Capítulo 5

Expressões Regulares

As Expressões Regulares são uma outra forma de expressar linguagens regulares, sendo uma alternativa de representação aos autómatos vistos até aqui, e constituindo uma forma mais compacta de representação.

Definição

Uma expressão regular simples pode ser obtida com um qualquer símbolo do alfabeto ou com o símbolo de cadeia vazia (ε). Adicionalmente, pode ser usado o símbolo $\emptyset$ para descrever a linguagem vazia. Agora, e de forma indutiva, uma nova expressão regular pode ser obtida por concatenação (representada por um '.', normalmente omitido), união (representada por um '+') ou fecho (representado por um *) de expressões existentes.

Vamos olhar para alguns exemplos no alfabeto binário (símbolos 0 e 1).

No caso de querermos reconhecer a cadeia 10, a expressão regular que o permite fazer será apenas 10, sendo aqui omitido o símbolo de concatenação.

Se quisermos reconhecer as cadeias 10 ou 01, podemos fazer a união das expressões para cada uma das opções, e ficamos com a expressão 10 + 01.

Se quisermos reconhecer cadeias iniciadas por 1, necessitamos de poder dizer que depois do 1 podem existir zero ou mais ocorrências dos símbolos 0 ou 1. Assim, podemos usar o fecho para permitir essa repetição, usando parêntesis para indicar a associação dos operadores, ficando com a expressão 1(0+1)*.

Exercício 1

Obtenha uma expressão regular para a linguagem das cadeias do alfabeto

{0,1} que quando interpretadas como um número em binário são múltiplos de 4.

Dizer que uma cadeia em binário representa números múltiplos de 4 é equivalente a dizer que as cadeias devem terminar em 00. Assim, a expressão regular torna-se simples de escrever:

(0+1)*00

Permite-se assim reconhecer qualquer sequência de zeros e uns inicialmente, mas obriga-se a que a cadeia termine em dois zeros consecutivos.

Exercício 2

Obtenha uma expressão regular para a linguagem das cadeias sobre o alfabeto {a,b} cujo comprimento seja múltiplo de 3.

Este problema pode resolver-se de forma simples, permitindo repetições de 3 símbolos do alfabeto:

((a+b)(a+b)(a+b))*

Com esta expressão, garante-se que cada bloco existente na cadeia final tem exatamente 3 símbolos, o que permite formar cadeias de comprimento múltiplo de 3 (incluindo a cadeia vazia).

Exercício 3

Obtenha uma expressão regular para a linguagem das cadeias do alfabeto {a,b} em que cada sequência de dois a's é obrigatoriamente seguida de uma sequência de 3 b's.

Para resolver o problema, necessitamos de restringir a quantidade de a's consecutivos e no caso de existirem dois a's, deve-se forçar a existência de 3 b's. Isto pode ser conseguido com a seguinte expressão:

(ab + aabbb + b)* (a + ε)

Olhando para esta solução, podemos verificar que as possibilidades de existência de a's estão sempre associadas a ocorrências de b's: se tivermos dois a's, obriga-se a ter três b's; se existir apenas um a, obriga-se a ter um b, de forma a evitar a's consecutivos. Ao adicionar a possibilidade de ter um b em qualquer ponto, e usando o fecho para a união destas três hipóteses, obtém-se uma solução parcial para o problema. Falta apenas permitir que as cadeias terminem em a, o que é conseguido com a parte $(a + \varepsilon)$ no final (repare-se que não há possibilidade de ter dois a's consecutivos).

Uma outra possível solução seria:
b* (abb* + aabbbb*)* (a + ε)
Nesta solução, permite-se o início da cadeia com a's ou b's, e um qualquer número de b's após o número obrigatório de b's. No final, a componente (a + ε) permite novamente que a cadeia termine possivelmente em a.

Exercício 4

Obtenha uma expressão regular para a linguagem das cadeias sobre o alfabeto {a,b} em que o número de a's é par.

Para resolver este problema, é necessário que cada a seja acompanhado obrigatoriamente de um outro a, garantindo assim a condição de paridade dos a's. Vamos ver algumas possibilidade de resposta.

b* (ab*ab*)*
Esta solução garante que a existência de a's é sempre em número par, permitindo um qualquer número de b's entre os a's. O b* inicial assegura que a cadeia possa iniciar em b, sendo que a parte da direita da expressão já permite cadeias terminadas em a ou em b.

(b + ab*a)*
Esta solução garante que os a's aparecem sempre em número par, e por outro lado permite que existam b's em qualquer ponto da cadeia (o fecho da expressão permite a existência de qualquer número de b's antes ou depois do par de a's, e o b* entre os a's permite também ter b's entre os a's de cada par de a's).

Exercício 5 (Desafio 2014/15)

Encontre uma expressão regular sobre o alfabeto {a, b, c} que permita reconhecer cadeias em que existem pelo menos dois caracteres idênticos consecutivos. Por exemplo, as cadeias 'aba' e 'acaba' não pertencem à linguagem, enquanto que as cadeias 'abba' e 'baaca' pertencem à linguagem.

Tendo o alfabeto 3 símbolos, existem três hipóteses de ter caracteres idênticos consecutivos: aa, bb ou cc. Então, essa parte será descrita pela sub-expressão aa+bb+cc. Temos ainda de permitir que o resto da cadeia tenha qualquer sequência de símbolos do alfabeto, o que pode ser conseguido com a sub-expressão (a+b+c)*. Juntando tudo, obtemos a expressão

(a+b+c)* (aa+bb+cc) (a+b+c)*

Exercício 6 (Exame 2014/15)

Apresente uma expressão regular que permita reconhecer códigos postais nacionais com 7 dígitos. Considere o símbolo 'D' como representando qualquer dígito (0 a 9), 'M' como representando uma letra maiúscula, 'm' como representando uma letra minúscula (incluindo letras acentuadas), 'E' como representando um espaço e 'T' como representando um traço. A expressão deve permitir reconhecer códigos postais como '4200-465 Porto', '4400-017 Vila Nova de Gaia' ou '4490-001 A Ver-o-Mar'.

A parte inicial da expressão é relativamente simples de obter (assumindo que não são colocadas restrições adicionais, como por exemplo o código postal não poder começar com 0): DDDDTDDDE. Falta agora a parte da expressão que permita reconhecer nomes de localidades. Estes nomes devem ter pelo menos uma palavra (embora possam ter mais), sendo que a primeira palavra deve começar por maiúscula, e as palavras devem ser separadas por um espaço ou um traço. Assim, uma possibilidade será Mm*((E+T)(M+m)m*)*. Assim, a expressão final será:

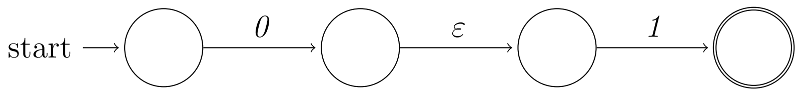
DDDDTDDDEMm*((E+T)(M+m)m*)*

Conversão de Expressões Regulares em Autómatos

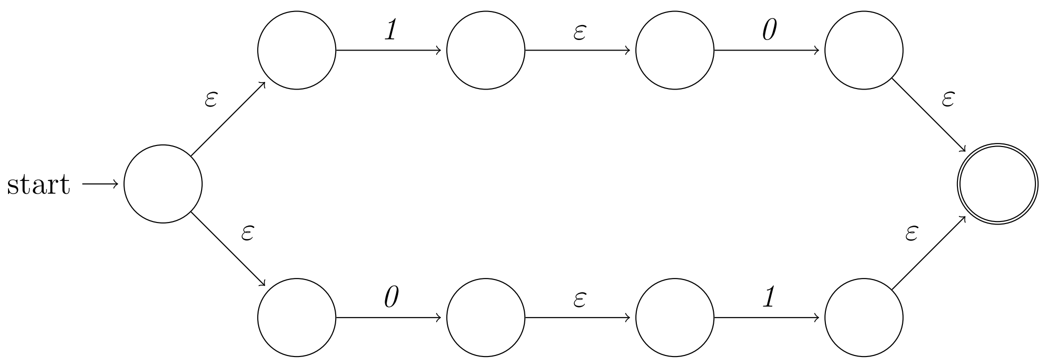
As expressões regulares podem ser convertidas em autómatos (ε-NFAs) através do uso de *templates* para cada uma das possibilidades da definição indutiva das expressões regulares. Recorde-se que, a partir daí, é possível converter para um DFA.

