Capítulo 2

Autómatos finitos

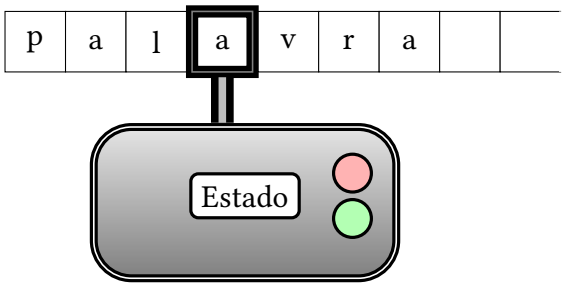
Neste capítulo formalizamos a noção de autómato finito como um dispositivo abstracto cuja função é decidir se uma palavra pertence ou não a uma determinada linguagem.

Nesta exposição seguimos uma vertente maioritariamente teórica. No entanto, os autómatos finitos têm múltiplas aplicações práticas: no desenvolvimento de software para a análise léxica em compiladores, na pesquisa de palavras e padrões em páginas de texto ou na Internet, no desenho de circuitos digitais, etc.

Embora existam outros modelos de autómatos finitos, pela sua simplicidade, vamos considerar apenas autómatos cuja função de saída é booleana. Esta decisão é natural uma vez que o nosso interesse primordial é a questão do reconhecimento de linguagens.

Assim, não estudaremos modelos mais complexos, em que os resultados são palavras sobre um determinado alfabeto, como as máquinas de Mealy[24] (George H. Mealy, Estados Unidos da América, 1927 - 2010) ou os autómatos de Moore (Edward F. Moore, Estados Unidos da América, 1925 - 2003).

Começamos com uma descrição e ilustração das diversas componentes de um autómato finito, bem como o seu "modus operandi".



Um autómato trabalha sobre uma fita dividida em células, no início da qual se escreve uma palavra, um símbolo por célula. É constituído por uma "cabeça" de

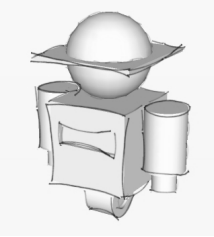
leitura que se move ao longo da fita, sempre para a direita, avançando de célula em célula. Possui ainda duas lâmpadas que se acendem no final do processamento de uma palavra: uma verde (que indica o reconhecimento de uma palavra) e uma vermelha (que indica a rejeição de uma palavra).

Internamente, um autómato finito armazena, em cada momento, um determinado estado (de entre um conjunto finito de estados), o qual pode mudar na transição de um momento para o seguinte em função do símbolo inscrito na fita sob a "cabeça" de leitura.

No início do processamento, um autómato encontra-se num estado especial, designado estado inicial, e a "cabeça" de leitura encontra-se posicionada sob o primeiro símbolo da palavra inscrita na fita.

A evolução de um autómato ao longo do tempo consiste numa sequência finita de passos determinados pelo avanço da "cabeça" de leitura de uma posição na fita para a posição seguinte, parando quando não consegue mudar de estado ou quando a "cabeça" de leitura ultrapassa o último símbolo da palavra inscrito na fita.

2.1 Autómatos deterministas



O determinismo é geralmente associado a uma doutrina filosófica adepta da negação do livre-arbítrio.
No contexto deste livro, significa tão somente a existência de uma única opção ou caminho.

Um autómato é determinista sempre que, estando num determinado estado e perante um qualquer símbolo de entrada inscrito na fita, admite um e um só estado para o qual pode transitar. Vamos formalizar a noção de autómato determinista bem como de reconhecimento de uma linguagem por um autómato determinista. Antes disso, começamos com um exemplo ilustrativo da resolução de um problema por um autómato finito.

Exemplo 2.1.1

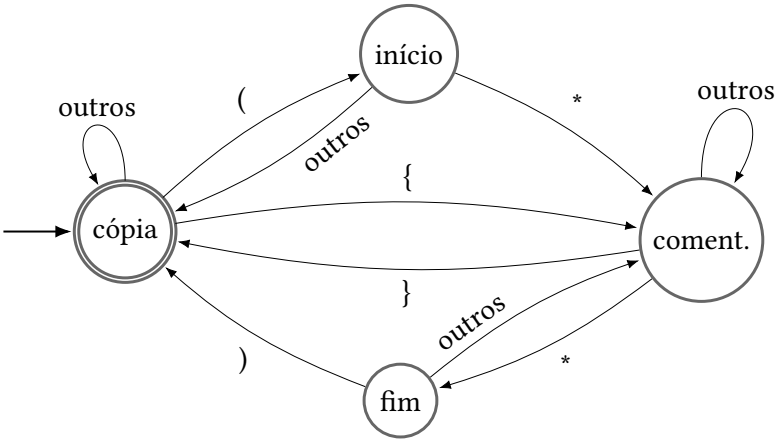
O ficheiro "origem" contém um programa escrito na linguagem Pascal. Os comentários podem estar nas formas $(*\ldots*)$ e $\{\ldots\}$, ou ainda nas formas, $(*\ldots\}$ e $\{\ldots*)$.

Problema. Remover todos os comentários e registar a nova versão do programa no ficheiro "destino".

Resolução por um Autómato. Ignorando as questões da leitura/escrita em fichei-

ros, desenhamos um autómato com quatro estados: *cópia*, *comentário*, *início* e *fim*. A leitura de cada carácter provoca uma transição de estado e uma operação associada ao novo estado.

Representamos o autómato no diagrama seguinte:



O estado *cópia* é o estado inicial, assinalado com uma seta. O estado de aceitação é também o estado *cópia*, assinalado com uma borda dupla. Os comentários estão sintacticamente correctos se e só se, partindo do estado *cópia*, o autómato voltar a este mesmo estado após ter lido o último carácter do ficheiro.

Definição 2.1.2 Autómato Finito Determinista.

Um **autómato finito determinista** (AFD) é um quíntuplo ordenado

$$A = (Q, \Sigma, \delta, s, F)$$

onde:

- Q é um conjunto finito, não vazio, de **estados**;
- Σ é um alfabeto de *símbolos de entrada*;
- $\delta : Q \times \Sigma \to Q$ é uma **função de transição** (parcial) ou de mudança de estado;
- $s \in Q$ é o **estado inicial**;
- $F \subseteq Q$ é o conjunto dos **estados de aceitação**.

Intuitivamente, uma palavra faz parte da linguagem de um autómato se e só se, partindo do estado inicial e do início da palavra, o autómato atinge um estado de aceitação quando completa a leitura da palavra.

Exemplo 2.1.3

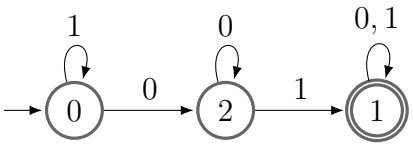
Consideremos a linguagem

$$L = \{w \in \Sigma^\star : \quad \text{a palavra } 01 \text{ é parte de } w\},$$

definida sobre o alfabeto $\Sigma = \{0, 1\}$.

Problema. Construir um *autómato reconhecedor das palavras de* L.

Solução. O autómato $A = \big(\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\}\big)$, onde a função de transição δ é definida pelo *diagrama de transições*



ou pela *tabela de transições*

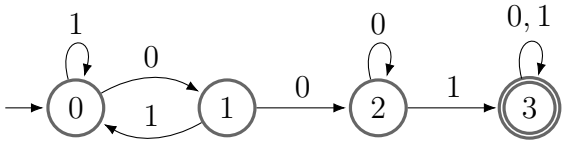
δ	0	1
$\to q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

Exemplo 2.1.4

Consideremos a linguagem $L = \{w \in \Sigma^\star : \quad \text{a sequência } 001 \text{ é parte de } w\}$, definida sobre o alfabeto $\Sigma = \{0, 1\}$.

Problema. Construir um *autómato reconhecedor das palavras de* L.

Solução. O autómato $A = \big(\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_3\}\big)$ onde a função de transição δ é dada pelo *diagrama de transições*



Definição 2.1.5 Função de Transição Estendida de um AFD.

A **função de transição estendida** de um AFD $A = \big(Q, \Sigma, \delta, s, F\big)$ é

$$\delta^\star : Q \times \Sigma^\star \to Q$$

definida por:

Caso base. Para $q \in Q$, $\delta^\star(q, \varepsilon) = q$;

Passo indutivo. Para $q \in Q$ e $w = xa$, com $x \in \Sigma^\star$ e $a \in \Sigma$,

$$\delta^\star(q, xa) = \delta\left(\delta^\star(q, x), a\right).$$

A função $\delta^\star$ representa efectivamente todos os possíveis passeios ao longo do diagrama de transições do autómato, isto é, se $w = a_1 a_2 \cdots a_n$ e $\delta(p_i, a_{i+1}) = p_{i+1}$, para $i = 0, 1, \ldots, n-1$ então $\delta^\star(p_0, w) = p_n$.

Definição 2.1.6 Palavra Reconhecida por um AFD.

Um AFD $A = \big(Q, \Sigma, \delta, s, F\big)$ *reconhece* (ou aceita) *a palavra* $w \in \Sigma^\star$ se $\delta^\star(s, w) \in F$.

Definição 2.1.7 Linguagem Reconhecida por um AFD.

