingíveis é $\{S, X, Y\}$. Assim, eliminamos as produções que envolvem a variável W, obtendo a gramática equivalente:

$$S \to XbS \mid YaS \mid \varepsilon \qquad X \to aS \qquad Y \to bS. \tag{5.6.1}$$

Eliminamos os terminais não isolados nos lados direitos das produções:

$$S \to XBS \mid YAS \mid \varepsilon \qquad X \to AS \qquad Y \to BS$$
$$A \to a \qquad B \to b.$$

Reduzimos os lados direitos das produções a duas variáveis:

$$S \to XZ \mid YW \mid \varepsilon \qquad X \to AS \qquad Y \to BS$$
$$Z \to BS \qquad W \to AS$$
$$A \to a \qquad B \to b.$$

Após eliminar as produções ε, obtemos:

$$S \to XZ \mid YW \qquad X \to AS \mid A \qquad Y \to BS \mid B$$
$$Z \to BS \mid B \qquad W \to AS \mid A$$
$$A \to a \qquad B \to b.$$

Eliminando as produções unitárias, chegamos à forma normal de Chomsky:

$$S \to XZ \mid YW \qquad X \to AS \mid a \qquad Y \to BS \mid b$$
$$Z \to BS \mid b \qquad W \to AS \mid a$$
$$A \to a \qquad B \to b.$$

(d) Mostre (por indução) que para $n \in \mathbb{N}$, $a^n b^n \in \mathcal{L}(G)$.

Resolução. Vamos considerar a gramática (5.6.1) e mostrar por indução que para $n \in \mathbb{N}$, $S \overset{\star}{\Rightarrow} a^n b^n$.

Caso base. Para $n = 1$, temos que $S \Rightarrow XbS \Rightarrow aSbS \Rightarrow abS \Rightarrow ab$, usando as produções $S \to XbS$, $X \to aS$, $S \to \varepsilon$ e $S \to \varepsilon$.

Passo indutivo. Consideramos como hipótese de indução que $S \overset{\star}{\Rightarrow} a^n b^n$ e vamos mostrar a tese $S \overset{\star}{\Rightarrow} a^{n+1} b^{n+1}$.

Aplicando as produções $S \to XbS$, $X \to aS$ e $S \to \varepsilon$ ao último S, derivamos $S \Rightarrow XbS \Rightarrow aSbS \Rightarrow aSb$. Pela hipótese de indução, concluímos que $S \overset{\star}{\Rightarrow} aa^n b^n b = a^{n+1} b^{n+1}$.

(e) Identifique a linguagem $\mathcal{L}(G)$.

Solução. Mostra-se que $\mathcal{L}(G) = \{w \in \{a, b\}^\star \colon \#_a(w) = \#_b(w)\}$. Deixamos a demonstração ao cuidado do leitor.

5.6.12. Usando o lema da bombagem para linguagens independentes do contexto, mostre que qualquer linguagem finita é independente do contexto.

Resolução. Se uma linguagem L é finita então o tamanho das palavras é limitado, pelo que existe $\max\{|w|\colon w \in L\}$. Fixando $n = 1 + \max\{|w|\colon w \in L\}$, concluímos que todas as condições do lema da bombagem são, de forma trivial, satisfeitas, uma vez que não há palavras em L com tamanho maior ou igual a n.

5.6.13. Usando indução estrutural, mostre que as linguagens associadas às expressões regulares são geradas por gramáticas independentes do contexto.

Resolução. Vamos usar indução estrutural sobre a definição de expressão regular sobre um alfabeto Σ e linguagem associada.

Casos base. A linguagem associada à ER $\emptyset$ é $\emptyset$ e uma gramática geradora desta linguagem é

$$G = (\{S\}, \Sigma, \emptyset, S).$$

A linguagem associada à ER ε é $\{\varepsilon\}$ e uma gramática geradora desta linguagem é

$$G = (\{S\}, \Sigma, \{S \to \varepsilon\}, S).$$

A linguagem associada à ER a, com $a \in \Sigma$, é $\{a\}$ e uma gramática geradora desta linguagem é

$$G = (\{S\}, \Sigma, \{S \to a\}, S).$$

Passos indutivos. Sejam $G_E = (V_E, \Sigma, P_E, S_E)$ e $G_F = (V_F, \Sigma, P_F, S_F)$ gramáticas geradoras das linguagens associadas às expressões regulares E e F, respectivamente.

A linguagem associada à ER (E) é $\mathcal{L}(E)$ e uma gramática geradora desta linguagem é G_E.

A linguagem associada à ER $(E + F)$ é $\mathcal{L}(E) \cup \mathcal{L}(F)$ e uma gramática geradora desta linguagem é

$$G_{E+F} = (V_E \cup V_F \cup \{S_{E+f}\}, \Sigma, P_E \cup P_F \cup \{S_{E+F} \to S_E \mid S_F\}, S_{E+F}).$$

A linguagem associada à ER (EF) é $\mathcal{L}(E)\mathcal{L}(F)$ e uma gramática geradora desta linguagem é

$$G_{EF} = (V_E \cup V_F \cup \{S_{EF}\}, \Sigma, P_E \cup P_F \cup \{S_{EF} \to S_E S_F\}, S_{EF}).$$

A linguagem associada à ER (E^*) é $\mathcal{L}(E)\star$ e uma gramática geradora desta linguagem é

$$G_{E^*} = (V_E \cup \{S_{E^*}\}, \Sigma, P_E \cup \{S_{E*} \to S_E S_{E^*} \mid \varepsilon\}, S_{E^*}).$$

5.6.14. Considere o contexto da demonstração do Teorema 5.3.4. Verifique que, para qualquer palavra $w \in \Sigma^+$, $Y \in \delta^\star(X, w)$ se e só se $X \overset{\star}{\Rightarrow} wY$. Em seguida, prove que, para qualquer $w \in \Sigma^\star$, $\delta^\star(s, w) \in F$ se e só se $S \overset{\star}{\Rightarrow} w$.

5.6.15. Obtenha gramáticas lineares geradoras das seguintes linguagens:

(a) $L_1 = \{a^n b^n c^k \colon n, k > 0\}$.

Solução. As produções de uma gramática linear geradora de L_1 são

$$S \to Sc \mid Xc \qquad\qquad X \to aXb \mid ab.$$

(b) $L_2 = \{a^k b^n c^n \colon n, k \geq 0\}$.

Solução. As produções de uma gramática linear geradora de L_2 são

$$S \to aS \mid X \qquad\qquad X \to bXc \mid \varepsilon.$$

5.6.16. Considere a LIC $L = \{a^n b^n \colon n \geq 0\}$.

(a) Construa uma GIC G geradora de L.

Solução. A gramática $G = (\{S\}, \{a, b\}, P_G, S)$, onde o conjunto de produções P_G é constituído pelas produções:

$$S \to aSb \mid \varepsilon,$$

é geradora de L.

(b) Mostre que, para $k \geq 1$, L^k é uma LIC.

Solução. Para $k \geq 1$ a gramática $G^k = (\{S', S\}, \{a, b\}, P_{G^k}, S')$ gera L^k, onde o conjunto de produções P_{G^k} é constituído pelas seguintes produções:

$$S' \to \overbrace{S \cdots S}^{k \text{ vezes}} \qquad S \to aSb \mid \varepsilon.$$

(c) Mostre que $L^\star$ é uma LIC.

Solução. A gramática $G^\star = (\{S', S\}, \{a, b\}, P_{G^\star}, S')$ gera $L^\star$, onde o conjunto de produções $P_{G^\star}$ é constituído pelas seguintes produções:

$$S' \to SS' \mid \varepsilon \qquad S \to aSb \mid \varepsilon.$$

(d) Mostre que $\mathcal{L}(a^*b^*) \setminus L$ é uma LIC.

Resolução. Temos que $\mathcal{L}(a^*b^*) \setminus L = \{a^n b^m \colon m \neq n \text{ e } m, n \geq 0\}$. Uma gramática geradora desta linguagem é $G' = (\{S, A, B\}, \{a, b\}, P_{G'}, S)$, onde o conjunto de produções $P_{G'}$ é constituído pelas seguintes produções:

$$S \to aSb \mid A \mid B \qquad A \to aA \mid A \qquad B \to bB \mid B.$$

5.6.17. Aplique o Lema da Bombagem para LIC para provar que as seguintes linguagens não são independentes do contexto:

(a) $L = \{ww \colon w \in \{a, b\}^\star\}$.

Resolução. Suponhamos que L é independente do contexto. De acordo com o lema da bombagem, existe um inteiro n tal que toda a palavra $z \in L$ de tamanho $|z| \geq n$ pode ser decomposta em $z = uvwxy$ de tal forma que: (i) $|vwx| \leq n$; (ii) $|vx| \geq 1$; (iii) uv^kwx^ky é uma palavra da linguagem L, qualquer que seja $k \geq 0$.

Consideremos a palavra $z = a^nb^na^nb^n \in L$ de tamanho $4n \geq n$ e uma qualquer decomposição $z = uvwxy$.

Um e um só dos 3 casos seguintes ocorre: ou vwx está inteiramente contida na primeira metade de z ou vwx contém o centro de z ou vwx está inteiramente contida na segunda metade de z.

Se vwx está inteiramente contida na primeira metade de z então o centro da palavra uv^2wx^2y é deslocado à esquerda em pelo menos uma posição e no máximo em n posições. Assim, a segunda metade de uv^2wx^2y começa com b enquanto a primeira continua a começar com a. Logo, uv^2wx^2y não pertence a L, o que é uma contradição.

Se vwx está inteiramente contida na segunda metade de z então, de forma análoga, o centro da palavra uv^2wx^2y é deslocado à direita em pelo menos uma posição e no máximo em n posições. Assim, a primeira metade de uv^2wx^2y termina em a enquanto a segunda continua a terminar em b. Logo, uv^2wx^2y não pertence a L, o que é uma contradição.

Finalmente, se vwx contém o centro de z, consideremos $k = 0$ e a palavra uwy. Esta palavra é obtida removendo pelo menos uma letra e o número de a´s no início da palavra mantém-se, bem como o número de b´s no final da palavra. Assim, $uwy = a^nb^pa^qb^n$ em que $p < n$ ou $q < n$. Logo, uwy não é da forma ww qualquer que seja a palavra w, pelo que uwy não pertence a L, o que é uma contradição.

Em qualquer dos casos, chegamos a uma contradição. Concluímos que L não pode ser independente do contexto.

(b) $L = \{a^ib^jc^{ij} \colon i, j \in \mathbb{N}_0\}$.

Resolução. Admitamos que L é uma LIC. Seja $n > 0$ o parâmetro especificado no lema da bombagem e seja $z = a^nb^nc^{n^2} \in L$, de tamanho $|z| = n^2 + 2n \geq n$.

Consideremos uma qualquer partição $z = uvwxy$ tal que (i) $|vwx| \leq n$ e (ii) $|vx| \geq 1$. Pelo lema da bombagem, (iii) uv^kwx^ky é uma palavra da linguagem L, qualquer que seja $k \geq 0$.

Começamos por observar que nem v nem x podem ser da forma $a^p b^q$, com $p, q \geq 1$. De facto, nesse caso, para $k = 2$, a palavra uv^2wx^2y não pertenceria a L pois conteria a subpalavra $a^p b^q a^p b^q$.

De modo análogo, nem v nem x podem ser da forma $b^p c^q$, com $p, q \geq 1$, caso contrário, para $k = 2$, a palavra uv^2wx^2y conteria a subpalavra $b^p c^q b^p c^q$, pelo que não pertenceria a L.

Assim, tanto v como x só podem conter a´s ou b´s ou c´s. Resta-nos analisar os seguintes casos (nos quais consideramos $k = 0$):

(a) Se $v = a^p$ e $x = a^q$ com $p + q \geq 1$ então

$$uv^0wx^0y = a^{n-(p+q)}b^n c^{n^2} \notin L,$$

uma vez que $(n - (p + q))n < n^2$.

(b) Se $v = a^p$ e $x = b^q$ com $p + q \geq 1$ então

$$uv^0wx^0y = a^{n-p}b^{n-q}c^{n^2} \notin L,$$

uma vez que $(n - p)(n - q) < n^2$.

(c) O caso $v = a^p$ e $x = c^q$ não pode ocorrer, caso contrário w conteria n b´s e portanto $|vwx| > n$ em contradição com (i).

(d) Se $v = b^p$ e $x = b^q$ com $p + q \geq 1$ então

$$uv^0wx^0y = a^n b^{n-(p+q)}c^{n^2} \notin L,$$

uma vez que $n(n - (p + q)) < n^2$.

(e) Se $v = b^p$ e $x = c^q$ com $p + q \geq 1$ então $uv^0wx^0y = a^n b^{n-p}c^{n^2-q}$. Temos que $n(n - p) = n^2 - q \iff q = np$. Para $p = 0$ vem $q = 0$ o que é uma contradição com $p + q \geq 1$. Para $p \geq 1$ vem $q \geq n$ e portanto $|vwx| \geq n + 1$ em contradição com (i). Em qualquer dos casos, concluímos que $uv^0wx^0y \notin L$.

(f) Se $v = c^p$ e $x = c^q$ com $p + q \geq 1$ então

$$uv^0wx^0y = a^n b^n c^{n^2-(p+q)} \notin L,$$

uma vez que $n^2 < n^2 - (p + q)$.

Em qualquer dos casos, chegamos a uma contradição com a condição (iii) do lema da bombagem, pelo que L não pode ser independente do contexto.

