Capítulo 6

Propriedades das Linguagens Regulares

Neste capítulo exploram-se algumas propriedades das lingugens regulares.

6.1 Propriedades de Fecho

Referem-se aqui um conjunto de operações que quando aplicadas sobre linguagens regulares preservam a regularidade da linguagem resultado.

Alguns exemplos dessas operações são a Concatenação, Fecho, União, Intersecção, Diferença, Complemento, Reverso, Homomorfismo e Homomorfismo Inverso.

Para os operadores de Complemento e Reverso, as operações podem ser feitas diretamente no DFA da linguagem:

- Para o complemento, basta trocar todos os estados de aceitação para não-aceitação e vice-versa, passando assim o autómato a aceitar todas as cadeias que antes rejeitava. Note que esta operação tem de ser realizada no DFA completo (de forma a que o estado morto, se existir, passe a ser um estado de aceitação do novo autómato).

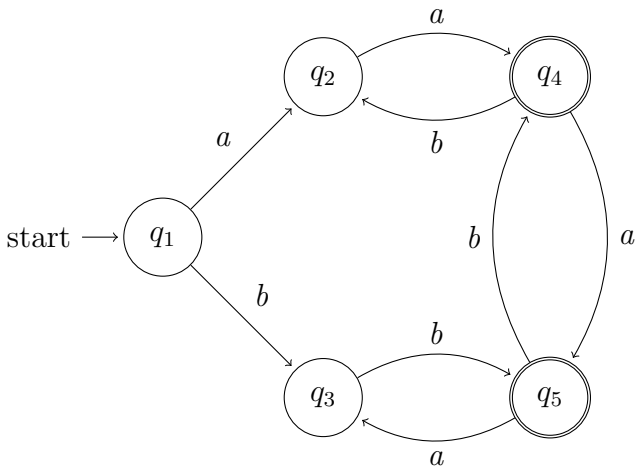
- Para o reverso, deve-se inverter a direção de todas as setas de transições, acrescentar um novo estado, que será o estado inicial do novo auómato, com transições ε para os estados de aceitação do autómato original (os quais devem deixar de ser estados de aceitação), e tornando o estado inicial do autómato original como único estado de aceitação do novo autómato.

Para os operadores de União, Interseção e Diferença, as operações podem ser feitas a partir dos DFAs das linguagens originais, fazendo o produto

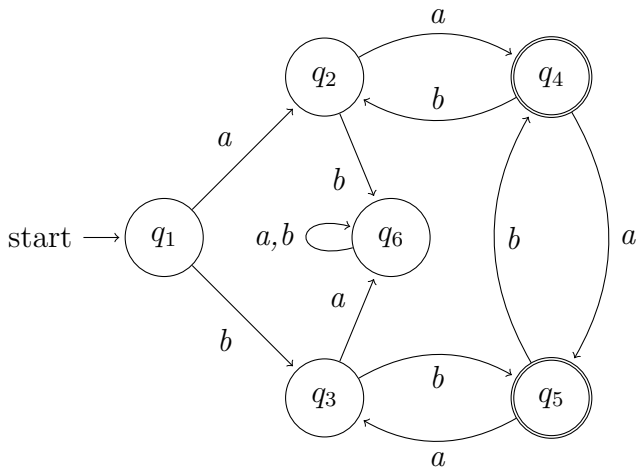
dos dois autómatos, e verificando quais os estados de aceitação para o novo autómato. No caso da união, os estados de aceitação devem incluir os estados de aceitação tanto de um autómato como de outro; para a interseção, apenas aqueles que incluam estados de aceitação de ambos os autómatos originais; para a diferença, apenas os estados que sejam de aceitação para o autómato da primeira linguagem mas não para o da segunda linguagem.

Exercício 1

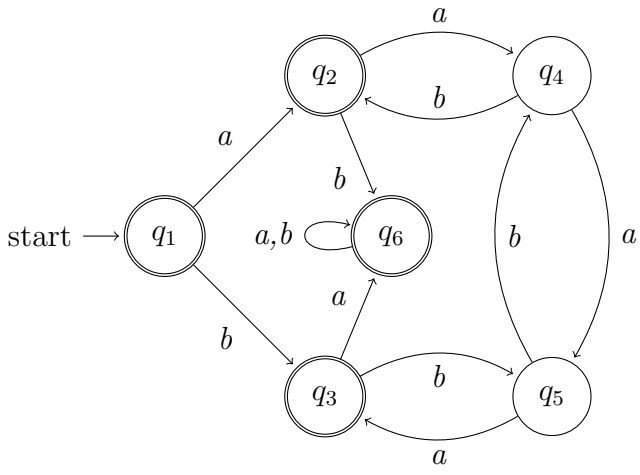
Obtenha o autómato para o complemento da linguagem representada pelo seguinte autómato.



Para obter o complemento da linguagem, é necessário estar a trabalhar sobre o DFA completo, para garantir que qualquer 'estado morto', caso exista, está presente no autómato (uma vez que este será um estado de aceitação no autómato para o complemento da linguagem). Assim, acrescentamos o estado morto, para ficarmos com o DFA completo.



