4 Autómatos finitos

Neste capítulo vamos introduzir outras estruturas que permitem caracterizar as linguagens regulares. A principal vantagem, destas novas estruturas, sobre a representação com expressões regulares é a de terem, naturalmente, associado um algoritmo que testa (em tempo linear) a pertença de uma palavra à linguagem que representam. Para além desta característica, a expressividade aparente dos autómatos, isto é, a facilidade com que somos capazes de os escrever, é muito maior do que a das expressões regulares.

4.1 Autómatos finitos determinísticos

Definição 4.1 (DFA) *Um autómato finito determinístico (DFA) é um quintúplo ordenado* $\langle S, \Sigma, \delta, s_0, F\rangle$ *em que:*

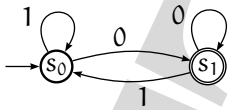
- S *é um conjunto finito, não vazio, a que chamamos o* conjunto dos estados*;*
- Σ *é um conjunto finito, não vazio, que constitui o* alfabeto do autómato*;*
- δ *é uma função total* $\delta : S \times \Sigma \to S$*, chamada* função de transição*;*
- s_0*, com* $s_0 \in S$*, é o* estado inicial*;*
- F*, com* $F \subseteq S$*, é o* conjunto dos estados finais*.*

Um DFA é muitas vezes representado informalmente por um *diagrama*, por um digrafo com estados e arcos etiquetados. Os arcos dos autómatos costumam designar-se por *transições*.

Exemplo 4.2 *O autómato* $A = \langle \{s_0, s_1\}, \{0,1\}, \delta, s_0, \{s_1\} \rangle$ *com*

$$\delta(s_0, 0) = s_1 \quad \delta(s_0, 1) = s_0 \quad \delta(s_1, 0) = s_1 \quad \delta(s_1, 1) = s_0,$$

é representado pelo seguinte diagrama:



Informalmente, para verificarmos se uma dada palavra pertence à linguagem definida por um dado autómato, se uma palavra é "reconhecida" ou "aceite" pelo autómato, procedemos da seguinte forma:

1. Começamos por nos "situarmos" no estado inicial do autómato. Aquele que tem, por convenção, uma pequena seta a apontar para ele.

2. Para cada uma das letras da palavra, e por ordem, "passamos" para o estado do autómato para o qual "aponta" a transição com o nome da letra em causa, a partir do estado em que "estamos".

3. Quando esgotamos as letras da palavra, usando o procedimento do passo anterior, o estado em que nos "encontramos" determina o sucesso, ou não, da "aceitação" da palavra: se o estado for final (um daqueles que está representado com linha dupla) a palavra é "aceite", e não o é, caso contrário.

Para definir rigorosamente este processo e assim obtermos uma definição formal da linguagem associada a um DFA, comecemos por definir indutivamente, para cada autómato, uma nova função $\widehat{\delta}$ como extensão de δ:

$$\widehat{\delta} : S \times \Sigma^\star \longrightarrow S$$

em que

$$\begin{aligned} (\forall s)\ (s \in S \ &\Rightarrow\ \widehat{\delta}(s, \varepsilon) = s) \\ (\forall s\ \forall \sigma\ \forall \rho)\ ((s \in S \wedge \sigma \in \Sigma \wedge \rho \in \Sigma^\star) \ &\Rightarrow\ \widehat{\delta}(s, \sigma\rho) = \widehat{\delta}(\delta(s, \sigma), \rho)). \end{aligned} \tag{4.1}$$

Definição 4.3 (Linguagem representada por um DFA) *Seja* $A = \langle S, \Sigma, \delta, s_0, F \rangle$ *um DFA. A linguagem* $\mathcal{L}(A)$ *representada por* A *define-se como*

$$\mathcal{L}(A) = \{\rho \in \Sigma^\star \mid \widehat{\delta}(s_0, \rho) \in F\}, \tag{4.2}$$

em que $\widehat{\delta}$ *é definida como atrás.*

Exemplo 4.4 *A palavra* 100 *pertence à linguagem definida pelo autómato* A *(Exemplo 4.2) porque*

$$\begin{aligned}\widehat{\delta}(s_0, 100) &= \widehat{\delta}(\delta(s_0, 1), 00) = \widehat{\delta}(s_0, 00)\\ &= \widehat{\delta}(\delta(s_0, 0), 0) = \widehat{\delta}(s_1, 0)\\ &= \widehat{\delta}(\delta(s_1, 0), \epsilon) = \widehat{\delta}(s_1, \epsilon)\\ &= s_1 \in F.\end{aligned}$$

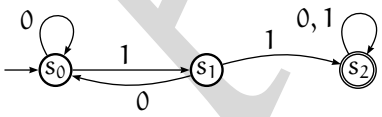
A palavra 01 *não pertence à mesma linguagem porque*

$$\begin{aligned}\widehat{\delta}(s_0, 01) &= \widehat{\delta}(\delta(s_0, 0), 1) = \widehat{\delta}(s_1, 1)\\ &= \widehat{\delta}(\delta(s_1, 1), \epsilon) = \widehat{\delta}(s_0, \epsilon)\\ &= s_0 \notin F.\end{aligned}$$

Problema 27 Seja B a linguagem das palavras de alfabeto $\{0, 1\}$ que representam em binário números múltiplos de 5.

1. Descreve um autómato finito determinístico que reconheça esta linguagem.
2. Para o autómato encontrado, prova que o mesmo reconhece a linguagem B.

Problema 28 Considera o autómato finito representado na figura.



1. Constrói uma descrição formal para este autómato como um tuplo $\mathcal{A} = (Q, \Sigma, \delta, s_0, F)$;
2. Indica quais das seguintes palavras são aceites por este autómato: 101001, 111111, 11001010111 e 0000011000;
3. Diz (em português) qual a propriedade que uma palavra de $\{0, 1\}^\star$ têm de ter para ser aceite por este autómato.

Problema 29 Considera os seguintes autómatos finitos do alfabeto $\Sigma = \{0, 1\}$,

$$\begin{aligned}\mathcal{A} &= (\{s_0, s_1\}, \Sigma, \delta_A, s_0, \{s_1\})\\ \mathcal{B} &= (\{s_0, s_1, s_2\}, \Sigma, \delta_B, s_0, \{s_1, s_2\})\end{aligned}$$

com as funções de transição dadas por:

$$\delta_A(s_0,0) = s_0 \quad \delta_A(s_0,1) = s_1$$
$$\delta_A(s_1,0) = s_0 \quad \delta_A(s_1,1) = s_1$$

$$\delta_B(s_0,0) = s_1 \quad \delta_B(s_0,1) = s_2$$
$$\delta_B(s_1,1) = s_2 \quad \delta_B(s_2,0) = s_1$$

1. Representa cada um dos autómatos por um digrafo.
2. Diz quais das seguintes palavras são aceites por algum dos autómatos:

$$\begin{array}{llll} \epsilon & 101 & 111 \\ 11001 & 01010 & 00011 \end{array}$$

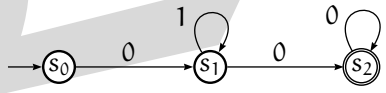
3. Diz quais as linguagens reconhecidas pelos autómatos.

Definição 4.5 (Equivalência de autómatos) *Dois autómatos dizem-se equivalentes se representarem a mesma linguagem.*

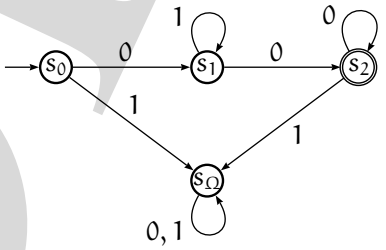
Por vezes, por comodidade, relaxa-se um pouco a definição de DFA não exigindo que a função $\delta : S \times \Sigma \to S$ seja total. Ou seja, permitindo que para algum estado não esteja definida a transição para algum símbolo. Um autómato finito determinístico deste tipo $A = \langle S, \Sigma, \delta, s_0, F\rangle$ em que a função de transição não é total, normalmente designado como *não completo*, considera-se como abreviatura do DFA *completo* $A' = \langle S \cup \{s_\Omega\}, \Sigma, \delta', s_0, F\rangle$ em que $s_\Omega \notin S$ e definindo

$$\begin{array}{rcl} \delta' : (S \cup \{s_\Omega\}) \times \Sigma & \longrightarrow & S \cup \{s_\Omega\} \\ (s,\sigma) & \longmapsto & \begin{cases} \delta(s,\sigma) & \text{quando } \delta(s,\sigma) \text{ está definido} \\ s_\Omega & \text{nos outros casos.} \end{cases} \end{array} \tag{4.3}$$

Exemplo 4.6 *O seguinte autómato* não completo



*é a abreviatura deste outro autómato (*completo*)*



Problema 30 Descreve um autómato finito que reconheça a linguagem das palavras de $\{0,1\}^\star$ que...

1. não têm nenhum "1";
2. são diferentes de "1";
3. contêm pelo menos algum "0" e algum "1";
4. têm comprimento não inferior a 2;
5. não contêm "101" como subpalavra-palavra;
6. terminam em "1";
7. terminam em "1" mas não em "111";
8. têm pelo menos dois "0" consecutivos;
9. terminam em "1" e têm pelo menos dois "0" consecutivos;
10. têm um número ímpar de "0" ou um número par de "1";
11. têm no máximo um par de "0" e um par de "1" consecutivos;
12. são representação binária de inteiros positivos múltiplos de 4;
13. são representação binária de inteiros positivos múltiplos de 2 mas não de 3;
14. contêm (algures) pelo menos três "0" seguidos, mas não contêm dois ou mais "1" seguidos'
15. se têm algum par de "0" adjacentes, este aparece antes de qualquer par de "1" adjacentes;
16. não terminam em "1101" nem em "1011";
17. têm igual número de "0" e "1" e nenhum seu prefixo tem um número de "0" que excede em dois o número de "1", nem um número de "1" que excede em dois o número de "0"

Problema 31 Seja $\mathcal{A} = (\Sigma, S, \delta, s_0, F)$ um autómato finito determinístico e s um estado de $\mathcal{A}$, tal que $\delta(s, a) = s$, $\forall a \in \Sigma$. Mostra por indução no comprimento de ρ, que $\forall \rho \in \Sigma^\star, \widehat{\delta}(s, \rho) = s$.

Problema 32 Dada uma linguagem L, seja $L^R = \{\rho^R \mid \rho \in L\}$. Mostra que se L for aceite por um autómato finito , então L^R também o é.

Problema 33 Descreve um autómato finito determinístico que reconheça a linguagem A das palavras de alfabeto $\{0,1\}$ em que não ocorrem sequências pares de 0's imediatamente à esquerda de sequências ímpares de 1's.

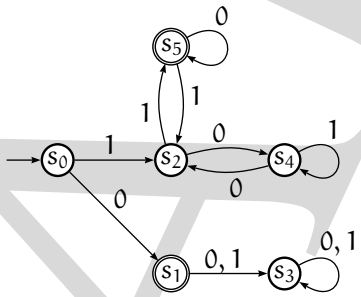
Problema 34 Seja B a linguagem das palavras de alfabeto $\{0,1\}$ que representam em binário números múltiplos de 5.

1. Descreve um autómato finito determinístico que reconheça esta linguagem.
2. Para o autómato encontrado, prova que o mesmo reconhece a linguagem B.

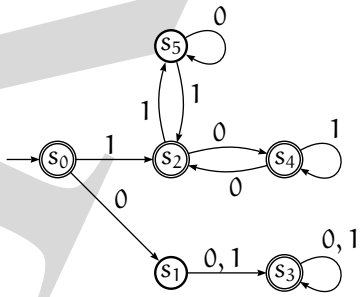
4.1.1 O autómato complementar

Dada uma linguagem representada por uma expressão regular, é relativamente difícil encontrar a expressão regular que representa a sua linguagem complementar. Esta operação é particularmente simples se se tratar de autómatos determinísticos finitos.