(c) $L = \{a^n b^n a^n b^n : n \in \mathbb{N}_0\}$.

Resolução. Admitamos que L é uma LIC. Seja $n > 0$ o parâmetro especificado no lema da bombagem e seja $z = a^n b^n a^n b^n \in L$, de tamanho $|z| = 4n \geq n$.

Consideremos uma qualquer partição $z = uvwxy$ tal que (i) $|vwx| \leq n$ e (ii) $|vx| \geq 1$. Pelo lema da bombagem, (iii) $uv^k wx^k y$ é uma palavra da linguagem L, qualquer que seja $k \geq 0$.

Consideremos as seguintes regiões de w :

$$\underbrace{a^n}_{R1}\underbrace{b^n}_{R2}\underbrace{a^n}_{R3}\underbrace{b^n}_{R4}$$

onde as regiões R_1 e R_3 estão associadas à letra a e as regiões R_2 e R_4 estão associadas à letra b.

Por (i) concluímos que vwx só pode estar incluída em no máximo duas regiões consecutivas e para $k = 2$ o número de letras nas regiões que contêm vwx aumenta enquanto o número de letras nas regiões associadas se mantém. Logo, $uv^2wx^2y \notin L$, o que é uma contradição com (iii), pelo que L não pode ser independente do contexto.

(d) $L = \{a^{n^2} : n \geq 1\}$.

Resolução. Admitamos que L é independente do contexto. Seja $n \geq 1$ o parâmetro especificado no lema da bombagem e seja $z = a^{n^2} \in L$.

Consideremos uma qualquer partição $z = uvwxy$ tal que (i) $|vwx| \leq n$ e (ii) $|vx| \geq 1$. Pelo lema da bombagem, (iii) $uv^k wx^k y$ é uma palavra da linguagem L, qualquer que seja $k \geq 0$.

Temos que $v = a^p$ e $x = a^q$, com $1 \leq p + q \leq n$ (por (i) e (ii)).

Para $k = 2$, por (iii), concluímos que $uv^2wx^2y \in L$. Mas $uv^2wx^2y = a^{n^2+p+q}$. No entanto, $n^2 < n^2 + p + q \leq n^2 + n < (n + 1)^2$, pelo que $uv^2wx^2y \notin L$ o que é uma contradição. Concluímos que L não pode ser independente do contexto.

(e) $L = \{a^p : p$ é um número primo$\}$.

Resolução. Admitamos que L é independente do contexto. Seja $n > 0$ o parâmetro especificado no lema da bombagem. Uma vez que o conjunto dos números primos é infinito, seja $p \geq n$ um número primo. Então $z = a^p \in L$.

Consideremos uma qualquer partição $z = uvwxy$ tal que (i) $|vwx| \leq n$ e (ii) $|vx| \geq 1$. Pelo lema da bombagem, (iii) $uv^k wx^k y$ é uma palavra da linguagem L, qualquer que seja $k \geq 0$.

Temos que $v = a^q$ e $x = a^r$, com $1 \leq q + r \leq n$ (por (i) e (ii)).

Para $k = p + 1$, por (iii), concluímos que $uv^{p+1}wx^{p+1}y \in L$. Mas

$$uv^{p+1}wx^{p+1}y = uv^pvwxx^py = a^{p+qp+rp}.$$

No entanto, $p+qp+rp = p(1+q+r)$ não é primo uma vez que $1+q+r \geq 2$ e portanto $uv^{p+1}wx^{p+1}y \notin L$ o que é uma contradição. Concluímos que L não pode ser independente do contexto.

(f) $L = \{w \in \{a,b,c\}^\star : \#_a(w) < \#_b(w) < \#_c(w)\}$.

Resolução. Admitamos que L é uma LIC. Seja $n > 0$ o parâmetro especificado no lema da bombagem e seja $z = a^n b^{n+1} c^{n+2} \in L$, de tamanho $|z| = 3n + 3 \geq n$.

Consideremos uma qualquer partição $z = uvwxy$ tal que (i) $|vwx| \leq n$ e (ii) $|vx| \geq 1$. Pelo lema da bombagem, (iii) $uv^k wx^k y$ é uma palavra da linguagem L, qualquer que seja $k \geq 0$.

Por (i) concluímos que vx não pode conter simultaneamente a´s e c´s. Temos os seguintes casos:

(a) Se vx contêm apenas a´s então para $k = 2$ o número de a´s aumenta em pelo menos uma unidade enquanto o número de b´s se mantém. Logo, $uv^2wx^2y \notin L$.

(b) Se vx contém a´s e b´s então para $k = 2$ o número de b´s aumenta em pelo menos uma unidade enquanto o número de c´s se mantém. Logo, $uv^2wx^2y \notin L$.

(c) Se vx contêm apenas b´s então para $k = 2$ o número de b´s aumenta em pelo menos uma unidade enquanto o número de c´s se mantém. Logo, $uv^2wx^2y \notin L$.

(d) Se vx contém b´s e c´s então para $k = 0$ o número de b´s diminui em pelo menos uma unidade enquanto o número de a´s se mantém. Logo, $uv^0wx^0y \notin L$.

(e) Se vx contêm apenas c´s então para $k = 0$ o número de c´s diminui em pelo menos uma unidade enquanto o número de b´s se mantém. Logo, $uv^0wx^0y \notin L$.

Em qualquer dos casos possíveis, chegamos a uma contradição com (iii), pelo que L não pode ser independente do contexto.

(g) $L = \{w \in \{a,b,c\}^\star : \#_b(w) > \#_a(w)$ e $\#_c(w) > \#_a(w)\}$.

Resolução. Admitamos que L é uma LIC. Seja $n > 0$ o parâmetro especificado no lema da bombagem e seja $z = a^n b^{n+1} c^{n+1} \in L$, de tamanho $|z| = 3n + 2 \geq n$.

Consideremos uma qualquer partição $z = uvwxy$ tal que (i) $|vwx| \leq n$ e (ii) $|vx| \geq 1$. Pelo lema da bombagem, (iii) $uv^k wx^k y$ é uma palavra da linguagem L, qualquer que seja $k \geq 0$.

Por (i) concluímos que vx não pode conter simultaneamente a´s e c´s. Temos os seguintes casos:

(a) Se vx contêm a´s então para $k = 2$ o número de a´s aumenta em pelo menos uma unidade enquanto o número de c´s se mantém. Logo,

$uv^2wx^2y \notin L$.

(b) Se vx contém b´s ou c´s então para $k = 0$ o número de b´s ou de c´s diminui em pelo menos uma unidade enquanto o número de a´s se mantém. Logo, $uv^0wx^0y \notin L$.

Em qualquer dos casos possíveis, chegamos a uma contradição com (iii), pelo que L não pode ser independente do contexto.

5.6.18. Determine gramáticas geradoras das seguintes linguagens independentes do contexto:

(a) $^{***}L = \{w \in \{a,b\}^\star \colon \#_a(w) = \#_b(w)\}$.

Resolução. Uma GIC geradora de L é $G = (\{S\}, \{a,b\}, P, S)$ definida pelo seguinte conjunto de P de produções:

$$S \to aSb \mid bSa \mid SS \mid \varepsilon.$$

Notamos que há outras formas de gerar L, por exemplo, usando as produções

$$S \to aSbS \mid bSaS \mid \varepsilon.$$

Vejamos os detalhes de uma outra solução. Excepto a palavra vazia, qualquer palavra de L é da forma $w = ax$ com $x \in L_a = \{w \in \{a,b\}^\star \colon \#_a(x) = \#_b(x) - 1\}$ ou da forma $w = by$ com $y \in L_b = \{w \in \{a,b\}^\star \colon \#_a(x) = \#_b(x)+1\}$. Sejam X uma variável geradora de L_a e Y uma variável geradora de L_b. Então as produções $S \to aX \mid bY \mid \varepsilon$ geram L.

Vejamos como obter uma gramática geradora da linguagem L_a. Seja $x \in L_a$. Então $x = bw$, em que $\#_a(w) = \#_b(w)$ ou então $x = az$, em que $\#_a(z) = \#_b(z) - 2$. No primeiro caso, $w \in L$ e podemos considerar a produção $X \to bS$. No segundo caso, seja z_i o prefixo de tamanho i de z e consideremos a função $f(z_i) = \#_a(z_i) - \#_b(z_i)$. Temos que $f(z_0) = 0$, $f(z_n) = -2$, onde $n = |z|$, e $f(z_{i+1}) = f(z_i) \pm 1$, para $i = 0, \ldots, n-1$. Logo, terá de existir um valor de i tal que $f(z_i) = -1$. Assim, $z = z_i v$ com $z_i \in L_A$ e $v \in L_A$. Chegamos assim à produção $X \to aXX$.

Com uma análise semelhante, concluímos que as palavras da linguagem L_b são geradas pelas produções $Y \to bYY \mid aS$.

Em conclusão, L é gerada pelas produções

$$S \to aX \mid bY \mid \varepsilon \qquad X \to aXX \mid bS \qquad Y \to bYY \mid aS.$$

(b) $^{***}L = \{w \in \{a,b\}^\star \colon \#_a(w) = 2\#_b(w)\}$.

Resolução. Uma GIC geradora de L é dada pelas seguintes produções

$$S \to aSaSbS \mid aSbSaS \mid bSaSaS \mid \varepsilon.$$

O conjunto de produções seguintes também permite gerar L :

$$S \to aSaSb \mid aSbSa \mid bSaSa \mid \varepsilon.$$

Vamos analisar uma outra solução:

$$S \to SS \mid aaSb \mid aSbSa \mid bSaa \mid \varepsilon.$$

Dizemos que uma palavra $w \in L$ é atómica se nenhum dos seus prefixos próprios é uma palavra da linguagem, isto é, se $w = xy$ com $y \neq \varepsilon$ então $x \notin L$.

Começamos por observar que se w não é atómica então $w = xy$ com $x \in L$ e $y \in L$, pelo que a produção $S \to SS$ permite gerar as palavras não atómicas de L.

Consideremos a função $f(w) = \#_a(w) - 2\#_b(w)$. Notamos que $f(xy) = f(x) + f(y)$ para quaisquer palavras x e y. Para além disso, se $w \in L$ então $f(w) = 0$, $f(ax) = f(xa) = 1 + f(x)$ e $f(bx) = f(xb) = -2 + f(x)$.

Suponhamos então que w é atómica.

Se w é da forma $aaxa$, para alguma palavra x, então existem y e z tais que $aaxa = aaybza$. De facto, uma vez que $f(aay) \neq 0$ para qualquer prefixo y de x e como $f(aa) = 2 > 0$ e $f(aax) = -1 < 0$ terão de existir dois prefixos consecutivos tais que a função f passa de $+1$ a -1. Assim, $f(ay) = f(z) = 0$ pelo que $ay \in L$ e $z \in L$. Logo, $aaxa$ é gerada usando a produção $S \to aSbSa$.

Se w é da forma $aaxb$ então $x \in L$ e w é gerada usando a produção $S \to aaSb$.

Se w é da forma $abxa$ então $x \in L$ e w é gerada usando a produção $S \to aSbSb$ e $S \to \varepsilon$ aplicado ao primeiro S.

Suponhamos agora que w é da forma $abxb$. Uma vez que $f(ab) = -1$ e $f(abx) = +2$, terá de existir um prefixo y de x tal que $f(aby) = 0$. Mas então w não seria atómica.

Se w é da forma $baxa$ então temos 3 casos:

(a) $w = baa$ e as produções $S \to bSaa$ e $S\ to\varepsilon$ permitem gerar w.

(b) $w = bayaa$ e uma vez que $f(ay) = 0$, w é gerada usando a produção $S \to bSaa$.

(c) $w = bayba$ então como $f(ba) = -1 < 0$ e $f(bay) = +1 > 0$ então existem palavras z e t tal que $w = baztba$ com $f(baz) = f(tba) = 0$ e w não seria composta.

Se w é da forma $baxb$ então como $f(ba) = -1 < 0$ e $f(bax) = +2 > 0$ então existem palavras y e z tais que $w = bayzb$ com $f(bay) = f(zb) = 0$ e w não seria composta.

Se w é da forma $bbxaa$ então $f(bx) = 0$ pelo que w é gerada usando a produção $S \to bSaa$.

Se w é da forma $bbxba$ então como $f(b) = -2 < 0$ e $f(bbx) = +1 > 0$ então existem y e z tais que $w = bbyzba$ com $f(bby) = f(zba) = 0$ e w não seria atómica.

Se w é da forma $bbxb$ então como $f(b) = -2 < 0$ e $f(bbx) = +2 > 0$ então existem y e z tais que $w = bbyzb$ com $f(bby) = f(zb) = 0$ e w não seria atómica.

(c) $L = \{w \in \{a,b\}^\star : \#_a(w) < \#_b(w)\}$.

Resolução. Seja $w \in L$ e seja S o axioma de uma GIC geradora de L.

Se $w = az$ então $\#_a(z) + 2 \leq \#_b(z)$ e é possível mostrar que $z = uv$, com $u, v \in L$. Assim, a produção $S \to aSS$ permite gerar w.

Se $w = bz$ então ou $\#_a(z) < \#_b(z)$ ou $\#_a(z) = \#_b(z)$. No primeiro caso a produção $S \to bS$ permite gerar w. No segundo caso usamos a produção $S \to bX$ em que X é uma variável de uma gramática que permita gerar a linguagem da alínea (a).