A conversão faz-se então substituíndo os elementos básicos (cada símbolo do alfabeto, cadeia vazia, ou símbolo de linguagem vazia) pelos templates equivalentes, e usando depois os templates das operações de concatenação, união e fecho para ir construíndo um ε-NFA de sub-expressões cada vez mais complexas, até atingir a expressão completa.

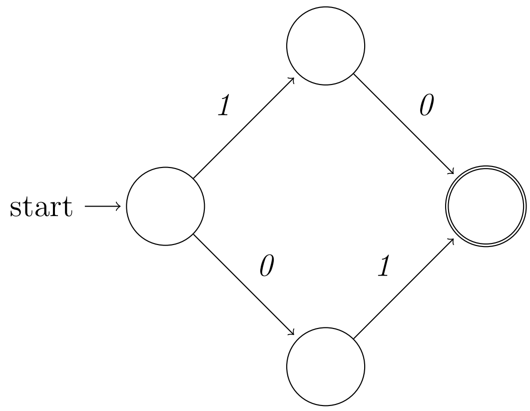
Vamos ver o exemplo da expressão 01 + 10. As imagens abaixo mostram os vários passos para a criação do autómato par a expressão. Na primeira imagem, vemos as representações dos símbolos 0 e 1 (cada um dos símbolos representado por um autóato). Na imagem seguinte, pode ver-se a construção do autómato para a expressão 01 (a construção do autómato para a expressão 10 seria equivalente), podendo ver-se como é feita a concatenação dos autómatos de duas expressões previamente convertidas.



Na figura seguinte mostra-se o resultado final da união das duas opções (10 + 01), sendo possível ver como é feita a união de dois autómatos previamente obtidos.

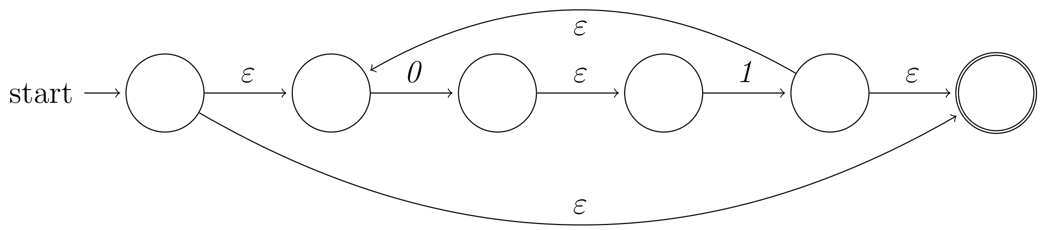


Podemos ver que esta solução tem um conjunto de transições ε excessivo e desnecessário. Podemos simplificar o autómato acima para o seguinte:

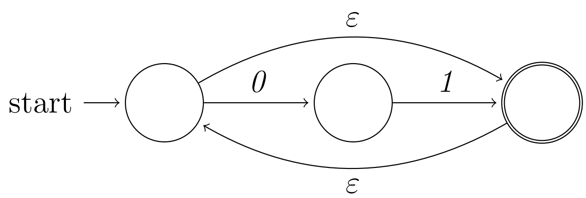


Podemos ver neste autómato que continua a aceitar as sequências 10 e 01

Vamos ver ainda um exemplo com o fecho, para a expressão (01)*. O autómato da expressão 01 é idêntico ao obtido anteriormente e mostrado acima. Ao realizar o fecho, o autómato ficará assim:



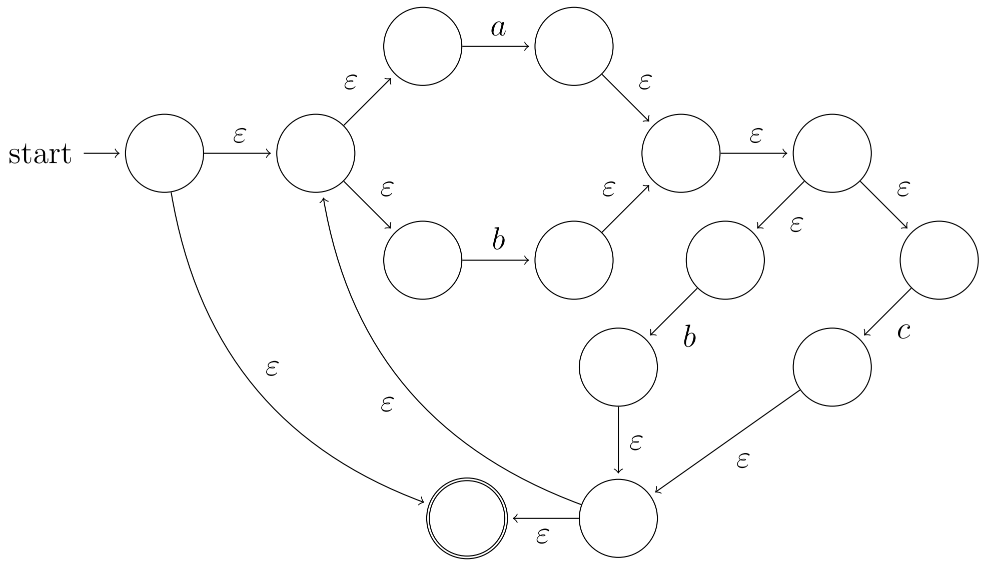
Novamente, este autómato tem ainda um conjunto de transições ε que aparentam ser desnecessárias. Podemos então simplificar o autómato, ficando com a seguinte versão:



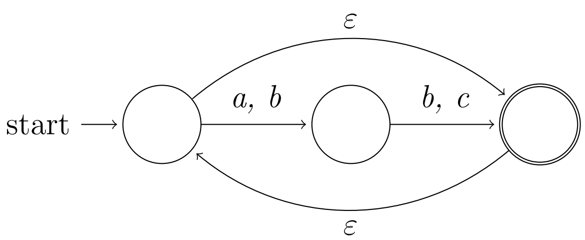
Exercício 7

Converta a expressão regular ((a+b)(b+c))* num autómato.

Para realizar a conversão, podemos optar pela utilização dos *templates*, obtendo o seguinte ε-NFA para a expressão:



Novamente, podemos ver que o autómato obtido por este método tem várias transições desnecessárias. Podemos agora tentar simplificar este autómato (ou alternativamente, tentar produzir diretamente um autómato a partir da expressão regular). Uma possível solução (mais compacta) seria:

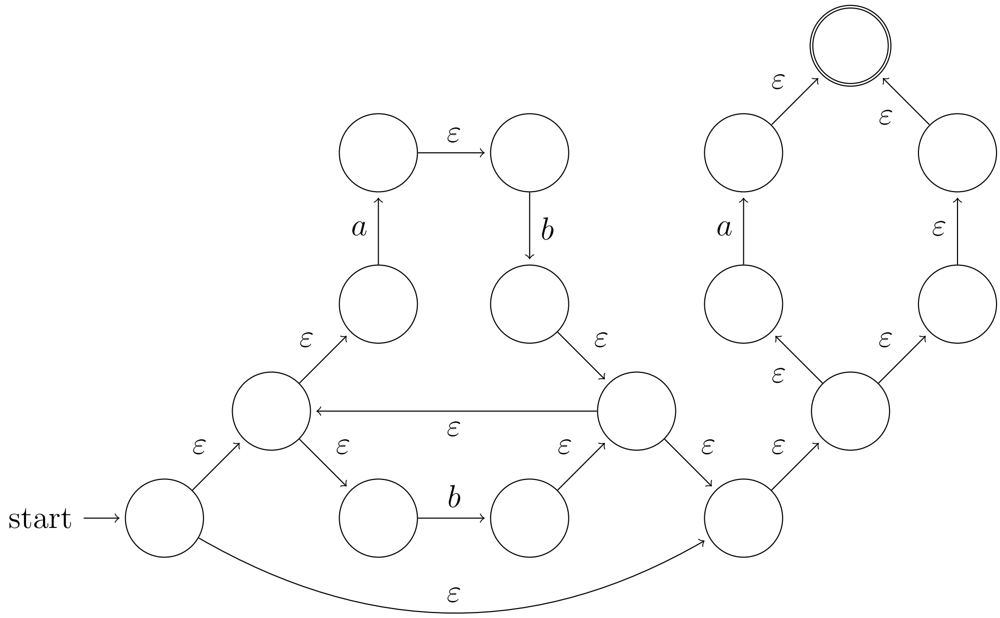


Repare-se que este autómato é estruturalmente equivalente ao visto anteriormente para a expressão (10)* dado que as duas expressões são também semelhantes.