Exemplo 4.7 *Consideremos a linguagem,* L*, das palavras de alfabeto* $\Sigma = \{0,1\}$ *que correspondem a representações binárias de inteiros múltiplos de* 3*. Um DFA que represente* L *pode ser o representado pelo seguinte diagrama:*



A linguagem complementar desta, $\overline{\text{L}}$*, a linguagem das palavras que correspondem a inteiros que* ***não são múltiplos de*** 3*, será a linguagem gerada pelo autómato anterior, mas em que os estados finais e estados não finais foram trocados. Ou seja:*



Claro que este reconhece ε *com pertencendo a* $\overline{\text{L}}$ *apesar de este não representar um número em binário. Mas isso é consequência da forma como definimos a linguagem.*

Teorema 4.8 (complementar duma linguagem dada por um DFA) *O complementar de uma linguagem dada por um DFA é também uma linguagem representável por um DFA. Para além disso se* $L = \mathcal{L}(A)$ *para um DFA completo* $A = \langle S, \Sigma, \delta, s_0, F\rangle$*, então a linguagem complementar* $\overline{L} = \mathcal{L}(\overline{A})$ *com*

$$\overline{A} = \langle S, \Sigma, \delta, s_0, S \setminus F\rangle.$$

Dem. *A demonstração é trivial pois, pela Definição 4.3,*

$$\begin{aligned} \rho \in \overline{L} \Leftrightarrow \rho \notin L &\Leftrightarrow \widehat{\delta}(s_0, \rho) \notin F \\ &\Leftrightarrow \widehat{\delta}(s_0, \rho) \in S \setminus F \\ &\Leftrightarrow \rho \in \mathcal{L}(\overline{A}). \end{aligned}$$

■

4.1.2 O autómato produto

Sejam $\mathcal{A}_1 = \langle S_1, \Sigma, \delta_1, i_1, F_1\rangle$ e $\mathcal{A}_2 = \langle S_2, \Sigma, \delta_2, i_2, F_2\rangle$ dois autómatos finitos determinísticos. Vamos construir um **autómato produto** $\mathcal{A}_3$ que vai simular o funcionamento simultâneo de $\mathcal{A}_1$ e $\mathcal{A}_2$. Esta construção vai permitir encontrar um DFA que represente $\mathcal{L}(A_1) \cup \mathcal{L}(A_2)$, o mesmo se passando para $\mathcal{L}(A_1) \cap \mathcal{L}(A_2)$. Formalmente, $\mathcal{A}_3 = \langle S_3, \Sigma, \delta_3, i_3, F_3\rangle$ onde

$$\begin{aligned} S_3 &= S_1 \times S_2 = \{(s_1, s_2) \mid s_1 \in S_1 \text{ e } s_2 \in S_2\} \\ i_3 &= (i_1, i_2) \\ \delta_3((s_1, s_2), a) &= (\delta_1(s_1, a), \delta_2(s_2, a)) \qquad \forall\, s_1 \in S_1, s_2 \in S_2, a \in \Sigma \end{aligned}$$

A definição de F_3 indicará se a linguagem de $\mathcal{A}_3$ será a intersecção ou a reunião das linguagens de $\mathcal{A}_1$ e de $\mathcal{A}_2$. Se definirmos

$$F_3 = F_1 \times F_2 = \{(s_1, s_2) \mid s_1 \in F_1 \ \wedge\ s_2 \in F_2\}$$

O autómato $\mathcal{A}_3$ vai aceitar uma palavra se e só se $\mathcal{A}_1$ e $\mathcal{A}_2$ também a aceitarem, i.e., $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = L(\mathcal{A}_3)$.

Problema 35 Mostra por indução em $|\rho|$, que $\forall\, \rho \in \Sigma^\star,\ \ \widehat{\delta_3}((s_1, s_2), \rho) = (\widehat{\delta_1}(s_1, \rho), \widehat{\delta_2}(s_2, \rho))$

Proposição 4.9 $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

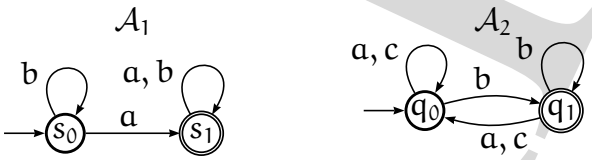
Dem.

$$\begin{aligned}
\rho \in L(\mathcal{A}_3) &\iff \widehat{\delta_3}(i_3, \rho) \in F_3 \\
&\iff \widehat{\delta_3}((i_1, i_2), \rho) \in F_1 \times F_2 \\
&\iff (\widehat{\delta_1}(i_1, \rho), \widehat{\delta_2}(i_2, \rho)) \in F_1 \times F_2 \\
&\iff \widehat{\delta_1}(i_1, \rho) \in F_1 \ \wedge \ \widehat{\delta_2}(i_2, \rho) \in F_2 \\
&\iff \rho \in L(\mathcal{A}_1) \ \wedge \ \rho \in L(\mathcal{A}_2) \\
&\iff \rho \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)
\end{aligned}$$

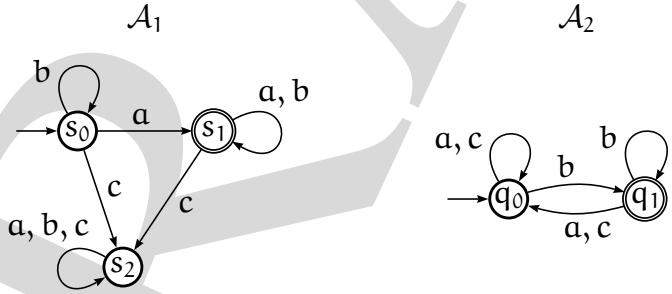
■

Como no caso da construção por subconjuntos para a obtenção de um autómato finito determinístico equivalente a um não determinístico, a construção da função de transição pode só envolver os estados do autómato produto que são atingíveis do estado inicial.

Exemplo 4.10 *Considera os seguintes autómatos finitos determinísticos:*



Como os autómatos não têm o mesmo alfabeto, antes de calcular o autómato produto é necessário transformar o primeiro autómato sem mudar a linguagem representada por ele.



Sejam então $\mathcal{A}_1 = \langle S_1, \{a, b, c\}, \delta_1, s_0, , F_1\rangle$ *e* $\mathcal{A}_1 = \langle S_2, \{a, b, c\}, \delta_2, q_0, , F_2\rangle$ *onde* $S_1 = \{s_0, s_1, s_2\}$, $S_2 = \{q_0, q_1\}$, $F_1 = \{s_1\}$, $F_2 = \{q_1\}$, *e as funções de transição* δ_1 *e* δ_2 *são respectivamente,*

δ_1	a	b	c
$\rightarrow s_0$	s_1	s_0	s_2
$\star s_1$	s_1	s_1	s_2
s_2	s_2	s_2	s_2

δ_2	a	b	c
$\rightarrow q_0$	q_0	q_1	q_0
$\star q_1$	q_0	q_1	q_0

Para o autómato produto $\mathcal{A}_3 = \langle S_3, \{a, b, c\}, \delta_3, (s_0, q_0), F_3 \rangle$*, tem-se*

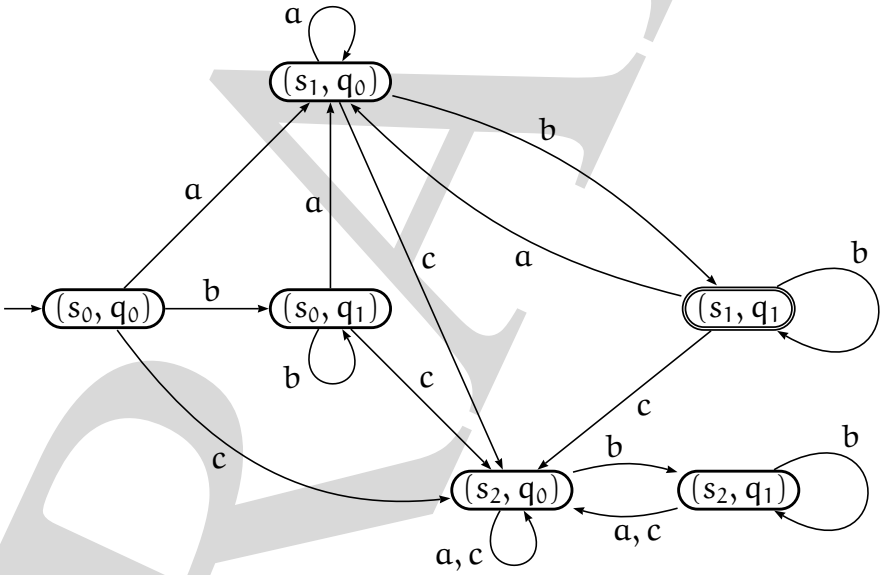
$$S_3 = \{(s_0, q_0), (s_1, q_0), (s_2, q_0), (s_0, q_1), (s_1, q_1), (s_2, q_1)\}$$

e a função delta δ_3 *é dada por*

δ_3	a	b	c
$\rightarrow (s_0, q_0)$	(s_1, q_0)	(s_0, q_1)	(s_2, q_0)
(s_1, q_0)	(s_1, q_0)	(s_1, q_1)	(s_2, q_0)
(s_2, q_0)	(s_2, q_0)	(s_2, q_1)	(s_2, q_0) .
(s_0, q_1)	(s_1, q_0)	(s_0, q_1)	(s_2, q_0)
(s_1, q_1)	(s_1, q_0)	(s_1, q_1)	(s_2, q_0)
(s_2, q_1)	(s_2, q_0)	(s_2, q_1)	(s_2, q_0)

Se o autómato produto representar a intersecção das linguagens de $\mathcal{A}_1$ *e* $\mathcal{A}_2$*, temos que* $F_3 = F_1 \times F_2 = \{(s_1, q_1)\}$*. O diagrama do autómato produto* $\mathcal{A}_3$ *é:*

$\mathcal{A}_3$



O autómato $\mathcal{A}_3$ *aceita a linguagem constituída pelas palavras de alfabeto* $\{a, b, c\}$ *que têm algum* a*, terminam em* b *e não têm* c*'s.*

Se num autómato produto se modificar o valor de F_3 podemos obter um autómato que reconhece a reunião das linguagens de $\mathcal{A}_1$ e $\mathcal{A}_2$. O autómato $\mathcal{A}_3$ aceita uma palavra se pelo menos um dos autómatos $\mathcal{A}_1$ ou $\mathcal{A}_2$ aceitar:

$$F_3 = \{(s_1, s_2) \mid s_1 \in F_1 \text{ ou } s_2 \in F_2\}.$$

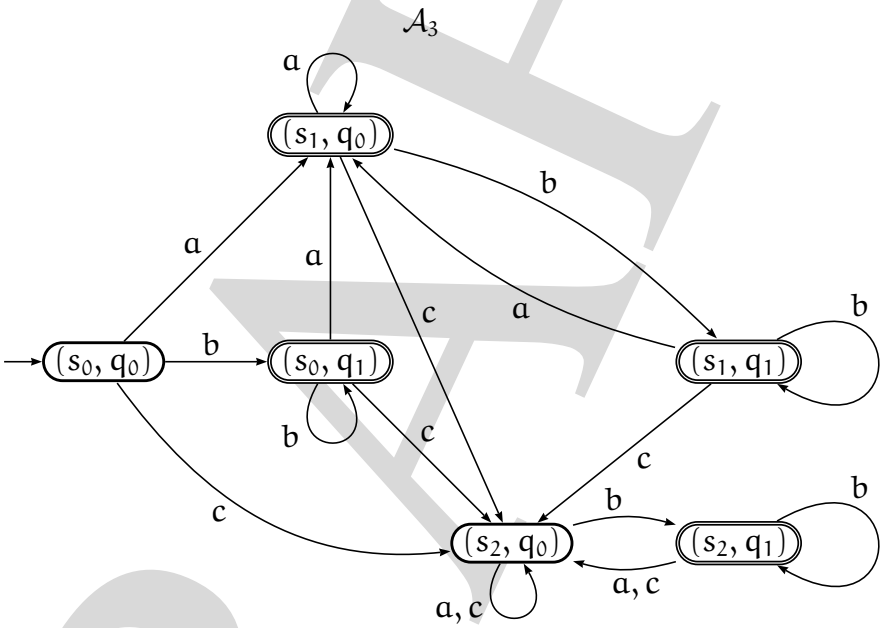
Proposição 4.11 $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

Problema 36 Demonstra a Proposição 4.11.

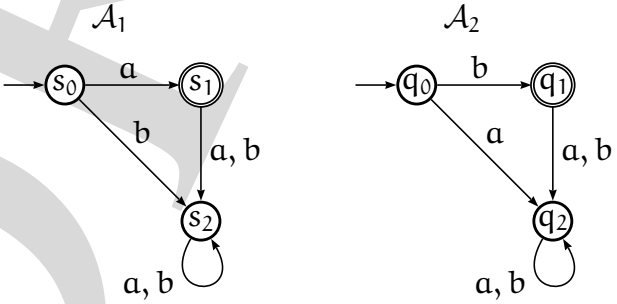
Exemplo 4.12 *Para o exemplo do Exercício 4.10 para se obter um autómato para a reunião da linguagens, $\mathcal{A}_3$, basta considerar o conjunto de estados finais constituído pelos pares em que pelo menos um dos estados seja final,*

$$F_3 = \{(s_1, q_0), (s_0, q_1), (s_1, q_1), (s_2, q_1)\}.$$

O diagrama do autómato da reunião $\mathcal{A}_3$ fica:



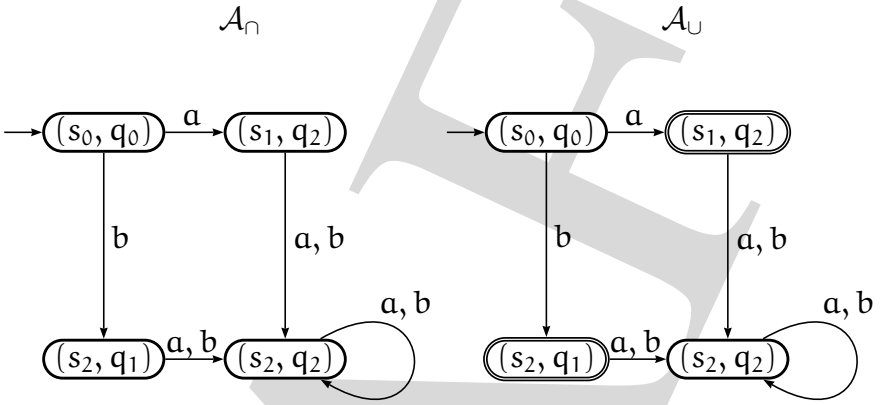
Exemplo 4.13 *Considera os seguintes autómatos finitos determinísticos:*



Como os autómatos não são completos, a função de transição para o autómato produto é:

δ_3	a	b
$\rightarrow (s_0, q_0)$	(s_1, q_2)	(s_2, q_1)
(s_1, q_2)	(s_2, q_2)	(s_2, q_2)
(s_2, q_1)	(s_2, q_2)	(s_2, q_2)
(s_2, q_2)	(s_2, q_2)	(s_2, q_2)

O autómato para a intersecção $\mathcal{A}_\cap$ reconhece a linguagem $\{a\} \cap \{b\}$ (ou seja, $\emptyset$) e o da reunião $\mathcal{A}_\cup$, reconhece a linguagem $\{a\} \cup \{b\}$. Os diagramas para o autómato da intersecção e da reunião, são respectivamente:



Problema 37 Exemplifica todas as construções anteriores considerando as linguagens $A = \{aa\}$ e $B = \{bbb\}$ de alfabeto $\{a, b\}$ e os respectivos autómatos.