A **linguagem reconhecida** (ou aceite) por um AFD $A = \big(Q, \Sigma, \delta, s, F\big)$ é

$$\mathcal{L}(A) = \{w \in \Sigma^\star : \ \delta^\star(s, w) \in F\}.$$

A linguagem de um AFD decorre do seu diagrama de transições, como o conjunto das sequências de símbolos associadas a todos os passeios possíveis, desde o estado inicial até um dos estados de aceitação.

Notamos que a Definição 2.1.2, p. 31 permite que a função de transição de um AFD seja apenas parcial, pelo que é possível que o autómato pare sem que seja percorrida toda a palavra inscrita na fita.

O que fazer então quando um autómato está num estado para o qual não existe transição pelo próximo símbolo na fita? Neste caso, é evidente que a palavra que está presente na fita não pode ser aceite, mesmo que o estado seja de aceitação, porque não foi toda lida.

Por outro lado, se um autómato pára porque leu todos os símbolos da palavra inscrita na fita então aceita ou rejeita consoante o estado em que parou é ou não de aceitação.

Quando a função de transição de um AFD é parcial então o mesmo se passa com a função de transição estendida. Assim, quando o autómato pára, é apenas a sub-palavra efectivamente lida que é aceite ou rejeitada.

Definição 2.1.8 Autómato Completo.

Um autómato finito é **completo** se a sua função de transição for total.

Sempre que a função de transição é parcial o autómato é **incompleto**.

Se um autómato é incompleto, existe pelo menos um estado e pelo menos um símbolo para os quais a transição a partir desse estado e por esse símbolo não está definida. Nesta situação, podemos assumir que o autómato transita efectivamente para um estado indefinido, que não é de aceitação, e do qual não pode sair (por mais símbolos que se leiam em seguida).

A este estado indefinido, que é sempre de rejeição, chamamos *estado ratoeira*. Os estados ratoeira permitem completar os autómatos incompletos preservando as linguagens reconhecidas.

Exemplo 2.1.9

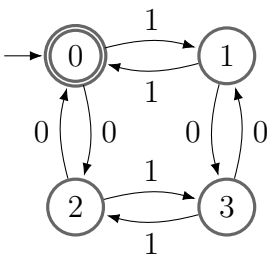
Seja L a linguagem das palavras binárias que têm um número par de 0´s e um número par de 1´s.

Problema. Pretendemos construir um *autómato reconhecedor das palavras de* L.

Solução. A estratégia consiste em contar os 0´s e os 1´s (módulo 2):

1. os 0´s e os 1´s já contados são em número par;
2. foram já contados um número par de 0´s e um número ímpar de 1´s;
3. foram já contados um número par de 1´s e um número ímpar de 0´s;
4. os 0´s e os 1´s já contados são em número ímpar.

Usando esta estratégia, obtemos o autómato seguinte, ilustrado pelo diagrama de transições (à esquerda) e pela respectiva tabela de transições (à direita).



δ	0	1
$* \rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Verificação indutiva para a palavra 110101. Temos que:

- $\delta^\star(q_0, \varepsilon) = q_0$
- $\delta^\star(q_0, 1) = \delta\left(\delta^\star(q_0, \varepsilon), 1\right) = \delta(q_0, 1) = q_1$
- $\delta^\star(q_0, 11) = \delta\left(\delta^\star(q_0, 1), 1\right) = \delta(q_1, 1) = q_0$
- $\delta^\star(q_0, 110) = \delta\left(\delta^\star(q_0, 11), 0\right) = \delta(q_0, 0) = q_2$
- $\delta^\star(q_0, 1101) = \delta\left(\delta^\star(q_0, 110), 1\right) = \delta(q_2, 1) = q_3$
- $\delta^\star(q_0, 11010) = \delta\left(\delta^\star(q_0, 1101), 0\right) = \delta(q_3, 0) = q_1$
- $\delta^\star(q_0, 110101) = \delta\left(\delta^\star(q_0, 11010), 1\right) = \delta(q_1, 1) = q_0$

portanto $110101 \in L$.

Exemplo 2.1.10

Sobre o alfabeto $\Sigma = \{0, 1\}$, consideremos a linguagem L das palavras que são representações binárias de múltiplos de 5.

Problema. Pretendemos construir um *autómato reconhecedor das palavras de* L.

Estratégia. Em cada momento do reconhecimento, precisamos de conhecer o valor do resto da divisão por 5 do número decimal correspondente à palavra binária já lida.

Solução. Sendo w a representação binária do número n (decimal), então

- a palavra $w0$ representa $2 \times n$
- a palavra $w1$ representa $2 \times n + 1$.

Para além disso, é válida a propriedade

$$(a \times b + c) \bmod 5 = (a \times (b \bmod 5) + c) \bmod 5.$$

Por exemplo, para a palavra 11001 temos

- $1 \mapsto 1_{(10)}$
  $$1 \bmod 5 = 1$$
- $11 \mapsto 2 \times 1 + 1 = 3_{(10)}$
  $$3 \bmod 5 = 3$$
- $110 \mapsto 2 \times 3 = 6_{(10)}$
  $$6 \bmod 5 = 1$$

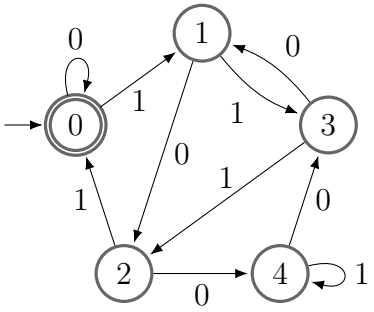
- $1100 \mapsto 2 \times 6 = 12_{(10)}$

$$\begin{aligned} 12 \bmod 5 &= 2 \times 6 \bmod 5 \\ &= (2 \times (6 \bmod 5)) \bmod 5 \\ &= 2 \bmod 5 = 2 \end{aligned}$$

- $11001 \mapsto 2 \times 12 + 1 = 25_{(10)}$

$$\begin{aligned} 25 \bmod 5 &= (2 \times 12 + 1) \bmod 5 \\ &= (2 \times (12 \bmod 5) + 1) \bmod 5 \\ &= 5 \bmod 5 = 0. \end{aligned}$$

Constatamos que existem 5 casos distintos a considerar, um para cada um dos possíveis restos da divisão módulo 5. Chegamos assim ao AFD seguinte, representado pelo diagrama e pela correspondente tabela de transições.



δ	0	1
$* \to q_0$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

Uma vez que o estado inicial do autómato obtido no exemplo anterior é de aceitação, esse autómato aceita a palavra vazia. Mas a palavra vazia não corresponde a um número binário e o autómato não a deve aceitar! Assim, esta solução não é ainda a correcta. Precisamos pois de construir um autómato em que estado inicial não seja de aceitação.

O mesmo autómato aceita ainda representações binárias com zeros à esquerda e em particular a representação binária do número 0. Devemos considerar o número 0 como um múltiplo de 5?

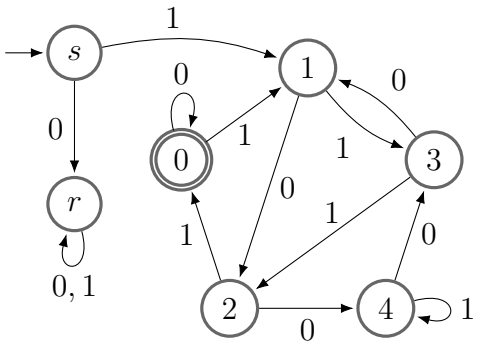
Estas observações remetem para a importância da formalização dos problemas e da necessidade de verificar as soluções nos "casos extremos". No exemplo seguinte apresentamos uma solução para as duas anomalias identificadas.

Exemplo 2.1.11

Seja L a linguagem das palavras binárias que começam por 1 e são representações binárias de múltiplos de 5.

Construímos um *autómato reconhecedor das palavras de* L efectuando uma restrição ao autómato do exemplo anterior:

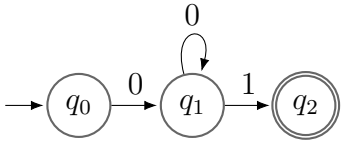
1. Se a palavra começar por 0, então não pertence à linguagem.
2. As palavras que começam por 0 são conduzidas ao *estado ratoeira* (r).
3. Um novo estado de partida (s) permite essa verificação e também *rejeitar a palavra vazia*.



Os autómatos definidos nos exemplos anteriores são todos completos. Neste último exemplo, o autómato contém um *estado ratoeira* (um estado de não aceitação em que todas as transições apontam para si mesmo). Geralmente omitimos os estados ratoeira, sobretudo nos diagramas de transição, conforme ilustrado no exemplo seguinte.

Exemplo 2.1.12

O diagrama de transições seguinte representa um autómato incompleto que reconhece a linguagem $L = \{0^n 1 : \ n \in \mathbb{N}\}$.



A possibilidade da função de transição ser parcial não acrescenta qualquer "poder adicional" em termos de reconhecimento de linguagens. De facto, é sempre possível construir um AFD completo que reconhece a mesma linguagem que a reconhecida por um AFD incompleto, bastando introduzir um estado ratoeira para o completar.