Concluímos que uma GIC geradora de L é dada pelas produções

$$\begin{aligned} S &\to aSS \mid bS \mid bX \\ X &\to XX \mid aXb \mid bXa \mid \varepsilon. \end{aligned}$$

(d) $L = \{w \in \{a,b\}^\star : \#_a(w) \leq \#_b(w)\}$.

Resolução. Seja $w \in L$ e seja S o axioma de uma GIC geradora de L.

Seja X uma variável que permite gerar as palavras w tais que $\#_a(w) = \#_b(w)$.

Se $\#_a(w) < \#_b(w)$ então a gramática da alínea anterior permite gerar w.

Se $\#_a(w) = \#_b(w)$ então a produção $S \to X$ permite gerar w.

Concluímos que uma GIC geradora de L (depois de simplificada) é dada pelas produções

$$\begin{aligned} S &\to aSS \mid bS \mid X \\ X &\to XX \mid aXb \mid bXa \mid \varepsilon. \end{aligned}$$

(e) $L = \{w \in \{a,b\}^\star : \#_a(w) \neq \#_b(w)\}$.

Resolução. Basta observar que $L = \{w \in \{a,b\}^\star \colon \#_a(w) < \#_b(w)\} \cup \{w \in \{a,b\}^\star \colon \#_b(w) < \#_a(w)\}$ e, em seguida, aplicar a alínea (c).

Concluímos que uma GIC geradora de L é dada pelas produções

$$\begin{aligned} S &\to S_1 \mid S_2 \\ S_1 &\to aS_1S_1 \mid bS_1 \mid bX \mid X \\ S_2 &\to bS_2S_2 \mid aS_2 \mid aX \mid X \\ X &\to XX \mid aXb \mid bXa \mid \varepsilon. \end{aligned}$$

(f) $^{***}L = \{z \in \{a,b\}^\star \colon$ não existe $w \in \{a,b\}^\star$ tal que $z = ww\}$.

Resolução. A linguagem complementar da linguagem L é $\overline{L} = \{ww \colon w \in \{a,b\}^\star\}$. Embora $\overline{L}$ não seja independente do contexto, L é independente do contexto.

Para construir uma gramática geradora de L é necessário analisar todas as formas possíveis de obter uma palavra da forma ww, com $w \in \{a,b\}^\star$ e, em seguida, estabelecer um processo para quebrar esta regra. Uma GIC geradora de L é, por exemplo:

$$\begin{aligned} S &\to E \mid U & E &\to AB \mid BA \\ A &\to ZAZ \mid a & B &\to ZBZ \mid b \\ U &\to ZUZ \mid Z & Z &\to a \mid b. \end{aligned}$$

5.6.19. Existe um erro na seguinte demonstração de que a linguagem

$$L = \{w \in \{a,b\}^\star \colon \#_a(w) < \#_b(w) < 2\#_a(w)\}$$

não é independente do contexto. Descubra-o.

Demonstração: Seja $n \in \mathbb{N}$ o parâmetro especificado no lema da bombagem que garante que qualquer palavra de tamanho maior ou igual do que n pode ser bombeada.

Consideremos a palavra $z = a^{n+1}b^{n+2} \in L$, de tamanho $|z| = 2n+3 \geq n$.

Seja $z = uvwxy$ uma qualquer partição da palavra z nas condições do lema. Há 3 casos a considerar:

Caso 1: Se vwx só tem a´s, seja $p = \#_a(vx) \geq 1$. Então $\#_a(uv^kwx^ky) = n+1+p(k-1)$ e fixando $k = 2$ concluímos que $\#_a(uv^kwx^ky) \geq n+2 = \#_b(uv^kwx^ky)$.

Caso 2: Se vwx só tem b´s, seja $p = \#_b(vx) \geq 1$. Então $\#_b(uv^kwx^ky) = n+2+p(k-1)$ e fixando $k = 0$ concluímos que $\#_b(uv^kwx^ky) \leq n+1 = \#_a(uv^kwx^ky)$.

Caso 3: Se vwx contém a´s e b´s, sejam $p = \#_a(vx) \geq 1$ e $q = \#_b(vx) \geq 1$. Então $\#_a(uv^kwx^ky) = n+1+p(k-1)$ e $\#_b(uv^kwx^ky) = n+2+q(k-1)$.

Aqui temos duas possibilidades:

(a) Se $p \leq q$ então, fixando $k = 0$, concluímos que $\#_a(uv^kwx^ky) \geq \#_b(uv^kwx^ky)$.

(b) Se $p > q$ então, fixando $k = 2$, concluímos que $\#_a(uv^kwx^ky) \geq \#_b(uv^kwx^ky)$.

Em qualquer dos casos, existe um valor $k \in \mathbb{N}_0$ tal que $uv^kwx^ky \notin L$. Concluímos que L não satisfaz o lema da bombagem e, consequentemente, não é independente do contexto.

Resolução. O erro encontra-se no Caso 3(a). De facto, sempre que $p = q$, $\#_a(uv^kwx^ky) < \#_b(uv^kwx^ky)$. Para além disso, $\#_b(uv^kwx^ky) \geq \#_a(uv^kwx^ky)$ apenas quando $k = -1$ e $p < n$, o que, dependendo da partição, pode não ocorrer.

5.6.20. Considere a GIC G com as seguintes produções:

$$S \to ASA \mid aB \qquad A \to B \mid S \qquad B \to b \mid \varepsilon.$$

(a) Obtenha uma gramática equivalente, G', na forma normal de Chomsky.

Resolução. Todos os símbolos são atingíveis e geradores, pelo que G não tem produções inúteis.

Eliminação de terminais não isolados no lado direito das produções:

$$S \to ASA \mid XB \qquad A \to B \mid S \qquad B \to b \mid \varepsilon \qquad X \to a.$$

Redução dos tamanhos dos lados direitos das produções:

$$S \to AY \mid XB \qquad A \to B \mid S \qquad B \to b \mid \varepsilon \qquad X \to a \qquad Y \to SA.$$

Eliminação de produções ε (variáveis anuláveis A, B):

$$S \to AY \mid Y \mid XB \mid X \qquad A \to B \mid S \qquad B \to b$$
$$X \to a \qquad Y \to SA \mid S.$$

Eliminação de produções unitárias:

$$S \to AY \mid XB \mid SA \mid a \qquad A \to AY \mid XB \mid SA \mid a \mid b$$
$$B \to b \qquad X \to a \qquad Y \to SA \mid AY \mid XB \mid a.$$

(b) Seja $w \in \mathcal{L}(G')$ uma qualquer palavra de tamanho $n = |w|$. Determine, como função de n, o número de passos da derivação de w usando a gramática G'.

Resolução. O número de passos utilizados para derivar uma qualquer palavra de tamanho $n > 0$ usando as produções de uma gramática na forma normal de Chomsky é $2n-1$. Partindo de S, a aplicação de uma produção da forma $X \to YZ$ incrementa o tamanho do lado direito da sentença derivada em um símbolo de variável. Assim, para obter uma sentença com n símbolos de variável são necessários $n - 1$ passos. Finalmente, usando produções da forma $X \to a$, são necessários mais n passos para substituir cada símbolo de variável por um terminal.

5.6.21. Mostre que a GIC G definida pelas seguintes produções é ambígua:

$$S \to aSbS \mid bSaS \mid \varepsilon.$$

Em seguida, determine uma gramática não ambígua geradora de $\mathcal{L}(G)$.

Resolução. Por exemplo, a palavra $abab$ tem duas derivações à esquerda distintas,

$$S \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSbS \; derababS \Rightarrow abab$$

e

$$S \Rightarrow aSbS \Rightarrow abSaSbS \Rightarrow abaSbS \Rightarrow ababS \Rightarrow abab.$$

Uma gramática não ambígua geradora de L é

$$S \to aBS \mid bAS \mid \varepsilon \quad A \to a \mid bAA \quad B \to b \mid aBB.$$

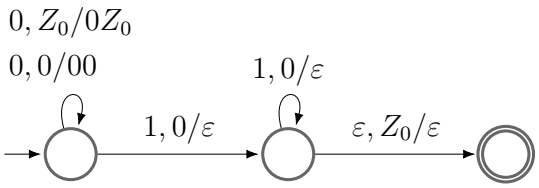
6 Autómatos de Pilha

6.5 Exercícios

6.5.1. Verifique que as seguintes linguagens são Independentes do Contexto, construindo autómatos de pilha que as reconheçam, se possível deterministas:

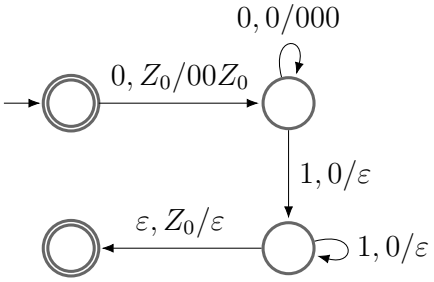
(a) $\{0^n 1^n : n > 0\}$;

Resolução. O autómato seguinte funciona tanto na modalidade de reconhecimento por pilha vazia como na de reconhecimento por estados de aceitação. Para além disso, o AP é determinista.



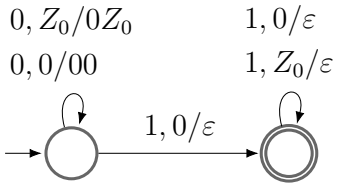
(b) $\{0^n 1^{2n} : n \geq 0\}$;

Resolução. A estratégia utilizada no AP determinista a seguir apresentado consiste em: começar por empilhar dois 0´s por cada 0 que apareça; se o número de 1´s for o dobro do número de 0´s então, ao desempilhar um 0 por cada 1 que apareça, no final a pilha ficará apenas com o símbolo Z_0.

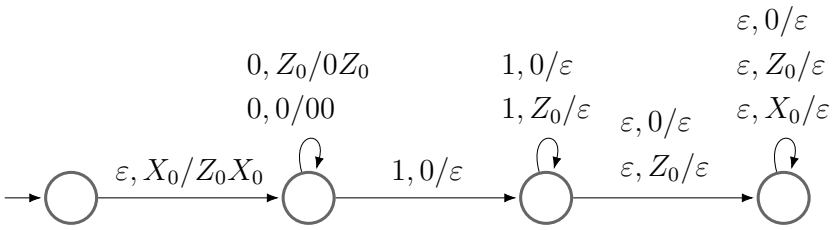


(c) $\{0^m 1^n : m \geq n > 0\}$;

Resolução. A estratégia consiste em empilhar os 0´s da palavra e, em seguida, desempilhar um 0 por cada 1 que apareça. No final, se o número de 0´s for maior ou igual ao número de 1´s então ficará um 0 no topo da pilha (quando os 1´s sejam menos do que os 0´s) ou o símbolo de pilha inicial (quando o número de 1´s seja igual ao número de 0´s). O AP seguinte é determinista e reconhece as palavras na modalidade de estados de aceitação.



O AP seguinte é não determinista e reconhece a linguagem na modalidade de pilha vazia. (É possível juntar os dois estados mais à direita num único estado.)



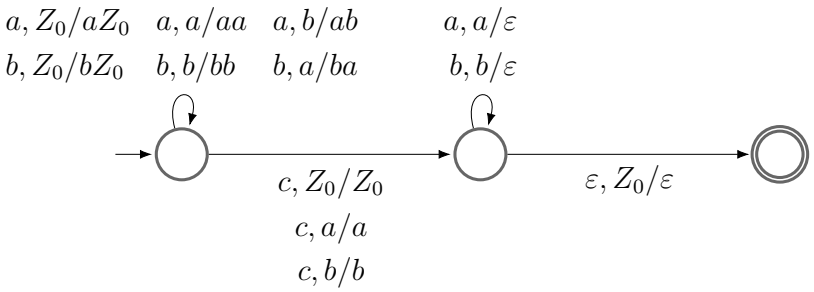
Notamos ainda que não é possível definir um APD reconhecedor desta linguagem na modalidade de pilha vazia, pois a linguagem não é livre de prefixos (por exemplo $001 \in L$ é um prefixo de $0011 \in L$).

(d) $\{wcw^{-1} : w \in \{a, b\}^\star\}$;

Resolução. A estratégia consiste em:

- Ir lendo a´s e b´s e colocá-los na pilha até aparecer a letra c.
- Ir lendo a´s e b´s e retirá-los da pilha verificando se são iguais.
- A palavra é aceite se e só se a pilha fica vazia no final da leitura.

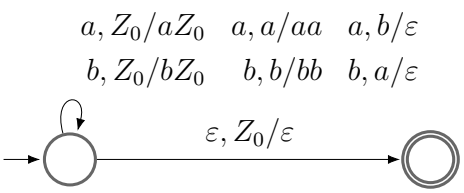
O autómato seguinte é determinista e reconhece a linguagem quer por pilha vazia, quer por estados de aceitação.



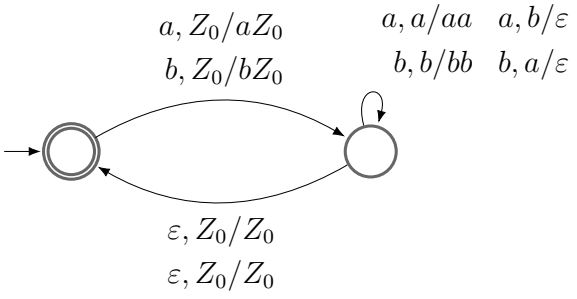
(e) $\{w \in \{a,b\}^\star \colon \#_a(w) = \#_b(w)\}$

Resolução. Se a letra lida (a ou b) é diferente da que está no topo da pilha então temos um par de letras que se anulam (pelo que retiramos a letra que está no topo da pilha). No fim da leitura de w, se $\#_a(w) = \#_b(w)$ então no topo da pilha deve estar o símbolo Z_0.