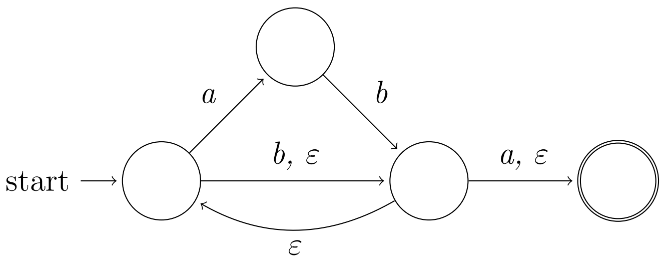
Exercício 8

Obtenha um autómato para a expressão regular (ab+b)*(a+ε).

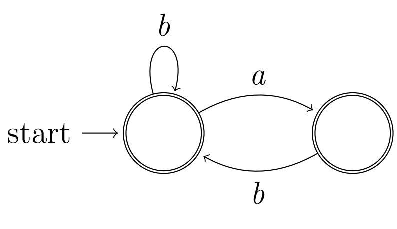
Usando novamente os *templates*, chegamos ao seguinte autómato:



Novamente, este autómato tem várias transições ε desnecessárias. Podemos tentar simplificar o autómato, produzindo uma representação mais compacta, como por exemplo:



Podíamos ainda tentar condensar mais o autómato, e representá-lo apenas com dois estados:



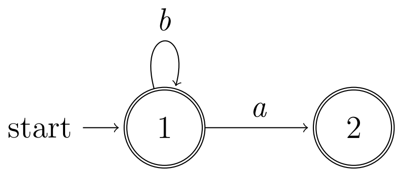
Conversão de Autómatos em Expressões Regulares

Vamos ver dois métodos para realizar a conversão em sentido contrário, denominados de construção de caminhos e eliminação de estados.

Construção de Caminhos

Para obter a expressão regular a partir de um autómato pelo método de construção de caminhos, começamos por numerar os nós do autómato de 1 até N, em que N é o número de estados do autómato. Depois, vão sendo obtidos caminhos sucessivamente mais complexos, no formato $R^k_{i,j}$, representando os caminhos possíveis do nó i até ao nó j passando no máximo pelo nó numerado como k. No final, a expressão regular para a linguagem é obtida pela união de todos os caminhos desde o estado inicial até cada um dos estados finais com k igual a N.

Vamos ver um exemplo para o seguinte autómato:



Os caminhos para $k = 0$ são obtidos vendo as ligações entre i e j que não passam por mais nenhum nó. Assim, temos os seguintes caminhos:

$R^0_{1,1} = b + \varepsilon$; $R^0_{1,2} = a$; $R^0_{2,1} = \emptyset$; $R^0_{2,2} = \varepsilon$

Note-se que existe sempre a possibilidade de transitar para o próprio nó com ε, e que quando não existe caminho direto entre nós distintos é usado o símbolo $\emptyset$ para o indicar.

Os caminhos de índice k mais elevado são obtidos usando a fórmula:

$$R^k_{i,j} = R^{k-1}_{i,j} + R^{k-1}_{i,k}(R^{k-1}_{k,k})^* R^{k-1}_{k,j}$$

Assim, podemos calcular os caminhos para $k = 1$:

Podemos agora calcular os caminhos para $k = 2$. Como apenas interessam os caminhos que levam do estado inicial a cada um dos estados finais, basta calcular $R^2_{1,1}$ e $R^2_{1,2}$, pelo que podemos evitar alguns cálculos desnecessários.

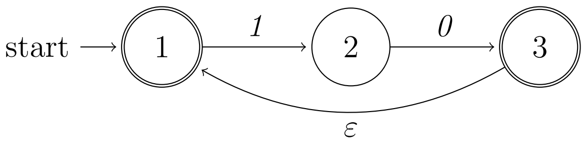
$R^1_{1,1}$	$R^0_{1,1} + R^0_{1,1}(R^0_{1,1})^*R^0_{1,1} = (b+\varepsilon) + (b+\varepsilon)(b+\varepsilon)^*(b+\varepsilon) = b^*$
$R^1_{1,2}$	$R^0_{1,2} + R^0_{1,1}(R^0_{1,1})^*R^0_{1,2} = a + (b+\varepsilon)(b+\varepsilon)^*a = b^*a$
$R^1_{2,1}$	$R^0_{2,1} + R^0_{2,1}(R^0_{1,1})^*R^0_{1,1} = \emptyset + \emptyset(b+\varepsilon)^*(b+\varepsilon) = \emptyset$
$R^1_{2,2}$	$R^0_{2,2} + R^0_{2,1}(R^0_{1,1})^*R^0_{1,2} = \varepsilon + \emptyset(b+\varepsilon)^*a = \varepsilon$

$R^2_{1,1}$	$R^1_{1,1} + R^1_{1,2}(R^1_{2,2})^*R^1_{2,1} = b^* + b^*a\varepsilon^*\emptyset = b^*$
$R^2_{1,2}$	$R^1_{1,2} + R^1_{1,2}(R^1_{2,2})^*R^1_{2,2} = b^*a + b^*a\varepsilon^*\varepsilon = b^*a$

A expressão final será a união destes dois termos, ficando b* + b*a, ou, escrito de outra forma, b*(a+ε).

Exercício 9

Use o método de construção de caminhos para encontrar uma expressão regular para a linguagem definida pelo seguinte autómato.



A tabela abaixo contém as expressões que representam os caminhos parciais para $0 \leq k \leq 2$.

A expressão final será $R^3_{1,1} + R^3_{1,3}$, pelo que, usando a fórmula, podemos desdobrar na seguintes expressão:

$R^2_{1,1} + R^2_{1,3}(R^2_{3,3})^*R^2_{3,1} \ + \ R^2_{1,3} + R^2_{1,3}(R^2_{3,3})^*R^2_{3,3}$

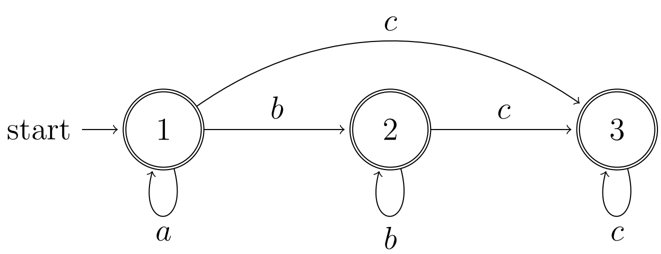
A expressão final será então $\varepsilon + 10(\varepsilon + 10)^*\varepsilon + 10 + 10(\varepsilon + 10)^*(\varepsilon + 10)$ podendo ser simplificada para (10)*.

Podemos, de forma a poupar esforço, começar pelo final, isto é, olhar para os termos de $k = 3$ que constituem a expressão regular da linguagem, e verificar queis os termos de $k = 2$ necessários, fazendo depois o mesmo para $k = 1$.