4.2 Autómatos finitos não determinísticos (NFA)

Muitas vezes, para facilidade formal de demonstração ou por uma expressividade mais evidente, optamos por outro tipo de autómatos finitos, os autómatos não determinísticos, para representar linguagens regulares. Estes, como vamos ver, têm uma composicionalidade evidente, tornando as operações elementares das expressões regulares (concatenação, disjunção e fecho de Kleene) em transformações simples nesta classe de autómatos.

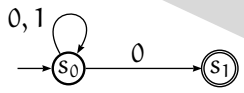
Em vez de termos um autómato, em que não temos "que tomar decisões" sobre qual o caminho que vamos tomar ao ler uma palavra, passamos a ter um autómato em que de um estado, e lendo um dado símbolo, podemos ir para diversos estados. Mesmo o estado em que começamos a ler uma palavra pode não ser único. Um NFA pode ter um conjunto não singular de estados iniciais e o estado que se usa para começar a ler uma palavra é uma das escolhas que temos que fazer.

Definição 4.14 (NFA) *Um autómato finito não determinístico (NFA) é um quíntuplo ordenado $\langle S, \Sigma, \delta, I, F \rangle$ em que:*

- S *é um conjunto finito, não vazio, a que chamamos o conjunto dos estados;*
- Σ *é um conjunto finito, não vazio, que constitui o alfabeto;*
- δ *é uma função parcial* $\delta : S \times \Sigma \to \mathcal{P}(S)$*, chamada função de transição;*
- I*, com* $I \subseteq S$ *e* $I \neq \emptyset$*, é o conjunto de estados iniciais;*
- F*, com* $F \subseteq S$*, é o conjunto dos estados finais.*

Exemplo 4.15 *Considera o NFA* $A = \langle \{s_0, s_1\}, \{0, 1\}, \delta, \{s_0\}, \{s_1\} \rangle$ *com*

$$\delta(s_0, 0) = \{s_0, s_1\} \quad \delta(s_0, 1) = \{s_0\}$$



Como o estado do autómato, resultante da leitura de uma palavra, passa a ser então dependente da (ou das) escolha(s) que tenhamos feito ao longo desse processo, dizemos que uma palavra é "aceite" por um NFA, se existir um conjunto de escolhas que nos conduza a um estado final. Para captar esta possibilidade de escolha dos diversos percursos que podemos percorrer com a leitura de uma mesma palavra, a função de transição de um NFA, não faz corresponder a cada par (estado, letra) um só novo estado mas sim um conjunto de estados. A função de transição estendida vai fazer corresponder para cada palavra, não um conjunto de estados para cada estado de partida, mas um conjunto de estados para cada conjunto de estados que representa as diversas situações resultantes das diferentes possíveis escolhas até aí feitas.

Então, da mesma forma que foi feito para os autómatos determinísticos (DFA) podemos definir uma função de transição estendida

$$\widehat{\delta} : \mathcal{P}(S) \times \Sigma \longrightarrow \mathcal{P}(S)$$

com a seguinte definição recursiva:

$$\begin{gathered} (\forall X)\ (X \in \mathcal{P}(S)\ \Rightarrow\ \widehat{\delta}(X, \epsilon) = X) \\ (\forall X\ \forall \sigma\ \forall \rho)\ ((X \in \mathcal{P}(S) \wedge \sigma \in \Sigma \wedge \rho \in \Sigma^\star)\ \Rightarrow \widehat{\delta}(X, \sigma\rho) = \widehat{\delta}(\bigcup_{s \in X} \delta(s, \sigma), \rho)). \end{gathered} \tag{4.4}$$

Podemos então definir a linguagem representada por um NFA.

Definição 4.16 (Linguagem representada por um NFA) *Seja* $A = \langle S, \Sigma, \delta, I, F\rangle$ *um NFA. A linguagem* $\mathcal{L}(A)$ *representada por* A *define-se como*

$$\mathcal{L}(A) = \{\rho \in \Sigma^\star \mid \widehat{\delta}(I, \rho) \cap F \neq \emptyset\}, \tag{4.5}$$

em que $\widehat{\delta}$ *é definida como atrás.*

Uma outra formulação pode ser dada a esta definição 4.16 que, por vezes, se torna bastante cómoda para provar certas proposições. Em vez de utilizarmos os possíveis conjuntos de estados correspondentes à avaliação sucessiva de $\widehat{\delta}(I, \rho)$, podemos referir directamente o conjunto de estados que conduzem, com sucesso, a avaliação de uma palavra ρ de um estado inicial a um estado final.

Lema 4.17 *Seja* $A = \langle S, \Sigma, \delta, I, F\rangle$ *um NFA e* $\rho = \rho_1\rho_2\ldots\rho_{|\rho|} \in \Sigma^\star$*, é condição necessária e suficiente para que* $\rho \in \mathcal{L}(A)$*, que*

$$(\exists s_0, s_1, \ldots, s_{|\rho|} \in S)\ (s_0 \in I \wedge s_{|\rho|} \in F \wedge (\forall i \in [0, |\rho|[\ s_i \in \delta(s_{i-1}, \rho_i))). \tag{4.6}$$

Dem. *A demonstração sai directa, da Definição 4.16 e do enunciado do lema.* ■

Problema 38 Demonstra o Lema 4.17.

Lema 4.18 *Seja* $A = \langle S, \Sigma, \delta, I, F\rangle$ *uma NFA e* $\rho \in \Sigma^\star$ *com* $\rho = \rho'\sigma$ *para algum* $\rho' \in \Sigma^\star$ *e* $\sigma \in \Sigma$*. Então, para qualquer conjunto de estados* $X \subseteq S$*,*

$$\widehat{\delta}(X, \rho) = \widehat{\delta}(\widehat{\delta}(X, \rho'), \sigma).$$

Dem. *A demonstração segue por indução sobre o comprimento de* $|\rho|$*. Se* $|\rho| = 0$ *a proposição não faz sentido, pelo que comecemos por provar para* ρ *de comprimento unitário, ou seja para* $\rho \in \Sigma$*. Nesse caso temos trivialmente que, para qualquer conjunto de estados* X*,*

$$\widehat{\delta}(X, \varepsilon\rho) = \widehat{\delta}(\widehat{\delta}(X, \varepsilon), \rho).$$

Suponhamos, então, que para qualquer palavra ρ*, com* $|\rho| < n$*, se se tiver* $\rho = \rho'\sigma$ *então, para qualquer conjunto de estados* X*,* $\widehat{\delta}(X, \rho'\sigma) = \widehat{\delta}(\widehat{\delta}(X, \rho'), \sigma)$*. Seja então* ρ *uma palavra tal que* $|\rho| = n$ *e* $\rho = \rho'\sigma$*. Então seja* $\sigma' \in \Sigma$ *e* $\rho'' \in \Sigma^\star$ *tal que* $\rho' = \sigma'\rho''$*. Então* $\rho = \sigma'\rho''\sigma$ *com* $|\sigma'\rho''| < n$*. Então*

$$\begin{aligned}
\widehat{\delta}(X, \rho) &= \widehat{\delta}(X, \sigma'\rho''\sigma) \\
&= \widehat{\delta}(\widehat{\delta}(X, \sigma'), \rho''\sigma) && \textit{(pela definição de } \widehat{\delta}\ (4.4)\textit{)} \\
&= \widehat{\delta}(\widehat{\delta}(\widehat{\delta}(X, \sigma'), \rho''), \sigma) && \textit{(pela hipótese de indução)} \\
&= \widehat{\delta}(\widehat{\delta}(X, \sigma'\rho''), \sigma) && \textit{(pela definição de } \widehat{\delta}\ (4.4)\textit{)} \\
&= \widehat{\delta}(\widehat{\delta}(X, \rho'), \sigma).
\end{aligned}$$

Portanto o Lema é válido para qualquer palavra ρ de qualquer comprimento. ■

Definição 4.19 (linguagem reversa) *Seja L uma linguagem de alfabeto Σ, e $R : \Sigma^\star \to \Sigma^\star$ a função dada pela seguinte definição indutiva (ver 2.9):*

$$(\forall \sigma)\ (\sigma \in \Sigma \cup \{\varepsilon\} \ \Rightarrow\ \sigma^R = \sigma)$$
$$(\forall \sigma\, \forall \rho)\ ((\sigma \in \Sigma \wedge \rho \in \Sigma^\star) \ \Rightarrow\ (\sigma\rho)^R = \rho^R\sigma).$$

A linguagem L^R é a linguagem de alfabeto Σ, definida como:

$$L^R = \{\rho \in \Sigma^\star \mid \rho^R \in L\}.$$

Definição 4.20 (autómato reverso) *Seja $A = \langle S, \Sigma, \delta, I, F\rangle$ um NFA, definimos como o seu autómato reverso, o NFA $A^R = \langle S, \Sigma, \delta^R, F, I\rangle$ com*

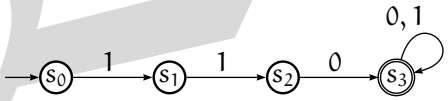
$$\begin{array}{rcl} \delta^R : S \times \Sigma & \longrightarrow & \mathcal{P}(S) \\ (s, \sigma) & \longmapsto & \{x \in S \mid \delta(x, \sigma) \ni s\} \end{array}$$

Proposição 4.21 *Seja A um NFA, então*

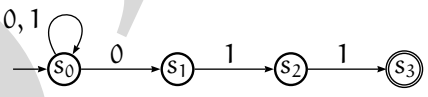
$$\mathcal{L}(A)^R = \mathcal{L}(A^R).$$

Dem. *A demonstração é trivial, usando o Lema 4.17.* ■

Exemplo 4.22 *Consideremos o NFA (neste caso até um DFA) dado pelo diagrama seguinte, que representa a linguagem das palavras de alfabeto $\Sigma = \{0, 1\}$ que têm 110 como prefixo:*



A linguagem das palavras, com o mesmo alfabeto, que têm 011 como sufixo, tem o seguinte NFA como representante.



Problema 39 Demonstra a Proposição 4.21.

Teorema 4.23 (Método de construção de subconjuntos) *Seja A um autómato finito não determinístico, então existe um autómato finito determinístico A' tal que*

$$\mathcal{L}(A) = \mathcal{L}(A').$$

Dem. *Seja $A = \langle S, \Sigma, \delta, I, F\rangle$ o NFA. Consideremos o DFA $A' = \langle \mathcal{P}(S), \Sigma, \delta', I, F'\rangle$ em que*

$$F' = \{X \subseteq S \mid X \cap F \neq \emptyset\}$$

e

$$\begin{array}{rcl} \delta' : \mathcal{P}(S) \times \Sigma & \longrightarrow & \mathcal{P}(S) \\ (X, \sigma) & \longmapsto & \bigcup_{s \in X} \delta(s, \sigma). \end{array}$$

É trivial verificar que este novo DFA está bem definido. Necessitamos provar somente a equivalência entre os dois autómatos.