O autómato seguinte é não determinista e reconhece a linguagem nas duas modalidades (pilha vazia ou estados de aceitação).



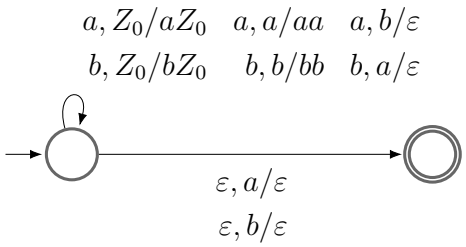
Um autómato determinista que reconhece a linguagem na modalidade de estados de aceitação é o seguinte:



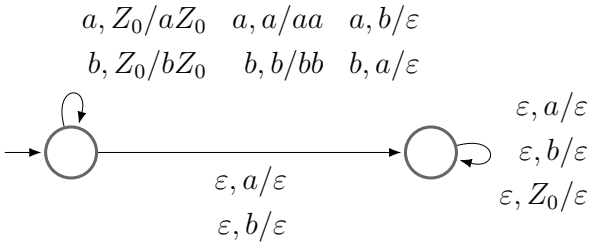
Notamos que o estado inicial serve para aceitar a palavra vazia e que não é possível construir um APD que reconheça a linguagem por pilha vazia (pois ε é um prefixo próprio de qualquer palavra diferente da palavra vazia).

(f) $\{w \in \{a,b\}^\star \colon \#_a(w) \neq \#_b(w)\}$

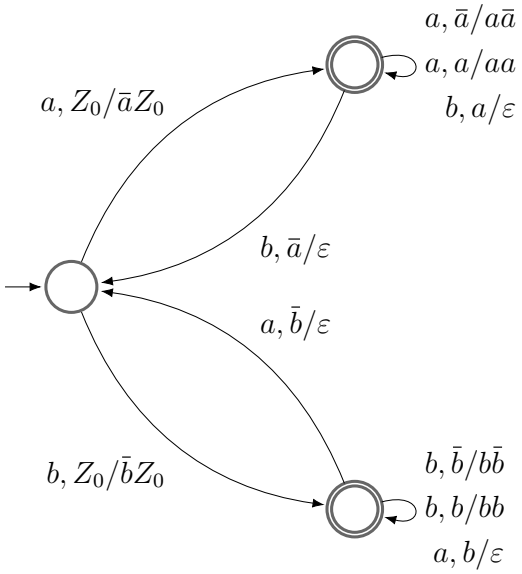
Resolução. Se no final da leitura de w ainda restar algum a ou b na pilha então é porque $\#_a(w) \neq \#_b(w)$. O AP seguinte é não determinista e reconhece a linguagem apenas na modalidade de estados de aceitação.



O AP seguinte é não determinista e reconhece a linguagem na modalidade de pilha vazia.



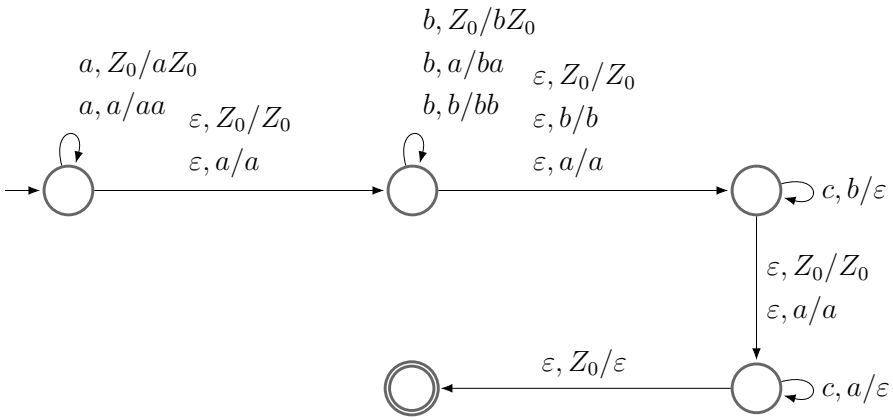
O AP seguinte é determinista e reconhece a linguagem na modalidade de estados de aceitação. A estratégia consiste em marcar a primeira letra lida (e guardada na pilha) para que se possa identificar quando o número de a´s lidos é igual ao número de b´s lidos.



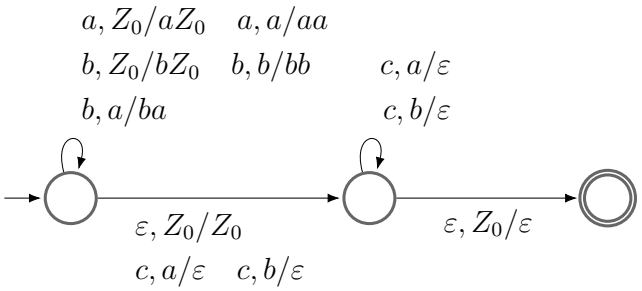
(g) $\{a^m b^n c^{m+n} : m, n \geq 0\}$

Resolução. Construímos um AP não determinista que começa por empilhar os a´s. Em seguida, empilha os b´s e por fim retira da pilha os b´s e

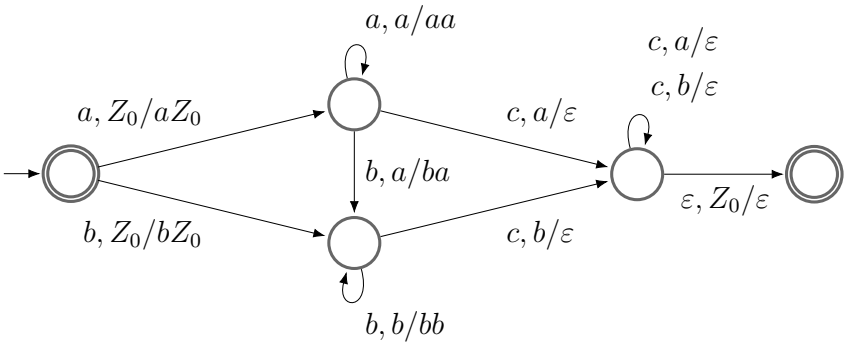
os a´s empilhados, um a um, por cada c que encontra.



Notamos que o autómato anterior pode ser simplificado para o seguinte:



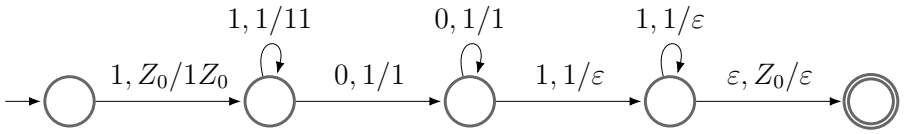
Um APD que reconhece a linguagem por estados de aceitação é o seguinte:



Notamos que na modalidade de pilha vazia o APD acima reconhece todas as palavras exceto a palavra vazia e que não é possível construir um APD que reconheça toda a linguagem na modalidade de pilha vazia.

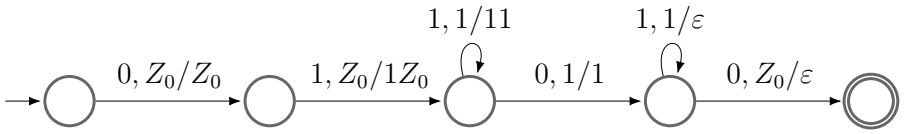
(h) $\{1^m 0^n 1^m : m, n \geq 1\}$

Resolução. O autómato seguinte é determinista e funciona nas duas modalidades de reconhecimento.



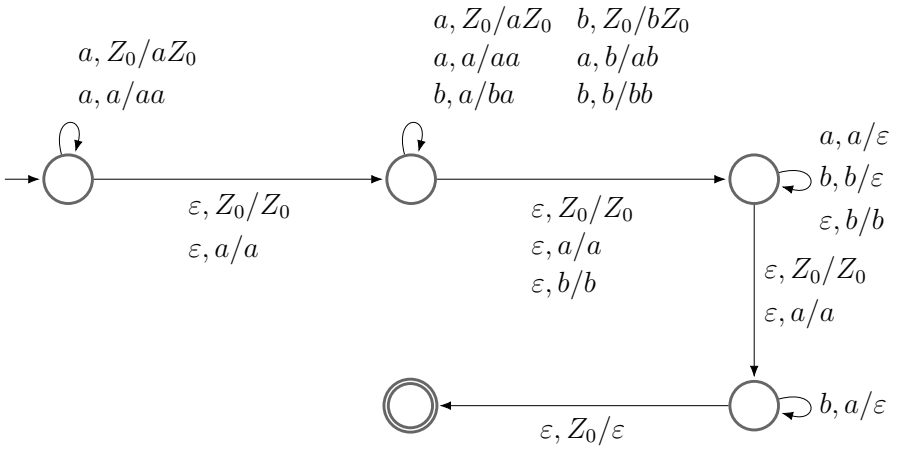
(i) $\{01^n01^n0 \colon n \geq 1\}$

Resolução. O autómato seguinte é determinista e funciona nas duas modalidades de reconhecimento.



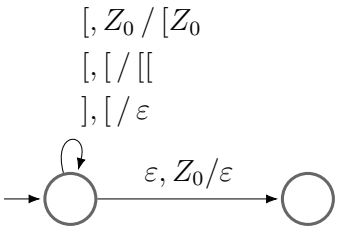
(j) $\{a^n w w^{-1} b^n \colon w \in \{a, b\}^\star, n \geq 0\}$

Resolução. Um AP não determinista é:



6.5.2. Construa um AP, se possível determinista, que reconheça a linguagem das sequências de parêntesis rectos correctamente equilibradas. Por exemplo: [[][[]]][].

Resolução. A estratégia consiste em colocar na pilha cada parêntesis esquerdo que apareça e por cada parêntesis direito que apareça retirar um parêntesis esquerdo do topo da pilha. Assumimos que ε faz parte da linguagem. O autómato reconhece por pilha vazia e é não determinista.



6.5.3. Sejam $L_1 = \{(11)^n(00)^n : n \geq 0\}$, $L_2 = L_1^\star$ e $L_3 = \{\left((11)^n(00)^n\right)^m : m, n \geq 0\}$.

(a) Mostre, usando o teorema da substituição, que L_1 e L_2 são independentes do contexto.

Resolução. Sabemos que a classe das LIC é fechada para substituições por LIC.

As linguagens $\{00\}$ e $\{11\}$ são independentes do contexto pois são finitas. Para além disso, a linguagem $L_4 = \{a^n b^n : n \geq 0\}$ é também independente do contexto. Considerando a substituição definida por $s(a) = \{00\}$ e $s(b) = \{11\}$ concluímos que $L_1 = s(L_4)$ é independente do contexto.

A linguagem L_5 associada à expressão regular a^* é uma LIC pois é regular. Assim, considerando a substituição $s(a) = L_1$ concluímos que $L_2 = s(L_5)$ é uma LIC.

(b) Construa GIC geradoras de L_1 e de L_2 e reduza-as à forma normal de Chomsky.

Resolução. Uma GIC geradora de $L_4 = \{a^n b^n : n \geq 0\}$ é $S \to aSb \mid \varepsilon$. Uma GIC geradora da linguagem $\{00\}$ é $X \to 00$ e uma GIC geradora da linguagem $\{11\}$ é $Y \to 11$. Assim, uma GIC geradora de L_1 é:

$$\begin{gathered} S \to XSY \mid \varepsilon \\ X \to 00 \\ Y \to 11 \end{gathered}$$

Uma GIC geradora da linguagem L_5 associada à expressão regular a^* é $Z \to aZ \mid \varepsilon$. Assim, uma GIC geradora de L_2 é:

$$\begin{gathered} Z \to SZ \mid \varepsilon \\ S \to XSY \mid \varepsilon \\ X \to 00 \\ Y \to 11 \end{gathered}$$

(c) Construa autómatos de pilha reconhecedores de L_1 e de L_2.

Sugestão. Basta usar o método de conversão de uma GIC num autómato de pilha (não determinista) com um único estado, indicado na secção Subsecção 6.2.1.

(d) Mostre que L_3 não é uma LIC.

Sugestão. Seja n o inteiro indicado no lema da bombagem para LIC. Considere a palavra $z = (00)^n(11)^n \in L_3$ de tamanho $4n \geq n$. Dada uma qualquer partição de z da forma $z = uvwxy$, com $|vwx| \leq n$ e $|vx| > 0$, mostre que existe um valor de k tal que $uv^kwx^ky \notin L_3$.

6.5.4. Considere a linguagem $L = \{w \in \{a,b\}^+ : \#_a(w) = \#_b(w)\}$.

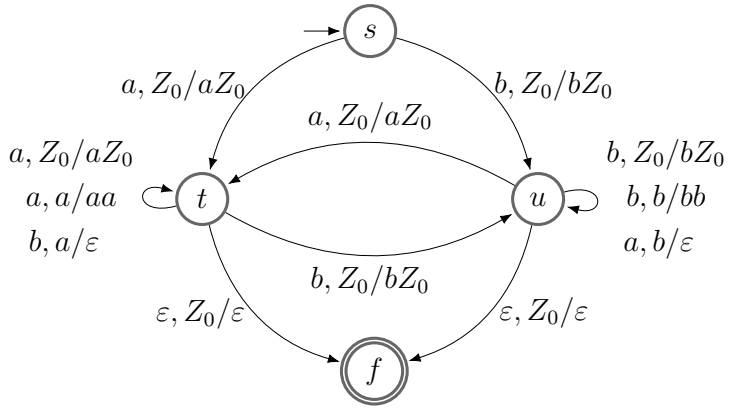
(a) Comente a afirmação "Uma vez que a palavra $ab \in L$ é um prefixo próprio da palavra $abba \in L$, não existe um autómato de pilha que reconheça L por pilha vazia."