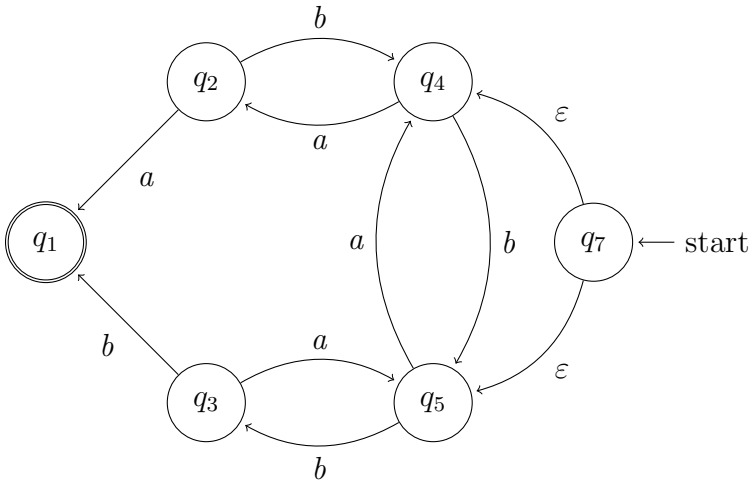
Agora, necessitamos apenas de trocar a aceitação dos estados, fazendo com que as cadeias aceites pela linguagem original deixem de o ser, e todas as cadeias não aceites pela linguagem original passem a ser aceites. Ficamos então com o seguinte DFA:



Exercício 2

Obtenha o autómato para o reverso da linguagem representada pelo autómato do exercício anterior.

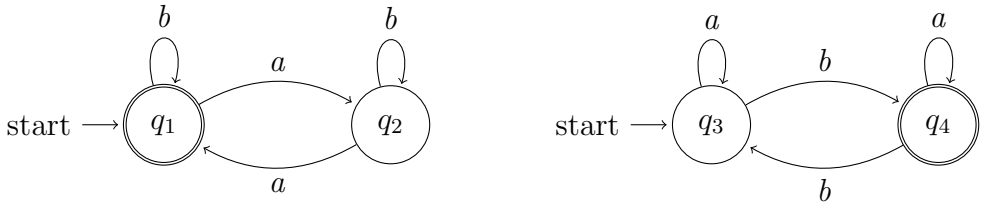
Para obter uma solução, necessitamos então de realizar 3 alterações ao autómato original: acrescentar um novo estado inicial com transições ε para cada um dos estados finais originais; fazer com que o estado inicial do autómato original passe a ser o único estado de aceitação do novo autómato; e inverter o sentido de todas as transições. Note-se que para obter o reverso não é necessário trabalhar com o DFA completo – uma vez que o estado morto do DFA original só tem transições de entrada e nenhuma de saída, invertendo o sentido das transições ficaríamos com um estado só com transições de saída mas nenhuma de entrada, ou seja, um estado não atingível. Fazendo as alterações necessárias, ficamos então com o seguinte autómato como resultado:



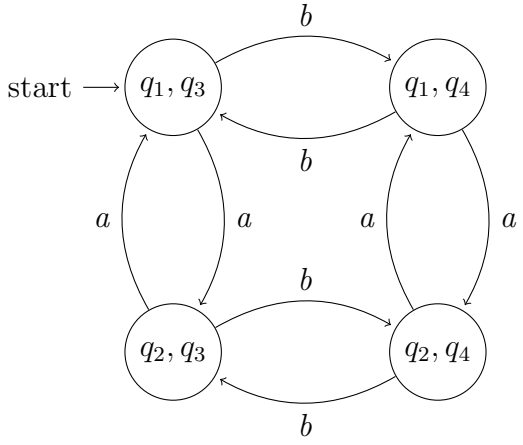
Exercício 3

Considere as linguagens A e B sobre o alfabeto {a,b}, em que A representa cadeias com número par de a's, e B representa cadeias com número ímpar e b's. Obtenha autómatos para as linguagens $C = A \cup B$, $D = A \cap B$, $E = A - B$ e $F = B - A$.

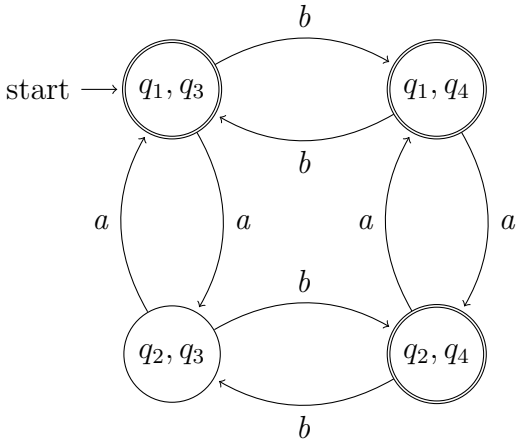
Vamos começar por representar cada um dos DFAs para as linguagens A e B. Ficamos então com os seguintes dois autómatos:



Ao fazer o produto destes dois autómatos, ficamos com um autómato com os estados compostos por um estado de cada um dos autómatos originais, sendo as transições calculadas com base nos estados de destino de cada um dos autómatos. O estado inicial será o estado composto pelos estados iniciais de cada um dos autómatos. Assim, ficamos com o seguinte autómato produto:

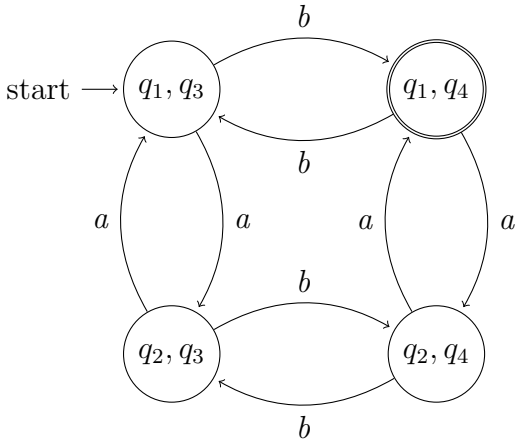


Para obter cada uma das linguagens pretendidas, será apenas necessário indicar quais os estados finais em cada caso. No caso de $C = A \cup B$, os estados finais serão todos aqueles que incluam estados que eram finais em A ou em B, ou seja, todos os estados que contenham q_1 (número par de a's) ou q_4 (número ímpar de b's). Assim, o autómato para C será:

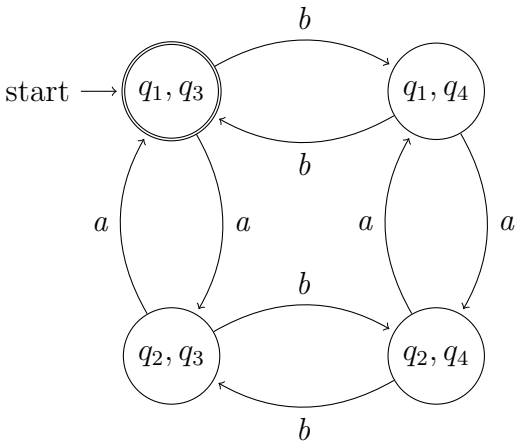


Repare-se que o único estado não final é aquele composto por q_2 e q_3, ou seja, o estado que representa cadeias em que o número de a's é ímpar e o número de b's é par. Os outros estados, representando número par de a's e/ou número ímpar de b's são todos estados de aceitação.

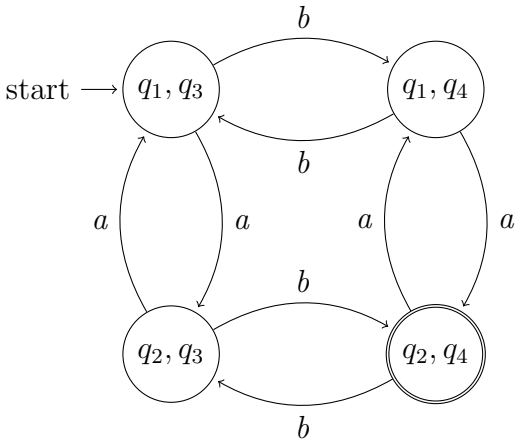
Para $D = A \cap B$, será necessário ter como estados finais apenas aqueles compostos por estados finais tanto em A como em B. Neste caso, será o estado composto por q_1 e q_4, resultando no seguinte autómato:



Para a linguagem $E = A - B$, será necessário que os estados finais sejam aqueles que são finais em A mas não o sejam em B. Neste caso, serão finais os estados que contenham q_1, mas não q_4 (ou seja, estados em que o número de a's seja par, mas garantindo não aceitar cadeias quando o número de b's é ímpar), resultando no seguinte autómato:



Finalmente, para a linguagem $F = B - A$, será necessário fazer algo semelhante ao que foi feito para E, mas em que neste caso os estados finais serão aqueles que contêm q_4 mas não q_1:

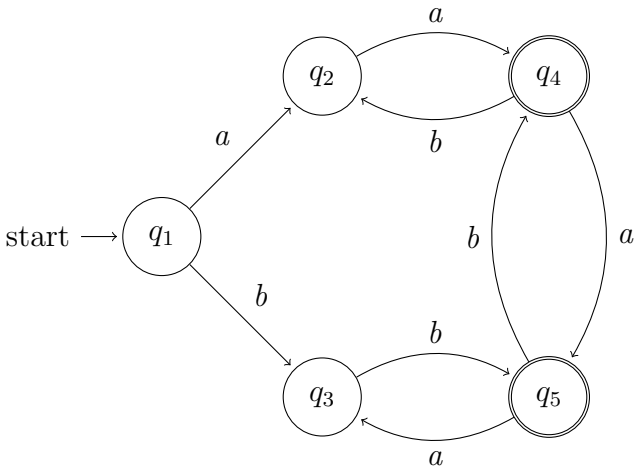


6.2 Propriedades de Decisão

Referem-se aqui algumas questões que podem ser colocadas relativamente a linguagens regulares, nomeadamente determinar se uma dada linguagem é vazia, determinar se uma cadeia pertence a uma linguagem, e determinar se duas representações de linguagens correspondem à mesma linguagem (para esta última, vamos olhar para a possibilidade de minimização de autómatos).

Exercício 1

Indique, das seguintes cadeias, quais pertencem à linguagem representada pelo autómato abaixo: abba, aaba, baba, abab, bbab, baab, baaa, aaab, bbba. Indique ainda se a linguagem representada é vazia.