Comecemos por definir $\widehat{\delta'} : \mathcal{P}(S) \times \Sigma^\star \to \mathcal{P}(S)$, indutivamente, como

$$\begin{array}{rcl} \widehat{\delta'}(X, \varepsilon) & = & X \\ \widehat{\delta'}(X, \sigma\rho) & = & \widehat{\delta'}(\delta'(X, \sigma), \rho), \end{array}$$

e provemos, por indução sobre o comprimento de ρ ($\rho \in \Sigma^\star$), que

$$\widehat{\delta'}(I, \rho) = \widehat{\delta}(I, \rho).$$

Se $|\rho| = 0$, e portanto $\rho = \varepsilon$, tem-se, pela definição, que

$$\widehat{\delta}(I, \varepsilon) = I = \widehat{\delta'}(I, \varepsilon).$$

Suponhamos então que

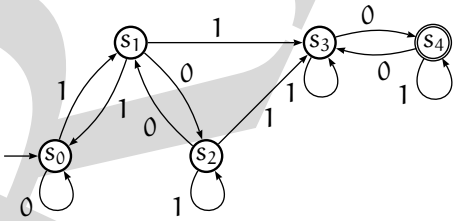
$$(\forall \rho\ (|\rho| < n \ \Rightarrow\ \widehat{\delta}(I, \rho) = \widehat{\delta'}(I, \rho))).$$

Seja $\rho \in \Sigma^\star$ *com* $|\rho| = n$. *Então* $\rho = \rho'\sigma$ *para algum* $\rho' \in \Sigma^\star$ *com* $|\rho'| = n-1$.

$$
\begin{aligned}
\widehat{\delta}(I,\rho) &= \widehat{\delta}(I,\rho'\sigma) \\
&= \widehat{\delta}(\widehat{\delta}(I,\rho'),\sigma) && \textit{(pelo Lema 4.18)} \\
&= \widehat{\delta}(\widehat{\delta'}(I,\rho'),\sigma) && \textit{(pela hipótese de indução)} \\
&= \widehat{\delta}\left(\bigcup_{s\in\widehat{\delta'}(I,\rho')} \delta(s,\sigma),\varepsilon\right) && \textit{(pela definição de } \widehat{\delta}\textit{)} \\
&= \bigcup_{s\in\widehat{\delta'}(I,\rho')} \delta(s,\sigma) && \textit{(pela definição de } \widehat{\delta}\textit{)} \\
&= \delta'(\widehat{\delta'}(I,\rho'),\sigma) \\
&= \widehat{\delta'}(I,\rho'\sigma) \\
&= \widehat{\delta'}(I,\rho)
\end{aligned}
$$

Logo $\mathcal{L}(A) = \mathcal{L}(A')$. ■

Exemplo 4.24 *Consideremos o NFA representado pelo seguinte diagrama:*

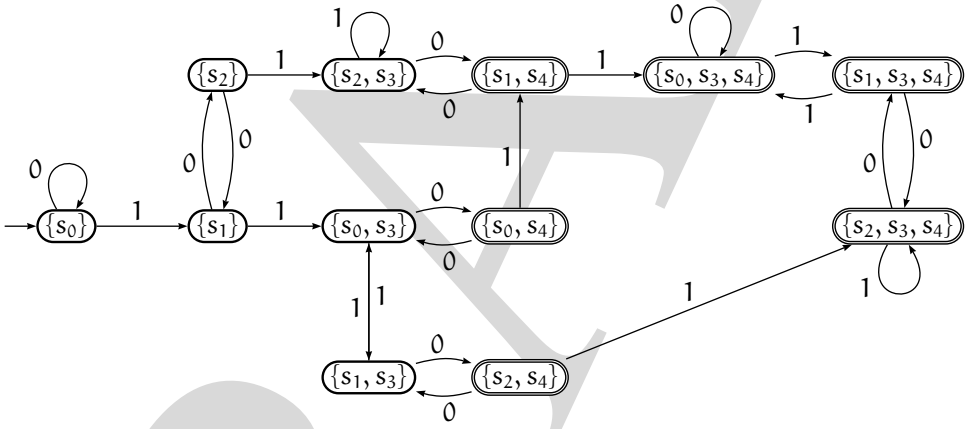


Este autómato representa a linguagem das representações binárias que correspondem a inteiros que não são múltiplos de 3, *concatenada com a linguagem das palavras que começam com um* 1 *e têm um número ímpar de* 0*'s.*

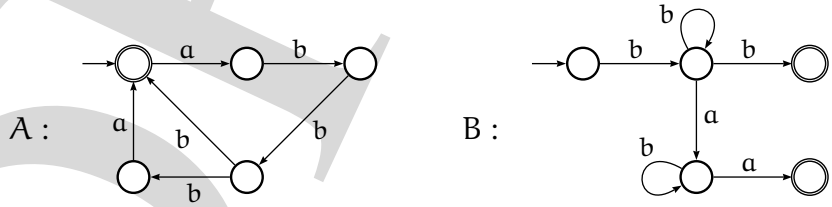
Para obtermos um DFA equivalente a este NFA, prossigamos com a construção dos subconjuntos que resulta do Teorema anterior. Tem-se

δ	0	1
$\{s_0\}$	$\{s_0\}$	$\{s_1\}$
$\{s_1\}$	$\{s_2\}$	$\{s_0, s_3\}$
$\{s_2\}$	$\{s_1\}$	$\{s_2, s_3\}$
$\{s_0, s_3\}$	$\{s_0, s_4\}$	$\{s_1, s_3\}$
$\{s_2, s_3\}$	$\{s_1, s_4\}$	$\{s_2, s_3\}$
$\{s_1, s_3\}$	$\{s_2, s_4\}$	$\{s_0, s_3\}$
$\{s_0, s_4\}$	$\{s_0, s_3\}$	$\{s_1, s_4\}$
$\{s_1, s_4\}$	$\{s_2, s_3\}$	$\{s_0, s_3, s_4\}$
$\{s_2, s_4\}$	$\{s_1, s_3\}$	$\{s_2, s_3, s_4\}$
$\{s_0, s_3, s_4\}$	$\{s_0, s_3, s_4\}$	$\{s_1, s_3, s_4\}$
$\{s_1, s_3, s_4\}$	$\{s_2, s_3, s_4\}$	$\{s_0, s_3, s_4\}$
$\{s_2, s_3, s_4\}$	$\{s_1, s_3, s_4\}$	$\{s_2, s_3, s_4\}$

O estado inicial é o estado $\{s_0\}$, e são finais todos os estados a cujo nome pertence s_4. O diagrama do DFA resultante é, então, o seguinte.



Problema 40 Considera os autómatos finitos *não-determinísticos* representados pelos seguintes diagramas:



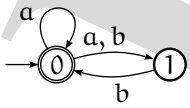
Diz quais das seguintes palavras são aceites por $\mathcal{A}$ ou $\mathcal{B}$:

1. ϵ

2. aa
3. aba
4. abba
5. bba
6. abab

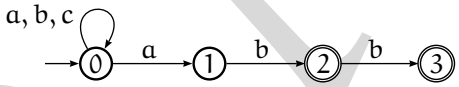
Problema 41 Constrói um autómato finito *não-determinístico* que reconheça a linguagem do alfabeto $\Sigma = \{0, 1\}$ das palavras com um 1 na terceira posição a contar do fim.

Problema 42 Considera o seguinte autómato finito não-determinístico representado pelo seguinte diagrama:

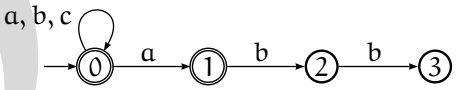


Converte, pela construção dos subconjuntos, o autómato num autómato finito *determinístico*.

Problema 43 Seja $\mathcal{A}$ o autómato finito de alfabeto $\{a, b, c\}$ representado pelo diagrama seguinte.



1. Qual é a linguagem reconhecida pelo autómato $\mathcal{A}$? Porquê?
2. Usando o método da construção de subconjuntos, determina um autómato determinístico que seja equivalente a $\mathcal{A}$.
3. Recorda que se um dado autómato determinístico $\langle S, \Sigma, \delta, s_0, F\rangle$ em que δ é uma função total (não encrava), reconhece L, então o autómato $\langle S, \Sigma, \delta, s_0, S \setminus F\rangle$ reconhece $\Sigma^\star \setminus L$ (isto é, a linguagem complementar de L).
   Por que é que a linguagem reconhecida pelo autómato seguinte não é a complementar da linguagem reconhecida por $\mathcal{A}$?

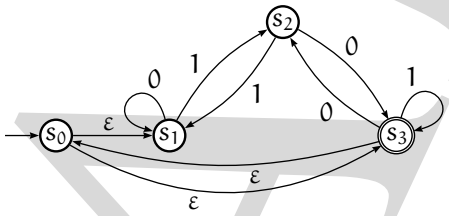


4.3 Autómatos não determinísticos com transições por ε (NFAε)

Por vezes, seja pela sua expressividade intrínseca seja por facilitar construções formais, usamos NFAs que admitem transições por ε. À classe destes autómatos designamos normalmente por NFAε.

Num NFAε, de um estado seu não só podem partir transições por um mesmo símbolo para diversos estados (como num NFA) como podem existir transições para outros estados, que em vez de "consumirem" um caracter de Σ, consomem ε. Quando do estado s_1 temos uma transição por ε para o estado s_2, isso significa que quando "chegamos" a s_1 ("vindos" de outro estado) podemos optar por "ir" (ou não) para o estado s_2, sem que para isso tenhamos que "consumir" algum caracter.

Exemplo 4.25 *O NFAε seguinte representa o Fecho de Kleene da linguagem das representações binárias dos inteiros congruentes com 2, módulo 3.*



Definição 4.26 (NFAε) *Um autómato finito não-determinístico com transições por ε (NFAε) é um quíntuplo ordenado $\langle S, \Sigma, \delta, I, F\rangle$ em que*

- *S é um conjunto finito, não vazio, a que chamamos o conjunto dos estados;*
- *Σ é um conjunto finito, não vazio, que constitui o alfabeto;*
- *δ é uma função parcial $\delta : S \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(S)$, chamada função de transição;*
- *I, com $I \subseteq S$ e $I \neq \emptyset$, é o conjunto de estados iniciais;*
- *F, com $F \subseteq S$, é o conjunto dos estados finais.*

O conceito de Fecho por ε, a seguir formalizado, corresponde ao conjunto de estados que se podem atingir a partir de um dado estado "viajando" somente por transições por ε.

Definição 4.27 (Fecho por ε de um conjunto de estados) *Seja $A = \langle S, \Sigma, \delta, I, F\rangle$ um NFAε, e seja $X \subseteq S$ um conjunto de estados de A. Definimos indutivamente o*

seguinte conjunto:

$$\begin{aligned} F_\varepsilon^0(X) &= X, \\ F_\varepsilon^1(X) &= \bigcup_{s\in X} \delta(s,\varepsilon), \\ F_\varepsilon^n(X) &= \bigcup_{s\in F_\varepsilon^{n-1}(X)} \delta(s,\varepsilon). \end{aligned}$$

Então o fecho por ε *de* X *(*$F_\varepsilon(X)$*) fica definido como:*

$$F_\varepsilon(X) = \bigcup_{n\geq 0} F_\varepsilon^n(X).$$

Como se trata de um autómato finito, isto é, com um número finito de estados, este conjunto está trivialmente bem definido. Quando $X = \{s\}$ *também se denomina* $F_\varepsilon(X)$ *por* fecho por ε de s *e pode-se designar por* $F_\varepsilon(s)$*.*

Exemplo 4.28 *Considerando o NFA*ε *do Exemplo 4.25 podemos calcular* $F_\varepsilon(\{s_0\})$*. Tem-se*

$$\begin{aligned} F_\varepsilon^0(\{s_0\}) &= \{s_0\}, \\ F_\varepsilon^1(\{s_0\}) &= \delta(s_0,\varepsilon) \\ &= \{s_1, s_3\}, \\ F_\varepsilon^2(\{s_0\}) &= \delta(s_1,\varepsilon) \cup \delta(s_3,\varepsilon) \\ &= \{s_0\} \end{aligned}$$

Donde, $F_\varepsilon(\{s_0\}) = \bigcup_{n=0}^{2} F_\varepsilon^n(\{s_0\}) = \{s_0, s_1, s_2\}$*.*

Para definir formalmente a linguagem representada por um NFAε, comecemos, como fizemos para os NFA, por estender a função de transições δ.

Então, seja $A = \langle S, \Sigma, \delta, I, F\rangle$ um NFAε, definimos $\widehat{\delta}$ como

$$\widehat{\delta} : \mathcal{P}(S) \times \Sigma^\star \longrightarrow \mathcal{P}(S)$$

com a seguinte definição recursiva:

$$(\forall X)(X \in \mathcal{P}(S) \;\Rightarrow\; \widehat{\delta}(X,\varepsilon) = F_\varepsilon(X)), \tag{4.7}$$

$$(\forall X\ \forall\rho\forall\sigma)((X \in \mathcal{P}(S) \wedge \sigma \in \Sigma \wedge \rho \in \Sigma^\star) \;\Rightarrow\; \widehat{\delta}(X,\sigma\rho) = \widehat{\delta}\left(\bigcup_{s\in X} F_\varepsilon(\delta(s,\sigma)),\rho\right)), \tag{4.8}$$

com $F_\varepsilon(X)$ como está definido em 4.27.

Definição 4.29 (linguagem representada por um NFAε) *Seja* $A = \langle S, \Sigma, \delta, I, F\rangle$ *um NFAε. A linguagem,* $\mathcal{L}(A)$*, representada por* A *define-se como*

$$\mathcal{L}(A) = \{\rho \in \Sigma^\star \mid \widehat{\delta}(F_\varepsilon(I), \rho) \cap F \neq \emptyset\}, \tag{4.9}$$

em que $\widehat{\delta}$ *e* F_ε *são definidos como atrás.*

Podemos, como fizemos para a conversão de NFAs em DFAs, aplicar a definição de linguagem representada por um NFAε para obter um algoritmo que encontre um DFA equivalente. Em vez disso, vamos usar a definição de fecho por ε de um estado para obter uma transformação de um NFAε num NFA equivalente.

