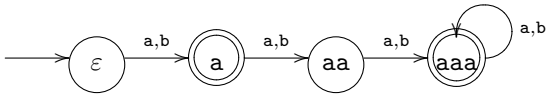
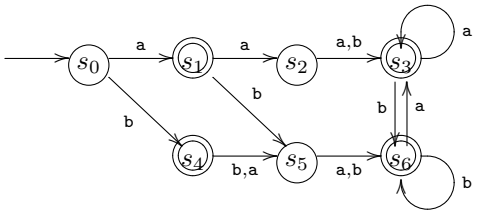
Folha Prática 5

Para revisão...

Qualquer AFD que reconheça $L = \{x \mid x \in \{\mathtt{a},\mathtt{b}\}^\star,\ |x| \neq 2 \text{ e } |x| \neq 0\}$ tem de separar as palavras de comprimento 0, 1, 2 e maior do que 2. O AFD mínimo para L tem quatro estados:

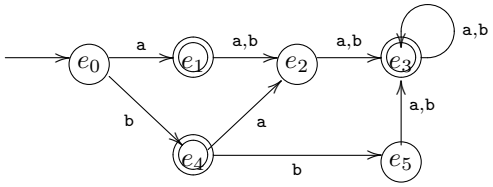


Os dois AFDs representados abaixo aceitam L mas não são mínimos (subdividem alguns dos grupos referidos e ficam com mais estados do que os necessários).



$[\varepsilon] = \mathcal{C}_{s_0}$ $\quad$ $[\mathtt{a}] = \mathcal{C}_{s_1} \cup \mathcal{C}_{s_4}$

$[\mathtt{aa}] = \mathcal{C}_{s_2} \cup \mathcal{C}_{s_5}$ $\quad$ $[\mathtt{aaa}] = \mathcal{C}_{s_3} \cup \mathcal{C}_{s_6}$



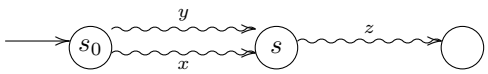
$[\varepsilon] = \mathcal{C}_{e_0}$ $\quad$ $[\mathtt{a}] = \mathcal{C}_{e_1} \cup \mathcal{C}_{e_4}$

$[\mathtt{aa}] = \mathcal{C}_{e_2} \cup \mathcal{C}_{e_5}$ $\quad$ $[\mathtt{aaa}] = \mathcal{C}_{e_3}$

Estamos a usar a notação $\mathcal{C}_s$ para denotar o conjunto das palavras de $\Sigma^\star$ que levam o AFD do estado inicial ao estado s, sendo totalmente consumidas. As palavras de $\mathcal{C}_s$ são indistinguíveis (isto é, são equivalentes) para o autómato.

Palavras indistinguíveis para um dado AFD A

Seja $A = (S, \Sigma, \delta, s_0, F)$ um AFD e sejam x e y duas palavras de $\Sigma^\star$ que levam o AFD A do estado inicial s_0 a um mesmo estado s. A partir do estado s, o AFD A **não conseguirá distinguir** xz de yz, qualquer que seja $z \in \Sigma^\star$.



Para formalizar esta propriedade, definimos a função $\hat{\delta}$ por $\hat{\delta}(s, \varepsilon) = s$ e $\hat{\delta}(s, aw) = \hat{\delta}(\delta(s, a), w)$, para todo $s \in S$, $a \in \Sigma$ e $w \in \Sigma^\star$, para indicar o estado em o AFD fica se consumir uma certa palavra a partir de um certo estado (em particular, $\hat{\delta}(s, aw)$ designa o estado em que o AFD A fica se consumir a palavra aw a partir do estado s).

Podemos mostrar que $\hat{\delta}$ satisfaz: $\hat{\delta}(s, wv) = \hat{\delta}(\hat{\delta}(s, w), v)$, para todo $w, v \in \Sigma^\star$ e $s \in S$.