Para verificar quais as cadeias que pertencem à linguagem, basta verificar quais são aceites pelo autómato, simulando o comportamento deste à medida que processa a cadeia de entrada. Neste caso, são aceites as cadeias aaba, bbab, aaab e bbba. As cadeias abba, baba, abab, baab e baab não são aceites pelo autómato, e portanto não pertencem à linguagem. Podemos ver, por exemplo, que o *percurso* do autómato para a cadeia aaba será $q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_4 \xrightarrow{b} q_2 \xrightarrow{a} q_4$; sendo q_4 um estado final, a cadeia é aceite. Já para a cadeia abba, depois de transitar de q_1 para q_2 com o primeiro a, o autómato 'morre' pois não existe transição possível de q_2 com b.

Para a linguagem ser vazia, não poderiam existir no autómato estados finais alcançáveis a partir do estado inicial. Neste caso, existem estados finais alcançáveis a partir do estado inicial, pelo que a linguagem não é vazia (como, aliás, se verificou pelo facto de existirem cadeias pertencentes à linguagem).

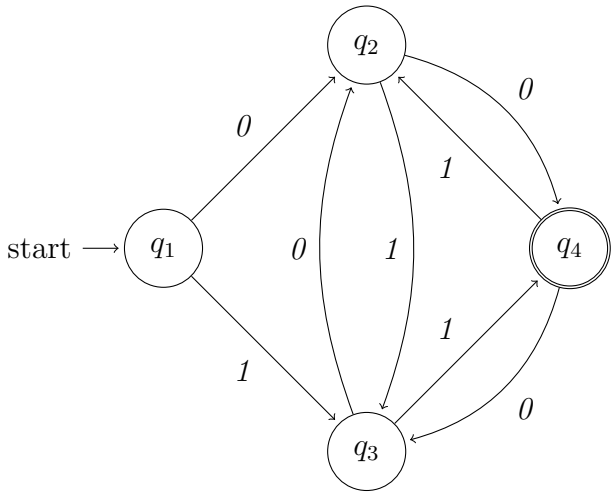
6.2.1 Minimização de Autómatos

A minimização de autómatos serve não só para encontrar uma representação mais compacta para determinada linguagem / autómato, mas pode também

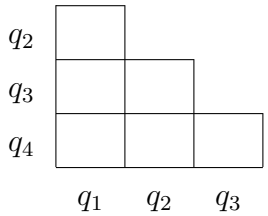
ser usada para determinar se duas representações correspondem à mesma linguagem, verificando se os dois autómatos minimizados são estruturalmente equivalentes (caso o sejam, correspondem à mesma linguagem).

A minimização de autómatos é feita recorrendo a uma tabela triangular de equivalências de estados, a qual é preenchida iterativamente com condições de equivalência, e impossibilidades de equivalência de estados.

Vamos ver um exemplo de construção desta tabela para o seguinte autómato:



Começa-se por desenhar uma tabela triangular como apresentada abaixo, em que quer num eixo quer no outro se representam os estados. As células representam a equivalência (ou não) entre os estados cuja interseção resulta nessa célula. Por exemplo, a célula que se encontra na interseção entre q_2 e q_3 vai conter informação sobre equivalência entre esses dois estados. Repare-se que não é representada a linha diagonal da tabela, e portanto no eixo vertical começamos pelo segundo estado, e no eixo horizontal não representamos o último estado.



O próximo passo será assinalar na tabela quais os estados que à partida não são equivalentes, que são as intereseções entre um estado que seja de aceitação e um estado que não seja de aceitação. Neste caso, temos apenas um estado de aceitação, q_4, e marcamos então com um X todas as células

que resultam da interseção de q_4 com qualquer um dos outros 3 estados do autómato:

q_2			
q_3			
q_4	X	X	X
	q_1	q_2	q_3

De sequida, preenchem-se as células ainda vazias com as condições necessárias para que os estados sejam equivalentes. Estas condições são a equivalência dos estados de destino de cada um dos estados para cada um dos símbolos da linguagem. Neste caso, temos apenas os símbolos 0 e 1, pelo que existem apenas duas condições em cada célula. Por exemplo, para a célula que resulta da interseção de q_1 com q_2, temos a condição de $q_2 = q_4$ uma vez que q_1 transita para q_2 com 0, e q_2 transita para q_4 com 0. Depois de preencher as células vazias com estas condições, ficamos então com a seguinte tabela:

q_2	$q_2 = q_4$ $q_3 = q_3$		
q_3	$q_2 = q_2$ $q_3 = q_4$	$q_4 = q_2$ $q_3 = q_4$	
q_4	X	X	X
	q_1	q_2	q_3