Teorema 4.30 *Dado um NFAε* A*, existe um NFA,* A'*, tal que*

$$\mathcal{L}(A) = \mathcal{L}(A').$$

Dem. *Seja* $A = \langle S, \Sigma, \delta, I, F\rangle$ *o NFAε. Consideremos o NFA* $A' = \langle \mathcal{P}(S), \Sigma, \delta', F_\varepsilon(I), F'\rangle$ *em que*

$$F' = \bigcup_{X \in F} F_\varepsilon(X)$$

e

$$\begin{aligned} \delta' : \mathcal{P}(S) \times \Sigma &\longrightarrow \mathcal{P}(S) \\ (X, \sigma) &\longmapsto F_\varepsilon\left(\bigcup_{s \in X} \delta(s, \sigma)\right). \end{aligned}$$

O NFA A' *está trivialmente bem definido, e a equivalência dos autómatos segue directamente das definições das linguagens representadas por estes autómatos, respectivamente 4.29 e 4.16.* ■

Em muitos casos, e em especial para simplificar as demonstrações, é conviniente considerar NFAεs com um só estado inicial e um só estado final. O Lema seguinte mostra que tal é sempre possível.

Lema 4.31 *Seja* $A = \langle S, \Sigma, \delta, I, F\rangle$ *um NFAε, existe um NFAε,* $A' = \langle S', \Sigma, \delta', I', F'\rangle$*, equivalente a* A*, tal que:*

1. *tem somente um estado inicial* $I' = \{s'_0\}$*;*
2. *não há transições que "cheguem" do estado inicial* $(\forall s \in S' \forall \sigma \in \Sigma)(s'_0 \notin \delta'(s, \sigma) \wedge s'_0 \notin \delta'(s, \varepsilon))$*;*

3. *tem somente um estado final* $F = \{s'_f\}$;

4. *não há transições que "partam" do estado final* $(\delta'(s'_f, \varepsilon) = \emptyset) \wedge (\forall \sigma \in \Sigma\ \delta'(s'_f, \sigma) = \emptyset)$.

Um autómato que verifique estas propriedades denomina-se normalizado.

Dem. *Se o autómato* A *já estiver nas condições requeridas, nada há a demonstrar. Caso isso não aconteça, seja* $S' = S \cup \{s'_0, s'_f\}$, *com* $s'_0, s'_f \notin S$. *Façamos* $I' = \{s'_0\}$ *e* $F' = \{s'_f\}$ *e tomemos* δ' *como extensão de* δ

$$\begin{aligned} \delta' : S' \times \Sigma &\longrightarrow S' \\ s \in S,\ \sigma \in \Sigma \cup \{\varepsilon\}\ (s, \sigma) &\longmapsto \delta(s, \sigma) \\ (s'_0, \varepsilon) &\longmapsto F_\varepsilon(I) \\ s \in F_\varepsilon(F)\ (s, \varepsilon) &\longmapsto \{s'_f\}. \end{aligned} \tag{4.10}$$

■

Definição 4.32 (linguagem direita) *Seja* $A = \langle S, \Sigma, \delta, s_0, F\rangle$ *uma autómato e* s *um seu estado. A linguagem direita de* S *é*

$$\{\rho \in \Sigma^\star \mid \widehat{\delta}(s, \rho) \in F\}.$$

Definição 4.33 (linguagem esquerda) *Seja Seja* $A = \langle S, \Sigma, \delta, s_0, F\rangle$ *uma autómato e* s *um seu estado. A linguagem esquerda de* S *é*

$$\{\rho \in \Sigma^\star \mid \widehat{\delta}(s_0, \rho) = s\}.$$

4.4 Equivalência entre Linguagens Regulares e Linguagens Aceites por Autómatos Finitos

As linguagens aceites por autómatos finitos são precisamente as linguagens regulares, i.e. as descritas por expressões regulares.

Teorema 4.34 (Kleene[Kle56]) *Uma linguagem é reconhecível por um autómato finito se e só se for regular. Ou seja, o conjunto das linguagens representadas pelas expressões regulares é o mesmo que o representado por DFA.*

Para a demonstração deste teorema é necessário mostrar que:

- Qualquer linguagem descrita por uma expressão regular é aceite por um autómato finito.

- Qualquer linguagem aceite por um autómato finito é descrita por uma expressão regular.

Existem vários modos de conversão entre estes dois modelos. Vamos para cada caso analisar alguns e indicar quais as suas características e vantagens.

4.4.1 De Expressões Regulares para Autómatos Finitos

Neste caso temos de demonstrar que:

Teorema 4.35 *Seja* L *uma linguagem representada por uma expressão regular* r, $\mathrm{L} = \mathcal{L}(\mathrm{r})$*, então existe um DFA* A*, tal que* $\mathrm{L} = \mathcal{L}(\mathrm{A})$.

Pretendemos obter para cada expressão regular r um DFA equivalente. Alguns dos métodos convertem a expressão regular r num autómato finito não determinístico equivalente, NFA. Como, pelo Teorema 4.23 apresentado na Secção 4.2, podemos transformar este NFA num DFA, temos construído o autómato pretendido.

As conversões distinguem-se ainda pelo facto dos NFAs obtidos terem ou não transições por ε.

Temos, entre outros, os seguintes algoritmos (a que habitualmente se associa um autómato com o mesmo nome):

- Algoritmo de Thompson: produz um NFAε.
- Algoritmo de Brzozowski: produz um DFA, denominado autómato de derivavas.
- Algoritmo de Glushkov: produz um NFA, denominado autómato de posições.
- Algoritmo de Antimirov: produz um NFA, denominado autómato de derivadas parciais.

Vamos considerar os dois primeiros: algoritmo de Thompson e algoritmo de Brzozowski.

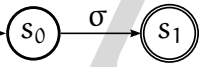
4.4.1.1 Algoritmo de Thompson

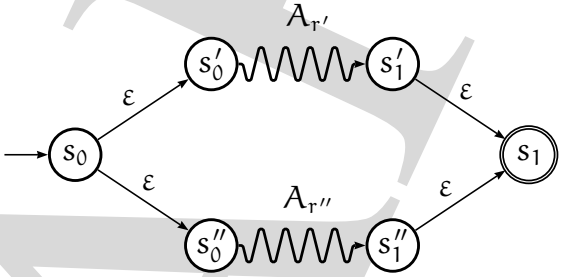
O algoritmo de Thompson [?] constrói um NFAεs indutivamente na definição das expressões regulares. Para cada regra indutiva da Definição 3.1, e supondo que temos um NFAε correspondente a cada uma das expressões regulares (se for esse o caso) constituintes, mostramos como se pode construir um NFAε que corresponde à expressão regular final. Construamos então cada um dos NFAε A_{r} correspondentes a cada uma das expressões regulares r obtidas pelas regras de 3.1. Em cada caso supomos conhecido o alfabeto, Σ.

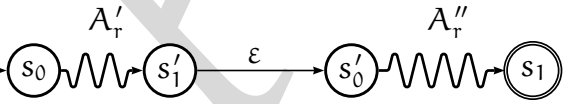
De notar, que depois da aplicação de cada passo deste método, o autómato resultante está normalizado como indicado no Lema 4.31, isto é, possui um só estado final do qual não partem transições, assim como um estado inicial ao qual não chega qualquer transição. Pelo que fica assegurada a aplicabilidade recursiva do método.

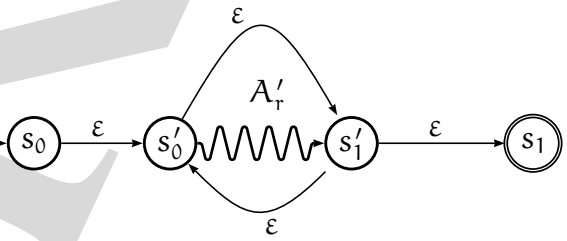
i) $r = \emptyset$ $\quad A_\emptyset :$ 

ii) $r = \varepsilon$ $\quad A_\varepsilon :$ 

iii) $r = \sigma, \sigma \in \Sigma$ $\quad A_\sigma :$ 

iv) $r = r' + r''$ $\quad A_{r'+r''} :$ 

v) $r = r'r''$ $\quad A_{r'r''} :$ 

vi) $r = r'^\star$ $\quad A_{r'^\star} :$ 

Proposição 4.36 (Correção do Algoritmo de Thompson) *Seja* r *um expressão regular. O Algoritmo de Thompson produz um NFA*ε A_r *tal que* $\mathcal{L}(r) = \mathcal{L}(A_r)$.

Dem. *A correção de cada uma das construções é simples à luz da definição de linguagem associada a uma expressão regular (ver também a Definição 3.1).*

i) $r = \emptyset$

Neste caso $\mathcal{L}(r) = \emptyset$ *e* $A_\emptyset = \langle \{s_0\}, \Sigma, \delta, \{s_0\}, \emptyset \rangle$. *Como o conjunto de estados finais é vazio temos* $\mathcal{L}(A_\emptyset) = \emptyset = \mathcal{L}(r)$..

ii) $r = \varepsilon$ *Neste caso* $\mathcal{L}(r) = \{\varepsilon\}$ *e* $A_\varepsilon = \langle \{s_0\}, \Sigma, \{s_0\}, \delta, \{s_0\} \rangle$. *Como a função de transição* δ *não está definida para nenhum par (estado,símbolo), mas o estado inícial é final temos, pela definição de* $\widehat{\delta}$, *que* $\mathcal{L}(A_\varepsilon) = \{\varepsilon\} = \mathcal{L}(r)$.

iii) $r = \sigma, \sigma \in \Sigma$

Neste caso $\mathcal{L}(r) = \{\sigma\}$ *e* $A_\varepsilon = \langle \{s_0, s_1\}, \Sigma, \{s_0, s_1\}, \delta, \{s_1\} \rangle$, *com* $\delta(s_0, \sigma\rangle = \{s_1\}$. *Pela função de transição* δ *e sendo* s_1 *um estado final, temos que* $\mathcal{L}(A_\sigma) = \{\sigma\} = \mathcal{L}(r)$.

iv) $r = r' + r''$

Seja $A_{r'} = \langle S', \Sigma, \delta', \{s_0'\}, \{s_1'\} \rangle$ *e* $A_{r''} = \langle S'', \Sigma, \delta'', \{s_0''\}, \{s_1''\} \rangle$. *Temos que* $\mathcal{L}(r) = \mathcal{L}(r + r') = \mathcal{L}(r') \cup \mathcal{L}(r'')$ *e, por hipótese de indução,* $\mathcal{L}(r') = \mathcal{L}(A_{r'})$ *e* $\mathcal{L}(r'') = \mathcal{L}(A_{r''})$. *Seja* $A_r = \langle S, \Sigma, \delta, \{s_0\}, \{s_1\} \rangle$, *com*

- $S = S' \cup S'' \cup \{s_0, s_1\}$
- *a função* δ *é definida por:*
  - $\delta(s_0, \varepsilon) = \{s_0', s_0''\}$
  - $\delta(s_1', \varepsilon) = \{s_1\}$ *e* $\delta(s_1'', \varepsilon) = \{s_1\}$
  - *para* $s' \in S'$ *e* $\sigma \in \Sigma \cup \{\varepsilon\}$, $\delta(s', \sigma) = \delta'(s', \sigma)$.
  - *para* $s'' \in S''$ *e* $\sigma \in \Sigma \cup \{\varepsilon\}$, $\delta(s'', \sigma) = \delta''(s'', \sigma)$.

Queremos provar que $\mathcal{L}(r) = \mathcal{L}(A_r)$.

Se $\rho \in \mathcal{L}(r)$ *então* $\rho \in \mathcal{L}(r')$ *ou* $\rho \in \mathcal{L}(r'')$, *e por hipótese de indução* $\rho \in \mathcal{L}(A_{r'})$ *ou* $\rho \in \mathcal{L}(A_{r''})$, *respectivamente. Se* $\rho \in \mathcal{L}(A_{r'})$ *então* $\widehat{\delta'}(\{s_0'\}, \rho) \supseteq \{s_1'\}$. *E, do mesmo modo, se* $\rho \in \mathcal{L}(A_{r''})$ *então* $\widehat{\delta''}(\{s_0''\}, \rho) \supseteq \{s_1''\}$. *Em ambos os casos, como* $F_\varepsilon(\{s_1'\}) = F_\varepsilon(\{s_1''\}) = \{s_1\}$, *temos que* $\rho \in \mathcal{L}(A_r)$. *Suponhamos que* $\rho \in \mathcal{L}(A_{r'})$. *Então,*

$$\begin{aligned}
\widehat{\delta}(F_\varepsilon(\{s_0\}), \rho) &= \widehat{\delta}(\{s_0\} \cup F_\varepsilon(\{s_0'\}) \cup F_\varepsilon(\{s_0''\}), \rho) \\
&\supseteq \widehat{\delta}(F_\varepsilon(\{s_0'\}), \rho) \\
&\supseteq F_\varepsilon(\{s_1'\}) \\
&= \{s_1\}
\end{aligned}$$

Analogamente, se $\rho \in \mathcal{L}(A_{r''})$. *Logo* $\mathcal{L}(A_{r'}) \cup \mathcal{L}(A_{r''}) \subseteq \mathcal{L}(A_r)$.