Na Secção 1.2, p. 8 discutimos várias operações sobre linguagens. Dado um autómato que reconhece uma linguagem L, será que é possível construir um autómato que reconheça a linguagem reversa L^{-1} ou ainda a linguagem complementar $\overline{L}$? Como veremos, estas questões têm uma resposta afirmativa. De forma análoga, será que partindo de dois autómatos que reconheçam as linguagens L_1 e L_2 é possível construir um autómato que reconheça a linguagem união ($L_1 \cup L_2$) ou a linguagem intersecção ($L_1 \cap L_2$)? Uma vez mais, estas questões têm resposta afirmativa. Nos exercícios 2.6.14, p. 62 e 2.6.15, p. 62 indicamos como construir autómatos para a intersecção e união de linguagens.

Com o resultado seguinte mostramos como construir um AFD para a linguagem complementar da linguagem reconhecida por um AFD.

Teorema 2.1.13

Se $A = (Q, \Sigma, \delta, s, F)$ é um AFD completo que reconhece a linguagem L então

$$B = (Q, \Sigma, \delta, s, Q \setminus F)$$

é um AFD que reconhece $\overline{L}$.

Demonstração.

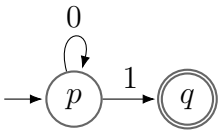
Se $w \in \overline{L}$ então $w \notin L$ e portanto w é rejeitada pelo autómato A. Assim, $\delta^\star(s, w) \notin F$, ou seja, $\delta^\star(s, w) \in Q \setminus F$ e portanto w é aceite pelo autómato B. Isto mostra que $L \subseteq \mathcal{L}(B)$.

Prova-se de modo análogo que $\mathcal{L}(B) \subseteq L$.

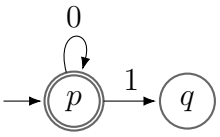
Alertamos para o facto de que para construir o autómato reconhecedor da linguagem complementar, é essencial partir de um autómato *determinista* e *completo*.

Exemplo 2.1.14

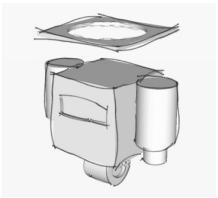
Consideremos o autómato determinista não completo, A, reconhecedor da linguagem $\mathcal{L}(A) = \{0^n 1 : n \in \mathbb{N}_0\}$:



Ao aplicar o método do complementar, obtemos o autómato seguinte, B, que reconhece a linguagem $\mathcal{L}(B) = \{0^n : n \in \mathbb{N}_0\} \neq \overline{\mathcal{L}(A)}$.



2.2 Autómatos não deterministas



O conceito de *Não Determinismo* num autómato significa que as transições de estado, por leitura de um símbolo da palavra de entrada, podem conduzir a múltiplos estados ou mesmo a nenhum estado.

Podem ainda ocorrer transições por ε, em que o estado do autómato muda sem que seja lido qualquer símbolo da palavra. Por enquanto, não vamos considerar este caso.

No autómato representado na figura seguinte existem transições para os estados q e r a partir do estado p, por leitura da letra a. Existe também uma transição para o estado r a partir do estado q sem que seja lida qualquer letra (uma transição ε).

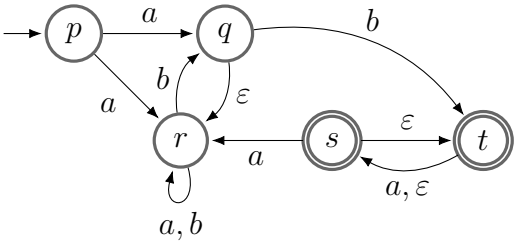


Figura 2.2.1 Um autómato não determinista

Definição 2.2.2 Autómato Finito Não Determinista.

Um **autómato finito não determinista** (AFND) é um quíntuplo ordenado

$$A = (Q, \Sigma, \delta, s, F)$$

onde:

- Q é um conjunto finito, não vazio, de estados;
- Σ é um alfabeto de símbolos de entrada;
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ é uma função (parcial) de transição ou de mudança de estado;
- $s \in Q$ é o estado inicial;
- $F \subseteq Q$ é o conjunto dos estados de aceitação.

Continuando com o AFND ilustrado na Figura 2.2.1, p. 39, suponhamos que este se encontra no estado p e que o próximo símbolo de entrada é a. Podendo escolher entre transitar para o estado q e transitar para o estado r, vamos considerar que o autómato opta pela "melhor opção" que possa conduzir a uma eventual aceitação da palavra presente na fita.

Se todas as alternativas conduzem a um estado de rejeição então o autómato rejeita a palavra lida. Caso contrário, ou seja, quando existe uma possibilidade de parar num estado de aceitação após completar a leitura da palavra então o autómato aceita.

Exemplo 2.2.3

Sobre o alfabeto $\Sigma = \{0, 1\}$, consideremos a linguagem

$$L = \{w \in \Sigma^\star : \ w \text{ termina em } 01\}.$$

Problema. Pretendemos construir um autómato finito não determinista reconhecedor da linguagem L.

Solução. A ideia do funcionamento do autómato é que este vai lendo os símbolos da palavra até ao momento em que "adivinha" que faltam apenas dois símbolos, verificando então se estes são o 0 seguido do 1.

O diagrama de transições é assim o seguinte:

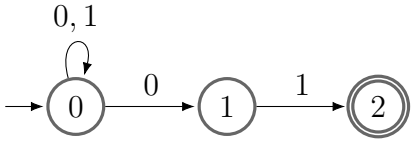


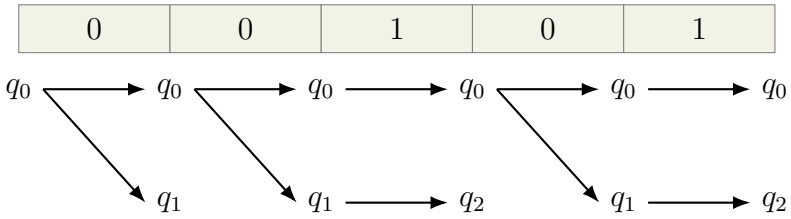
Figura 2.2.4 Diagrama de transições do AFND

O diagrama anterior corresponde à seguinte tabela de transições:

Tabela 2.2.5 Tabela de transições do AFND

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

A palavra 00101 tem várias possibilidades de reconhecimento:



Uma vez que há um caminho que termina no estado de aceitação q_2, a palavra 00101 pertence à linguagem.

A estrutura de reconhecimento em árvore, ilustrada no exemplo anterior, está na origem da definição da extensão da função de transição para AFND.

Definição 2.2.6 Função de Transição Estendida.

A **função de transição estendida** de um AFND $A = (Q, \Sigma, \delta, s, F)$,

$$\delta^\star : Q \times \Sigma^\star \to \mathcal{P}(Q)$$

é definida indutivamente por:

Caso base. Para $q \in Q$, $\delta^\star(q, \varepsilon) = \{q\}$.

Passo indutivo. Para $q \in Q$ e $w = xa$, com $x \in \Sigma^\star$ e $a \in \Sigma$, se

$$\delta^\star(q, x) = \{p_1, p_2, \ldots, p_k\}$$

e se

$$\bigcup_{i=1,\ldots,k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$$

então $\delta^\star(q, w) = \{r_1, r_2, \ldots, r_m\}$.

Notamos que para $w = xa$, com $x \in \Sigma^\star$ e $a \in \Sigma$,

$$\delta^\star(q, w) = \delta^\star(q, xa) = \bigcup_{p \in \delta^\star(q,x)} \delta(p, a).$$

Exemplo 2.2.7

Usando a Tabela 2.2.5, p. 41 do exemplo anterior, constatamos que o desenvolvimento de $\delta^\star(q_0, w)$ para a palavra $w = 00101$, semelhante a uma árvore, é o seguinte:

- $\delta^\star(q_0, \varepsilon) = \{q_0\}$
- $\delta^\star(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\delta^\star(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\delta^\star(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
- $\delta^\star(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\delta^\star(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

A noção de transição estendida permite-nos indicar formalmente que uma palavra w é aceite por um AFND se e só se $\delta^\star(s, w) \cap F \neq \emptyset$, onde s é o estado inicial e F é o conjunto de estados de aceitação.

Assim, uma palavra w é aceite por um AFND quando o conjunto $\delta^\star(s, w)$ contém *pelo menos um* dos estados de aceitação.

Exemplo 2.2.8

No Exemplo 2.2.7, p. 42 vimos que $\delta^\star(q_0, 00101) = \{q_0, q_2\}$ e como $q_2 \in F$, a palavra $w = 00101$ é aceite. Vimos também que $\delta^\star(q_0, 00) = \{q_0, q_1\}$ e como $\{q_0, q_1\} \cap F = \emptyset$, a palavra 00 é rejeitada.

A linguagem reconhecida por um AFND é naturalmente constituída por todas as palavras que são aceites.

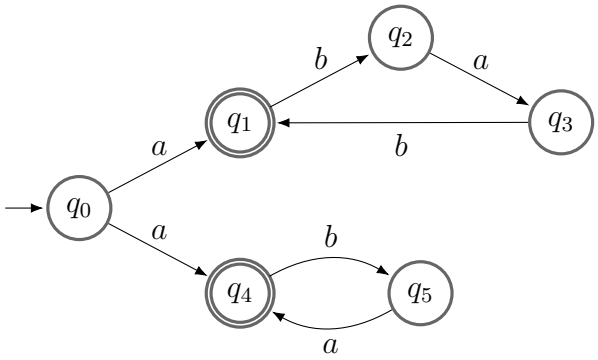
Definição 2.2.9 Linguagem Reconhecida por um AFND.