Resolução. Qualquer LIC é reconhecível por pilha vazia por um autómato de pilha. Esta linguagem é independente do contexto (como se prova na alínea seguinte). Logo, possui um autómato de pilha que a reconhece por pilha vazia e concluímos que a afirmação é falsa.

No entanto, esta linguagem não é reconhecível por pilha vazia por um autómato de pilha determinista. De facto, uma das condições para que exista um tal autómato é que a linguagem seja livre de prefixos, o que não se verifica.

(b) Construa um autómato de pilha que reconheça L por estados de aceitação.

Resolução. Podemos definir vários autómatos de pilha que reconhecem L. A ideia do AP ilustrado em seguida é: (1) cada estado sabe qual o símbolo $x \in \{a,b\}$ que se está a armazenar na pilha; (2) retiramos um símbolo x da pilha sempre que aparece um símbolo diferente de x; (3) no fim do reconhecimento, a pilha fica vazia se e só se o número de a's é igual ao número de b's.



O AP ilustrado funciona com as duas modalidades de reconhecimento, quer

por estado de aceitação, quer por pilha vazia. Este autómato é não determinista. Por exemplo, no estado t existem as duas possibilidades de transição por ε e pela letra a com Z_0 no topo da pilha.

É ainda possível construir um APD reconhecedor desta linguagem, conforme ilustrado na resolução do exercício Tarefa 6.5.1.e.

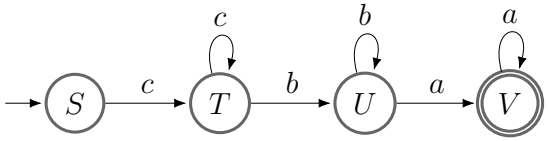
6.5.5. Considere a linguagem L associada à expressão regular $a^+b^+c^+$.

(a) Construa um autómato finito, A^{-1}, com 4 estados, que reconheça L^{-1}.

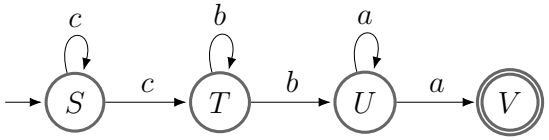
Resolução. Uma expressão regular associada à linguagem L^{-1} é $c^+b^+a^+$ e um AFD (AFND) que reconhece L^{-1} é

$$A^{-1} = (\{S, T, U, V\}, \{a, b, c\}, \delta, S, \{V\})$$

onde a função de transição δ é definida pelo diagrama (AFD) seguinte



ou pelo diagrama seguinte (AFND)



(b) A partir do autómato A^{-1} construído na alínea anterior obtenha uma gramática Linear à Direita geradora de L^{-1}.

Resolução. Uma gramática linear à direita geradora de L^{-1} é

$$G = (\{S, T, U, V\}, \{a, b, c\}, P, S)$$

onde o conjunto de produções é (AFD)

$$\begin{aligned} S &\to cT \\ T &\to cT \mid bU \\ U &\to bU \mid aV \\ V &\to aV \mid \varepsilon \end{aligned}$$

ou (AFND)

$$\begin{aligned} S &\to cS \mid cT \\ T &\to bT \mid bU \\ U &\to aU \mid aV \\ V &\to \varepsilon \end{aligned}$$

(c) A partir da gramática obtida na alínea anterior obtenha uma gramática Linear à Esquerda geradora de L.

Resolução. Uma gramática linear à esquerda geradora de $L = (L^{-1})^{-1}$ é (AFD)

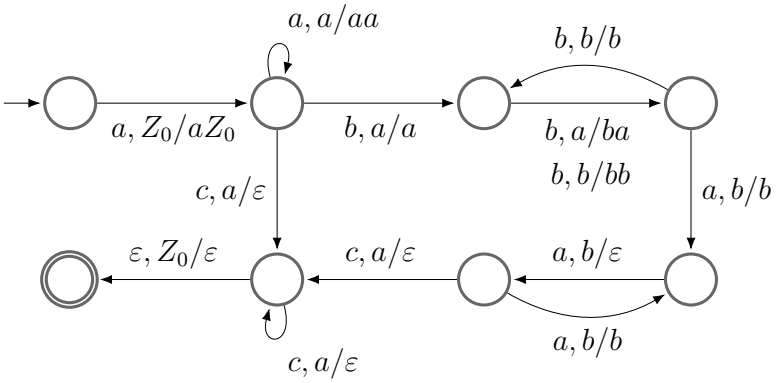
$$\begin{aligned} S &\to Tc \\ T &\to Tc \mid Ub \\ U &\to Ub \mid Va \\ V &\to Va \mid \varepsilon \end{aligned}$$

ou (AFND)

$$\begin{aligned} S &\to Sc \mid Tc \\ T &\to Tb \mid Ub \\ U &\to Ua \mid Va \\ V &\to \varepsilon \end{aligned}$$

(d) Construa um AP reconhecedor de $L_I = \{a^n b^m a^m c^n : n > 0 \text{ e } m \text{ par}\}$.

Resolução. Um autómato de pilha determinista (nas duas modalidades de reconhecimento) é:



(e) Construa um AP reconhecedor de $L \cap L_I$.

Sugestão. Usar o método do autómato produto.

6.5.6. Sabe-se que a classe das LIC não é fechada para a intersecção, isto é, existem LIC L_1 e L_2 tais que $L_1 \cap L_2$ não é uma LIC. Usando este facto, mostre que a classe das LIC também não é fechada para o complementar.

Resolução. Sejam L_1 e L_2 LIC tais que $L_1 \cap L_2$ não é uma LIC. Se a classe das LIC fosse fechada para o complementar, teríamos que $\overline{L_1}$ e $\overline{L_2}$ seriam LIC. Como a classe das LIC é fechada para a união, $\overline{L_1} \cup \overline{L_2}$ seria uma LIC. Assim, se a classe

das LIC fosse fechada para o complementar, também $\overline{\overline{L_1} \cup \overline{L_2}}$ seria uma LIC. Mas, $\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2$ não é uma LIC. Logo, a classe das LIC não pode ser fechada para o complementar.

6.5.7. Mostre que se $L = \mathcal{N}(A)$ para algum APD A então L é livre de prefixos.

Sugestão. Faça a prova por contradição, assumindo que L não é livre de prefixos.

Resolução. Seja $A = (Q, \Sigma, \Gamma, \delta, s, Z_0)$ um APD que reconhece L por pilha vazia. Suponhamos que L não é livre de prefixos. Então existem palavras x, y com $y \neq \varepsilon$ tais que $x \in L$ e $w = xy \in L$ (x é um prefixo próprio de w).

Uma vez que $x \in L$, existe $q \in Q$ tal que $(s, x, Z_0) \vdash^\star (q, \varepsilon, \varepsilon)$ e, como o autómato é determinista, $(s, w, Z_0) = (s, xy, Z_0) \vdash^\star (q, y, \varepsilon)$, pelo que a palavra w não é reconhecida, ou seja, $w \notin L$, o que é uma contradição.

6.5.8. Considere a linguagem $L = \{0^n 1^n : n \geq 0\}$.

(a) Indique uma GIC G geradora de L e, em seguida, prove que $\mathcal{L}(G) = L$.

Resolução. Uma gramática geradora de L é $G = (\{S\}, \{0, 1\}, P, S)$ definida pelas produções $S \to 0S1 \mid \varepsilon$. Vamos mostrar que $L = \mathcal{L}(G)$.

$L \subseteq \mathcal{L}(G)$. Vamos mostrar por indução em $\mathbb{N}_0$ que $\forall n \in \mathbb{N}_0, a^n b^n \in \mathcal{L}(G)$.

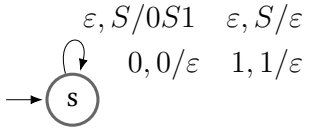
Caso base. Para $n = 0$, $a^n b^n = \varepsilon$ e $S \Rightarrow \varepsilon$ usando a produção $S \to \varepsilon$.

Passo Indutivo. Consideremos por hipótese que $S \stackrel{\star}{\Rightarrow} a^n b^n$. Aplicando a produção $S \to aSb$ obtemos $S \Rightarrow aSb$. Aplicando a hipótese concluímos que $S \stackrel{\star}{\Rightarrow} aa^n b^n b = a^{n+1} b^{n+1}$.

$\mathcal{L}(G) \subseteq L$. Por indução sobre o número de passos de derivação mostra-se que, $\forall n \in \mathbb{N}_0$, se $S \stackrel{\star}{\Rightarrow} w$ então $w = a^n b^n$.

(b) Obtenha um AP com um único estado reconhecedor de L. Esse autómato é determinista? Justifique.

Resolução. A conversão da gramática G produz o seguinte autómato de pilha:



O autómato é não determinista porque $\delta(s, \varepsilon, S) = \{(s, 0S1)(s, \varepsilon)\}$ tem dois elementos.

(c) Construa um APD que reconheça L por estados de aceitação.

Sugestão. Ver o Exemplo 6.1.5.

(d) Caso seja possível construa um APD que reconheça L por pilha vazia, senão indique a razão pela qual tal não é possível.

Resolução. A linguagem L não é livre de prefixos porque $\varepsilon \in L$ é um prefixo próprio das restantes palavras de L. Assim, não existe um APD que reconheça L por pilha vazia.

6.5.9. Sabendo que a linguagem $L_1 = \{a^n b^n c^n \colon n \in \mathbb{N}\}$ não é independente do contexto mostre que a linguagem $L_2 = \{w \in \{a,b,c\}^+ \colon \#_a(w) = \#_b(w) = \#_c(w)\}$ também não é independente do contexto.

Resolução. Consideremos a LR $L = \mathcal{L}(a^+b^+c^+)$. Uma vez que a intersecção de uma LIC com uma LR é uma LIC, se L_2 fosse uma LIC então $L_2 \cap L$ seria uma LIC. Mas $L_2 \cap L = L_1$ que sabemos não ser uma LIC. Logo, L_2 não pode ser uma LIC.

6.5.10. Sabendo que a linguagem $L_1 = \{a^n b^m a^n b^m \colon n, m \geq 0\}$ não é independente do contexto, mostre que $L_2 = \{ww \colon w \in \{a,b\}^\star\}$ também não é uma LIC.

Resolução. Se L_2 fosse uma LIC então a sua intersecção com a linguagem regular L_3 associada à expressão regular $a^*b^*a^*b^*$ seria uma LIC. Mas $L_2 \cap L_3 = L_1$ que não é independente do contexto. Logo, L_2 não pode ser uma LIC.

6.5.11. Sabendo que a linguagem $L_1 = \{ww^{-1} \colon w \in \{a,b\}^\star\}$ é independente do contexto, mostre que a linguagem $L = \{xx^{-1}y^{-1}y \colon x, y \in \{a,b\}^\star\}$ é também independente do contexto.

Resolução. Basta observar que $L = L_1 L_1$ e que a classe das LIC é fechada para a concatenação de linguagens.

Note que $L_2 = \{y^{-1}y \colon y \in \{a,b\}^\star\} = \{yy^{-1} \colon y \in \{a,b\}^\star\} = L_1$. Mesmo sem usar esta observação, chegamos à mesma conclusão uma vez que a linguagem reversa de L_2 é L_1 e a classe das LIC é fechada para esta operação.

6.5.12. Seja $A = \big(Q_A, \Sigma, \Gamma, \delta_A, s_A, Z_0, F_A\big)$ um AP reconhecedor de uma LIC L_I, por estados de aceitação. Seja $B = \big(Q_B, \Sigma, \delta_B, s_B, F_B\big)$ um AFD reconhecedor de uma LR L_R.

Mostre que a linguagem $L_I \cap L_R$ é reconhecida pelo autómato de pilha $A_\cap = (Q_A \times Q_B, \Sigma, \Gamma, \delta_\cap, s_\cap, F_\cap)$, produto dos autómatos A e B, onde:

- $s_\cap = (s_A, s_B)$;
- $F_\cap = F_A \times F_B$;
- Se $(p_A, \gamma) \in \delta_A(q_A, a, Z)$ e $p_B = \delta_B(q_B, a)$, para $p_A, q_A \in Q_A$, $p_B, q_B \in Q_B$, $a \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$ e $\gamma \in \Gamma^\star$ então

$$((p_A, p_B), \gamma) \in \delta_\cap((q_A, q_B), a, Z).$$

Sugestão. Use indução estrutural sobre $w \in \Sigma^\star$.