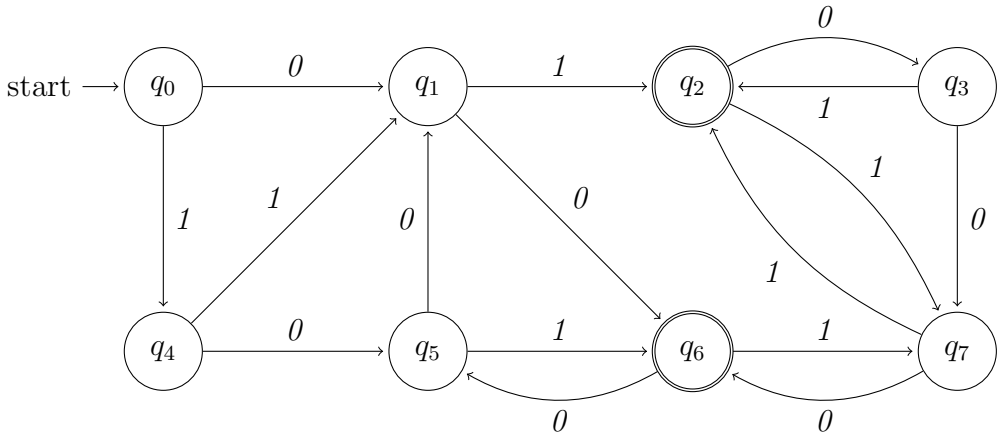
O próximo passo é ir verificando, de uma forma sistemática (por exemplo, linha por linha ou coluna por coluna, repetindo o processo até não haver mais alterações), quais os estados com pelo menos uma condição de equivalência falsa, o que leva a que os estados não possam ser considerados equivalentes, marcando então a célula com um X. Neste caso, logo na primeira iteração, temos condições em todas as 3 células que envolvem q_4, pelo que nenhum par de estados é equivalente, ficando então com a seguinte tabela:

Podemos então concluir que o autómato apresentado é já o autómato mínimo para a linguagem que representa, não podendo ser simplificado.

q_2	X		
q_3	X	X	
q_4	X	X	X
	q_1	q_2	q_3

Exercício 2 (Desafio 2014/15)

Obtenha o DFA mínimo para a linguagem representada pelo seguinte autómato.



Para obter o DFA mínimo, usamos a tabela de equivalência de estados. Na tabela abaixo, temos já marcados com um 'X' as células onde se intersetam um estado final com um estado não-final. Neste caso, tendo q_2 e q_6 como estados finais, serão marcadas todas as células de interseção entre um destes estados e um dos estados não finais.

O próximo passo será então a colocação das condições de equivalência necessárias para que cada par de estados possa ser considerado equivalente. Tendo dois símbolos na linguagem (0 e 1), cada célula terá duas condições (correspondentes à equivalência dos estados de destino das transições com cada símbolo da linguagem). Por questões de organização da tabela, e de forma a facilitar a leitura da mesma, representa-se sempre na linha de cima de cada célula a condição referente ao símbolo 0 e na linha de baixo a condição referente ao símbolo 1. Adicionalmente, e em cada condição, o estado à esquerda é o estado de destino considerando o estado na vertical, e o estado à direita é o estado de destino considerando o estado na horizontal. Assim, a título de exemplo, na célula de interseção entre q_3 e q_5, na linha de cima

	q_0	q_1	q_2	q_3	q_4	q_5	q_6
q_1							
* q_2	X	X					
q_3			X				
q_4			X				
q_5			X				
* q_6	X	X		X	X	X	
q_7			X				X

temos $q_7 = q_1$, sendo que q_7 é o estado de destino da transição com o símbolo 0 a partir de q_3, e q_1 é o estado de destino da transição com o símbolo 1 a partir de q_5. A tabela abaixo mostra então este preenchimento.

	q_0	q_1	q_2	q_3	q_4	q_5	q_6
q_1	$q_1 = q_6$ $q_4 = q_2$						
* q_2	X	X					
q_3	$q_1 = q_7$ $q_4 = q_2$	$q_6 = q_7$ $q_2 = q_2$	X				
q_4	$q_1 = q_5$ $q_4 = q_1$	$q_6 = q_5$ $q_2 = q_1$	X	$q_7 = q_5$ $q_2 = q_1$			
q_5	$q_1 = q_1$ $q_4 = q_6$	$q_6 = q_1$ $q_2 = q_6$	X	$q_7 = q_1$ $q_2 = q_6$	$q_5 = q_1$ $q_1 = q_6$		
* q_6	X	X	$q_3 = q_5$ $q_7 = q_7$	X	X	X	
q_7	$q_1 = q_6$ $q_4 = q_2$	$q_6 = q_6$ $q_2 = q_2$	X	$q_7 = q_6$ $q_2 = q_2$	$q_5 = q_6$ $q_1 = q_2$	$q_1 = q_6$ $q_6 = q_2$	X

O próximo passo será o de marcar com um X as células com condições de equivalência já marcadas com um X. Para isso, vamos percorrer a tabela (neste caso, foi percorrida coluna a coluna, da esquerda para a direita, e em cada coluna de cima para baixo), e verificar o estado das células indicadas