A **linguagem reconhecida** por um AFND $A = \big(Q, \Sigma, \delta, s, F\big)$ é

$$\mathcal{L}(A) = \{w \in \Sigma^\star : \ \delta^\star(s, w) \cap F \neq \emptyset\}.$$

Exemplo 2.2.10

Pretendemos caracterizar a linguagem reconhecida pelo seguinte AFND.



Neste autómato existem duas formas de aceitar a palavra a: uma pela transição do estado inicial q_0 para o estado de aceitação q_1 e outra pela transição do estado inicial para o estado de aceitação q_4.

Seguindo a transição de q_0 para q_1 constatamos que o autómato aceita também a palavra a concatenada com uma ou mais repetições da palavra bab.

Por outro lado, seguindo a transição de q_0 para q_4 constatamos que aceita ainda a palavra a concatenada com uma ou mais repetições da palavra ba.

Assim, a linguagem reconhecida por este AFND é

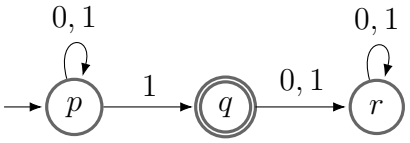
$$L = \{a\}\{bab\}^\star \cup \{a\}\{ba\}^\star.$$

Vimos já no Teorema 2.1.13, p. 38 e no Exemplo 2.1.14, p. 38 que para obter um autómato para a linguagem complementar da linguagem reconhecida por um autómato

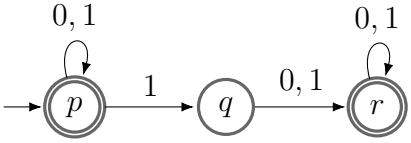
determinista é essencial que ele seja completo. Mostramos agora, por intermédio de um exemplo, que não é possível aplicar o método do complementar quando o autómato não é determinista.

Exemplo 2.2.11

Consideremos o autómato não determinista (completo), A, reconhecedor da linguagem $\mathcal{L}(A) = \{0,1\}^\star\{1\}$.



Quando aplicamos o método do complementar, trocando os estados de aceitação com os de rejeição, obtemos o autómato, B, reconhecedor da linguagem $\mathcal{L}(B) = \{0,1\}^\star \neq \overline{\mathcal{L}(A)}$.



2.3 Equivalência entre AFND e AFD

Dado um autómato não determinista $N = \left(Q_N, \Sigma, \delta_N, s_N, F_N\right)$ vamos construir um autómato determinista $D = \left(Q_D, \Sigma, \delta_D, s_D, F_D\right)$ que é equivalente a N, isto é, que reconhece a mesma linguagem que a reconhecida por N.

Definição 2.3.1 Determinização de um AFND.

A determinização de um AFND $N = (Q_N, \Sigma, \delta_N, s_N, F_N)$ é o autómato finito determinista $D = (Q_D, \Sigma, \delta_D, s_D, F_D)$, onde:

- $s_D = \{s_N\}$;
- $Q_D = \mathcal{P}(Q_N)$;
- $F_D = \{X \in \mathcal{P}(Q_N) : \ X \cap F_N \neq \emptyset\}$;
- para $X \in \mathcal{P}(Q_N)$ e $a \in \Sigma$,

$$\delta_D(X, a) = \bigcup_{p \in X} \delta_N(p, a).$$

Teorema 2.3.2

Seja $D = (Q_D, \Sigma, \delta_D, s_D, F_D)$ *a determinização de um autómato não determinista* $N = \left(Q_N, \Sigma, \delta_N, s_N, F_N\right)$. *Nestas condições, para* $w \in \Sigma^\star$,

$$\delta^\star{}_D\left(s_D, w\right) = \delta^\star{}_N\left(s_N, w\right).$$

Demonstração.

Vamos usar indução estrutural sobre as palavras de $\Sigma^\star$.

Caso base. Para $w = \varepsilon$, aplicando a definição de s_D, obtemos $\delta^\star{}_D\left(s_D, \varepsilon\right) = s_D = \{s_N\} = \delta^\star{}_N\left(s_N, \varepsilon\right)$.

Passo indutivo. Admitindo como hipótese de indução que a propriedade é verdadeira para $w \in \Sigma^\star$, vamos mostrar que é também verdadeira para wa, com $a \in \Sigma$.

Pela definição de função de transição estendida de um AFD, temos que:

$$\delta^\star{}_D\left(s_D, wa\right) = \delta_D\left(\delta^\star{}_D(s_D, w), a\right).$$

Aplicando a hipótese de indução, obtemos:

$$\delta_D\left(\delta^\star{}_D(s_D, w), a\right) = \delta_D\left(\delta^\star{}_N\left(s_N, w\right), a\right).$$

Pela definição de δ_D, temos que:

$$\delta_D\left(\delta^\star{}_N\left(s_N, w\right), a\right) = \bigcup_{p \in \delta^\star{}_N(s_N, w)} \delta_N(p, a).$$

Finalmente, aplicando a definição de função de transição estendida de um AFND, obtemos:

$$\bigcup_{p \in \delta^\star{}_N(s_N, w)} \delta_N(p, a) = \delta^\star{}_N\left(s_N, wa\right).$$

Corolário 2.3.3

$$\mathcal{L}(D) = \mathcal{L}(N).$$

Demonstração.

$$
\begin{aligned}
& w \in \mathcal{L}(D) \\
\iff & \delta^\star{}_D(s_D, w) \in F_D && \text{(definição de linguagem de um AFD)} \\
\iff & \delta^\star{}_N(s_N, w) \in F_D && \text{(teorema anterior)} \\
\iff & \delta^\star{}_D(s_D, w) \cap F_N \neq \emptyset && \text{(definição de } F_D\text{)} \\
\iff & w \in \mathcal{L}(N) && \text{(definição de linguagem de um AFND)}
\end{aligned}
$$

Exemplo 2.3.4

A partir da Tabela 2.2.5, p. 41 de transições do autómato não determinista definido no Exemplo 2.2.3, p. 40,

$$N = \big(Q_N, \Sigma, \delta_N, s_N, F_N\big) = \big(\{q_0, q_1, q_2\}, \{0, 1\}, \delta_N, q_0, \{q_2\}\big)$$

construímos o autómato determinista

$$D = \big(Q_D, \Sigma, \delta_D, s_D, F_D\big)$$

representado pela seguinte tabela de transições:

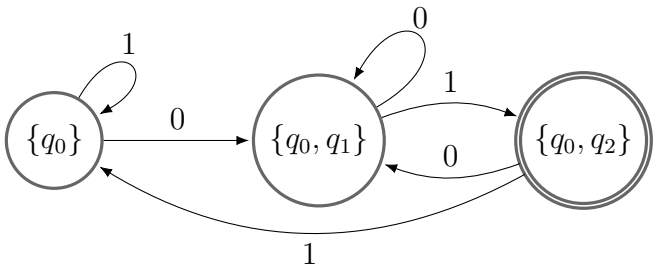
δ_D	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\to \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$*\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

Em seguida, calculamos todas as transições possíveis a partir do estado inicial:

- $\begin{cases} \delta_D(\{q_0\}, 0) = \delta_N(q_0, 0) = \{q_0, q_1\} \\ \delta_D(\{q_0\}, 1) = \delta_N(q_0, 1) = \{q_0\} \end{cases}$
- $\begin{cases} \delta_D(\{q_0, q_1\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0) = \{q_0, q_1\} \\ \delta_D(\{q_0, q_1\}, 1) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1) = \{q_0, q_2\} \end{cases}$
- $\begin{cases} \delta_D(\{q_0, q_2\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_2, 0) = \{q_0, q_1\} \\ \delta_D(\{q_0, q_2\}, 1) = \delta_N(q_0, 1) \cup \delta_N(q_2, 1) = \{q_0\} \end{cases}$

Concluímos que *apenas 3 dos 8 estados podem ser atingidos.*

Assim, eliminando os estados que não são atingíveis, simplificamos a tabela de transições e obtemos o seguinte autómato determinista:



Notamos que para cada AFND existe um AFD que lhe é equivalente. Contudo, se o AFND tiver n estados, o AFD equivalente pode ter 2^n estados. Mas, em muitos casos, é possível obter um AFD com um número de estados próximo do número de estados do AFND original. De facto, conforme ilustrado no exemplo anterior, existem muitos estados que não são atingíveis a partir do estado inicial.

Podemos construir directamente a tabela de transições de um AFD equivalente a um dado AFND de tal modo que todos os estados presentes nessa tabela são atingíveis (um autómato acessível). Evitamos assim o cálculo das transições correspondentes a estados não atingíveis. Esta tabela é construída incrementalmente, partindo do estado inicial, linha a linha. Só acrescentamos linhas à tabela quando no cálculo de uma linha surjam novos estados. Esta técnica é ilustrada no Exemplo 2.3.6, p. 48 e descrita no algoritmo seguinte.

Algoritmo 2.3.5 Método dos Subconjuntos.

Entrada: um AFND $N = \big(Q_N, \Sigma, \delta_N, s_N, F_N\big)$.

Saída: um AFD equivalente e acessível $D = (Q_D, \Sigma, \delta_D, s_D, F_D)$.