$k = 0$	$k = 1$	$k = 2$
$R^0_{1,1} = \varepsilon$	$R^1_{1,1} = \varepsilon + \varepsilon\varepsilon^*\varepsilon = \varepsilon$	$R^2_{1,1} = \varepsilon + 1\varepsilon^*\emptyset = \varepsilon$
$R^0_{1,2} = 1$	$R^1_{1,2} = 1 + \varepsilon\varepsilon^*1 = 1$	$R^2_{1,2} = 1 + 1\varepsilon^*\varepsilon = 1$
$R^0_{1,3} = \emptyset$	$R^1_{1,3} = \emptyset + \varepsilon\varepsilon^*\emptyset = \emptyset$	$R^2_{1,3} = \emptyset + 1\varepsilon^*0 = 10$
$R^0_{2,1} = \emptyset$	$R^1_{2,1} = \emptyset + \emptyset\varepsilon^*\varepsilon = \emptyset$	$R^2_{2,1} = \emptyset + \varepsilon\varepsilon^*\emptyset = \emptyset$
$R^0_{2,2} = \varepsilon$	$R^1_{2,2} = \varepsilon + \emptyset\varepsilon^*1 = \varepsilon$	$R^2_{2,2} = \varepsilon + \varepsilon\varepsilon^*\varepsilon = \varepsilon$
$R^0_{2,3} = 0$	$R^1_{2,3} = 0 + \emptyset\varepsilon^*\emptyset = 0$	$R^2_{2,3} = 0 + \varepsilon\varepsilon^*0 = 0$
$R^0_{3,1} = \varepsilon$	$R^1_{3,1} = \varepsilon + \varepsilon\varepsilon^*\varepsilon = \varepsilon$	$R^2_{3,1} = \varepsilon + 1\varepsilon^*\emptyset = \varepsilon$
$R^0_{3,2} = \emptyset$	$R^1_{3,2} = \emptyset + \varepsilon\varepsilon^*1 = 1$	$R^2_{3,2} = 1 + 1\varepsilon^*\varepsilon = 1$
$R^0_{3,3} = \varepsilon$	$R^1_{3,3} = \varepsilon + \varepsilon\varepsilon^*\emptyset = \varepsilon$	$R^2_{3,3} = \varepsilon + 1\varepsilon^*0 = \varepsilon + 10$

Exercício 10

Considere o autómato abaixo e calcule $R^3_{1,3}$



Começando pelo final, e usando a fórmula para determinar quais os termos para $k = 2$ necessários, temos então:
$R^3_{1,3} = R^2_{1,3} + R^2_{1,3}R^2_{3,3}{}^*R^2_{3,3}$

Aplicando a fórmula a cada um dos termos:
$R^2_{1,3} = R^1_{1,3} + R^1_{1,2}R^1_{2,2}{}^*R^1_{2,3}$
$R^2_{3,3} = R^1_{3,3} + R^1_{3,2}R^1_{2,2}{}^*R^1_{2,3}$

Aplicando novamente a fórmula aos termos distintos:
$R^1_{1,3} = R^0_{1,3} + R^0_{1,1}R^0_{1,1}{}^*R^0_{1,3} = c + (a + \varepsilon)(a + \varepsilon)^*c = a^*c$
$R^1_{1,2} = R^0_{1,2} + R^0_{1,1}R^0_{1,1}{}^*R^0_{1,2} = b + (a + \varepsilon)(a + \varepsilon)^*b = a^*b$

$R^1_{2,2} = R^0_{2,2} + R^0_{2,1}R^0_{1,1}{}^*R^0_{1,2} = (b+\varepsilon) + \emptyset(a+\varepsilon)^*b = b+\varepsilon$
$R^1_{2,3} = R^0_{2,3} + R^0_{2,1}R^0_{1,1}{}^*R^0_{1,3} = c + \emptyset(a+\varepsilon)^*c = c$
$R^1_{3,3} = R^0_{3,3} + R^0_{3,1}R^0_{1,1}{}^*R^0_{1,3} = (c+\varepsilon) + \emptyset(a+\varepsilon)^*c = c+\varepsilon$
$R^1_{3,2} = R^0_{3,2} + R^0_{3,1}R^0_{1,1}{}^*R^0_{1,2} = \emptyset + \emptyset(a+\varepsilon)^*b = \emptyset$

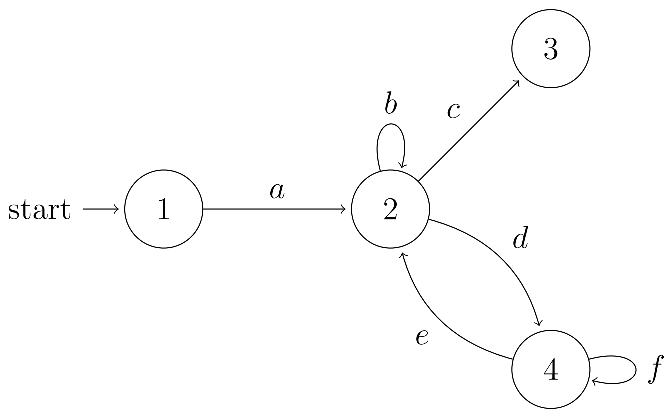
Podemos agora calcular os termos para $k = 2$:
$R^2_{1,3} = a^*c + a^*b(b+\varepsilon)^*c =$ a*b*c
$R^2_{3,3} = c + \varepsilon + \emptyset(b+\varepsilon)^*c = c+\varepsilon$

E agora podemos calcular o termo pretendido:
$R^3_{1,3} = a^*b^*c + a^*b^*c(c+\varepsilon)^*(c+\varepsilon)$ =a*b*cc*

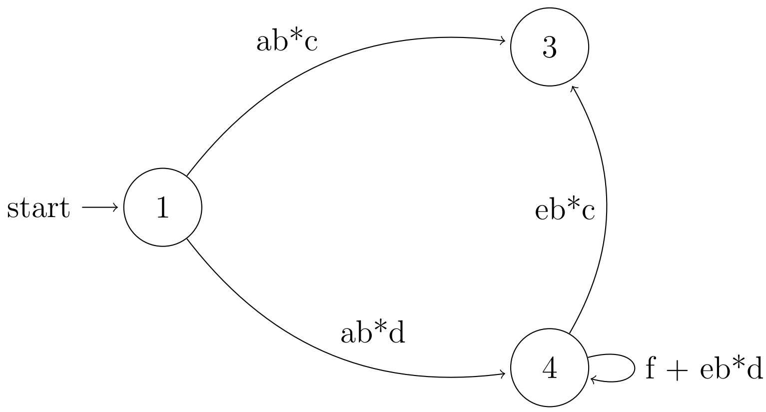
Eliminação de Estados

O segundo método consiste em eliminar sucessivamente estados não finais do autómato, substituindo-os pelas ligações equivalentes, até ficar apenas com o estado inicial e estados finais. A expressão regular da linguagem é obtida pela união das expressões que levam do estado inicial a cada um dos estados finais.