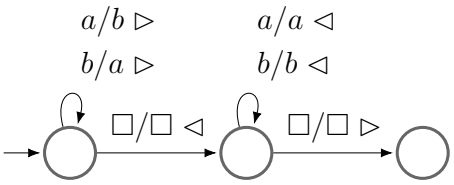
7 Máquinas de Turing

7.7 Exercícios

7.7.1. Considere o alfabeto $\Sigma = \{a, b\}$. Desenvolva máquinas de Turing para:

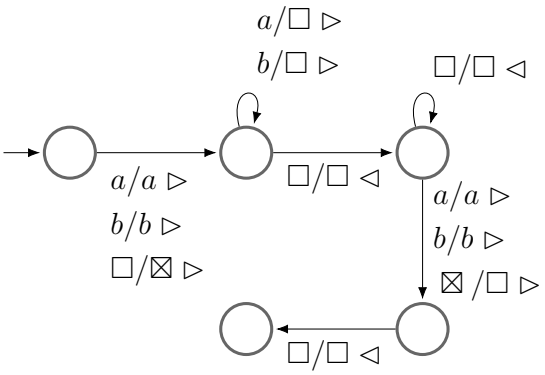
(a) Trocar a´s por b´s numa palavra;

Resolução. A MT seguinte troca a´s por b´s e volta ao início da palavra.



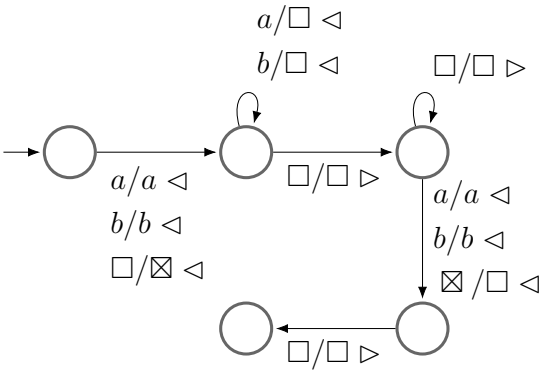
(b) Apagar a palavra situada à direita da posição actual;

Resolução. A MT seguinte apaga a palavra à direita da posição actual (até encontrar uma célula vazia) e volta à posição inicial.



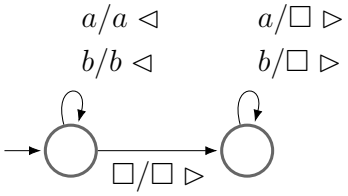
(c) Apagar a palavra situada à esquerda da posição actual;

Resolução. A MT seguinte apaga a palavra à esquerda da posição actual (até encontrar uma célula vazia) e volta à posição inicial.



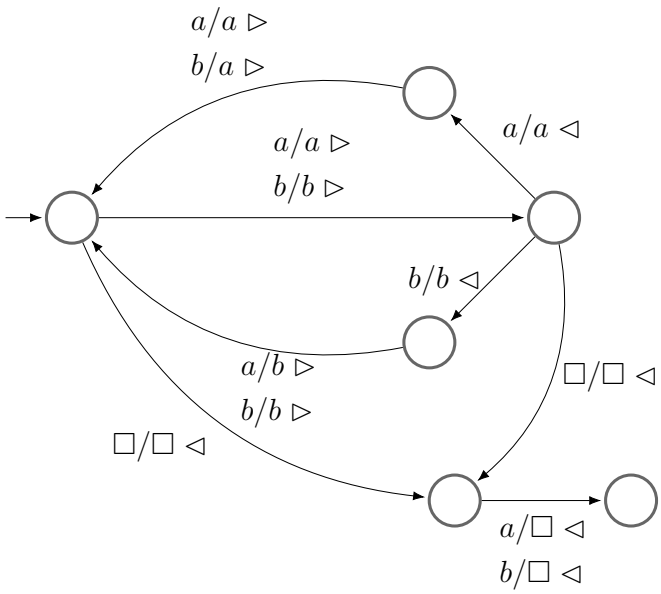
(d) Apagar a palavra inscrita na fita, assumindo que a posição actual corresponde a um qualquer símbolo da palavra;

Resolução. A MT seguinte começa por mover a cabeça de leitura/escrita para a posição do primeiro símbolo da palavra e, em seguida, apaga a palavra à direita dessa posição (até encontrar uma célula vazia). Termina na posição seguinte ao último símbolo da palavra original.

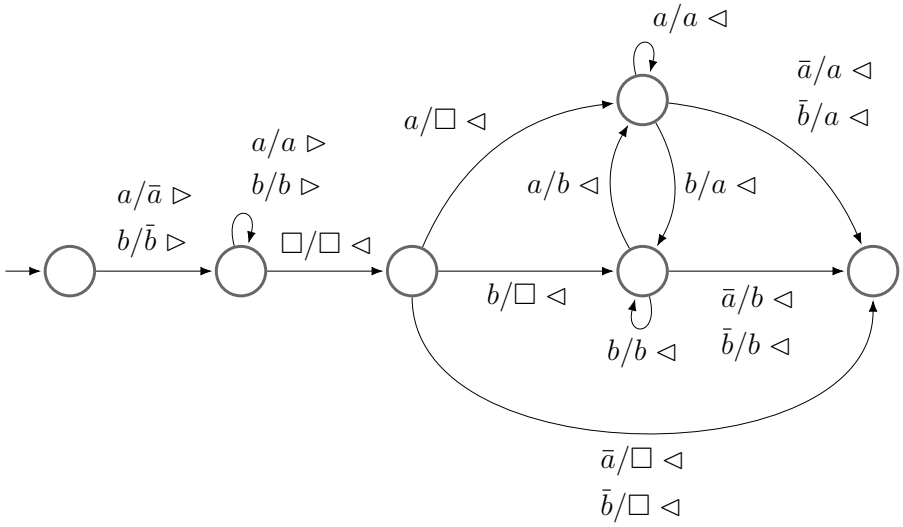


(e) Apagar o símbolo de uma palavra inscrito na posição actual, deslocando para a esquerda a palavra situada à direita da posição actual;

Resolução. Para cada posição, a seguinte MT copia o símbolo que está na posição seguinte (desloca-se uma posição para a direita, memoriza o símbolo nessa posição e, em seguida, volta à posição anterior). A máquina pára no último símbolo da nova palavra.



A MT seguinte começa por mover a cabeça de leitura/escrita para o final da palavra e, em seguida, desloca-se para a esquerda, substituindo cada símbolo pelo que se lhe seguia (memorizando o símbolo). A máquina pára na mesma posição em que começou.

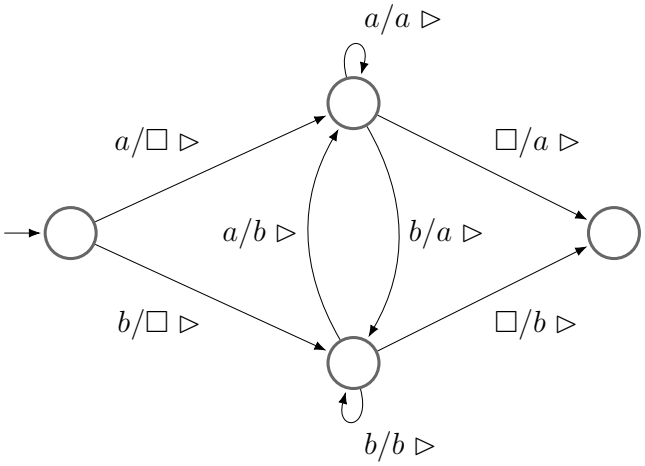


(f) Apagar o símbolo na posição actual, deslocando para a direita a palavra situada à esquerda da posição actual;

Solução. Ver a solução da alínea anterior e adaptar.

(g) Inserir um símbolo branco na posição actual, deslocando para a direita a palavra situada à direita da posição actual (inclusive);

Resolução. A MT seguinte começa por inserir o símbolo branco na posição actual, memorizando o símbolo que lá estava. Em seguida, substitui cada um dos símbolos para a direita pelo anterior. A máquina pára após o último símbolo da palavra modificada.

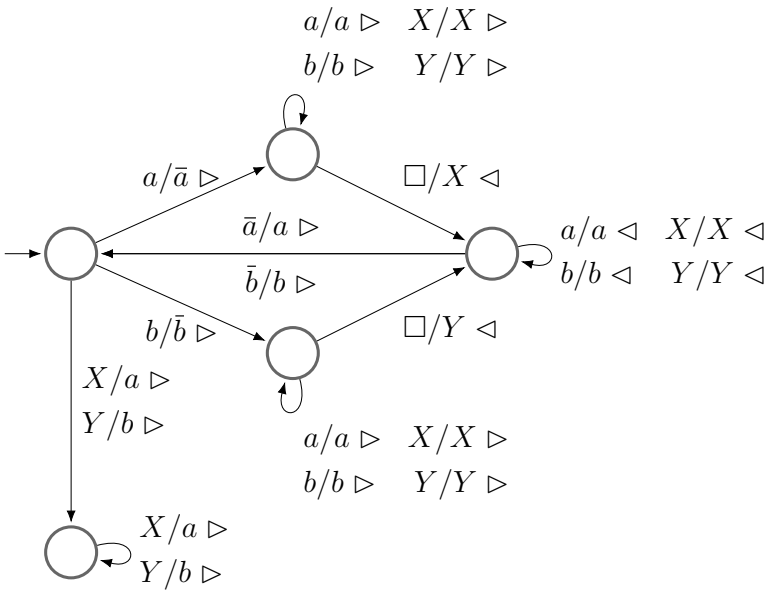


(h) Inserir um símbolo branco na posição actual, deslocando para a esquerda a palavra situada à esquerda da posição actual (inclusive);

Resolução. Ver a alínea anterior e adaptar.

(i) Duplicar uma palavra;

Resolução. A MT seguinte marca cada letra a copiar, desloca-se para o final, inserindo o símbolo X quando copia a letra a e o símbolo Y quando copia a letra b. Em seguida, volta atrás, desmarca a letra marcada e avança para a próxima letra a copiar. Após todas as letras terem sido copiadas, a máquina substitui cada X e cada Y pela letra correspondente.



(j) Reverter uma palavra.

Solução. Uma solução simples consiste em construir uma cópia à esquerda da palavra original (copiando a palavra original da esquerda para a direita) e no final apagar a palavra original.

Uma outra solução consiste em construir uma cópia à direita da palavra original separada por um marcador (copiando a palavra original da direita para a esquerda) e no final apagar a palavra original e o marcador.

É ainda possível construir a reversa no mesmo espaço ocupado pela palavra original, trocando a primeira letra não marcada com a última letra não marcada, tendo em atenção o caso em que o tamanho da palavra é ímpar.

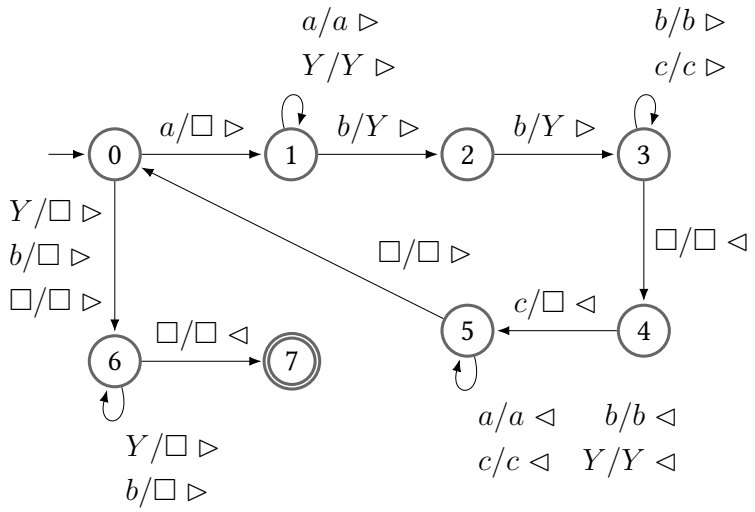
7.7.2. Mostre que as seguintes linguagens são recursivas, construindo máquinas de Turing que as decidam.

(a) $\{a^n b^m c^n : m \geq 2n \geq 0\}$;

Resolução. A estratégia usada na construção da MT seguinte consiste em repetir o seguinte processo:

por cada um dos a´s que é apagado marca dois b´s com Y e apaga um c, voltando ao início;

No fim da etapa anterior, se a palavra pertence à linguagem apenas ficam na fita Y´s e b´s, os quais são também apagados.



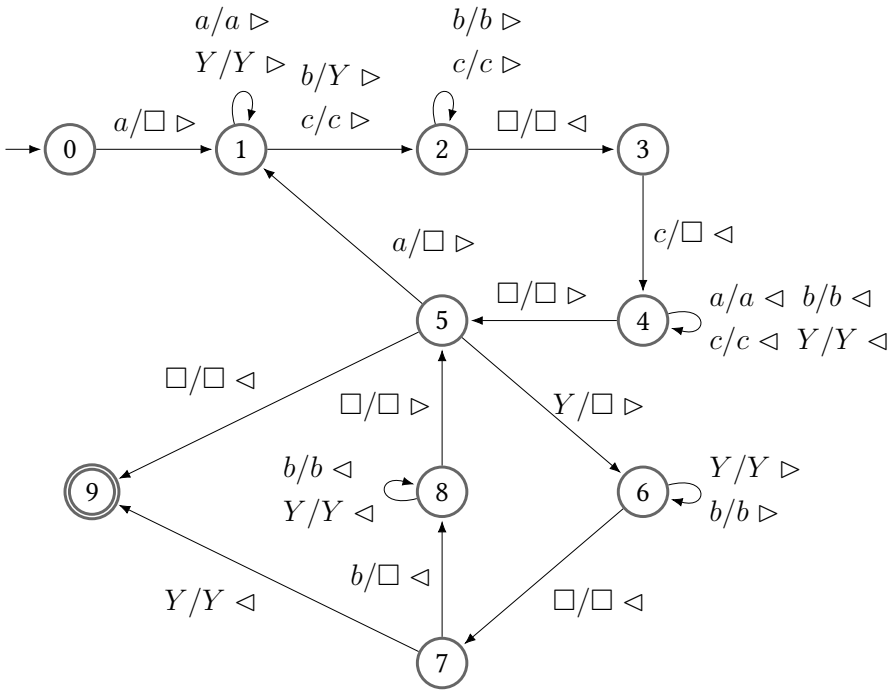
(b) $\{a^n b^m c^n : 0 \leq n < 2n\}$.