Assim, a propriedade que enunciámos acima corresponde a

$$\text{se } \hat{\delta}(s_0, x) = \hat{\delta}(s_0, y) \text{ então } \hat{\delta}(s_0, xz) = \hat{\delta}(s_0, yz), \text{ para todo } z \in \Sigma^\star,$$

e diz-nos que se as palavras x e y levam o AFD A do estado inicial ao mesmo estado então, qualquer que seja $z \in \Sigma^\star$, as palavras xz e yz levam o AFD A de s_0 a um mesmo estado. Portanto, ou xz e yz são *ambas aceites* pelo AFD A ou xz e yz são *ambas rejeitadas* pelo AFD A. Isto é, sendo L a linguagem que o AFD A reconhece, tem-se:

$$\textbf{se } \hat{\delta}(s_0, x) = \hat{\delta}(s_0, y) \textbf{ então } \forall z \in \Sigma^\star (xz \in L \Leftrightarrow yz \in L).$$

Palavras que todos os AFDs que aceitam L distinguem

O AFD mínimo que reconhece uma dada linguagem regular L satisfaz uma propriedade mais forte: distingue somente palavras que *todos* os AFDs que aceitam L são obrigados a distinguir. Para o AFD mínimo tem-se a condição seguinte:

$$\hat{\delta}(s_0, x) = \hat{\delta}(s_0, y) \textbf{ se e só se } \forall z \in \Sigma^\star\ (xz \in L \Leftrightarrow yz \in L).$$

o que quer dizer que $\hat{\delta}(s_0, x) \neq \hat{\delta}(s_0, y)$ **só se** existir $z \in \Sigma^\star$ tal que $xz \in L \wedge yz \notin L$ ou $xz \notin L \wedge yz \in L$.

Tal condição define a relação de equivalência R_L que carateriza os estados do AFD mínimo que reconhece L.

Corolário do teorema de Myhill-Nerode

O teorema de Myhill-Nerode diz que L é uma linguagem regular se e só se o conjunto das classes de equivalência da relação R_L é **finito**, sendo R_L dada por $R_L = \{(x, y) \mid x, y \in \Sigma^\star \text{ e } \forall z \in \Sigma^\star (xz \in L \Leftrightarrow yz \in L)\}$.

Da prova do teorema obtém-se a caraterização do AFD mínimo para L (que é único a menos das designações dos estados).

> *Corolário do teorema de Myhill-Nerode:*
> *Se L é uma linguagem regular, então o AFD mínimo que reconhece L é dado por $\mathcal{A} = (\Sigma^\star/R_L, \Sigma, \delta, [\varepsilon], F)$, com $F = \{[x] \mid x \in L\}$, e $\delta([x], a) = [xa]$, para todo $[x] \in \Sigma^\star/R_L$ e todo $a \in \Sigma$.*

O conjunto das classes de equivalência da relação R_L é denotado por $\Sigma^\star/R_L$ (isto é, $\frac{\Sigma^\star}{R_L}$). Usámos a notação $[x]$, para designar a classe de equivalência de x segundo R_L. Por definição de classe de equivalência, $[x] = \{y \mid y \in \Sigma^\star \text{ e } (x, y) \in R_L\}$, isto é, $[x]$ é o conjunto das palavras que são equivalentes a x segundo R_L. Qualquer elemento de uma classe pode ser usado como seu representante nas operações que precisarmos de efetuar.