$s_D \leftarrow \{s_N\}$; $Q_D \leftarrow \{s_D\}$;

Para cada estado $X \in Q_D$ ainda não visitado

visitar X;

Para cada símbolo $a \in \Sigma$

$Y \leftarrow \bigcup_{p \in X} \delta_N(p, a)$; $Q_D \leftarrow Q_D \cup \{Y\}$; $\delta_D(X, a) \leftarrow Y$;

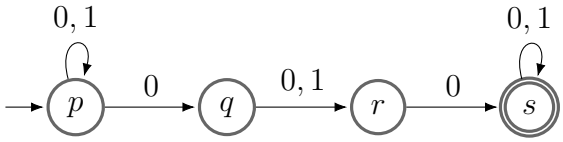
$F_D \leftarrow \emptyset$;

Para cada estado $X \in Q_D$

Se $X \cap F_N \neq \emptyset$ então $F_D \leftarrow F_D \cup \{X\}$;

Exemplo 2.3.6

Vamos determinizar o AFND representado pelo seguinte diagrama de transições:



A partir do AFND construímos directamente a tabela de um AFD equivalente na qual todos os estados são atingíveis (evitando o cálculo das transições correspondentes a estados que não atingíveis).

A tabela é construída linha a linha, partindo do estado inicial, e acrescentando apenas os novos estados que vão sendo calculados.

δ_D	0	1
$\rightarrow \{p\}$	$\{p,q\}$	$\{p\}$
$\{p,q\}$	$\{p,q,r\}$	$\{p,r\}$
$\{p,q,r\}$	$\{p,q,r,s\}$	$\{p,r\}$
$\{p,r\}$	$\{p,q,s\}$	$\{p\}$
$*\{p,q,r,s\}$	$\{p,q,r,s\}$	$\{p,r,s\}$
$*\{p,q,s\}$	$\{p,q,r,s\}$	$\{p,r,s\}$
$*\{p,r,s\}$	$\{p,q,s\}$	$\{p,s\}$
$*\{p,s\}$	$\{p,q,s\}$	$\{p,s\}$

Analisámos apenas o problema da conversão de autómatos não deterministas em autómatos deterministas. Uma vez que a conversão de um autómato determinista num autómato não determinista é *trivial* (porquê?), concluímos que a classe das linguagens reconhecidas por AFD é igual à classe das linguagens reconhecidas por AFND.

2.4 Autómatos com transições ε

Consideramos agora a possibilidade da função de transição de um autómato permitir transições de um estado para outro sem avançar na leitura da palavra. A este tipo de transições, chamamos *transições ε*.

Exemplo 2.4.1

O autómato representado na figura seguinte reconhece representações decimais de números reais. As transições por *dígito* condensam as 10 transições possíveis pelos dígitos decimais $0, 1, \ldots, 9$ e a transição ε do estado inicial (estado q_0) para o estado q_1 permite o reconhecer números sem sinal.

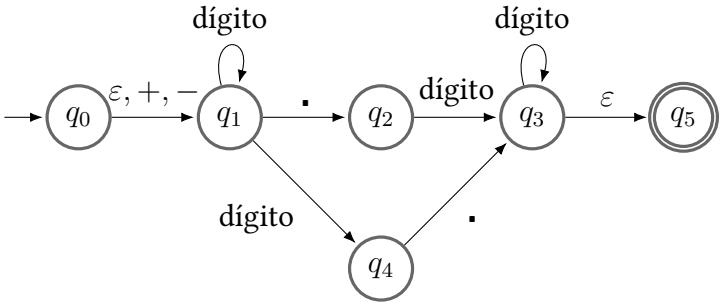


Figura 2.4.2 Diagrama de transições de um autómato não determinista com transições ε.

A função de transição deste autómato tem em conta a existência de transições ε, sendo dada pela seguinte tabela:

Tabela 2.4.3 Tabela de transições de um autómato não determinista com transições ε.

δ	ε	$+, -$	.	dígito
$\rightarrow q_0$	$\{q_1\}$	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$	$\{q_1, q_4\}$
q_2	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_3	$\{q_5\}$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_4	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
$*q_5$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

As definições e propriedades apresentadas nas secções anteriores não são válidas para esta classe de autómatos. Vamos assim generalizá-las de tal modo que os AFND sem transições ε são considerados como AFND com transições ε em que $\delta(q, \varepsilon) = \emptyset$, para todos os estados q.

Definição 2.4.4 Autómato Finito Não Determinista com Transições ε.

Um **autómato finito não determinista com transições ε** (AFNDε) ε é um quíntuplo ordenado $A = (Q, \Sigma, \delta, s, F)$, onde:

- Q é um conjunto finito, não vazio, de estados;
- Σ é um alfabeto de símbolos de entrada;
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ é a função (parcial) de transição ou de mudança de estado;
- $s \in Q$ é o estado inicial;
- $F \subseteq Q$ é o conjunto dos estados de aceitação.

A existência de transições ε num autómato modifica naturalmente o conjunto de estados que são atingíveis a partir de um dado estado e, consequentemente, a definição da função de transição estendida.

Formalizamos agora a noção do conjunto de estados que um autómato pode atingir a partir de um determinado estado utilizando apenas transições ε.

Definição 2.4.5 Fechoε.

Seja p um qualquer estado de um AFNDε $A = (Q, \Sigma, \delta, s, F)$. O fecho$\varepsilon$ de p é definido indutivamente por:

Caso base. $p \in \text{fecho}\varepsilon(p)$;

Passo indutivo. Se $q \in \text{fecho}\varepsilon(p)$ e se existe $r \in Q$ tal que $r \in \delta(q, \varepsilon)$ então $r \in \text{fecho}\varepsilon(p)$.

Da definição anterior, resulta que um estado q pertence ao fechoε de um estado p sempre que no diagrama de transições exista um passeio de p para q com os arcos rotulados apenas por ε.

Se $P \subseteq Q$ é um conjunto de estados então

$$\text{fecho}\varepsilon(P) = \bigcup_{p \in P} \text{fecho}\varepsilon(p).$$

O algoritmo seguinte permite calcular o fechoε de um conjunto de estados.

Algoritmo 2.4.6 Cálculo do fechoε.

Entrada: um AFNDε $E = (Q, \Sigma, \delta, s, F)$ e um conjunto $P \subseteq Q$.

Saída: fecho$\varepsilon(P)$.

$X \leftarrow P$; fecho$\varepsilon(P) \leftarrow P$;

Enquanto $X \neq \emptyset$

seleccionar $p \in X$;

$X \leftarrow X \setminus \{p\}$;

Para cada estado $q \in \delta(p, \varepsilon)$

Se $q \notin$ fecho$\varepsilon(P)$ então

fecho$\varepsilon(P) \leftarrow$ fecho$\varepsilon(P) \cup \{q\}$;

$X \leftarrow X \cup \{q\}$;

Exemplo 2.4.7

Consideremos o AFNDε representado pelo seguinte diagrama.

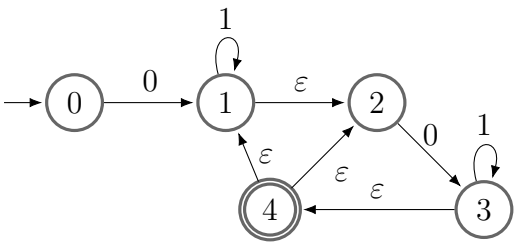


Figura 2.4.8 Autómato não determinista com transições ε

O fechoε de cada um dos estados deste autómato é:

- fecho$\varepsilon(q_0) = \{q_0\}$;
- fecho$\varepsilon(q_1) = \{q_1, q_2\}$;
- fecho$\varepsilon(q_2) = \{q_2\}$;

- fecho$\varepsilon(q_3) = \{q_1, q_2, q_3, q_4\}$;
- fecho$\varepsilon(q_4) = \{q_1, q_2, q_4\}$.

Para os autómatos com transições ε, é também a função de transição estendida que formaliza o modo de funcionamento do autómato no processo de aceitação/rejeição das palavras. Nesta classe de autómatos, é necessário ter em conta as possíveis mudanças de estado por transições ε, tanto antes como depois de uma transição por uma letra.

Definição 2.4.9 Função de Transição Estendida de um AFNDε.