Vamos ver um exemplo de eliminação de um estado, no seguinte autómato:

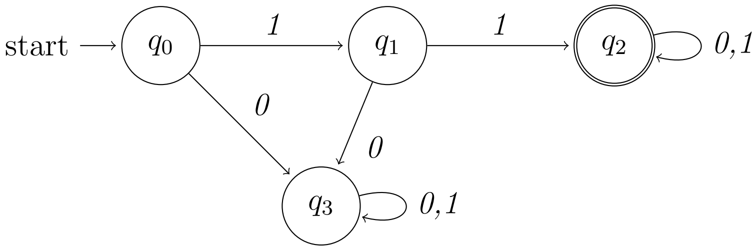


Se escolhermos eliminar o estado 2, será necessário inserir novas ligações que substituam todas aquelas em que o estado 2 participa. Então, vamos ver todas as ligações de entrada no estado 2, e todas as ligações de saída do estado 2, acrescentando depois novas ligações entre cada par de estados (entrada, saída). Temos entradas a partir dos estados 1 e 4, e saídas para os estados 3 e 4, pelo que vamos ter ligações entre os pares de estados (1,3), (1,4), (4,3) e (4,4). A expressão em cada nova ligação é obtida concatenando a expressão do arco que ligava ao estado a ser eliminado com o fecho do *loop* do estado a ser eliminado com a expressão do arco que liga ao arco de destino. Eliminando o estado 2, ficamos então com o seguinte autómato:

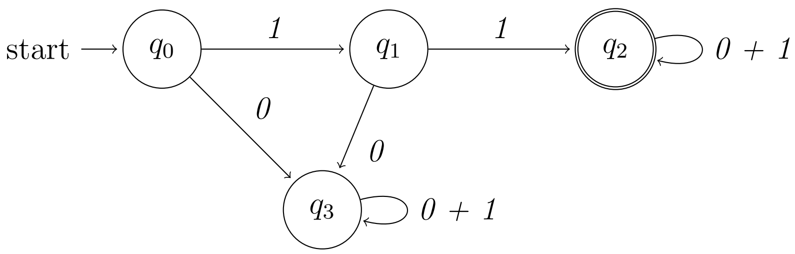


Exercício 11

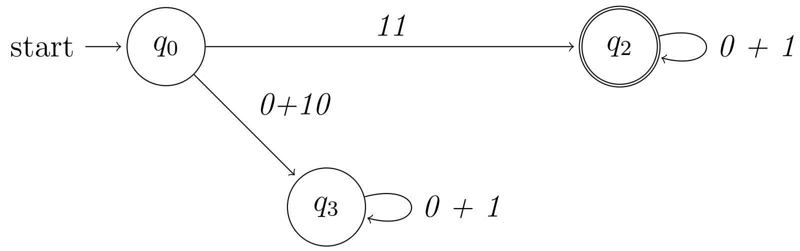
Use o método de eliminação de estados para obter uma expressão regular para a linguagem representada pelo seguinte autómato:



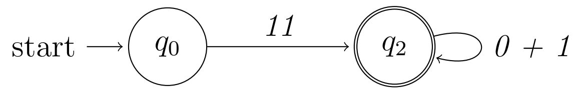
Temos de começar por substituir as transições nos arcos por expressões regulares equivalentes:



Vamos começar por eliminar q_1. Temos uma transição de entrada, vinda de q_0, e duas de saída, dirigidas a q_2 e q_3. Ao eliminar q_1, ficamos então com o seguinte autómato:



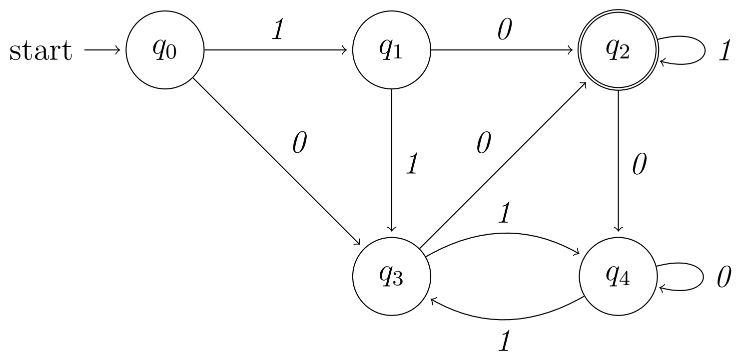
Podemos eliminar q_3 sem necessitar de adicionar qualquer nova transição, uma vez que não existem arcos de saída de q_3.



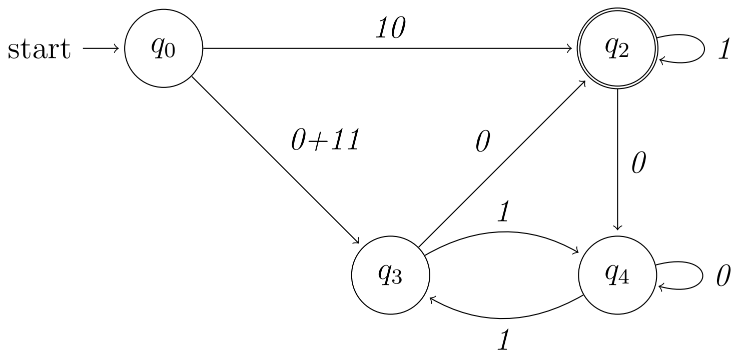
A expressão regular da linguagem será então 11(0+1)*.

Exercício 12

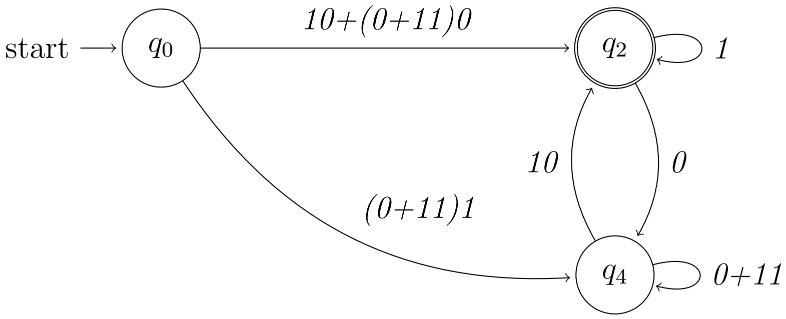
Use o método de eliminação de estados para obter uma expressão regular para a linguagem representada pelo seguinte autómato:



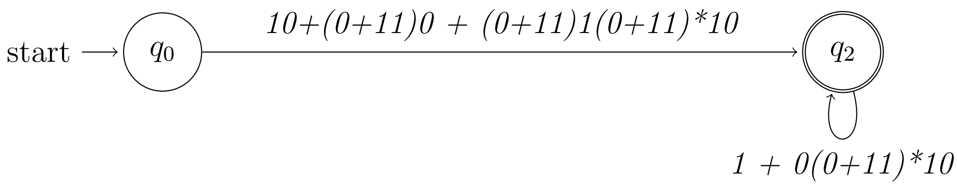
Olhando para este autómato, podemos ver que temos 3 estados a eliminar: q_1, q_3 e q_4. Para eliminar q_1, vamos precisar de duas novas ligações (temos uma de entrada, vinda de q_0, e duas de saída, para q_2 e q_3), ficando assim com o seguinte autómato:



Para eliminar q_3, olhamos para as transições de entrada (q_1 e q_4) e para as transições de saída (q_2 e q_4), ficando assim com 4 novas transições.



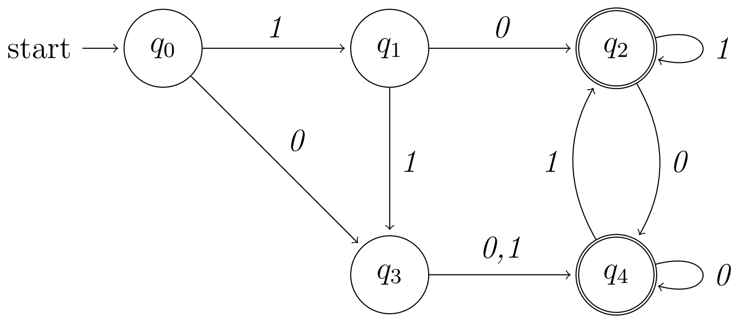
Finalmente, para eliminar q_4, é necessário ver as transições de entrada (q_0 e q_2) e as transições de saída (q_2), ficando assim com duas novas transições.



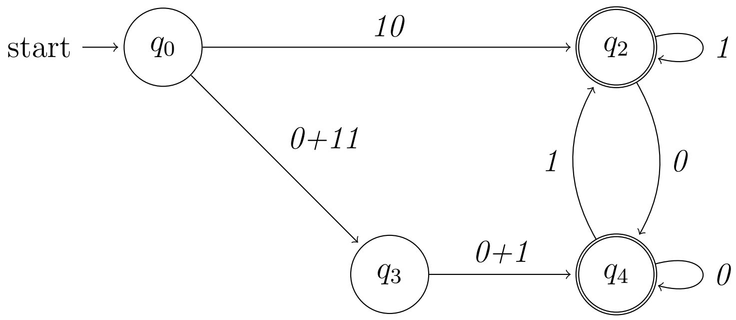
A expressão final será então (10+(0+11)0+(0+11)1(0+11)*10) (1+0(0+11)*10)*.