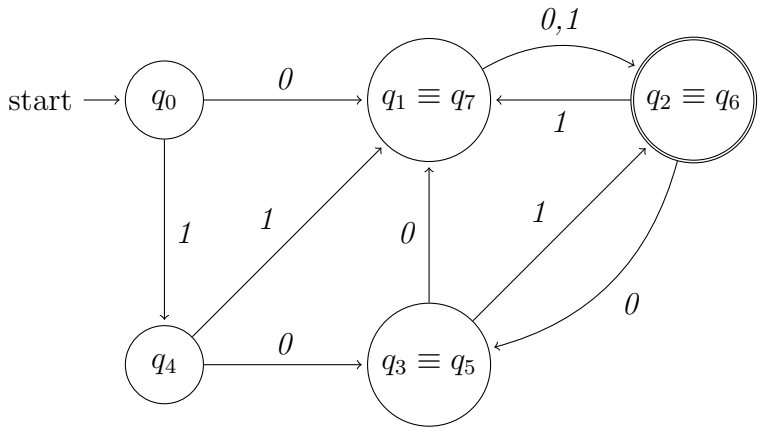
pelas condições de equivalência; se pelo menos uma dessas células estiver marcada com um X, então marcamos também a célula atual com um X, significando que também os estados que se intersetam nesta célula não serão equivalentes.

q_1	X						
* q_2	X	X					
q_3	X	X	X				
q_4	$q_1 = q_5$ $q_4 = q_1$	X	X	X			
q_5	X	X	X	$q_7 = q_1$ $q_2 = q_6$	X		
* q_6	X	X	$q_3 = q_5$ $q_7 = q_7$	X	X	X	
q_7	X	$q_6 = q_6$ $q_2 = q_2$	X	X	X	X	X
	q_0	q_1	q_2	q_3	q_4	q_5	q_6

Como podemos ver, a grande maioria das células foi já marcada com um X. Na próxima iteração deste processo obtemos a tabela abaixo.

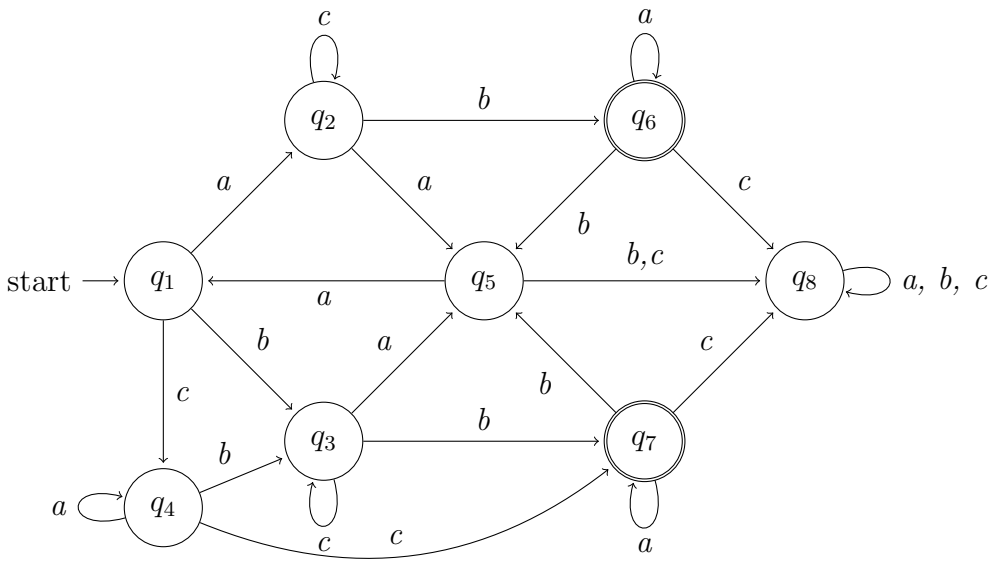
Agora já não existem mais alterações a realizar à tabela (novas passagens deixam a tabela inalterada), pelo que podemos verificar que o autómato original pode ser simplificado. As células que não estão marcadas com um X representam a equivalência dos estados respetivos. Por exemplo, a célula na interseção entre q_1 e q_7 não está marcada com um X, o que significa que q_1 e q_7 são dois estados equivalentes. Temos então as seguintes equivalências:
$q_1 \equiv q_7$
$q_2 \equiv q_6$
$q_3 \equiv q_5$
Podemos agora desenhar o novo DFA para a mesma linguagem, mas já com menos estados que o DFA original:

q_1	X						
* q_2	X	X					
q_3	X	X	X				
q_4	X	X	X	X			
q_5	X	X	X	$q_7 = q_1$ $q_2 = q_6$	X		
* q_6	X	X	$q_3 = q_5$ $q_7 = q_7$	X	X	X	
q_7	X	$q_6 = q_6$ $q_2 = q_2$	X	X	X	X	X
	q_0	q_1	q_2	q_3	q_4	q_5	q_6



Exercício 3

Simplifique o automato seguinte.



Para simplificar o autómato, vamos então usar a tabela de equivalência de estados. Na tabela abaixo, temos já marcados com um 'X' as células onde se intersetam um estado final com um estado não-final.

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_2							
q_3							
q_4							
q_5							
* q_6	X	X	X	X	X		
* q_7	X	X	X	X	X		
q_8						X	X

Neste caso, existem dois estados finais (q_6 e q_7), sendo que nas células de interseção entre um destes dois estados e dos um estado não finais é então marcada com um X.

O próximo passo será então o de colocar em cada célula as condições de equivalência necessárias para que os estados dessa célula sejam equivalentes. Como temos 3 símbolos na linguagem (a, b e c), cada célula terá 3 condições, como mostra a próxima iteração da tabela. De forma a simplificar

a interpretação, as linhas correspondem aos três símbolos, a, b e c, respetivamente, encontrando-se na esquerda das igualdades o estado de destino com a transição a partir do estado indicado na coluna respetiva, e no lado direito da igualdade o estado de destino com a transição a partir do estado indicado na linha respetiva.