A função de transição estendida de um AFNDε $A = (Q, \Sigma, \delta, s, F)$,

$$\delta^\star : Q \times \Sigma^\star \to \mathcal{P}(Q),$$

é definida indutivamente por

Caso base. Para $q \in Q$,

$$\delta^\star(q, \varepsilon) = \text{fecho}\varepsilon(q).$$

Passo indutivo. Para $q \in Q$, $x \in \Sigma^\star$ e $a \in \Sigma$, se

$$\delta^\star(q, x) = \{p_1, p_2, \ldots, p_k\}$$

e se

$$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$$

então

$$\delta^\star(q, xa) = \{\text{fecho}\varepsilon(r_1), \text{fecho}\varepsilon(r_2), \ldots, \text{fecho}\varepsilon(r_m)\}.$$

Para $a \in \Sigma$ e $x \in \Sigma^\star$, notamos que

$$\delta^\star(q, xa) = \bigcup_{p \in \delta^\star(q,x)} \text{fecho}\varepsilon(\delta(p, a)).$$

Por outro lado, se $R = \bigcup_{p \in \text{fecho}\varepsilon(q)} \delta(p, a)$ então

$$\delta^\star(q, ax) = \bigcup_{r \in R} \delta^\star(r, x).$$

Se $R \subseteq Q$ é um conjunto não vazio de estados e $a \in \Sigma \cup \{\varepsilon\}$ então as acções das funções de transição δ e de transição estendida $\delta^\star$ sobre o conjunto R são dadas

por

$$\delta(R,a) = \bigcup_{r\in R} \delta(r,a)$$

e

$$\delta^\star(R,w) = \bigcup_{r\in R} \delta^\star(r,w).$$

A convenção anterior permite-nos compactar a definição da função de transição estendida: para $q \in Q$, $a \in \Sigma$ e $x \in \Sigma^\star$,

$$\delta^\star(q,xa) = \text{fecho}\varepsilon\left(\delta\left(\delta^\star(q,x),a\right)\right) \tag{2.4.1}$$

e

$$\delta^\star(q,ax) = \delta^\star\left(\delta(\text{fecho}\varepsilon(q),a),x\right). \tag{2.4.2}$$

Exemplo 2.4.10

Usando a fórmula (2.4.2), o desenvolvimento da função de transição estendida do autómato não determinista do Exemplo 2.4.7, p. 51 com a palavra 0100 é:

$$\begin{aligned}
\delta^\star(q_0,0100) &= \delta^\star\left(\delta(\text{fecho}\varepsilon(q_0),0),101\right)\\
&= \delta^\star\left(\delta(\{q_0\},0),100\right)\\
&= \delta^\star\left(\{q_1\},101\right)\\
&= \delta^\star\left(\delta(\text{fecho}\varepsilon(q_1),1),00\right)\\
&= \delta^\star\left(\delta(\{q_1,q_2\},1),00\right)\\
&= \delta^\star\left(\{q_1\},00\right)\\
&= \delta^\star\left(\delta(\text{fecho}\varepsilon(q_1),0),0\right)\\
&= \delta^\star\left(\delta(\{q_1,q_2\},0),0\right)\\
&= \delta^\star\left(\{q_3\},1\right)\\
&= \delta^\star\left(\delta(\text{fecho}\varepsilon(q_3),0),\varepsilon\right)\\
&= \delta^\star\left(\delta(\{q_1,q_2,q_3,q_4\},0),\varepsilon\right)\\
&= \delta^\star\left(\{q_3\},\varepsilon\right) = \text{fecho}\varepsilon(q_3) = \{q_1,q_2,q_3,q_4\}.
\end{aligned}$$

Definição 2.4.11 Linguagem Reconhecida por um AFNDε.

A *linguagem reconhecida* por um AFNDε $A = \left(Q,\Sigma,\delta,s,F\right)$ é

$$\mathcal{L}(A) = \{w \in \Sigma^\star : \ \delta^\star(s,w) \cap F \neq \emptyset\}.$$

Exemplo 2.4.12

Conforme vimos no Exemplo 2.4.10, p. 53, $\delta^\star(0101,q_0) = \{q_1,q_2,q_3,q_4\}$. Uma vez que $F = \{q_4\}$, a palavra 0101 pertence a $\mathcal{L}(A)$.

Baseado na definição indutiva de função de transição estendida, o algoritmo seguinte determina o conjunto $\delta^\star(q, w)$, para uma palavra w e um estado $q \in Q$.

Algoritmo 2.4.13 Cálculo da Função Estendida de um AFNDε.

Entrada: um AFNDε, uma palavra $w = a_1 a_2 \cdots a_k$ e um estado $q \in Q$.

Saída: o conjunto $X = \delta^\star(q, w)$.

$X \leftarrow \text{fecho}_\varepsilon(q)$;

Para i desde 1 até k

$X \leftarrow \text{fecho}_\varepsilon\big(\bigcup_{p \in X} \delta(p, a_i)\big)$;

Em particular, uma palavra w é reconhecida por um AFNDε se só se, partindo do estado inicial, o conjunto X obtido com o algoritmo anterior satisfaz $X \cap F \neq \emptyset$.

Exemplo 2.4.14

O desenvolvimento de $\delta^\star(q_0, w)$ com $w = .2$ para o AFNDε ilustrado na Figura 2.4.2, p. 49 é o seguinte:

Passo 0.

$$X = \text{fecho}_\varepsilon(q_0) = \{q_0, q_1\};$$

Passo 1.

$$\begin{aligned} X &= \text{fecho}_\varepsilon(\delta(q_0, .) \cup \delta(q_1, .)) \\ &= \text{fecho}_\varepsilon(\{q_2\}) = \{q_2\}; \end{aligned}$$

Passo 2.

$$\begin{aligned} X &= \text{fecho}_\varepsilon(\delta(q_2, 2)) \\ &= \text{fecho}_\varepsilon(\{q_3\}) = \{q_3, q_5\}. \end{aligned}$$

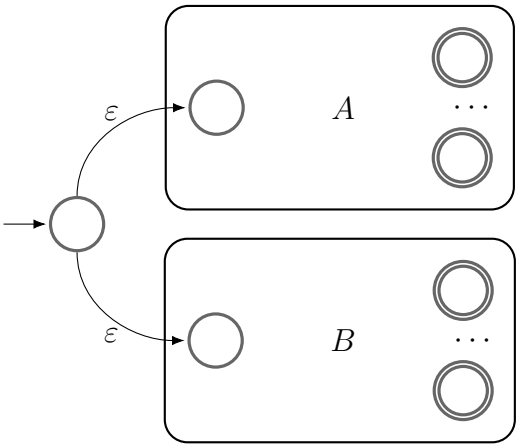
Uma vez que $X \cap F = \{q_3, q_5\} \cap \{q_5\} \neq \emptyset$, concluímos que $w = .2 \in \mathcal{L}(A)$.

A possibilidade de utilizar transições ε permite definir processos simples de construção de autómatos para a união e concatenação das linguagens de dois autóma-

tos, bem como para o fecho de Kleene da linguagem de um autómato.

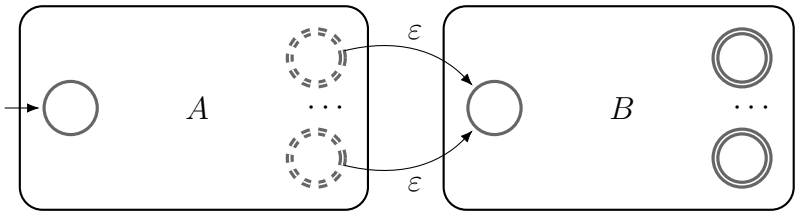
Exemplo 2.4.15 União de AFNDε.

Conforme ilustrado na figura seguinte, a partir de dois AFNDε A e B, construímos um AFNDε para a linguagem $\mathcal{L}(A) \cup \mathcal{L}(B)$, criando um novo estado inicial, de não aceitação, ligado por transições ε aos estados iniciais dos autómatos (os quais deixam de ser estados iniciais).



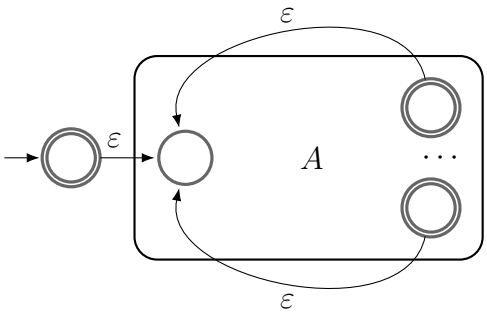
Exemplo 2.4.16 Concatenação de AFNDε.

Conforme ilustrado na figura seguinte, a partir de dois AFNDε A e B, construímos um AFNDε para a linguagem $\mathcal{L}(A)\mathcal{L}(B)$, ligando por transições ε os estados de aceitação do autómato A, os quais deixam de ser de aceitação, ao estado inicial do autómato B (o qual deixa de ser inicial).



Exemplo 2.4.17 Fecho de Kleene de AFNDε.

Conforme ilustrado na figura seguinte, a partir de um AFNDε A, construímos um AFNDε para a linguagem $\mathcal{L}(A)^\star$. Começamos por ligar, por transições ε, os estados de aceitação do autómato ao estado inicial. Em seguida, criamos um novo estado inicial, o qual é também de aceitação, ligado por uma transição ε ao estado inicial do autómato original (o qual deixa de ser inicial).



2.5 Equivalência entre AFNDε e AFD

Dado um autómato não determinista com transições ε vamos construir um autómato determinista equivalente, isto é, um AFD que reconhece a mesma linguagem. Neste caso, a determinização tem de ter em conta que podem existir transições ε tanto antes como após a transição por um símbolo do alfabeto.

Definição 2.5.1 Determinização de um AFNDε.

Se $E = (Q_E, \Sigma, \delta_E, s_E, F_E)$ é um AFNDε então a sua determinização é o AFD $D = (Q_D, \Sigma, \delta_D, s_D, F_D)$ definido por:

- $Q_D = \{P \in \mathcal{P}(Q_E) : \ P = \text{fecho}\varepsilon(P)\}$;
- $s_D = \text{fecho}\varepsilon(s_E)$;
- $F_D = \{P \in Q_D : \ P \bigcap F_E \neq \emptyset\}$;
- Para $P \in Q_D$ e $a \in \Sigma$,

$$\delta_D(P, a) = \text{fecho}\varepsilon\big(\bigcup_{p \in P} \delta_E(p, a)\big).$$

Teorema 2.5.2

Seja $E = (Q_E, \Sigma, \delta_E, s_E, F_E)$ um AFNDε e seja $D = (Q_D, \Sigma, \delta_D, s_D, F_D)$ a sua determinização. Nestas condições, para $w \in \Sigma^\star$,

$$\delta^\star{}_D(s_D, w) = \delta^\star{}_E(s_E, w).$$

Demonstração.