Exercício 13

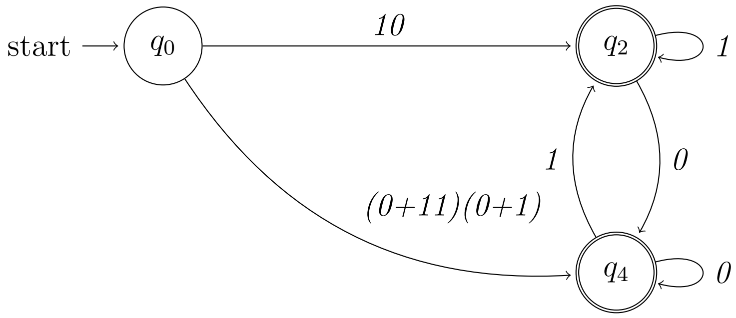
Use o método de eliminação de estados para obter uma expressão regular para a linguagem representada pelo seguinte autómato:



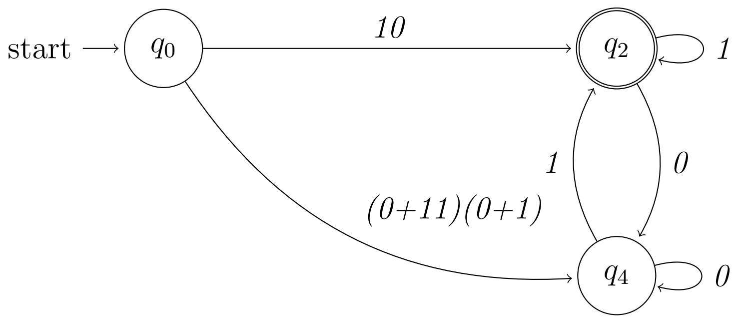
Olhando para o autómato, temos então dois estados a eliminar, q_1 e q_3. Começando por eliminar q_1, ficamos com o seguinte autómato:



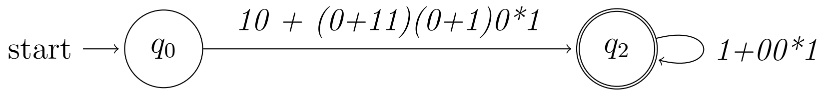
Eliminando agora q_3, ficamos com o seguinte autómato:



Neste momento, estão eliminados todos os estados exceto o estado inicial e estados finais. Para obter a expressão final será necessário reunir as expressões que levam do estado inicial a cada um dos estados finais, pelo que se torna necessário dividir o autómato em dois 'ramos', um para processar cada um dos estados finais. Ficamos assim por um lado com q_2 como estado final:

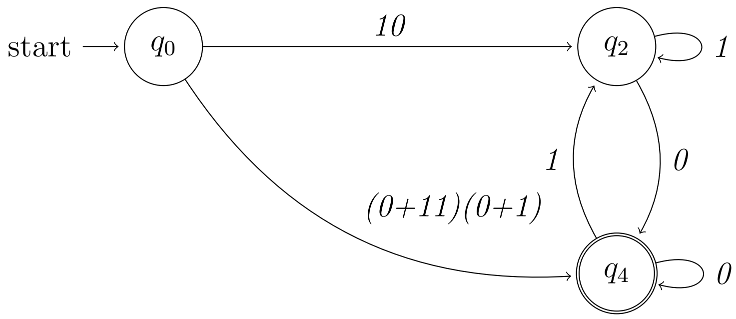


Neste caso, temos de eliminar q_4, ficando assim com o seguinte autómato:

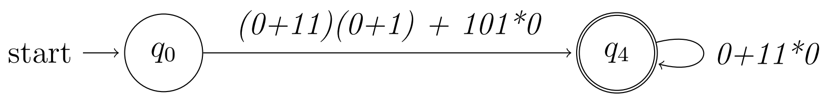


A expressão regular será assim (10 + (0+11)(0+1)0*1) (1+00*1)*.

Por outro lado, ficamos com q_4 como estado final:



Eliminando q_2 ficamos com o seguinte autómato:



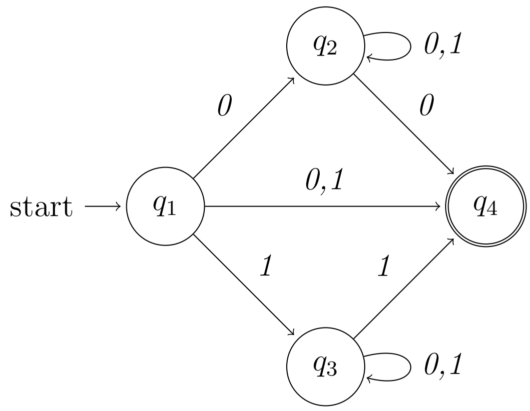
A expressão regular será assim ((0+11)(0+1) + 101*0) (0+11*0)*.

Quando se combinam as duas expressões parciais , obtém-se a expressão para a linguagem:

(10 + (0+11)(0+1)0*1) (1+00*1)* + ((0+11)(0+1) + 101*0) (0+11*0)*

Exercício 14 (TPC 2013/14)

Obtenha expressões regulares para a linguagem definida pelo seguinte autómato, usando o método de construção de caminhos, e o método de eliminação de estados.



Começando pelo método de construção de caminhos, aproveitando que os estados estão já numerados a partir de 1, e dado que temos apenas um estado final, a expressão final desejada será $R^4_{1,4}$, que podemos ir desenvolvendo para ver quais os termos necessários:

$R^4_{1,4} = R^3_{1,4} + R^3_{1,4}(R^3_{4,4})^* R^3_{4,4}$

Desenvolvendo cada um dos termos de ordem 3:

$R^3_{1,4} = R^2_{1,4} + R^2_{1,3}(R^2_{3,3})^* R^2_{3,4}$
$R^3_{4,4} = R^2_{4,4} + R^2_{4,3}(R^2_{3,3})^* R^2_{3,4}$

Desenvolvendo agora cada um dos termos de ordem 2:

$R^2_{1,3} = R^1_{1,3} + R^1_{1,2}(R^1_{2,2})^* R^1_{2,3}$
$R^2_{1,4} = R^1_{1,4} + R^1_{1,2}(R^1_{2,2})^* R^1_{2,4}$
$R^2_{3,3} = R^1_{3,3} + R^1_{3,2}(R^1_{2,2})^* R^1_{2,3}$
$R^2_{3,4} = R^1_{3,4} + R^1_{3,2}(R^1_{2,2})^* R^1_{2,4}$
$R^2_{4,3} = R^1_{4,3} + R^1_{4,2}(R^1_{2,2})^* R^1_{2,3}$
$R^2_{4,4} = R^1_{4,4} + R^1_{4,2}(R^1_{2,2})^* R^1_{2,4}$

Vendo agora os termos de ordem 1:

$R^1_{1,2} = R^0_{1,2} + R^0_{1,1}(R^0_{1,1})^* R^0_{1,2} = 0 + \varepsilon(\varepsilon)^* 0 = 0$
$R^1_{1,3} = R^0_{1,3} + R^0_{1,1}(R^0_{1,1})^* R^0_{1,3} = 1 + \varepsilon(\varepsilon)^* 1 = 1$
$R^1_{1,4} = R^0_{1,4} + R^0_{1,1}(R^0_{1,1})^* R^0_{1,4} = 0 + 1 + \varepsilon(\varepsilon)^*(0 + 1) = 0 + 1$
$R^1_{2,2} = R^0_{2,2} + R^0_{2,1}(R^0_{1,1})^* R^0_{1,2} = \varepsilon + 0 + 1 + \emptyset(\varepsilon)^* 0 = \varepsilon + 0 + 1$
$R^1_{2,3} = R^0_{2,3} + R^0_{2,1}(R^0_{1,1})^* R^0_{1,3} = \emptyset + \emptyset(\varepsilon)^* 1 = \emptyset$