Resolução. A estratégia usada na construção da MT seguinte consiste em repetir o seguinte processo:

por cada um dos a´s que é apagado marca um b´s com Y e apaga um c, voltando ao início.

No fim da etapa anterior, se a palavra pertence à linguagem apenas ficam na fita Y´s e b´s.

Se o número de Y´s for menor que o número de b´s então o número inicial de b´s é menor do que $2n$ e a palavra é reconhecida.



7.7.3. Construa máquinas de Turing para:

(a) converter um número em binário num número em unário;

Sugestão. Comece por desenvolver uma subrotina para calcular o dobro de um número em unário. Em seguida, use as propriedades $d(w0) = 2d(w)$ e $d(w1) = 2d(w) + 1$ para converter o número da esquerda para a direita. Exemplo:

$$\begin{gathered}\bar{1}011\#1\\ \bar{1}\bar{0}11\#11\\ \bar{1}\bar{0}\bar{1}1\#11111\\ \bar{1}\bar{0}\bar{1}\bar{1}\#11111111111\end{gathered}$$

(b) converter um número em unário num número em binário;

Sugestão. Realize sucessivas divisões por 2 (marcando um em cada dois 1´s) tendo em atenção se o resto é 0 ou 1. Para simplificar, coloque o resultado à esquerda do número dado. Exemplo:

$$\begin{gathered}\#11111111111\\ 1\#X1X1X1X1X1X\\ 11\#XXX1XXX1XXX\\ 011\#XXXXXXX1XXX\\ 1011\#XXXXXXXXXXX\end{gathered}$$

(c) somar dois números em unário;

Sugestão. É um problema simples de cópia das duas sequências de 1´s;

(d) subtrair dois números em unário;

Sugestão. Basta ir copiando e marcando 1´s até que uma das sequências se esgote.

(e) somar dois números em binário;

Sugestão. Neste problema é preciso ir calculando o bit de transporte para a próxima posição, começando na direita (bits menos significativos). É também preciso ter em atenção que os números podem ter tamanhos diferentes. Por exemplo, dados $x = 1110$ e $y = 0011$, vem que $t_0 = 0$, $z_0 = x_0+y_0+t_0 = 0+1+0 = 1$ e $t_1 = 0$; $z_1 = x_1 + y_1 + t_1 = 1 + 1 + 0 = 0$ e $t_2 = 1$; $z_2 = x_2+y_2+t_2 = 1+0+1 = 0$ e $t_3 = 1$; $z_3 = x_3+y_3+t_3 = 1+0+1 = 0$ e $t_4 = 1$; Logo, $z = x + y = 10001$.

(f) subtrair dois números em binário;

Sugestão. Problema semelhante ao da alínea anterior. Como representar números negativos?

7.7.4. Construa máquinas de Turing para decidir as seguintes linguagens (especifique a representação escolhida, estabeleça a caracterização formal da MT, tabela e diagrama de transições):

(a) $(a + b)a(a + b)^*$

Sugestão. Saltar a primeira letra, verificar se a segunda letra é um a e parar (aceitando ou rejeitando).

(b) $\{w \in \{a, b\}^\star \colon w = w^{-1}\}$

Sugestão. Ir comparando a letra mais à esquerda ainda não marcada com a letra mais à direita ainda não marcada (marcando as duas). Preste atenção aos palíndromos ímpares!

(c) $\{ww^{-1} \colon w \in \{a, b\}^\star\}$

Sugestão. Semelhante à alínea anterior. Note que as palavras da linguagem são apenas os palíndromos pares!

(d) $\{w \in \{0, 1\}^\star \colon \#_0(w) = \#_1(w)\}$

Sugestão. Implemente o seguinte algoritmo:

(1) Procurar um 0 não marcado;

Se não encontra então

Procurar um 1 não marcado

Se não encontra então aceita

senão rejeita;

senão marcar o 0 encontrado;

(2) Procurar um 1 não marcado;

Se não encontra então rejeita;

senão marca o 1 encontrado;

volta ao passo 1

(e) $\{a^n b^n a^n : n \geq 1\}$

Sugestão. Para cada a não marcado antes dos b´s marcar um b e marcar um a depois dos b´s. Repetir, tendo em atenção que no decorrer deste processo qualquer uma das 3 sequências se pode esgotar. Teste a máquina para as palavras ε, a, b, ab, ba, aa, $aaba$, $abba$, $abaa$.

(f) $\{a^n b^n c^n : n \geq 0\}$

Sugestão. Semelhante à alínea anterior. Tenha em atenção que a palavra vazia faz parte da linguagem!

7.7.5. Construa máquinas de Turing que cumpram as seguintes especificações:

(a) $1^n \vdash 1^{n^2}$;

(b) $1^n 0 1^m \vdash 1^n 0 1^m 0 1^{nm}$;

(c) $1^n \vdash 1^{2^n}$;

(d) $a^n b^m \vdash c^{\min(m,n)}$;

(e) $a^n b^m \vdash c^{\max(m,n)}$

(f) $1^n \vdash F(n)$, onde $F(n)$ é a sequência de Fibonnaci definida por: $F(0) = 0$, $F(1) = 1$ e $F(n) = F(n-1) \cdot F(n-2)$, para $n \geq 2$, onde $\cdot$ denota a operação de concatenação de palavras binárias.

7.7.6. Averigue se a classe das linguagens recorrentes (recorrentemente enumeráveis) é fechada para:

(a) a concatenação;

Resolução. Sejam L_1 e L_2 duas linguagens recorrentes. Então existem máquinas de Turing M_1 e M_2 que as decidem.

Dada uma qualquer palavra w, pretendemos averiguar se existem palavras

$x \in L_1$ e $y \in L_2$ tais que $w = xy$. Assim, basta definir uma máquina de Turing M que

Para cada $i = 0, 1, \ldots |w|$

Particiona a palavra w no prefixo x de tamanho i e no correspondente sufixo y de tamanho $|w| - i$;

Simula M_1 com x;

Se M_1 aceita x então

Simula M_2 com y

Se M_2 aceita y

então M pára e aceita w;

M pára e rejeita w.

Verifique se a construção anterior ainda funciona quando as linguagens são apenas recorrentemente enumeráveis.

(b) a operação estrela de Kleene;

Sugestão. Resolução semelhante à da alínea anterior, considerando agora todas as possíveis partições de uma palavra w em n partes ($n = 1, 2, \ldots, |w|$).

(c) imagens direta e inversa por intermédio de um homomorfismo.

Sugestão. A classe das linguagens recorrentemente enumeráveis é fechada para homomorfismos e para a imagem inversa por intermédio de um homomorfismo. No entanto, a classe das linguagens recorrentes não é fechada para homomorfismos (embora seja fechada para a imagem inversa por intermédio de um homomorfismo).

7.7.7. Considere linguagens recorrentemente enumeráveis $L_1, L_2, \ldots, L_n$, com $n \geq 2$, definidas sobre o mesmo alfabeto Σ, as quais formam uma partição de $\Sigma^\star$ (são disjuntas duas a duas e a sua união é $\Sigma^\star$). Nestas condições, mostre que são todas recorrentes.

Resolução. Para cada $i = 1, 2, \ldots, n$, seja M_i uma máquina de Turing que reconhece L_i. Vamos mostrar qe a linguagem L_j, $j = 1, 2, \ldots, n$ é recorrente. Definimos uma máquina de Turing que executa em paralelo as máquinas M_1, M_2, ..., M_n com uma palavra w (simula o primeiro passo de cada uma das máquinas com $w$depois simula os dois primeiros passos de cada uma das máquinas com w, etc.). Ora, ao fim de um número finito de passos, uma das máquinas terá parar e aceitar w. Assim, se a máquina que parou for M_j então o simulador pára e aceita w, senão pára e rejeita w.

7.7.8. Uma MT entra em ciclo com uma palavra w se a sequência de configurações associada à palavra w tem uma configuração repetida (e portanto um número infinito de repetições dessa configuração).

Seja $L \subseteq \Sigma^\star$ uma linguagem reconhecida por uma máquina de Turing M tal que para qualquer palavra $w \in \Sigma^\star$ a máquina M ou aceita w ou rejeita w ou entra ciclo com w. Mostre que L é decidível.

Sugestão. Basta construir uma MT que detecte configurações repetidas, com o auxílio de uma MT com duas fitas: uma fita principal usada para simular a execução da máquina original com uma palavra e uma fita auxiliar para guardar as sucessivas configurações.

No final de cada passo, o simulador acrescenta a configuração à fita auxiliar e verifica, em seguida, se esta contém duas configurações repetidas. Em caso afirmativo, pára e rejeita a palavra.

7.7.9. Mostre que as linguagens seguintes são decidíveis.

(a) $\{\langle A\rangle : A \text{ é um AFD e } \mathcal{L}(A) \text{ é infinita}\}$.

Sugestão. Como determinar se um estado de aceitação é atingível a partir do estado inicial? Como determinar se existe um ciclo que começa e termina num estado de aceitação?

(b) Sendo $L = \{w \in \{a,b\}^\star : \#_a(w) \text{ é ímpar}\}$,

$$\{\langle A\rangle : A \text{ é um AFD sobre } \{a,b\} \text{ e } \mathcal{L}(A) \cap L = \emptyset\}.$$

Sugestão. A linguagem L é regular. Logo, também $\mathcal{L}(A) \cap L$ é regular ....

(c) Considerando o alfabeto binário, $\Sigma = \{0,1\}$,

$$\{\langle G\rangle : G \text{ é uma GIC e } \{1\}^\star \cap \mathcal{L}(G) \neq \emptyset\}.$$

Sugestão. Como é que se constrói uma GIC geradora da intersecção de uma linguagem regular com uma GIC?

7.7.10. Desenvolva um procedimento que, dado $n \in \mathbb{N}$, permita enumerar os elementos de $\mathbb{Z}^n$ de uma forma sistemática. Por exemplo, um procedimento que permita listar, para $k = 0, 1, 2, \ldots$, todos os elementos do hipercubo $H_k \subset \mathbb{Z}^n$ de centro na origem e com lados de comprimento $2k$.

Sugestão. Investigue o método combinatório das estrelas e barras ("stars and bars").

7.7.11. Mostre que o problema de aceitação

$$\text{ACC}_{\text{MT}} = \{\langle M, w\rangle : M \text{ é uma máquina de Turing que aceita } w\}$$

é indecidível.

Resolução. Suponhamos que ACC_{MT} é decidível (recorrente). Então existe uma máquina de Turing M_A tal que, para cada entrada $\langle M, w\rangle$,

(a) Se M aceita w então M_A aceita $\langle M, w\rangle$ (pára num estado de aceitação);

(b) Se M não aceita w (pára num estado de rejeição ou não pára) então M_A rejeita $\langle M, w\rangle$ (pára num estado de rejeição).

Com base na máquina M_A, definimos a máquina M_X tal que, para cada entrada $\langle M\rangle$,

- Simula M_A com $\langle M, \langle M\rangle\rangle$;
- Se M_A aceita $\langle M, \langle M\rangle\rangle$ então M_X rejeita $\langle M\rangle$;
- Se M_A rejeita $\langle M, \langle M\rangle\rangle$ então M_X aceita $\langle M\rangle$.

Quando se executa M_X com a palavra $\langle M_X\rangle$ verificamos que M_X aceita $\langle M_X\rangle$ se e só se M_A rejeita $\langle M_X, \langle M_X\rangle\rangle$ se e só se M_X rejeita $\langle M_X\rangle$. Isto é uma Contradição! Logo, a máquina M_A não pode existir e o problema ACC_{MT} é indecidível.

7.7.12. Por redução do problema ACC_{MT} mostre que os seguintes problemas são indecidíveis:

(a) $\text{E}_{\text{MT}} = \{\langle M\rangle : M \text{ é uma máquina de Turing e } \mathcal{L}(M) = \emptyset\}$;

Resolução. Admitamos que E_{MT} é decidível. Então existe uma MT M_{E} tal que, com a entrada $\langle M\rangle$,

- se $\mathcal{L}(M) = \emptyset$ então M_{E} aceita $\langle M\rangle$;
- se $\mathcal{L}(M) \neq \emptyset$ então M_{E} rejeita $\langle M\rangle$.

Para cada palavra w e MT M seja M_w a MT tal que com uma entrada x,

- se $x \neq w$ então M_w rejeita x;
- se $x = w$ então M_w simula M com a entrada w:
  - se M aceita w então M_w aceita x;
  - senão M_w não aceita x (pára e rejeita ou não pára).

Consideremos a MT M_{A} tal que, com a entrada $\langle M, w\rangle$ para alguma MT M e palavra w,

(a) Usa a descrição da máquina M e a palavra w para construir a descrição $\langle M_w\rangle$ da máquina M_w.

(b) Simula a máquina M_E com a entrada $\langle M_w \rangle$.