Seja $\rho \in \mathcal{L}(A_r)$. *Então pela definição de linguagem aceite por um NFA*ε, $\widehat{\delta}(F_\varepsilon(\{s_0\}), \rho) \cap \{s_1\} \neq \emptyset$ *implica que* $\widehat{\delta}(F_\varepsilon(\{s_0\}), \rho) = \{s_1\}$. *Suponhamos, por*

absurdo, que $\rho \notin \mathcal{L}(A_{r'})$ *e* $\rho \notin \mathcal{L}(A_{r''})$. *Então,* $\widehat{\delta'}(F_\varepsilon(\{s_0'\}), \rho) \neq \{s_1'\}$ *e* $\widehat{\delta''}(F_\varepsilon(\{s_0''\}), \rho) \neq \{s_1''\}$. *Os estados* s_1' *e* s_1'' *são os únicos estados finais de* $A_{r'}$ *e* $A_{r''}$, *respectivamente, e dos únicos que se pode chegar a* s_1 *em* A_r *(e sem consumir símbolos). Isto é, não se poderia ter* $s_1 \in \widehat{\delta}(F_\varepsilon(\{s_0\}), \rho)$ *já que*

$$\begin{aligned}\widehat{\delta}(F_\varepsilon(\{s_0\}), \rho) &= \widehat{\delta}(\{s_0', s_0''\}, \rho) \\ &= F_\varepsilon(\widehat{\delta'}(F_\varepsilon(\{s_0'\}), \rho)) \cup F_\varepsilon(\widehat{\delta'}(F_\varepsilon(\{s_0'\}), \rho))\end{aligned}$$

e portanto tinhamos $\rho \notin \mathcal{L}(A_r)$ *(absurdo!).*

Mas então $\mathcal{L}(A_r) \subseteq \mathcal{L}(A_{r'}) \cup \mathcal{L}(A_{r''})$. *Concluímos que* $\mathcal{L}(A_r) = \mathcal{L}(A_{r'}) \cup \mathcal{L}(A_{r''}) = \mathcal{L}(r') \cup \mathcal{L}(r'') = \mathcal{L}(r' + r'') = \mathcal{L}(r)$.

v) $r = r'r''$

Seja $A_{r'} = \langle S', \Sigma, \delta', \{s_0\}, \{s_1'\}\rangle$ *e* $A_{r''} = \langle S'', \Sigma, \delta'', \{s_0'\}, \{s_1\}\rangle$. *Temos que* $\mathcal{L}(r) = \mathcal{L}(r') \cdot \mathcal{L}(r'')$ *e, por hipótese de indução,* $\mathcal{L}(r') = \mathcal{L}(A_{r'})$ *e* $\mathcal{L}(r'') = \mathcal{L}(A_{r''})$. *Seja* $A_r = \langle S' \cup S'', \Sigma, \delta, \{s_0\}, \{s_1\}\rangle$, *com a função* δ *é definida por:*

- $\delta(s_1', \varepsilon) = \{s_0'\}$
- *Para* $s' \in S'$ *e* $\sigma \in \Sigma \cup \{\varepsilon\}$, $\delta(s', \sigma) = \delta'(s', \sigma)$.
- *Para* $s'' \in S''$ *e* $\sigma \in \Sigma \cup \{\varepsilon\}$, $\delta(s'', \sigma) = \delta''(s'', \sigma)$.

Queremos provar que $\mathcal{L}(r) = \mathcal{L}(A_r)$.

Seja $\rho \in \mathcal{L}(r)$. *Então existem* $\rho', \rho'' \in \Sigma^\star$ *tal que* $\rho' \in \mathcal{L}(r')$ *e* $\rho'' \in \mathcal{L}(r'')$. *Tem-se que* $\rho' \in \mathcal{L}(A_{r'})$ *e* $\rho'' \in \mathcal{L}(A_{r''})$. *Como no caso anterior, tem-se que* $\widehat{\delta'}(\{s_0\}, \rho') = \{s_1'\}$ *e* $\widehat{\delta''}(\{s_0'\}, \rho'') = \{s_1\}$. *Então,*

$$\begin{aligned}\widehat{\delta}(F_\varepsilon\{s_0\}, \rho) &= \widehat{\delta}(F_\varepsilon\{s_0\}, \rho'\rho'') \\ &= \widehat{\delta}(\widehat{\delta'}(F_\varepsilon\{s_0\}, \rho'), \rho'') \\ &= \widehat{\delta}(F_\varepsilon(\{s_1'\}), \rho'') \\ &= \widehat{\delta'}(F_\varepsilon(\{s_0'\}), \rho'') \\ &= \{s_1\}\end{aligned}$$

Logo, $\rho \in \mathcal{L}(A_r)$ *e, assim,* $\mathcal{L}(r) \subseteq \mathcal{L}(A_r)$.

Seja agora $\rho \in \mathcal{L}(A_r)$. *Então sabemos que* $\widehat{\delta}(F_\varepsilon\{s_0\}, \rho) = \{s_1\}$. *Ao reconhecer a palavra* ρ *temos de passar necessariamente pelo estado* s_1'. *Seja* ρ' *o prefixo de* ρ *tal que*

$$\widehat{\delta}(F_\varepsilon(\{s_0\}), \rho') = \widehat{\delta'}(F_\varepsilon(\{s_0\}), \rho') = \{s_1'\}. \tag{4.11}$$

Não consumindo mais símbolos da palavra ρ *pode-se passar, por transições por* ε, *para o estado* s_0' *e continuar a reconhecer o "resto" da palavra* ρ, *seja*

ρ''. *Tem-se que*

$$\widehat{\delta}(F_\varepsilon(\{s_0'\}), \rho'') = \widehat{\delta'}(F_\varepsilon(\{s_0'\}), \rho'') = \{s_1\}. \tag{4.12}$$

Podemos concluir que $\rho = \rho'\rho''$ *tal que* $\rho' \in \mathcal{L}(A_{r'})$ *e* $\rho'' \in \mathcal{L}(A_{r''})$, *o que prova que* $\mathcal{L}(A_r) \subseteq \mathcal{L}(A_{r'})\mathcal{L}(A_{r''}) = \mathcal{L}(r')\mathcal{L}(r'') = \mathcal{L}(r)$.

vi) $r = r'^\star$

Seja $A_{r'} = \langle S', \Sigma, \delta', \{s_0'\}, \{s_1'\}\}\rangle$ *e, por hipótese de indução,* $\mathcal{L}(r') = \mathcal{L}(A_{r'})$. *Seja* $A_r = \langle S' \cup \{s_0, s_1\}, \Sigma, \delta, \{s_0\}, \{s_1\}\rangle$, *com a função* δ *é definida por:*

- $\delta(s_0, \varepsilon) = \{s_0'\}$
- $\delta(s_0', \varepsilon) = \{s_1'\}$
- $\delta(s_1', \varepsilon) = \{s_0'\}$
- $\delta(s_1', \varepsilon) = \{s_1\}$
- *Para* $s' \in S'$ *e* $\sigma \in \Sigma \cup \{\varepsilon\}$, $\delta(s', \sigma) = \delta'(s', \sigma)$.

Temos que $F_\varepsilon(s_0) \supseteq \{s_0, s_0', s_1', s_1\}$, $F_\varepsilon(s_0') \supseteq \{s_0', s_1', s_1\}$ *e* $F_\varepsilon(s_1') \supseteq \{s_0', s_1', s_1\}$.

Queremos provar que $\mathcal{L}(r) = \mathcal{L}(A_r)$.

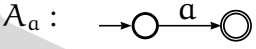
Seja $\rho \in \mathcal{L}(r)$. *Se* $\rho = \varepsilon$ *então, pela definição de* δ, $\widehat{\delta}$ *e* $F_\varepsilon(s_0)$, *é imediato que* $\rho \in \mathcal{L}(A_r)$.

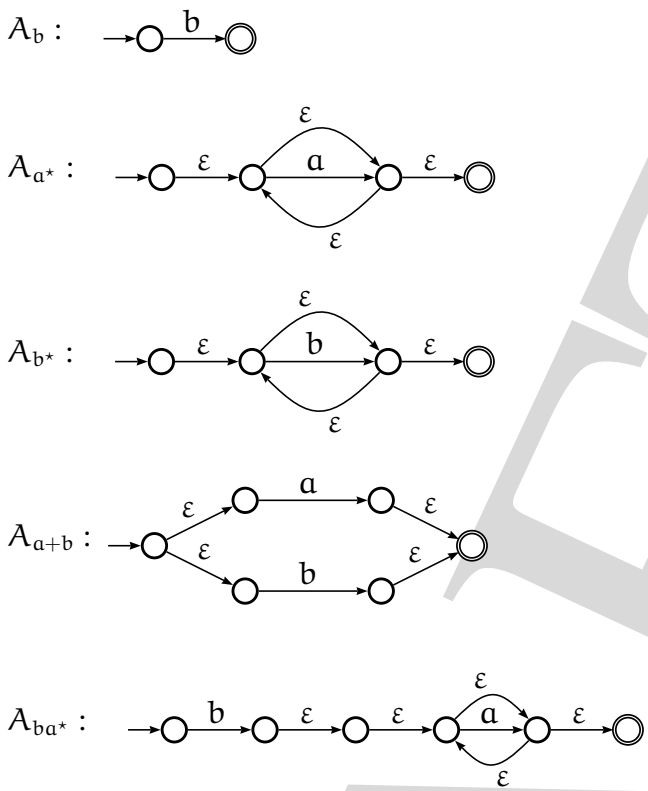
Senão, existe $n > 0$ *tal que* $\rho \in \mathcal{L}(r')^n$, *isto é,* $\rho = \rho_1 \dots \rho_n$ *tal que* $\rho_i \in \mathcal{L}(r')$, $1 \leq i \leq n$. *Provamos por indução em* n, *que* $\rho \in \mathcal{L}(A_r)$. *Se* $n = 1$, *então* $\rho \in \mathcal{L}(A_{r'})$ *e considerando* $F_\varepsilon(s_0)$ *e* $F_\varepsilon(s_1')$ *tem-se que* $\rho \in \mathcal{L}(A_r)$. *Suponhamos que a propriedade se verifica para* $\rho \in \mathcal{L}(r')^m$, *para* $m < n$. *Se* $\rho \in \mathcal{L}(r')^n$, *então* $\rho = \rho'\rho''$ *tal que* $\rho' \in \mathcal{L}(r)$ *e* $\rho'' \in \mathcal{L}(r')^{(n-1)}$. *Como no caso para* $n = 1$, *tem-se que* $\rho' \in \mathcal{L}(A_{r'})$, *e considerando o fecho por* ε *de* s_1' *e a hipótese de indução tem-se que* $\rho \in \mathcal{L}(A_r)$. *Concluímos que* $\mathcal{L}(r) \subseteq \mathcal{L}(A_r)$.

Seja agora $\rho \in \mathcal{L}(A_{r'^\star})$. *Se* $\rho = \varepsilon$ *ou* $\rho \in \mathcal{L}(A_{r'})$, *é imediato pela construção de* A_r *que* $\rho \in \mathcal{L}(r)$. *Caso contrário,* $\rho = \rho'\rho''$ *tal que* $\rho' \in \mathcal{L}(A_{r'})$ *e* ρ'' *tem de ter um prefixo* $\rho''' \in \mathcal{L}(A_{r'})$. *Continuando este processo concluí-se que existe* $n > 0$ *tal que* $\rho = \rho_1 \dots \rho_n$ *tal que* $\rho_i \in \mathcal{L}(A_{r'})$, *isto é,* $\rho_i \in \mathcal{L}(r')$, $i \leq 1 \leq n$. *Então* $\rho \in \mathcal{L}(r'^\star)$. *E, concluindo,* $\mathcal{L}(A_{r'^\star}) \subseteq \mathcal{L}(r'^\star)$.