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_2	$q_2 = q_5$ $q_3 = q_6$ $q_4 = q_2$						
q_3	$q_2 = q_5$ $q_3 = q_7$ $q_4 = q_3$	$q_5 = q_5$ $q_6 = q_7$ $q_2 = q_3$					
q_4	$q_2 = q_4$ $q_3 = q_3$ $q_4 = q_7$	$q_5 = q_4$ $q_6 = q_3$ $q_2 = q_7$	$q_5 = q_4$ $q_7 = q_3$ $q_3 = q_7$				
q_5	$q_2 = q_1$ $q_3 = q_8$ $q_4 = q_8$	$q_5 = q_1$ $q_6 = q_8$ $q_2 = q_8$	$q_5 = q_1$ $q_7 = q_8$ $q_3 = q_8$	$q_4 = q_1$ $q_3 = q_8$ $q_7 = q_8$			
* q_6	X	X	X	X	X		
* q_7	X	X	X	X	X	$q_6 = q_7$ $q_5 = q_5$ $q_8 = q_8$	
q_8	$q_2 = q_8$ $q_3 = q_8$ $q_4 = q_8$	$q_5 = q_8$ $q_6 = q_8$ $q_2 = q_8$	$q_5 = q_8$ $q_7 = q_8$ $q_3 = q_8$	$q_4 = q_8$ $q_3 = q_8$ $q_7 = q_8$	$q_1 = q_8$ $q_8 = q_8$ $q_8 = q_8$	X	X

Após uma primeira iteração, são marcadas com um X todas as células em que pelo menos uma das condições de equivalência está também marcada com um X. Neste caso, foi optou-se por percorrer a tabela coluna a coluna, de cima para baixo, da esquerda para a direita. Ficamos assim com a seguinte tabela após esta primeira iteração:

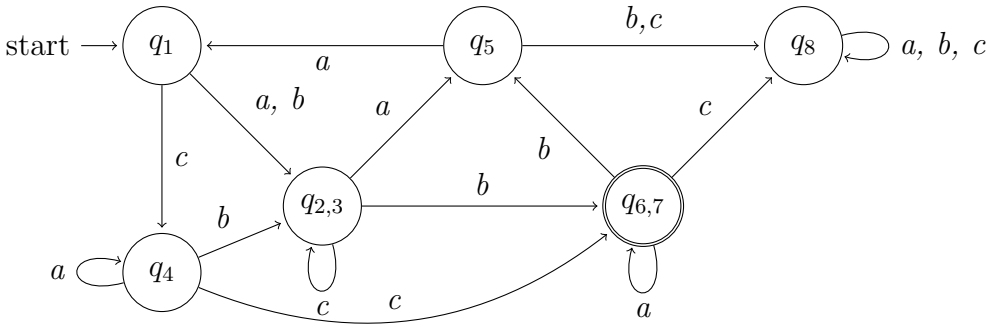
Após mais uma iteração, obtemos a seguinte tabela, que será já a tabela final (iterações seguintes não produzem mais alterações na tabela). Aqui podemos ver que apenas duas células não estão preenchidas com X, indicando

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_2	X						
q_3	X	$q_5 = q_5$ $q_6 = q_7$ $q_2 = q_3$					
q_4	X	X	X				
q_5	X	X	X	X			
* q_6	X	X	X	X	X		
* q_7	X	X	X	X	X	$q_6 = q_7$ $q_5 = q_5$ $q_8 = q_8$	
q_8	$q_2 = q_8$ $q_3 = q_8$ $q_4 = q_8$	X	X	X	$q_1 = q_8$ $q_8 = q_8$ $q_8 = q_8$	X	X

que os estados que deram origem a essas células são equivalentes. Neste caso, podemos constatar que q_2 e q_3 são equivalentes entre si, e que q_6 e q_7 são também equivalentes entre si.

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_2	X						
q_3	X	$q_5 = q_5$ $q_6 = q_7$ $q_2 = q_3$					
q_4	X	X	X				
q_5	X	X	X	X			
* q_6	X	X	X	X	X		
* q_7	X	X	X	X	X	$q_6 = q_7$ $q_5 = q_5$ $q_8 = q_8$	
q_8	X	X	X	X	X	X	X