Demonstração por indução sobre w.

Corolário 2.5.3

$$\mathcal{L}(D) = \mathcal{L}(E).$$

De forma semelhante ao método dos subconjuntos apresentado para AFND sem transições ε, o algoritmo seguinte constrói a tabela de transições do AFD de forma incremental, assegurando que todos os estados do autómato obtido são acessíveis.

Algoritmo 2.5.4 Método dos subconjuntos fechados.

Entrada: um AFNDε $E = (Q_E, \Sigma, \delta_E, s_E, F_E)$.

Saída: um AFD equivalente, $D = (Q_D, \Sigma, \delta_D, s_D, F_D)$.

$s_D \leftarrow \text{fecho}\varepsilon(s_E)$; $Q_D \leftarrow \{s_D\}$;

Para cada estado $P \in Q_D$ ainda não visitado

visitar P;

Para cada símbolo $a \in \Sigma$

$X \leftarrow \text{fecho}\varepsilon\left(\bigcup_{p \in P} \delta(p, a)\right)$;

$Q_D \leftarrow Q_D \cup \{X\}$;

$\delta_D(P, a) \leftarrow X$;

$F_D \leftarrow \emptyset$;

Para cada estado $P \in Q_D$

Se $P \cap F_E \neq \emptyset$ então $F_D \leftarrow F_D \cup \{P\}$;

Existe sempre a possibilidade do AFD obtido pelo método anterior ter 2^n estados, sendo n o número de estados do autómato não determinista. Para além disso, não é garantido que este algoritmo produza um AFD com o menor número de estados possível. No entanto, como veremos no capítulo seguinte, existem algoritmos para minimizar o número de estados dos AFD.

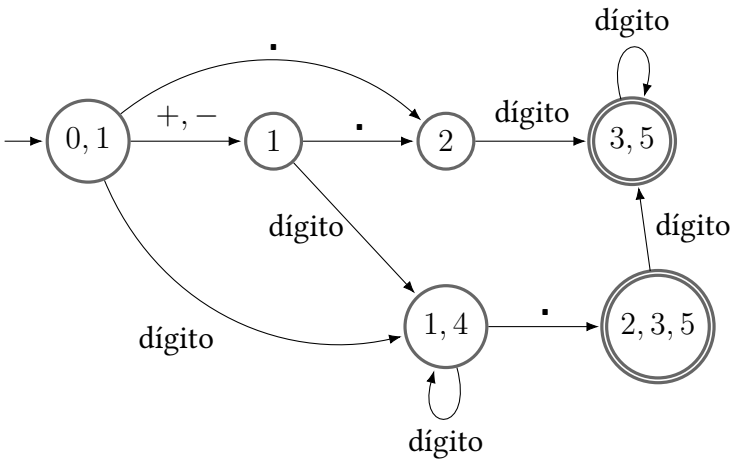
Exemplo 2.5.5

Consideremos a Tabela 2.4.3, p. 49 de transições do AFNDε do Exemplo 2.4.1, p. 49.

Utilizando o método acima descrito, obtemos a tabela de transições de um AFD equivalente:

δ_D	$+, -$	.	dígito
$\to \{q_0, q_1\}$	$\{q_1\}$	$\{q_2\}$	$\{q_1, q_4\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$	$\{q_1, q_4\}$
$\{q_2\}$	$\emptyset$	$\emptyset$	$\{q_3, q_5\}$
$\{q_1, q_4\}$	$\emptyset$	$\{q_2, q_3, q_5\}$	$\{q_1, q_4\}$
$*\{q_3, q_5\}$	$\emptyset$	$\emptyset$	$\{q_3, q_5\}$
$*\{q_2, q_3, q_5\}$	$\emptyset$	$\emptyset$	$\{q_3, q_5\}$

Na tabela anterior omitimos a linha correspondente ao estado $\emptyset$, correspondente ao estado ratoeira. O diagrama de transições deste autómato é assim:



2.6 Exercícios

Na resolução dos exercícios seguintes, para além do "desenho" dos autómatos, é importante apresentar uma explicação sucinta da ideia subjacente à sua construção.

Devemos sempre confirmar que as construções funcionam com palavras pequenas das linguagens em causa. Em particular, nunca esqueçamos de verificar se o autómato funciona para a palavra vazia!

1. Modifique o autómato apresentado no Exemplo 2.1.1, p. 30, de modo que as formas admitidas para comentários sejam apenas $(* \ldots *)$ e $\{\ldots\}$, pelo que não são admitidas as formas $(* \ldots\}$ e $\{\ldots *)$.

Sugestão. Basta duplicar o estado *comentário* separando o reconhecimento da forma $(*\ldots*)$ do reconhecimento da forma $\{\ldots\}$.

2. Construa autómatos finitos deterministas completos que reconheçam as seguintes linguagens definidas sobre o alfabeto $\Sigma = \{a, b\}$:

(a) $\{w \in \Sigma^\star : \ w \text{ tem uma letra}\}$

(b) $\{w \in \Sigma^\star : \ w \text{ não tem letras}\}$

(c) $\{w \in \Sigma^\star : \ w \text{ começa com } a\}$

(d) $\{w \in \Sigma^\star : \ w \text{ não começa com } a\}$

(e) $\{w \in \Sigma^\star : \ w \text{ começa com } ab\}$

(f) $\{w \in \Sigma^\star : \ w \text{ acaba em } a\}$

(g) $\{w \in \Sigma^\star : \ w \text{ não acaba em } a\}$

(h) $\{awa : \ w \in \Sigma^\star\}$

(i) $\{axaaya : \ x, y \in \Sigma^\star\}$

(j) $\{xwx \in \Sigma^\star : \ w \in \Sigma^\star, x \in \Sigma\}$

(k) $\{w \in \Sigma^\star : \ w \text{ tem exactamente um } a\}$

(l) $\{w \in \Sigma^\star : \ w \text{ tem pelo menos um } a\}$

(m) $\{w \in \Sigma^\star : \ w \text{ tem no máximo um } a\}$

3. Determine diagramas de transições de autómatos finitos deterministas que reconheçam as seguintes linguagens definidas sobre o alfabeto $\Sigma = \{a, b\}$:

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

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

(c) $\{a^n b^m : \ n, m > 0\}$

(d) $\{a^n : \ n \text{ é par}\}$

(e) $\{a^n : \ n \text{ é ímpar}\}$

(f) $\{a^n b^m : \ m \text{ e } n \text{ são números pares}\}$

(g) $\{w \in \{a, b\}^\star : \ |w| \text{ é par}\}$

(h) $\{w \in \{a, b\}^\star : \ |w| \bmod 3 = 0\}$

(i) $\{w \in \{a, b\}^\star : \ |w| \bmod 3 < 2\}$

(j) $\{w \in \{a, b\}^\star : \ \#_a(w) \bmod 3 = 0\}$

(k) $\{w \in \{a, b\}^\star : \ \#_{ab}(w) \bmod 2 = 0\}$

(l) $\{w \in \{a, b\}^\star : \ w \text{ não tem } a\text{'s consecutivos}\}$

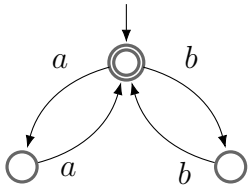
(m) $\{w \in \{a,b\}^\star : \ w$ não tem 3 a's consecutivos$\}$

(n) $\{w \in \{a,b\}^\star : \ w$ não tem duas letras iguais consecutivas$\}$

(o) $\{w \in \{a,b\}^\star : \ w$ não tem 3 letras iguais consecutivas$\}$

(p) $\{w \in \{a,b\}^\star : \ \#_{ab}(w) = \#_{ba}(w)\}$

4. Caracterize a linguagem reconhecida pelo seguinte AFD:



5. Determine um AFD reconhecedor da linguagem

$$\{w \in \{a,b\}^\star : \ \#_a(w)\#_b(w) \text{ é par}\}.$$

6. Mostre que qualquer AFD incompleto pode ser transformado num AFD completo que reconhece a mesma linguagem.

7. Aplique a fórmula (2.4.1) para desenvolver a função de transição estendida do autómato não determinista do Exemplo 2.4.7, p. 51 com a palavra 0101.

Confirme que obtém o mesmo resultado que o que se obtém com a fórmula (2.4.2).

8. Considere o AFNDε do Exemplo 2.4.7, p. 51.

(a) Mostre que todas as palavras começadas por 1 são rejeitadas.

(b) O autómato aceita a palavra vazia?

(c) Qual é a linguagem do autómato?

9. Para $k \in \mathbb{N}$ considere a linguagem

$$L_k = \{w \in \{0,1\}^\star : \ \text{o } k \text{ -ésimo dígito a contar da direita é } 1\}.$$

(a) Construa um AFND reconhecedor de L_k com $k+1$ estados.

(b) Construa um AFD reconhecedor de L_k com 2^k estados. (Sugestão: associe a cada estado uma palavra de k dígitos.)

(c) Mostre que qualquer AFD reconhecedor de L_k tem pelo menos 2^k estados.