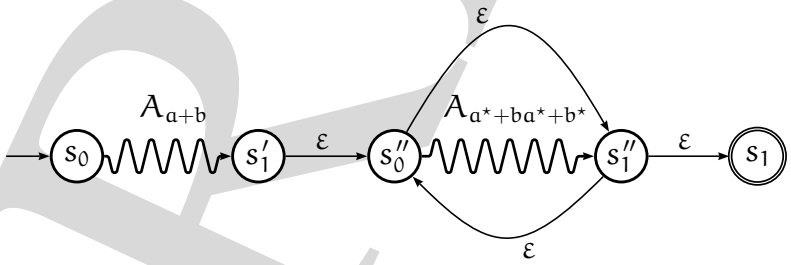
■

Exemplo 4.37 *Considera a expressão regular* $(a + b)(a^\star + ba^\star + b^\star)^\star$. *Podemos considerar os seguintes sub-autómatos:*

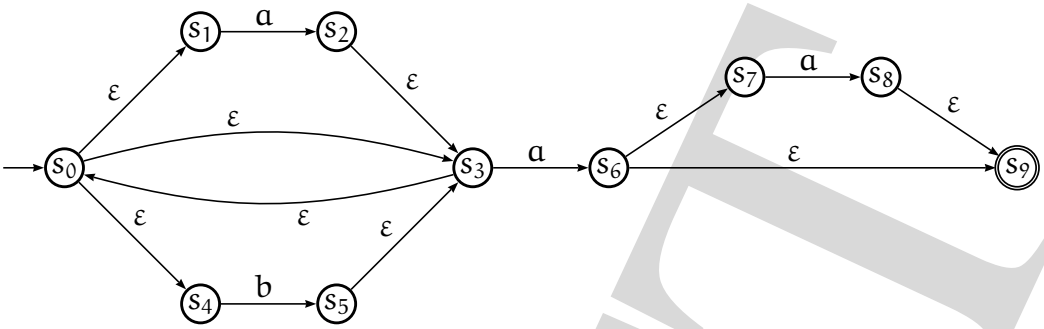
A_a : 



Denotando por $A_{a^\star+ba^\star+b^\star}$ o autómato obtido pela união de $A_{a^\star}$, $A_{ba^\star}$ e $A_{b^\star}$, e considerando mais uma vez as construções para a concatenação e o fecho de Kleene, obtemos o autómato $A_{(a+b)(a^\star+ba^\star+b^\star)^\star}$, correspondente à expressão regular dada:



Exemplo 4.38 *Se aplicarmos o método anterior para obter o autómato não determinístico que reconhece a linguagem correspondente à expressão regular $(a+b)^\star a(a+\varepsilon)$, obtemos (depois de eliminadas algumas transições por ε e respectivos estados, para simplificar):*



4.4.1.2 Derivadas e Algoritmo de Brzozowski

A noção de derivada duma expressão regular em relação a um símbolo, corresponde a uma representação da linguagem que se obtém quando se retira esse símbolo do início das palavras da linguagem que a expressão regular representa.

Dado um símbolo $\sigma \in \Sigma$ e uma linguagem $L \subseteq \Sigma^\star$ podemos definir a linguagem quociente de L por σ:

$$\sigma^{-1}L = \{\rho \mid \sigma\rho \in L\} \tag{4.13}$$

Começamos por associar a cada expressão regular r uma função que indica se $\mathcal{L}(r)$ contém ou não a palavra vazia, ε.

Definição 4.39 (Parte constante duma expressão regular) *Seja* $\varepsilon() : \mathsf{RE} \to \mathsf{RE}$ *a função* constante *que dada uma expressão regular* r *se define do seguinte modo:*

$$\varepsilon(r) = \begin{cases} \varepsilon & \text{se } \varepsilon \in \mathcal{L}(r), \\ \emptyset & \text{caso contrário.} \end{cases}$$

Lema 4.40 *A parte constante* $\varepsilon(r)$*, duma expressão regular* r *pode-se determinar indutivamente pelas seguintes regras:*

$$\begin{aligned}
\varepsilon(\varepsilon) &= \varepsilon \\
\varepsilon(\emptyset) &= \varepsilon(\sigma) = \emptyset, \ \forall \sigma \in \Sigma \\
\varepsilon(r' + r'') &= \begin{cases} \varepsilon(r'') & \text{se } \varepsilon(r') = \varepsilon, \\ \varepsilon(r') & \text{se } \varepsilon(r'') = \varepsilon, \\ \emptyset & \text{caso contrário.} \end{cases} \\
\varepsilon(r'.r'') &= \begin{cases} \varepsilon & \text{se } \varepsilon(r') = \varepsilon(r'') = \varepsilon, \\ \emptyset & \text{caso contrário.} \end{cases} \\
\varepsilon(r'^\star) &= \varepsilon
\end{aligned}$$

Problema 44 Prova, por indução na estrutura duma expressão regular, o lema anterior, Lema 4.40.

Definição 4.41 (Derivada de uma expressão regular por um símbolo) *Seja* r *uma expressão regular e* σ *um símbolo de* Σ*. A derivada de* r *em relação a* σ*, denotada por* $D_\sigma r$ *é a expressão regular que se obtém pela seguinte definição indutiva:*

$$D_\sigma\sigma = \varepsilon \tag{4.14}$$

$$D_\sigma\emptyset = D_\sigma\varepsilon = D_\sigma\sigma' = \emptyset \qquad \forall\sigma' \in \Sigma,\ \sigma' \neq \sigma \tag{4.15}$$

$$D_\sigma(r' + r'') = D_\sigma r' + D_\sigma r'' \tag{4.16}$$

$$D_\sigma(r'.r'') = (D_\sigma r').r'' + \varepsilon(r').D_\sigma r'' \tag{4.17}$$

$$D_\sigma r'^\star = (D_\sigma r').r'^\star \tag{4.18}$$

Exemplo 4.42 *As derivadas, em relação a* a *e* b*, para a expressão regular* r*,* $(a + b)^\star(a + ab^\star)$ *são as seguintes:*

$$\begin{aligned}
D_a r &= D_a(a + b)^\star.(a + ab^\star) + \varepsilon((a + b)^\star).D_a(a + ab^\star)\\
&= D_a(a + b).r + D_a(a) + D_a(ab^\star)\\
&= \varepsilon.r + \emptyset.r + \varepsilon + \varepsilon.b^\star\\
&= r + \varepsilon + b^\star\\
D_b r &= D_b(a + b)^\star.(a + ab^\star) + \varepsilon((a + b)^\star).D_b(a + ab^\star)\\
&= D_b(a + b).r + D_b(a) + D_b(ab^\star)\\
&= \varepsilon.r + \emptyset.r + \emptyset + \emptyset.b^\star\\
&= r
\end{aligned}$$

É importante notar que para obter as derivadas consideramos o facto de ε ser elemento neutro da concatenação e de $\emptyset$ ser elemento neutro da união e elemento absorvente da concatenação (ver Proposição 3.8). Também devem ser consideradas as propriedades associativa, comutativa e de idempotência da união, e que iremos denotar por propriedades ACI.

Exemplo 4.43 *As derivadas, em relação a* a *e a* b*, para a expressão regular* r*,* $(a + b)(a^\star + ba^\star + b^\star)^\star$ *são as seguintes:*

$$D_a r = D_a(a + b).(a^\star + ba^\star + b^\star)^\star = (a^\star + ba^\star + b^\star)^\star = D_b r$$

Lema 4.44 *Para toda a expressão regular* $r \in \mathrm{RE}$*, e para todo o símbolo* $\sigma \in \Sigma$*,* $\mathcal{L}(D_\sigma r) = \sigma^{-1}\mathcal{L}(r)$*.*

Problema 45 Prova, por indução na estrutura duma expressão regular, o lema anterior, Lema 4.44.

A noção de derivada em relação a um símbolo pode-se generalizar para palavras $\rho \in \Sigma^\star$.

Definição 4.45 (Derivada em relação a uma palavra) *Seja* r *uma expressão regular e* ρ *uma palavra de* $\Sigma^\star$*. A derivada de* r *em relação a* ρ *é uma expressão regular, denotada por* $D_\rho r$*, e que define-se indutivamente no tamanho de* ρ*:*

$$D_\varepsilon r = r \tag{4.19}$$

$$D_{\sigma\rho'} r = D_{\rho'}(D_\sigma(r)) \tag{4.20}$$

Exemplo 4.46 *Considera a expressão regular* r *dada por* $(a+b)(a^\star + ba^\star + b^\star)^\star$ *do Exemplo 4.43. Vamos calcular a derivada desta expressão regular* r *em relação à palavra* aab*. Pela Definição 4.45, tem-se que* $D_{aab} = D_b(D_a(D_a r)))$*.*

Pelo Exemplo 4.43, tem-se que $D_a r = (a^\star + ba^\star + b^\star)^\star$*. Derivando novamente por* a *tem-se:*

$$\begin{aligned} D_a(D_a r) =& (D_a(a^\star) + D_a(ba^\star) + D_a b^\star)(a^\star + ba^\star + b^\star) \\ =& (a^\star + \emptyset + \emptyset)(a^\star + ba^\star + b^\star)^\star \\ =& a^\star(a^\star + ba^\star + b^\star)^\star \end{aligned}$$

e, finalmente,

$$\begin{aligned} D_b(D_a(D_a r)) =& D_b(a^\star).(a^\star + ba^\star + b^\star)^\star + \varepsilon(a^\star) D_b((a^\star + ba^\star + b^\star)^\star) \\ =& D_b(a^\star + ba^\star + b^\star).(a^\star + ba^\star + b^\star)^\star \\ =& (a^\star + b^\star)(a^\star + ba^\star + b^\star)^\star \end{aligned}$$

Como no caso de um símbolo $\sigma \in \Sigma$, dada uma palavra $\rho \in \Sigma^\star$ e uma linguagem $L \subseteq \Sigma^\star$ podemos definir a linguagem quociente de L por ρ:

$$\rho^{-1}L = \{\rho' \mid \rho\rho' \in L\} \tag{4.21}$$

Lema 4.47 *Para toda a expressão regular* $r \in \mathrm{RE}$*, e para todo o símbolo* $\sigma \in \Sigma$*,* $\mathcal{L}(D_\rho r) = \rho^{-1}\mathcal{L}(r)$*.*

Problema 46 Prova, por indução na estrutura duma expressão regulare no tamanho da palavra, o lema anterior, Lema 4.47.

Definição 4.48 (expressões regulares similares) *Duas expressões regulares dizem-se* similares *se uma pode ser transformada na outra por aplicação das seguintes regras de equivalência:*

$$\begin{aligned} \varepsilon r = r\varepsilon &= r \\ \emptyset r = r\emptyset &= \emptyset \\ r + (r' + r'') &= (r + r') + r'' && \text{(A)} \\ r + r' &= r' + r && \text{(C)} \\ r + r &= r && \text{(I)} \end{aligned}$$

e duas expressões regulares dizem-se não similares *se não forem similares.*

Exemplo 4.49 *As seguintes expressões regulares são similares:* $(\emptyset.a^\star + \varepsilon.a^\star) + a^\star$ *e* $a^\star$*.*

Um facto importante demonstrado por J. Brzozowski [Brz64] é o de que o conjunto de derivadas duma expressão regular em relação a todas as palavras $\rho \in \Sigma^\star$ (ou, simplesmente o conjunto de derivadas duma expressão regular) é finito, desde que se considerem as propriedades algébricas atrás referidas.

Proposição 4.50 (Brzozowski) *O conjunto* D_r *de derivadas não similares duma expressão regular* $r \in RE$ *é finito.*

É importante constatar que se não se considerassem expressões regulares similares, o conjunto de derivadas duma expressão regular podia ser infinito.

Exemplo 4.51 *Seja a expressão regular* $a^{\star\star}$ *e considerem-se as derivadas em relação às palavras da forma* a^n*, para* $n \geq 1$*.*

$$\begin{aligned} D_a(a^{\star\star}) &= a^\star a^{\star\star} \\ D_{aa}(a^{\star\star}) &= a^\star a^{\star\star} + a^\star a^{\star\star} \\ D_{aaa}(a^{\star\star}) &= a^\star a^{\star\star} + a^\star a^{\star\star} + a^\star a^{\star\star} \\ &\vdots \end{aligned}$$

Exemplo 4.52 *Vamos calcular as derivadas da expressão regular,* r *dada por* $(a + b)(a^\star + ba^\star + b^\star)^\star$*. Para tal consideramos as palavras de* $\{a, b\}^\star$ *por ordem lexicográfica. Começamos por atribuir uma ordem aos símbolos do alfabeto, neste caso,* $a < b$*. Depois consideramos a palavra de tamanho* 0 *(*ε*), depois as de tamanho* 1*,* a *e* b*, em seguida as de tamanho* 2*, (*aa*,* ab*,* ba *e* bb*), e assim sucessivamente até não aparecerem novas derivadas (não similares). Pela Proposição 4.50 sabemos que o processo termina.*

Tem-se:

$$
\begin{aligned}
D_\varepsilon r =& r = (a+b)(a^\star + ba^\star + b^\star)^\star \\
D_a r =& (a^\star + ba^\star + b^\star)^\star = r' \\
D_b r =& r' \\
D_{aa} r =& D_a(D_a r) = D_a r' = D_a(a^\star + ba^\star + b^\star)r' = a^\star r' = r'' \\
D_{ab} r =& D_b(D_a r) = D_b r' = (a^\star + b^\star)r' = r''' \\
D_{ba} r =& D_a(D_b r) = D_a r' = r'' \\
D_{bb} r =& D_b(D_b r) = D_b r' = (a^\star + b^\star)r' = r''' \\
D_{aaa} r =& D_a(D_a(D_a r)) = D_a(D_a r') = D_a r'' = a^\star r' + \varepsilon(a^\star)D_a r' = r'' \\
D_{aab} r =& D_{ab}(D_a r) = D_b(D_a r') = D_b r'' = D_b r' = r''' \\
D_{aba} r =& D_{ba}(D_a r) = D_a(D_b r') = D_a r''' = r'' \\
D_{abb} r =& D_{bb}(D_a r) = D_b(D_b r') = D_b r''' = b^\star r' + r''' = r'''' \\
D_{abba} r =& D_{bba}(D_a r) = D_a(r'''') = r'' \\
D_{abbb} r =& D_{bbb}(D_a r) = D_b(r'''') = D_b(b^\star)r' + D_b r''' = r''''
\end{aligned}
$$

Como nestes duas últimas derivações não surgiram novas expressões, o processo termina e concluímos que $D_r = \{r, r', r'', r''', r''''\}$.