Em particular, importa salientar que:

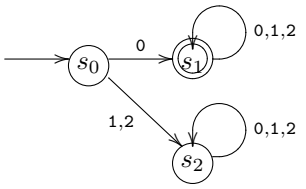
- O valor de $\delta([x], a)$ não depende do representante que usamos para designar a classe, o que é importante para que δ seja uma *função* de transição de $(\Sigma^\star/R_L) \times \Sigma$ em $\Sigma^\star/R_L$. Se $[x] = [y]$ então $[xa] = [ya]$, para todo $a \in \Sigma$. Portanto, se y fosse usado como representante de $[x]$, o valor de $\delta([y], a)$ seria igual a $\delta([x], a)$.
- No AFD mínimo, o estado inicial é $[\varepsilon]$ e tem-se $\hat{\delta}([\varepsilon], x) = [x]$, para todo $x \in \Sigma^\star$. Assim, vemos que $\hat{\delta}([\varepsilon], x) = \hat{\delta}([\varepsilon], y)$ se e só se $[x] = [y]$. Portanto, o AFD mínimo distingue x e y se e só se $(x, y) \notin R_L$. Ou seja, como notámos acima, o AFD mínimo não distingue x e y se $xz \in L \Leftrightarrow yz \in L$, para todo $z \in \Sigma^\star$.

Estamos a supor que nos é dada uma descrição de L e de Σ. Assim, dados x e y em $\Sigma^\star$:

- para concluirmos que $(x, y) \notin R_L$, temos que descobrir $z \in \Sigma^\star$ tal que $xz \in L \wedge yz \notin L$ ou $xz \notin L \wedge yz \in L$;
- se concluirmos que não pode existir z nessas condições, então $(x, y) \in R_L$. A justificação da não existência desse z obriga-nos a descobrir qual a condição mais geral sobre z para que $xz \in L$, a fazer o mesmo para $yz \in L$, e a verificar que as condições são iguais (ver exemplo a seguir). Notar que, se $xz \notin L$, para todo $z \in \Sigma^\star$, a condição referida para x seria $z \in \{\}$, como no exemplo.

Exemplo de aplicação do corolário do teorema de Myhill-Nerode para obter o AFD mínimo

Seja $L = \{x \mid x \text{ começa por } \mathtt{0}\}$, com $\Sigma = \{\mathtt{0}, \mathtt{1}, \mathtt{2}\}$. Não é díficil convencermo-nos de que qualquer AFD que aceite L tem pelo menos três estados, sendo o AFD mínimo o seguinte:



Se a palavra dada for ε, o autómato permanece no estado s_0. Como $\varepsilon \notin L$, o estado s_0 não é final. Nenhuma outra palavra leva este AFD de s_0 a s_0, o que é lógico, pois a análise do primeiro símbolo da palavra permite-nos decidir se a palavra é ou não é da linguagem L: se começar por 1 ou 2 terá de ser rejeitada, independentemente dos restantes símbolos que tiver (é o que s_2 faz); se começar por 0 será aceite, independentemente dos restantes símbolos que tiver (é o que faz s_1). À semelhança deste AFD, qualquer outro AFD que reconheça L deve distinguir as palavras ε, 0, e 1. Logo, o autómato representado é o AFD mínimo para L. Esta distinção corresponde à não equivalência das palavras ε, 0, e 1 segundo a relação R_L (o que implica que $[\varepsilon]$, $[0]$ e $[1]$ sejam classes distintas).

A relação R_L ajuda-nos a sistematizar e justificar a necessidade de criar cada um dos estados do AFD (classes de R_L).

Partindo do estado inicial $[\varepsilon]$ e, usando o corolário do Teorema de Myhill-Nerode, construimos o AFD mínimo para L assim:

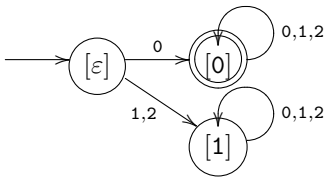
- $[\varepsilon]$ é o estado inicial; $[\varepsilon]$ não pertencerá ao conjunto de estados finais F pois $\varepsilon \notin L$.
- $\delta([\varepsilon], \mathtt{0}) \stackrel{\text{def}}{=} [\varepsilon\mathtt{0}] = [\mathtt{0}] \neq [\varepsilon]$, pois $(\mathtt{0}, \varepsilon) \notin R_L$, porque $\mathtt{0} \in L$ e $\varepsilon \notin L$.
  Notar que para ver que $(\mathtt{0}, \varepsilon) \notin R_L$, tomamos $z = \varepsilon$ para ter $\mathtt{0}z \in L$ e $\varepsilon z \notin L$, pois nada falta a 0 para ser de L.
  Consequentemente, $[\mathtt{0}]$ será um novo estado de $\mathcal{A}$ e, como $\mathtt{0} \in L$, teremos $[\mathtt{0}] \in F$.
- $\delta([\varepsilon], \mathtt{1}) \stackrel{\text{def}}{=} [\varepsilon\mathtt{1}] = [\mathtt{1}]$. Concluimos que $[\mathtt{1}]$ é um novo estado e não será final porque $\mathtt{1} \notin L$ e:
  $[\mathtt{1}] \neq [\mathtt{0}]$ pois, como $\mathtt{1} \notin L$ e $\mathtt{0} \in L$, temos $(\mathtt{1}, \mathtt{0}) \notin R_L$. Bastaria escolher $z = \varepsilon$.
  $[\mathtt{1}] \neq [\varepsilon]$ pois, embora $\mathtt{1} \notin L$ e $\varepsilon \notin L$, sabemos que $\mathtt{1}z \notin L$, para todo $z \in \Sigma^\star$, o que não é verdade para ε.
  De facto, por exemplo, para $z = \mathtt{0}$, temos $\varepsilon z = \varepsilon\mathtt{0} = \mathtt{0} \in L$ e $\mathtt{1}z = \mathtt{10} \notin L$.
- $\delta([\varepsilon], \mathtt{2}) \stackrel{\text{def}}{=} [\mathtt{2}] = [\mathtt{1}]$, porque se tem $\mathtt{2}z \notin L$, para todo $z \in \Sigma^\star$, à semelhança de 1.
  As palavras que comecem por 1 ou 2 são rejeitadas independentemente dos símbolos seguintes.

- $\delta([0],0) \stackrel{\text{def}}{=} [00] = [0]$, porque $00z \in L$, para todo $z \in \Sigma^\star$, à semelhança de 0.
  As palavras que comecem por 0 são aceites independentemente dos símbolos seguintes.
  $\delta([0],1) \stackrel{\text{def}}{=} [01] = [0]$, porque $01z \in L$, para todo $z \in \Sigma^\star$.
  $\delta([0],2) \stackrel{\text{def}}{=} [02] = [0]$, porque $02z \in L$, para todo $z \in \Sigma^\star$.
- $\delta([1],0) = \delta([1],1) = \delta([1],2) = [1]$, porque $[10] = [11] = [12] = [1]$.
  Isso resulta de $1z \notin L$, $10z \notin L$, $11z \notin L$, $12z \notin L$, para todo $z \in \Sigma^\star$.

Em suma, o AFD mínimo que reconhece $L = \{x \mid x \text{ começa por } 0\}$, de alfabeto $\Sigma = \{0,1,2\}$, é

$$\mathcal{A} = (\{\,[\varepsilon],[0],[1]\,\},\ \Sigma,\ \delta,\ [\varepsilon],\ \{\,[1]\,\})$$

para δ indicada acima, e o diagrama de transição de $\mathcal{A}$ é:



A menos da designação que escolhemos para os estados, o AFD mínimo é único. Nesse sentido, o AFD que obtivemos, não difere do que tinhamos anteriormente definido.

Observação: Se descartarmos o estado $[1]$, não teriamos um AFD, de acordo com a definição dada. De facto, tal definição obriga os AFDs a serem *completos* (para cada estado $s \in S$, há exatamente uma transição por cada símbolo $a \in \Sigma$; qualquer palavra de $\Sigma^\star$ pode ser processada pelo AFD até ao fim).