10. Construa autómatos não deterministas, possivelmente com transições ε, que reconheçam as linguagens seguintes.

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

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

(c) $\{a^m b^n : \ m, n \text{ são pares}\}$

(d) $\{a^m b^n : \ m, n \text{ são ímpares}\}$

(e) $\{awb : \ w \in \{a, b\}^\star\}$

(f) $\{w \in \{0, 1\}^\star : \ w \text{ é a representação binária de um par}\}$

(g) $\{w \in \{0, 1\}^\star : \ w \text{ acaba em } 01\}$

(h) $\{w \in \{0, 1\}^\star : \ w \text{ acaba em } 0 \text{ ou em } 11\}$

(i) $\{w \in \{0, 1\}^\star : \ w \text{ começa com } 00 \text{ ou com } 01\}$

(j) $\{w \in \{0, 1\}^\star : \ w \text{ acaba em } 00 \text{ ou em } 01\}$

(k) $\{w \in \{0, 1\}^\star : \ w \text{ começa com } 010 \text{ ou acaba em } 110\}$

(l) $\{w \in \{0, 1\}^\star : \ w \text{ começa com } 010 \text{ e acaba em } 110\}$

11. Construa autómatos deterministas equivalentes aos seguintes autómatos não deterministas:

(a)

δ	0	1
$\to p$	$\{p, q\}$	$\{p\}$
q	$\{r, s\}$	$\{t\}$
r	$\{p, r\}$	$\{t\}$
$*s$	$\emptyset$	$\emptyset$
$*t$	$\emptyset$	$\emptyset$

(b)

δ	0	1
$\to p$	$\{p, q\}$	$\{p, r\}$
q	$\{r, s\}$	$\emptyset$
r	$\{p, r\}$	$\emptyset$
$*s$	$\{p\}$	$\{q\}$

(c)

δ	a	b
$\to p$	$\{q\}$	$\{p\}$
q	$\{s\}$	$\{r, s\}$
r	$\{p, r\}$	$\{s\}$
$*s$	$\emptyset$	$\emptyset$

(d)

δ	ε	a
$\to p$	$\emptyset$	$\{q\}$
$*q$	$\{r\}$	$\emptyset$
r	$\{p\}$	$\emptyset$

(e)

δ	ε	0	1
$\to *p$	$\{r\}$	$\emptyset$	$\{q\}$
q	$\emptyset$	$\{p, r\}$	$\{r\}$
r	$\emptyset$	$\emptyset$	$\emptyset$

(f)

δ	ε	0	1
$\to p$	$\{q\}$	$\{q\}$	$\emptyset$
$*q$	$\{r\}$	$\{p, r\}$	$\{q, r\}$
r	$\emptyset$	$\{r\}$	$\{q\}$

12. Determinize o seguinte AFNDε:

δ	ε	a	b	c
$\to p$	$\{q, r\}$	$\emptyset$	$\{q\}$	$\{r\}$
q	$\emptyset$	$\{p\}$	$\{r\}$	$\{p, q\}$
$*r$	$\emptyset$	$\emptyset$	$\emptyset$	$\{q\}$

13.

(a) Mostre que para cada AFNDε existe um AFNDε equivalente com um único estado de aceitação.

(b) Será que qualquer qualquer AFD é equivalente a um AFD com apenas um estado de aceitação?

14. Sejam $A = \big(Q_A, \Sigma, \delta_A, s_A, F_A\big)$ e $B = \big(Q_B, \Sigma, \delta_B, s_B, F_B\big)$ dois AFD completos reconhecedores das linguagens L_A e L_B, respectivamente. Mostre que $L_A \cap L_B$ é reconhecida pelo AFD $A \cap B = \big(Q_A \times Q_B, \Sigma, \delta_{A\cap B}, s_{A\cap B}, F_{A\cap B}\big)$, onde:

- $s_{A\cap B} = (s_A, s_B)$;
- $F_{A\cap B} = F_A \times F_B$;
- para $a \in \Sigma$ e $(q_A, q_B) \in Q_A \times Q_B$,

$$\delta_{A\cap B}\big((q_A, q_B),\, a\big) = \big(\delta_A(q_A, a),\, \delta_B(q_B, a)\big).$$

15. Sejam $A = \big(Q_A, \Sigma, \delta_A, s_A, F_A\big)$ e $B = \big(Q_B, \Sigma, \delta_B, s_B, F_B\big)$ autómatos finitos deterministas completos reconhecedores das linguagens L_A e L_B, respectivamente. Mostre que $L_A \cup L_B$ é reconhecida pelo AFD $A \cup B = \big(Q_A \times Q_B,\, \Sigma,\, \delta_{A\cup B},\, s_{A\cup B},\, F_{A\cup B}\big)$, onde:

- $s_{A\cup B} = (s_A, s_B)$;
- $F_{A\cup B} = F_A \times Q_B \cup Q_A \times F_B$;
- para $a \in \Sigma$ e $(q_A, q_B) \in Q_A \times Q_B$,

$$\delta_{A\cup B}\big((q_A, q_B),\, a\big) = \big(\delta_A(q_A, a),\, \delta_B(q_B, a)\big).$$

16. Usando a construção do autómato produto, determine um AFD reconhecedor da linguagem das palavras binárias que têm pelo menos duas ocorrências de 01 e acabam em 11.

17. Usando a construção do autómato produto obtenha um AFD que reconheça a linguagem das palavras sobre o alfabeto $\Sigma = \{a, b\}$ que têm pelo menos dois a's e no máximo dois b's.

18. Considere o alfabeto $\Sigma = \{a, b, c\}$.

(a) Construa um AFNDε com apenas 3 estados e 4 transições que reconheça a linguagem $L = \{abc, ab\}^\star$.

(b) Construa um AFNDε que reconheça a linguagem $L^{-1}L$.

(c) Aplicando o algoritmo de determinização, construa um AFD equivalente ao AFNDε obtido na alínea anterior.

19. Diga, justificando, qual é o valor lógico de cada uma das seguintes afirmações:

(a) Se $A = \big(Q,\, \Sigma,\, \delta,\, s,\, F\big)$ é um autómato finito não determinista que reconhece uma linguagem L então $A^c = \big(Q,\, \Sigma,\, \delta,\, s,\, Q \setminus F\big)$ é um autómato que reconhece a linguagem $\Sigma^\star \setminus L$.

(b) Se $A = \big(Q,\, \Sigma,\, \delta,\, s,\, F\big)$ é um autómato finito determinista que reconhece uma linguagem L então $A^c = \big(Q,\, \Sigma,\, \delta,\, s,\, Q \setminus F\big)$ é um autómato que reconhece a linguagem $\Sigma^\star \setminus L$.

20. Sejam $A = \big(Q_A,\, \Sigma,\, \delta_A,\, s_A,\, F_A\big)$ e $B = (Q_B, \Sigma, \delta_B, s_B, F_B)$ AFD completos reconhecedores das linguagens L_A e L_B, respectivamente. Mostre que qualquer palavra da linguagem $L_A \setminus L_B$ é reconhecida pelo AFD

$$A \setminus B = \big(Q_A \times Q_B,\, \Sigma,\, \delta_{A\setminus B},\, s_{A\setminus B},\, F_{A\setminus B}\big)$$

, onde:

- $s_{A\setminus B} = (s_A, s_B)$;
- $F_{A\setminus B} = F_A \times (Q_B \setminus F_B)$;
- para $a \in \Sigma$ e $(q_A, q_B) \in Q_A \times Q_B$,

$$\delta_{A\setminus B}\big((q_A, q_B), a\big) = \big(\delta_A(q_A, a), \delta_B(q_B, a)\big).$$

21. Considere a seguinte proposta de extensão do método de construção do autómato produto para a união de linguagens de dois AFD referido no exercício Exercício 2.6.15, p. 62, à classe dos AFND:

Sejam $A = \big(Q_A,\, \Sigma,\, \delta_A,\, s_A,\, F_A\big)$ e $B = (Q_B, \Sigma, \delta_B, s_B, F_B)$ AFND reconhecedores das linguagens L_A e L_B, respectivamente. Definimos o autómato produto

$$A \cup B = \big(Q_A \times Q_B,\, \Sigma,\, \delta_{A\cup B},\, s_{A\cup B},\, F_{A\cup B}\big)$$

, onde:

- $s_{A\cup B} = (s_A, s_B)$;
- $F_{A\cup B} = F_A \times Q_B \cup Q_A \times F_B$;
- para $a \in \Sigma$ e $(q_A, q_B) \in Q_A \times Q_B$,

$$\delta_{A\cup B}\big((q_A, q_B),\, a\big) = \delta_A(q_A, a) \times \delta_B(q_B, a).$$

Verifique que com esta definição nem sempre é verdade que $\mathcal{L}(A \cup B) = \mathcal{L}(A) \cup \mathcal{L}(B)$.

22. Seja L a linguagem definida sobre um alfabeto Σ, reconhecida por um AFD A. Construa autómatos finitos reconhecedores das linguagens seguintes.

(a) Para $a \in \Sigma$ fixo, $a \,/\, L = \{x \in \Sigma^\star :\ ax \in L\}$;

(b) Para $a \in \Sigma$ fixo, $L \,/\, a = \{x \in \Sigma^\star :\ xa \in L\}$;

(c) metade$(L) = \{x \in \Sigma^\star :$ existe $y \in \Sigma^\star$ com $|x| = |y|$ e $xy \in L\}$.