(c)
- Se M_E aceita $\langle M_w \rangle$ então M_{A} rejeita $\langle M, w \rangle$;
- Se M_E rejeita $\langle M_w \rangle$ então M_{A} aceita $\langle M, w \rangle$.

Qual é o resultado da execução da máquina M_{A} com $\langle M, w \rangle$?

Se $w \in \mathcal{L}(M)$ então $\mathcal{L}(M_w) = \{w\}$ e a máquina M_{E} pára e rejeita M_w. Logo, a máquina M_{A} pára e aceita $\langle M, w \rangle$.

Se $w \notin \mathcal{L}(M)$ então $\mathcal{L}(M_w) = \emptyset$ e a máquina M_{E} pára e aceita M_w. Logo, a máquina M_{A} pára e rejeita $\langle M, w \rangle$.

Mas então a linguagem ACC_{MT} seria decidível o que não é o caso.

(b) $\text{REG}_{\text{MT}} = \{\langle M \rangle : M \text{ é uma máquina de Turing e } \mathcal{L}(M) \text{ é regular}\}$;

Resolução. Admitamos que REG_{MT} é decidível. Então existe uma MT M_{REG} tal que, com a entrada $\langle M \rangle$,

- se $\mathcal{L}(M)$ é regular então M_{REG} aceita $\langle M \rangle$;
- se $\mathcal{L}(M)$ não é regular então M_{REG} rejeita $\langle M \rangle$.

Para cada palavra w e MT M seja M_w a MT tal que com uma entrada x,

- se x é da forma $a^n b^n$ então M_w aceita x;
- se x não é da forma $a^n b^n$ então M_w simula M com a entrada w:
  - se M aceita w então M_w aceita x;
  - senão M_w não aceita x (pára e rejeita ou então não pára porque M não pára com w).

Consideremos MT M_{A} tal que, com a entrada $\langle M, w \rangle$ para alguma MT M e palavra w,

(a) Usa a descrição da máquina M e a palavra w para construir a descrição $\langle M_w \rangle$ da máquina M_w.

(b) Simula a máquina M_{REG} com a entrada $\langle M_w \rangle$.

(c)
- Se M_{REG} aceita $\langle M_w \rangle$ então M_{A} aceita $\langle M, w \rangle$;
- Se M_{REG} rejeita $\langle M_w \rangle$ então M_{A} rejeita $\langle M, w \rangle$.

Qual é o resultado da execução da máquina M_{A} com $\langle M, w \rangle$?

Se $w \in \mathcal{L}(M)$ então $\mathcal{L}(M_w) = \Sigma^\star$, que é uma linguagem regular, e a máquina M_{REG} pára e aceita M_w. Logo, a máquina M_{A} pára e aceita $\langle M, w \rangle$.

Se $w \notin \mathcal{L}(M)$ então $\mathcal{L}(M_w) = \{a^n b^n : n \geq 0\}$, que é uma linguagem não regular, e a máquina M_{REG} pára e rejeita M_w. Logo, a máquina M_{A} pára e rejeita $\langle M, w \rangle$.

Mas então a linguagem ACC_{MT} seria decidível o que não é o caso.

(c) $\text{FIN}_{\text{MT}} = \{\langle M\rangle : M \text{ é uma máquina de Turing e } \mathcal{L}(M) \text{ é finita}\}$;

Resolução. Admitamos que FIN_{MT} é decidível. Então existe uma MT M_{FIN} tal que, com a entrada $\langle M\rangle$,

- se $\mathcal{L}(M)$ é finita então M_{FIN} aceita $\langle M\rangle$;
- se $\mathcal{L}(M)$ não é infinita então M_{FIN} rejeita $\langle M\rangle$.

Para cada palavra w e MT M seja M_w a MT tal que com uma entrada x, simula M com a entrada w:

- se M aceita w então M_w aceita x;
- senão M_w não aceita x (pára e rejeita ou então não pára porque M não pára com w).

Consideremos a MT M_{A} tal que, com a entrada $\langle M, w\rangle$ para alguma MT M e palavra w,

(a) Usa a descrição da máquina M e a palavra w para construir a descrição $\langle M_w\rangle$ da máquina M_w.

(b) Simula a máquina M_{FIN} com a entrada $\langle M_w\rangle$.

(c)
- Se M_{FIN} aceita $\langle M_w\rangle$ então M_{A} rejeita $\langle M, w\rangle$;
- Se M_{FIN} rejeita $\langle M_w\rangle$ então M_{A} aceita $\langle M, w\rangle$.

Qual é o resultado da execução da máquina M_{A} com $\langle M, w\rangle$?

Se $w \in \mathcal{L}(M)$ então $\mathcal{L}(M_w) = \Sigma^\star$, que é uma linguagem infinita, e a máquina M_{FIN} pára e rejeita M_w. Logo, a máquina M_{A} pára e aceita $\langle M, w\rangle$.

Se $w \notin \mathcal{L}(M)$ então $\mathcal{L}(M_w) = \emptyset$, que é uma linguagem finita, e a máquina M_{FIN} pára e aceita M_w. Logo, a máquina M_{A} pára e rejeita $\langle M, w\rangle$.

Mas então a linguagem ACC_{MT} seria decidível o que não é o caso.

(d) $\text{ALL}_{\text{MT}} = \{\langle M\rangle : M \text{ é uma máquina de Turing e } \mathcal{L}(M) = \Sigma^\star\}$.

Resolução. Admitamos que ALL_{MT} é decidível. Então existe uma MT M_{ALL} tal que, com a entrada $\langle M\rangle$,

- se $\mathcal{L}(M) = \Sigma^\star$ então M_{ALL} aceita $\langle M\rangle$;
- se $\mathcal{L}(M) \neq \Sigma^\star$ então M_{ALL} rejeita $\langle M\rangle$.

Para cada palavra w e MT M seja M_w a MT tal que com uma entrada x, simula M com a entrada w:

- se M aceita w então M_w aceita x;

- senão M_w não aceita x (pára e rejeita ou então não pára porque M não pára com w).

Consideremos a MT M_{A} tal que, com a entrada $\langle M, w\rangle$ para alguma MT M e palavra w,

(a) Usa a descrição da máquina M e a palavra w para construir a descrição $\langle M_w\rangle$ da máquina M_w.

(b) Simula a máquina M_{ALL} com a entrada $\langle M_w\rangle$.

(c)
- Se M_{ALL} aceita $\langle M_w\rangle$ então M_{A} aceita $\langle M, w\rangle$;
- Se M_{ALL} rejeita $\langle M_w\rangle$ então M_{A} rejeita $\langle M, w\rangle$.

Qual é o resultado da execução da máquina M_{A} com $\langle M, w\rangle$?

Se $w \in \mathcal{L}(M)$ então $\mathcal{L}(M_w) = \Sigma^\star$ e a máquina M_{ALL} pára e aceita M_w. Logo, a máquina M_{A} pára e aceita $\langle M, w\rangle$.

Se $w \notin \mathcal{L}(M)$ então $\mathcal{L}(M_w) = \emptyset$ e a máquina M_{ALL} pára e rejeita M_w. Logo, a máquina M_{A} pára e rejeita $\langle M, w\rangle$.

Mas então a linguagem ACC_{MT} seria decidível o que não é o caso.

7.7.13. Por redução do problema E_{MT} mostre que

$$\text{EQ}_{\text{MT}} = \{\langle M_1, M_2\rangle : M_1 \text{ e } M_2 \text{ são MT e } \mathcal{L}(M_1) = \mathcal{L}(M_2)\}$$

é indecidível.

Resolução. Suponhamos que EQ_{MT} é decidível. Então existe uma MT M_{EQ} tal que dadas duas quaisquer máquinas de Turing M_1 e M_2:

- se $\mathcal{L}(M_1) = \mathcal{L}(M_2)$ então M_{EQ} pára e aceita $\langle M_1, M_2\rangle$;
- se $\mathcal{L}(M_1) \neq \mathcal{L}(M_2)$ então M_{EQ} pára e rejeita $\langle M_1, M_2\rangle$.

Seja $M_\emptyset$ uma qualquer máquina de Turing que pára com todas as entradas e rejeita (ou seja, $\mathcal{L}(M_\emptyset) = \emptyset$).

Consideremos a MT M_{E} tal que, dada uma qualquer codificação de uma máquina de Turing $\langle M\rangle$:

- Simula M_{EQ} com a entrada $\langle M, M_\emptyset\rangle$;
- Se M_{EQ} aceita $\langle M, M_\emptyset\rangle$ então M_{E} aceita $\langle M\rangle$;
- Se M_{EQ} rejeita $\langle M, M_\emptyset\rangle$ então M_{E} rejeita $\langle M\rangle$.

Assim definida, a máquina M_{E} decide se $\mathcal{L}(M) = \mathcal{L}(M_\emptyset) = \emptyset$, isto é, decide E_{MT}, que sabemos ser não decidível. Logo, a máquina de Turing, M_{E} não pode existir e portanto EQ_{MT} também não pode ser decidível.

7.7.14. Mostre que a linguagem $\overline{\text{HALT}}$ não é recorrentemente enumerável.

Sugestão. Basta argumentar de forma análoga à usada na Secção 7.5 para demonstrar que $\overline{\text{ACC}_{\text{MT}}}$ não é recorrentemente enumerável.

7.7.15. Considere o problema

$$\text{EQ}_{\text{MT}} = \{\langle M_1, M_2\rangle \colon M_1 \text{ e } M_2 \text{ são MT e } \mathcal{L}(M_1) = \mathcal{L}(M_2)\}.$$

(a) Mostre que EQ_{MT} não é recorrentemente enumerável.

Resolução. Basta reduzir a linguagem ACC_{MT}, que sabemos ser não recorrentemente enumerável à linguagem EQ_{MT}.

Condiremos a seguinte função computável que transforma uma instância de ACC_{MT} da forma $\langle M, w\rangle$ numa instância de EQ_{MT} da forma $\langle M_1, M_2\rangle$, onde:

- M_1 é uma máquina de Turing que aceita qualquer palavra, ou seja, $\mathcal{L}(M_1) = \Sigma^\star$.
- M_2 é a máquina de Turing que, dada uma qualquer palavra x, simula M com w e se M aceita w então M_2 aceita x. (Se M não aceita w então M_2 também não aceita x.)

Facilmente se verifica que $\langle M, w\rangle \in \text{ACC}_{\text{MT}}$ se e só se $\langle M_1, M_2\rangle \in \text{EQ}_{\text{MT}}$.

(b) Mostre que $\overline{\text{EQ}_{\text{MT}}}$ não é recorrentemente enumerável.

Resolução. Basta reduzir a linguagem ACC_{MT}, que sabemos ser não recorrentemente enumerável à linguagem $\overline{\text{EQ}_{\text{MT}}}$.

Condiremos a seguinte função computável que transforma uma instância de ACC_{MT} da forma $\langle M, w\rangle$ numa instância de $\overline{\text{EQ}_{\text{MT}}}$ da forma $\langle M_1, M_2\rangle$, onde:

- M_1 é uma máquina de Turing que rejeita todas as palavras, ou seja, $\mathcal{L}(M_1) = \emptyset$.
- M_2 é a máquina de Turing que, dada uma qualquer palavra x, simula M com w e se M aceita w então M_2 aceita x. (Se M não aceita w então M_2 também não aceita x.)

Facilmente se verifica que $\langle M, w\rangle \in \text{ACC}_{\text{MT}}$ se e só se $\langle M_1, M_2\rangle \in \overline{\text{EQ}_{\text{MT}}}$.

7.7.16. Aplique o Teorema de Rice para provar que as linguagens referidas no Exercício 7.7.12 não são recorrentes.

7.7.17. Considere a linguagem $L = \{ww \colon w \in \{a, b\}^\star\}$.

Este exercício é de longa resolução. Teste o funcionamento das máquinas de Turing que desenhar com palavras pequenas, por exemplo, ε, a, aa, ab, $abab$, $abba$, ...

(a) Construa uma máquina de Turing não determinista, M_N que reconheça L,

mas que não decida L.

Sugestão. Desenhe uma MT que: (i) de forma não determinista encontre o meio da palavra colocada na fita, marcando a primeira parte da palavra; (ii) verifique se a primeira parte (marcada) é igual à segunda parte (não marcada). Para além disso, como queremos que a máquina não decida a linguagem, esta não deverá parar para pelo menos uma palavra que não seja da forma ww.

(b) Esboce as árvores de derivação pela máquina M_N associadas às palavras $abbabb$ e $abbaba$.

Sugestão. Uma vez que as árvores de derivação não vão ser finitas, concentre-se apenas nos ramos que conduzem à aceitação / rejeição das palavras.

(c) Construa uma máquina de Turing não determinista que decida L.

Sugestão. Basta modificar a máquina anterior tratando os casos em que não pára, de modo a que, nesses casos, páre num estado de rejeição.

(d) Construa uma máquina de Turing determinista que decida L.

Sugestão. Desenhe uma MT que: (i) calcule de forma determinista o meio da palavra, por exemplo, transformando

$$w = s_1 s_2 \cdots s_n s_{n+1} \cdots s_{2n-2} s_{2n-1} s_{2n}$$

em

$$w = \bar{s}_1 \bar{s}_2 \cdots \bar{s}_n \bar{\bar{s}}_{n+1} \cdots \bar{\bar{s}}_{2n-2} \bar{\bar{s}}_{2n-1} \bar{\bar{s}}_{2n};$$

(ii) verifique se a primeira parte é igual à segunda parte.