$R^1_{2,4} = R^0_{2,4} + R^0_{2,1}(R^0_{1,1})^*R^0_{1,4} = 0 + \emptyset(\varepsilon)^*(0+1) = 0$
$R^1_{3,2} = R^0_{3,2} + R^0_{3,1}(R^0_{1,1})^*R^0_{1,2} = \emptyset + \emptyset(\varepsilon)^*0 = \emptyset$
$R^1_{3,3} = R^0_{3,3} + R^0_{3,1}(R^0_{1,1})^*R^0_{1,3} = \varepsilon + 0 + 1 + \emptyset(\varepsilon)^*1 = 0 + 1$
$R^1_{3,4} = R^0_{3,4} + R^0_{3,1}(R^0_{1,1})^*R^0_{1,4} = 1 + \emptyset(\varepsilon)^*(0+1) = 1$
$R^1_{4,2} = R^0_{4,2} + R^0_{4,1}(R^0_{1,1})^*R^0_{1,2} = \emptyset + \emptyset(\varepsilon)^*0 = \emptyset$
$R^1_{4,3} = R^0_{4,3} + R^0_{4,1}(R^0_{1,1})^*R^0_{1,3} = \emptyset + \emptyset(\varepsilon)^*1 = \emptyset$
$R^1_{4,4} = R^0_{4,4} + R^0_{4,1}(R^0_{1,1})^*R^0_{1,4} = \varepsilon + \emptyset(\varepsilon)^*(0+1) = \varepsilon$

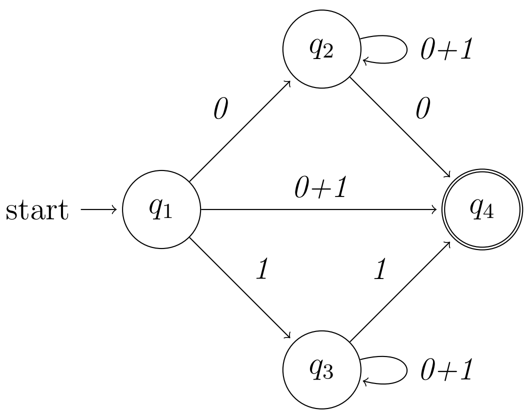
Podemos agora calcular as expressões para os termos de ordem 2:
$R^2_{1,3} = R^1_{1,3} + R^1_{1,2}(R^1_{2,2})^*R^1_{2,3} = 1 + 0(\varepsilon + 0 + 1)^*\emptyset = 1$
$R^2_{1,4} = R^1_{1,4} + R^1_{1,2}(R^1_{2,2})^*R^1_{2,4} = 0 + 1 + 0(\varepsilon + 0 + 1)^*0 = 0 + 1 + 0(0+1)^*0$
$R^2_{3,3} = R^1_{3,3} + R^1_{3,2}(R^1_{2,2})^*R^1_{2,3} = 0 + 1 + \emptyset(\varepsilon + 0 + 1)^*\emptyset = 0 + 1$
$R^2_{3,4} = R^1_{3,4} + R^1_{3,2}(R^1_{2,2})^*R^1_{2,4} = 1 + \emptyset(\varepsilon + 0 + 1)^*0 = 1$
$R^2_{4,3} = R^1_{4,3} + R^1_{4,2}(R^1_{2,2})^*R^1_{2,3} = \emptyset + \emptyset(\varepsilon + 0 + 1)^*\emptyset = \emptyset$
$R^2_{4,4} = R^1_{4,4} + R^1_{4,2}(R^1_{2,2})^*R^1_{2,4} = \varepsilon + \emptyset(\varepsilon + 0 + 1)^*0 = \varepsilon$

Calculamos agora as expressões para os termos de ordem 3:
$R^3_{1,4} = R^2_{1,4} + R^2_{1,3}(R^2_{3,3})^*R^2_{3,4} = 0 + 1 + 0(0+1)^*0 + 1(0+1)^*1$
$R^3_{4,4} = R^2_{4,4} + R^2_{4,3}(R^2_{3,3})^*R^2_{3,4} = \varepsilon + \emptyset(0+1)^*1 = \varepsilon$

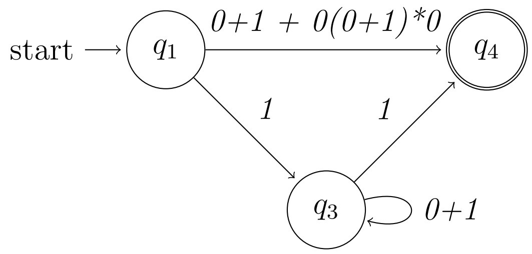
E finalmente podemos calcular a expressão final:
$R^4_{1,4} = R^3_{1,4} + R^3_{1,4}(R^3_{4,4})^*R^3_{4,4} = 0 + 1 + 0(0+1)^*0 + 1(0+1)^*1 + (0 + 1 + 0(0+1)^*0 + 1(0+1)^*1)(\varepsilon)^*\varepsilon = 0 + 1 + 0(0+1)^*0 + 1(0+1)^*1$

E temos então a expressão final para a linguagem definida pelo autómato.

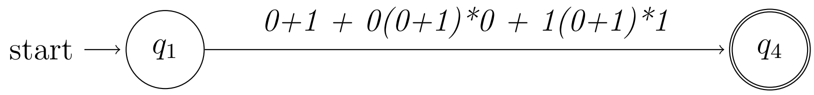
Usando agora o método de eliminação de estados, começamos por transformar as etiquetas das transições em expressões regulares:



De seguida, escolhemos um dos estados q_2 ou q_3 para eliminar primeiro. Eliminando q_2:



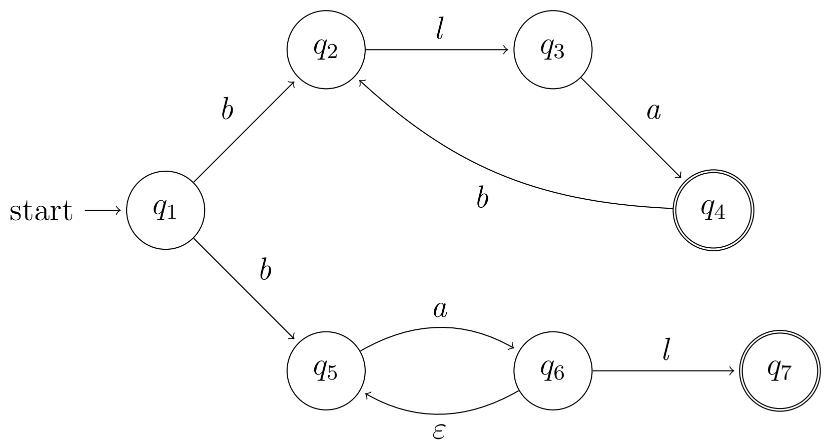
Eliminando agora o estado q_3:



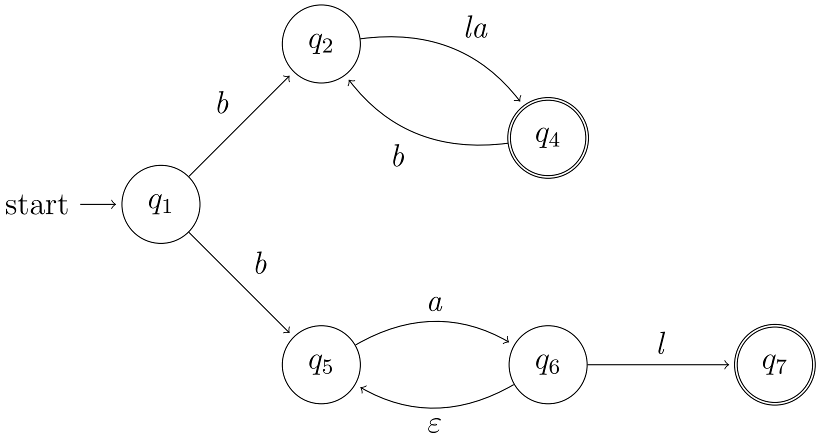
E chegamos então à expressão que nos define a linguagem:
0 + 1 + 0(0+1)*0 + 1(0+1)*1
Como podemos ver, ambos os métodos resultam na mesma expressão, embora o método de eliminação de estados o permita fazer com menos trabalho.