Algoritmo de Moore para minimização de um AFD dado

Dado um AFD $A = (S, \Sigma, \delta, s_0, F)$, podemos aplicar o algoritmo de Moore para decidir se A é ou não o AFD mínimo que reconhece $\mathcal{L}(A)$ e, simultaneamente, para obter o AFD mínimo que reconhece $\mathcal{L}(A)$, se A não for mínimo. Para isso, **irá identificar estados equivalentes em** $\mathcal{A}$: dois estados s e s' são equivalentes, i.e., *indistinguíveis para* $R_{\mathcal{L}(\mathcal{A})}$, sse **não** existir uma palavra $z \in \Sigma^\star$ tal que se consumir z a partir de s chega a estado final e se consumir z a partir de s' não chega a estado final, ou vice-versa. Na descrição do algoritmo, $\equiv$ denota a relação de equivalência entre estados.

Algoritmo de Moore:

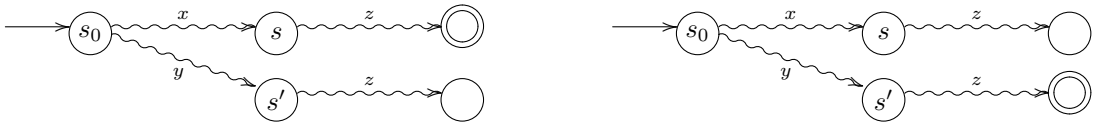
Começamos por retirar de S os estados não acessíveis de s_0. Seja $S' = \{s_0, s_1, \ldots, s_m\}$ o conjunto dos restantes. Como a relação $\equiv$ é simétrica, vamos considerar apenas os pares (s_i, s_j), para $0 \le i \le j \le m$. Para visualização, construimos uma tabela, onde os símbolos $\equiv$, X e ? denotam $\equiv$, $\not\equiv$ e **decisão pendente**. *Na aplicação do algoritmo, cada par pode ter uma lista de pares pendentes associada (a qual ficará na sua entrada na tabela). Inicialmente, essas listas são vazias. Aplicar o procedimento seguinte:*

- *Assinalar com $\equiv$ todas as entradas (s_i, s_i), para todo i.*
- *Para todo (s_i, s_j), com $s_i \in F \wedge s_j \notin F$ ou $s_i \notin F \wedge s_j \in F$, assinalar $s_i \not\equiv s_j$, colocando* X *em (s_i, s_j).*
- *Para $1 \le j \le m$ e $0 \le i < j$, se (s_i, s_j) não contém* X*, averiguar se já é conhecido que $\delta(s_i, a) \not\equiv \delta(s_j, a)$, para algum $a \in \Sigma$ (para isso, ver se existe* X *na entrada do par $(\delta(s_i, a), \delta(s_j, a))$.*
  - *Se, para algum $a \in \Sigma$, já for conhecido que $\delta(s_i, a) \not\equiv \delta(s_j, a)$, registar $s_i \not\equiv s_j$, assinalando (s_i, s_j) com* X*, e propagar a informação a todos os pares que estiverem na lista de pendentes de (s_i, s_j).* **Propagar** *significa assinalar com* X *cada um dos pares nessa lista e, recursivamente, propagar aos pares que estiverem nas listas de pendentes desses.*
  - *Se já se sabe que $\delta(s_i, a) \equiv \delta(s_j, a)$, para todo $a \in \Sigma$, isto é, todos já estão marcados com $\equiv$ na tabela, então registar $s_i \equiv s_j$, assinalando a entrada (s_i, s_j) com $\equiv$.*
  - *Nas restantes situações,* (s_i, s_j) aguardará as decisões *para $(\delta(s_i, a), \delta(s_j, a))$, com $a \in \Sigma$: para todos os pares $(\delta(s_i, a), \delta(s_j, a))$ sem marcação $\equiv$, acrescentar (s_i, s_j) à* **lista de pendentes** *de $(\delta(s_i, a), \delta(s_j, a))$ e assinalar a entrada (s_i, s_j) com o símbolo* ? *(fica pendente).*
- *Quando todos os pares estiverem analisados, substituir* ? *por $\equiv$ nas entradas que se mantiverem pendentes (cada entrada que não tem* X*, corresponde a um par de estados equivalentes).*
- *O conjunto de estados do AFD mínimo A' equivalente ao AFD A corresponde ao conjunto de classes de equivalência de $\equiv$ (restrita a $S' = \{s_0, s_1 \ldots, s_m\} \subseteq S$). Se $[s]$ denotar a classe do estado s, então a função de transição δ' é dada por $\delta'([s], a) = [\delta(s, a)]$, para todo $a \in \Sigma$. O estado inicial de A' é $[s_0]$ e o conjunto de estados finais é $F' = \{[s] \mid s \in F \cap S'\}$.*