É imediato, do exemplo anterior, que em vez de se ir considerando sucessivamente palavras de $\Sigma^\star$, é equivalente ir derivando em relação a cada símbolo do alfabeto as expressões regulares que vão aparecendo no processo.

Problema 47 Verifica o procedimento agora referido para o Exemplo 4.52.

Definição 4.53 (Autómato de Derivadas (ou de Brzozowski)) *Seja* r *uma expressão regular e* D_r *o conjunto seu conjunto de derivadas não similares. O autómato de derivadas é* $A_r = \langle D_r, \Sigma, \delta_r, \{r\}, F_r \rangle$ *onde* δ_r *e* F_r *são definidos por:*

$$
\begin{aligned}
\delta_r(s, \sigma) &= D_\sigma(s) && \forall s \in D_r, \forall \sigma \in \Sigma \\
F_r &= \{s \in D_r \mid \varepsilon(s) = \varepsilon\}
\end{aligned}
$$

Proposição 4.54 (Brzozowski) *Para toda a expressão regular* r, $\mathcal{L}(A_r) = \mathcal{L}(r)$.

Exemplo 4.55 *Seja mais uma vez a expressão regular* r, *dada por* $(a+b)(a^\star + ba^\star + b^\star)^\star$.

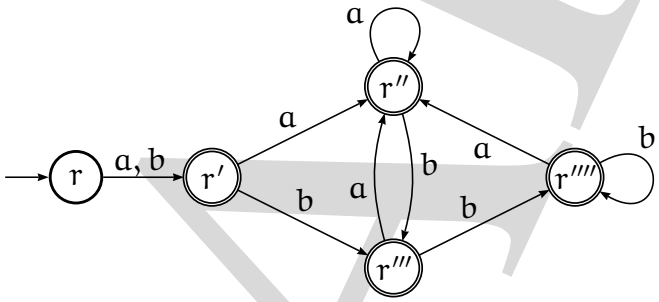
No Exemplo 4.52 já calculamos o conjunto das derivdas D_r*. O autómato das derivadas de* r *é* $A_r = \langle D_r, \{a, b\}, \delta_r, \{r\}, F_r \rangle$*, onde*

$$\begin{aligned} D_r &= \{(a+b)(a^\star + ba^\star + b^\star)^\star, (a^\star + ba^\star + b^\star)^\star, a^\star(a^\star + ba^\star + b^\star)^\star, \\ &\quad (a^\star + b^\star)(a^\star + ba^\star + b^\star)^\star, b^\star(a^\star + ba^\star + b^\star)^\star + (a^\star + b^\star)(a^\star + ba^\star + b^\star)^\star\} \\ &= \{r, r', r'', r''', r''''\} \end{aligned}$$

$$F_r = \{r', r'', r''', r''''\}$$

$$\begin{array}{llll} \delta_r(r, a) = D_a r = r' & & \delta_r(r, b) = D_b r = r' \\ \delta_r(r', a) = D_a r' = r'' & & \delta_r(r', b) = D_b r' = r''' \\ \delta_r(r'', a) = D_a r'' = r'' & & \delta_r(r'', b) = D_b r'' = r''' \\ \delta_r(r''', a) = D_a r''' = r'' & & \delta_r(r''', b) = D_b r''' = r'''' \\ \delta_r(r'''', a) = D_a r'''' = r'' & & \delta_r(r'''', b) = D_b r'''' = r'''' \end{array}$$

e com o seguinte diagrama:



É de salientar que este autómato é um DFA e, ainda assim é menor que o NFAε *obtido no Exemplo 4.37.*

Problema 48 Compara os autómatos obtidos nos Exemplos 4.37 e 4.55, obtendo um autómato determinístico equivalente ao NFAε do Exemplo 4.37.

4.4.2 De Autómatos Finitos para Expressões Regulares

Existem diversas formas de se obter uma expressão regular que represente a mesma linguagem que um dado autómato finito reconhece. Aqui exporemos um método conhecido como Algoritmo de Eliminação de Estados (AEE). Vamos supor que o autómato finito é um NFAε e está normalizado, de acordo com o Lema 4.31:

i) Asseguramos que existe um só estado inicial do autómato, s_0, tal que não existe nenhuma transição que termine nesse estado. Se não for esse o caso, acrescentamos um novo estado s_α, que passa a ser o estado inicial do autómato, com uma única transição-ε para o estado s_0.

ii) Asseguramos que existe um só estado final, e que deste estado não parte nenhuma transição. Se não for este o caso, acrescentamos um novo estado s_Ω, que passa a ser o único final, e, para cada estado final do autómato original, criamos uma transição-ε para este novo estado.

Generalizamos o conceito de autómato finito a um autómato cuja função de transição δ' passa a ser de uma diferente natureza:

$$\delta' : S \times RE \longrightarrow S$$

por forma a que as transições não sejam etiquetadas com símbolos do alfabeto Σ mas sim com expressões regulares com o mesmo alfabeto.

Definição 4.56 (Autómato Finito Generalizado (GFA)) *Um GFA é um quintúplo* $\langle S, \Sigma, \delta, s_0, F\rangle$ *em que:*

- S *é um conjunto finito, não vazio, a que chamamos o* conjunto dos estados*;*
- Σ *é um conjunto finito, não vazio, que constitui o* alfabeto do autómato*;*
- δ *é uma função total* $\delta : S \times S \to RE$ *chamada* função de transição*;*
- s_0*, com* $s_0 \in S$*, é o* estado inicial*;*
- F*, com* $F \subseteq S$*, é o* conjunto dos estados finais*.*

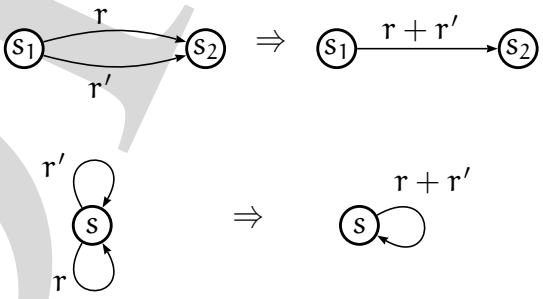
Vamos supor que $\delta(s, s') = \emptyset$ sempre que transição entre $s \in S$ e $s' \in S$ não seja dada explicitamente. Notar que qualquer DFA, NFA ou NFAε é um GFA.

Problema 49 Mostra a afirmação anterior.

4.4.2.1 Algoritmo de Eliminação de Estados (AEE)

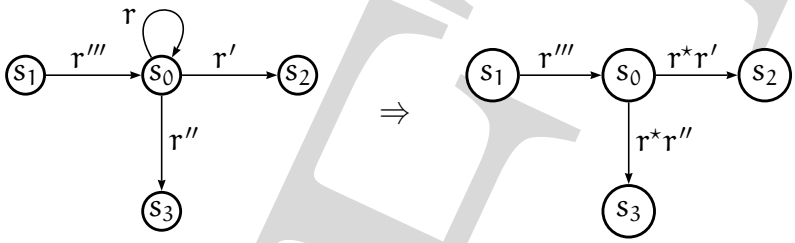
Seja A um autómato finito normalizado.

iii) Começamos por obter um GFA equivalente: fazem-se colapsar transições com as mesmas origens e destinos, sendo a transição restante etiquetada com a união das expressões regulares das transições agora eliminadas.



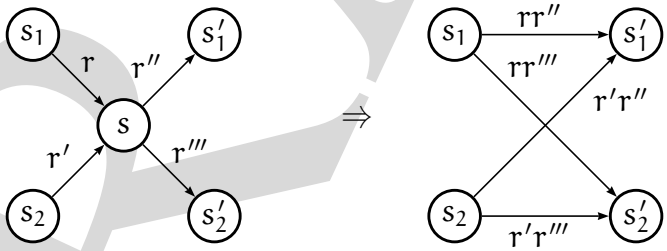
iv) Todos os estados, diferentes do inicial e do final, que não tenham, pelo menos, uma transição que deles parta e uma outra que a eles chegue, são eliminados e com eles as transições que os referem.

v) Nos restantes, eliminamos os lacetes passando a expressão regular que etiqueta cada saída a ser a concatenação do fecho de Kleene da expressão que constava no lacete agora eliminado com a expressão regular que constava nessa transição de saída.

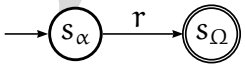


vi) Eliminamos sucessivamente todos os estados, exceptuando o inicial e o final, procedendo para cada um deles do seguinte modo.

Para cada estado s que queremos eliminar, para cada transição $s' \xrightarrow{r} s$ que chega a s ($\delta(s', s) = r$) e para cada transição $s \xrightarrow{r'} s''$ que parte de s ($\delta(s, s'') = r'$), construímos uma nova transição $s' \xrightarrow{rr'} s''$. Por fim, eliminamos o estado s e todas as transições que o referem. No final de cada passo deste é necessário aplicar de novo os passos iii) e v).

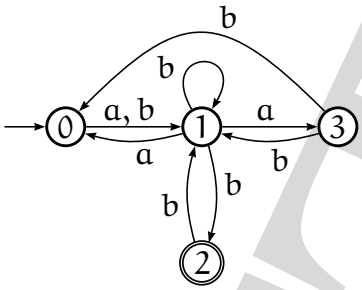


No fim deste processo, o autómato resultante deve ser do tipo

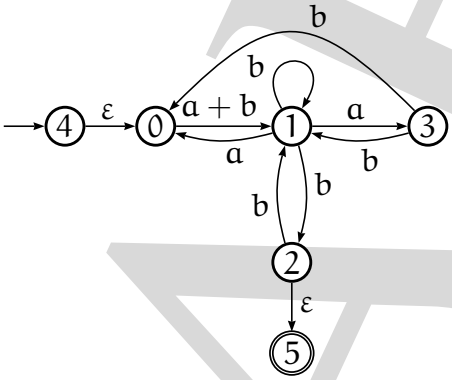


e a expressão regular que procurávamos é r.

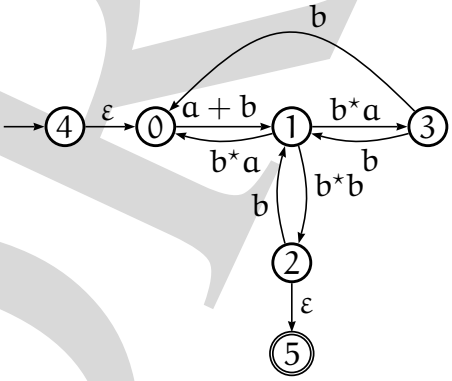
Exemplo 4.57 *Considera o seguinte NFA:*



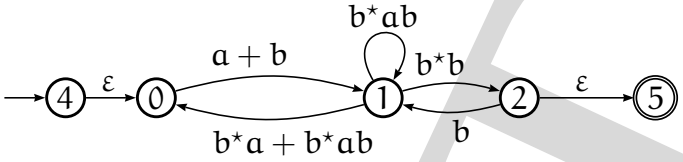
Normalizando e transformando em GFA tem-se



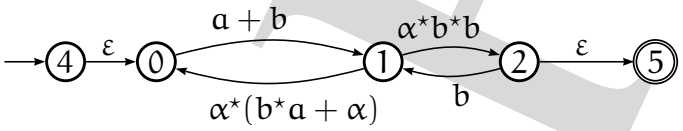
Não há estados que não são atingíveis ou de onde não partem transições. Eliminando os lacetes (neste caso só no estado 1*), tem-se o GFA seguinte*



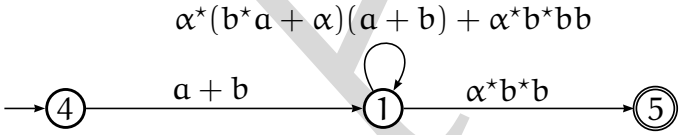
Começamos por eliminar o estado 3 *obtendo o GFA*



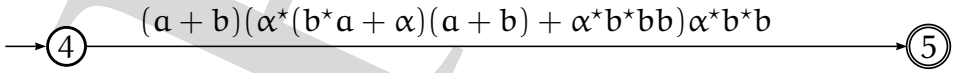
Consideremos, para encurtar um pouco as expressões, $\alpha = b^*ab$*. Então eliminando o lacete existente em* S1 *temos*



Eliminando, agora, os estados 0 *e* 2 *temos*



Eliminando primeiro o lacete em 1 *e depois o próprio estado* 1 *obtemos*



Donde podemos concluir que a expressão regular $(a + b)(\alpha^*(b^*a + \alpha)(a + b) + \alpha^*b^*bb)\alpha^*b^*b$ *é equivalente ao NFA dado.*

É de salientar que a ordem pela qual se eliminam os estados do GFA influencia a forma e o tamanho da expressão regular obtida. Diferentes ordenações odem conduzir a expressões regulares muito diferentes, embora todas equivalentes. Em cada passo também se poderá tentar simplificar as expressões regulares obtidas.

Problema 50 Obtem uma expressão regular equivalente ao NFA do Exemplo 4.57, utilizando uma ordem de eliminação de estados diferente da usada no exemplo referido.