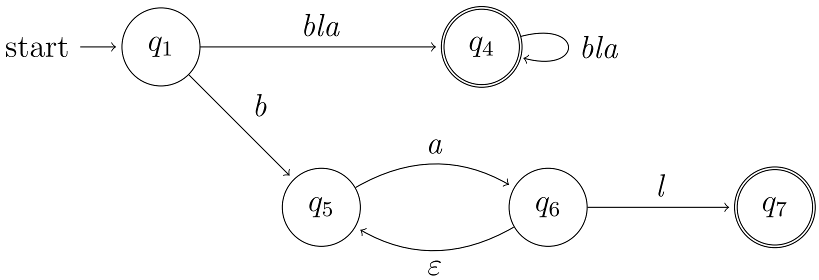
Exercício 15 (Desafio 2014/15)

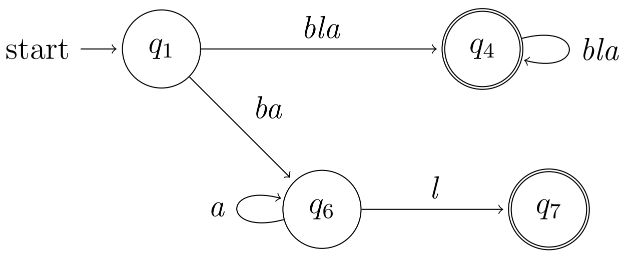
Converta o seguinte autómato (sobre o alfabeto {a, b, l}) na expressão regular equivalente, usando o método de eliminação de estados. Mostre todos os passos intermédios na sua solução.

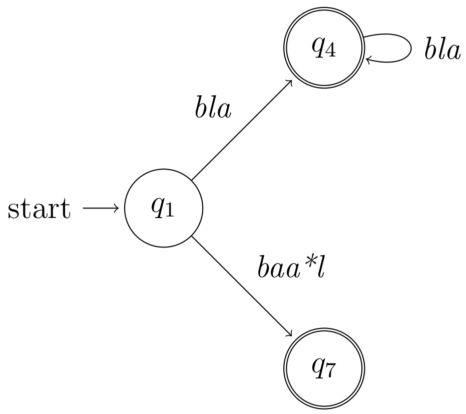


Como todas as transições têm apenas um símbolo, podemos começar já com a eliminação dos estados. Eliminando os estados q_3, q_2, q_5 e q_6 (por esta ordem), temos então a seguinte sequência de passos:

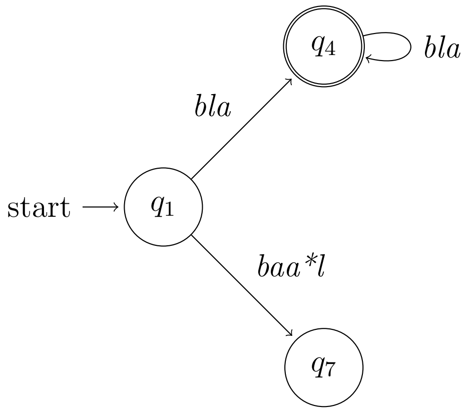


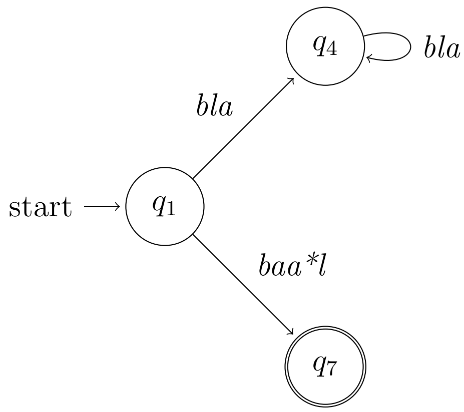


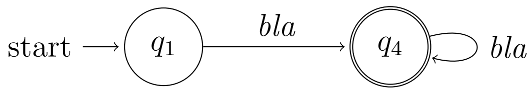


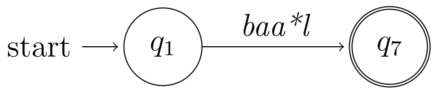


Dividindo agora o autómato nos dois estados de aceitação, temos:









Ficamos assim com a expressão bla(bla)* para o primeiro caso e baa*l para o segundo caso. Unindo as duas expressões, ficamos então com a seguinte expressão para a linguagem: bla(bla)* + baa*l

Leis Algébricas

As expressões regulares, enquanto representação algébrica de linguagens regulares, permitem ser-lhes aplicado um conjunto de leis algébricas, as quais podem ser úteis na simplificação de expressões, ou para verificar se duas expressões regulares representam a mesma linguagem.

Exercício 16

Indique se as expressões (0+11*0)* e (1*0)* representam a mesma linguagem.

Para verificar se as expressões representam a mesma linguagem, podemos tentar transformar uma na outra. Assim, começando pela primeira expressão, e aplicando sucessivamente leis algébricas das expressões regulares:

(0+11*0)*	
(ε0+11*0)*	(identidade da concatenação)
((ε+11*)0)*	(distributividade da concatenação)
(1*0)*	ε + 11* = 1*

Assim, podemos afirmar que as duas expressões representam a mesma linguagem.

Exercício 17

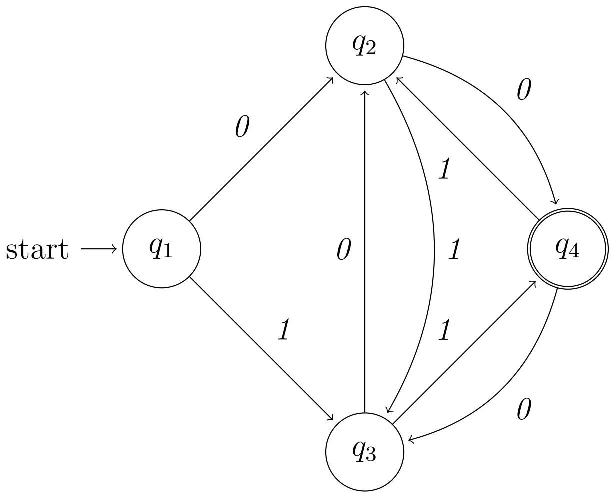
Simplifique a seguinte expressão regular:
(a+b+ε)* + ((a*)* + (b*)*)*

(a+b+ε)* + ((a*)* + (b*)*)*	
(a+b)* + ((a*)* + (b*)*)*	(L+ε)* = L*
(a+b)* + (a* + b*)*	(L*)* = L*

Temos agora uma expressão que pode ainda ser simplificada. Olhando para a parte da expressão (a* + b*)*, podemos tentar reescrever de uma forma simplificada: (a+b)* Vamos então tentar ver que cadeias fazem parte desta linguagem: da parte esquerda, são cadeias constituídas por 0 ou mais repetições de sequências de a's ou sequências de b's de comprimento maior ou igual a zero, ou seja, todas as cadeias sobre o alfabeto {a,b}. Do lado direito, são cadeias constituídas por 0 ou mais repetições de a ou de b, ou seja, novamente todas as cadeias sobre o alfabeto. Então, temos uma equivalência, pelo que podemos reescrever a expressão como (a+b)* + (a+b)* , a qual pode ser reescrita como (a+b)* (idempotência da união).

Exercício Proposto 1 Escreva uma expressão regular para reconhecer cadeias sobre o alfabeto {a,b} em que não existem subcadeias de 2 a's seguidas de 2 b's. Por exemplo, a cadeia abba pertence à linguagem, mas a cadeia baaabba já não pertence.

Exercício Proposto 2 Obtenha uma expressão regular para a linguagem representada pelo seguinte autómato.


start
q1
q2
q3
q4
0
1
0
1
1
1
0
0