Ideia que suporta o algoritmo:

Para cada par de estados (s, s'), ambos acessíveis de s_0, o algoritmo vai determinar se existe uma palavra $z \in \Sigma^\star$ que obrigue a manter s e s'. Se existir z tal que $\hat{\delta}(s, z) \in F$ e $\hat{\delta}(s', z) \notin F$ ou existir z tal que $\hat{\delta}(s, z) \notin F$ e $\hat{\delta}(s', z) \in F$, então s e s' não são equivalentes. Se não existir z nessas condições, então s e s' são equivalentes (podiamos uni-los num único estado).

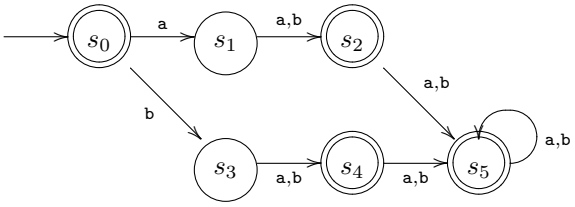
Os esquemas seguintes ilustram a situação.



Se $\hat{\delta}(s, z) \in F$ e $\hat{\delta}(s', z) \notin F$ então se x levar o AFD de s_0 a s e y levar o AFD de s_0 a s', tem-se $xz \in \mathcal{L}(A)$ e $yz \notin \mathcal{L}(A)$. Portanto, não se podia unir s e s' num estado só (esquema representado à esquerda). O caso $\hat{\delta}(s, z) \notin F$ e $\hat{\delta}(s', z) \in F$, representado à direita, é similar.

Exemplo:

Por aplicação do algoritmo de Moore, vamos averiguar se o AFD representado é mínimo.



	s_0	s_1	s_2	s_3	s_4	s_5
s_0	$\equiv$					
s_1	X	$\equiv$				
s_2		X	$\equiv$			
s_3	X		X	$\equiv$		
s_4		X		X	$\equiv$	
s_5		X		X		$\equiv$

A tabela inicial encontra-se acima à direita. Para os restantes pares, tem-se:

(s_0, s_2) : $s_0 \not\equiv s_2$ *porque* $\delta(s_0, \mathtt{a}) = s_1 \not\equiv s_5 = \delta(s_2, \mathtt{a})$.
(s_0, s_4) : $s_0 \not\equiv s_4$ *porque* $\delta(s_0, \mathtt{a}) = s_1 \not\equiv s_5 = \delta(s_4, \mathtt{a})$.
(s_0, s_5) : $s_0 \not\equiv s_5$ *porque* $\delta(s_0, \mathtt{a}) = s_1 \not\equiv s_5 = \delta(s_5, \mathtt{a})$.
(s_1, s_3) : $\delta(s_1, \mathtt{a}) = s_2 = \delta(s_1, \mathtt{b})$ *e* $\delta(s_3, \mathtt{a}) = s_4 = \delta(s_3, \mathtt{b})$.
Fica pendente. Assinalar dependência em (s_2, s_4).
(s_2, s_4) : $s_2 \equiv s_4$ *pois* $\delta(s_2, \mathtt{a}) = \delta(s_4, \mathtt{a})$ *e* $\delta(s_2, \mathtt{b}) = \delta(s_4, \mathtt{b})$.
(s_2, s_5) : $s_2 \equiv s_5$ *pois* $\delta(s_2, \mathtt{a}) = \delta(s_5, \mathtt{a})$ *e* $\delta(s_2, \mathtt{b}) = \delta(s_5, \mathtt{b})$.
(s_4, s_5) : $s_4 \equiv s_5$ *pois* $\delta(s_4, \mathtt{a}) = \delta(s_5, \mathtt{a})$ *e* $\delta(s_4, \mathtt{b}) = \delta(s_5, \mathtt{b})$.

	s_0	s_1	s_2	s_3	s_4	s_5
s_0	$\equiv$					
s_1	X	$\equiv$				
s_2	X	X	$\equiv$			
s_3	X	?	X	$\equiv$		
s_4	X	X	(s_1, s_3) $\equiv$	X	$\equiv$	
s_5	X	X	$\equiv$	X	$\equiv$	$\equiv$

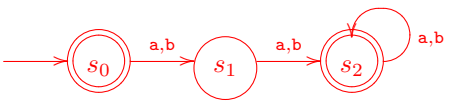
Se seguirmos o algoritmo passo a passo, a decisão (s_1, s_3) *fica pendente até ao fim, pois não há informação global sobre o número de pares que podem decidir a não equivalência de* (s_1, s_3). *Apenas no passo final se troca* ? *por* $\equiv$.

As classes de eqivalência de $\equiv$ *são:*

$\{s_0\}$, $\{s_1, s_3\}$, $\{s_2, s_4, s_5\}$.

	s_0	s_1	s_2	s_3	s_4	s_5
s_0	$\equiv$					
s_1	X	$\equiv$				
s_2	X	X	$\equiv$			
s_3	X	$\equiv$	X	$\equiv$		
s_4	X	X	$\equiv$	X	$\equiv$	
s_5	X	X	$\equiv$	X	$\equiv$	$\equiv$

O AFD mínimo equivalente ao AFD dado é:



A tabela final corresponde à (parte triangular inferior da) matriz da relação $\equiv$, se substituirmos $\equiv$ por 1 e X por 0.

Exercícios

1. Sejam $L_1 = \mathcal{L}(\mathtt{0} + (\mathtt{11})^\star)$ e $L_2 = \mathcal{L}((\mathtt{0} + \mathtt{1})^\star\mathtt{101})$ linguagens de alfabeto $\Sigma = \{\mathtt{0}, \mathtt{1}\}$.

a) Defina informalmente L_1 e L_2.

b) Sem aplicar explicitamente o corolário do teorema de Myhill-Nerode, apresente AFDs que reconheçam L_1 e L_2. Justifique a necessidade de cada estado que indicar.

c) Por aplicação do corolário do teorema de Myhill-Nerode, determine o AFD mínimo que reconhece L_1 e o AFD mínimo que reconhece L_2.

2. Para cada questão determine a resposta correta e justifique. $\#S$ denota o cardinal de S, i.e., $|S|$.

1. Seja $\mathcal{A} = (S, \Sigma, \delta, s_0, F)$ o AFD mínimo para $\mathcal{L}(\mathcal{A}) = \{aa\}\Sigma^\star$, com $a \in \Sigma$ fixo e $\Sigma\backslash\{a\} \neq \emptyset$.
   (a) $\#S = 3$, qualquer que seja Σ.
   (b) $\#S = 4$, qualquer que seja Σ.
   (c) $\#S \geq 1$ e nada mais se pode dizer sem conhecer Σ.

2. Seja $\mathcal{A} = (S, \Sigma, \delta, s_0, F)$ o AFD mínimo para $\mathcal{L}(\mathcal{A}) = \Sigma^\star\{aa\}$, com $a \in \Sigma$ fixo.
   (a) $\#S = 3$, qualquer que seja Σ.
   (b) $\#S = 4$, qualquer que seja Σ.
   (c) $\#S \geq 1$ e nada mais se pode dizer sem conhecer Σ.

3. Justifique a veracidade ou falsidade de cada uma das afirmações seguintes.

a) Para todo o alfabeto Σ, o AFD mínimo que reconhece a linguagem $\emptyset$ de $\Sigma^\star$ não tem estados finais.

b) Existe uma linguagem L de alfabeto $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ tal que o AFD mínimo que reconhece L não tem transições por $\mathtt{b}$ em nenhum estado.

4. Usando o corolário do teorema de Myhill-Nerode, que define o AFD mínimo para L e para $\overline{L}$, demonstre que, qualquer que seja a linguagem regular L de alfabeto Σ, o AFD mínimo para L e para $\overline{L}$ tem exatamente o mesmo conjunto de estados e a mesma função de transição, diferindo apenas no conjunto de estados finais: se $\mathcal{A} = (S, \Sigma, \delta, s_0, F)$ é o AFD mínimo que reconhece L então $\mathcal{A}' = (S, \Sigma, \delta, s_0, S \setminus F)$ é o AFD mínimo que reconhece a linguagem complementar de L.

5. Por aplicação do algoritmo de Moore, determine o AFD mínimo que reconhece $\mathcal{L}(A)$ para

$$A = (\{q_0, q_1, q_2, q_3, q_4\}, \Sigma, \delta, q_0, \{q_3, q_4\}),$$

com $\delta(q_0, 0) = q_1$, $\delta(q_0, 1) = q_2$, $\delta(q_1, 0) = \delta(q_1, 1) = q_1$, $\delta(q_2, 1) = q_4$, $\delta(q_2, 0) = q_3$, $\delta(q_3, 0) = q_1$, $\delta(q_4, 0) = q_1$, $\delta(q_4, 1) = q_2$, e $\delta(q_3, 1) = q_2$, de alfabeto $\Sigma = \{\mathtt{0}, \mathtt{1}\}$.

6. Por aplicação do método de Thompson, do método de conversão de um AFND-ε para AFD e do algoritmo de Moore, determine o AFD mínimo que reconhece a linguagem definida pela expressão regular, sobre $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, indicada em cada alínea.

a) $(\,(\,\mathtt{10}\,) + (\,\mathtt{11}\,)\,)$
b) $(\,(\,(\,\mathtt{10}\,) + (\,\mathtt{11}\,)\,)^\star\,)$
c) $(\,(\,\mathtt{1}(\,(\,\mathtt{10}\,)^\star\,)\,)^\star\,)$
d) $(\,(\,(\,(\mathtt{1}^\star) + (\mathtt{10})\,) + (\mathtt{11})\,)^\star\,)$

7. Seja L a linguagem das palavras de alfabeto $\{\texttt{0}, \texttt{1}\}$ que terminam em `010` ou em `101`. Determine o AFD mínimo que reconhece L, aplicando o método descrito em cada alínea.

a) Construa um AFND que reconheça L, tirando partido do não determinismo. Justifique sucintamente que está correto e, a seguir, converta-o para um AFD por aplicação do algoritmo de conversão. Finalmente, minimize tal AFD por aplicação do algoritmo de Moore.

b) Use a descrição de L e aplique o corolário do teorema de Myhill-Nerode para construir o AFD mínimo.

c) Partindo da descrição de L, construa um AFD que reconheça L tendo cuidado de justificar a necessidade de cada estado (e o que memoriza sobre a palavra consumida do estado inicial até esse estado). Essa justificação deverá constituir uma prova de correção do AFD (sucinta mas clara) bem como de ser mínimo.