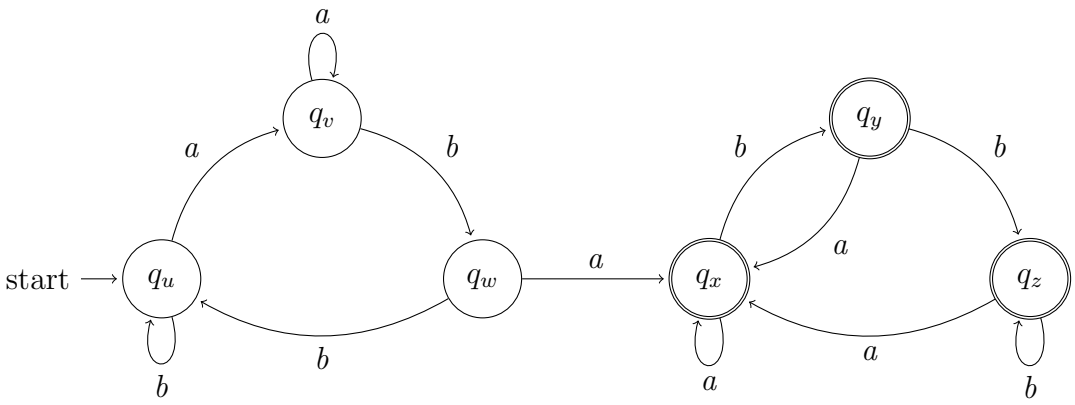
Então, o DFA mínimo para a linguagem terá apenas 6 estados, em comparação com os 8 do autómato inicial.



Exercício 4

Recorde o autómato obtido no **Exercício 5** do Capítulo 3, e minimize-o.

Recordando o DFA obtido no exercício, com os estados renomeados para simplificação:



Começamos então por construir a tabela de equivalência de estados, marcando já os estados não equivalentes (células na interseção de um estado final com um estado não final):

	q_u	q_v	q_w	q_x	q_y
q_v					
q_w					
* q_x	X	X	X		
* q_y	X	X	X		
* q_z	X	X	X		

Colocamos agora as condições de equivalência em cada célula não preenchida com um X:

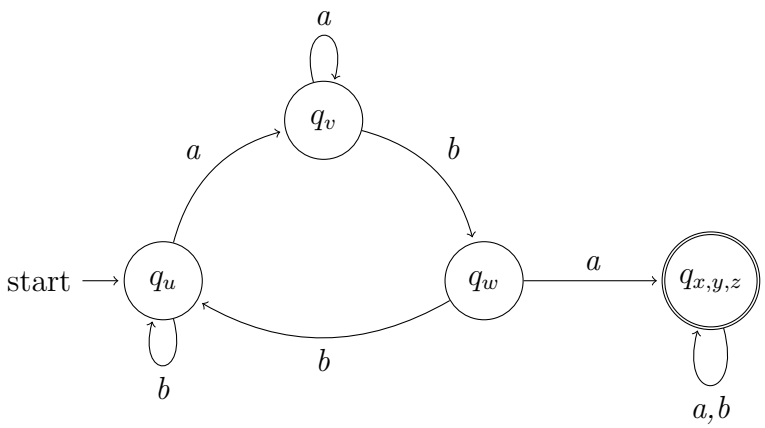
	q_u	q_v	q_w	q_x	q_y
q_v	$q_v = q_v$ $q_u = q_w$				
q_w	$q_v = q_x$ $q_u = q_u$	$q_v = q_x$ $q_w = q_u$			
* q_x	X	X	X		
* q_y	X	X	X	$q_x = q_x$ $q_y = q_z$	
* q_z	X	X	X	$q_x = q_x$ $q_y = q_z$	$q_x = q_x$ $q_z = q_z$

Após duas iterações do processo (marcar células com X quando pelo menos uma das células referenciadas nas condições de equivalência está já marcada com X), chegámos à seguinte tabela:

Temos então as seguintes equivalências: $q_x \equiv q_y \equiv q_z$.

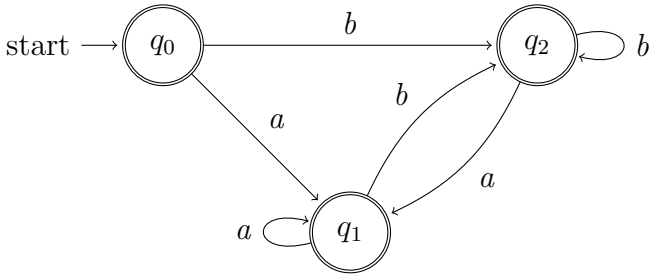
Podemos agora desenhar o novo DFA minimizado:

	q_u	q_v	q_w	q_x	q_y
q_v	X				
q_w	X	X			
* q_x	X	X	X		
* q_y	X	X	X	$q_x = q_x$ $q_y = q_z$	
* q_z	X	X	X	$q_x = q_x$ $q_y = q_z$	$q_x = q_x$ $q_z = q_z$



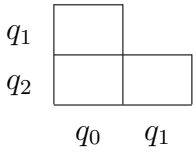
Exercício 5 (Exercício 2015/16)

Minimize o seguinte DFA, desenhando o DFA resultante, e indique qual a expressão regular que representa a linguagem por ele aceite.



Começamos então por construir a tabela de equivalências, cortando as células que representam interseção entre um estado final e um estado não final. Note-se que, dado que os três estados são estados finais, não há nenhuma célula cortada inicialmente (tabela da esquerda).

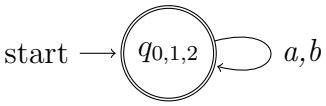
Colocamos agora as condições de equivalências necessárias para cada uma das células da tabela (tabela da direita).



q_1	$q_1 = q_1$ $q_2 = q_2$	
q_2	$q_1 = q_1$ $q_2 = q_2$	$q_1 = q_1$ $q_2 = q_2$
	q_0	q_1

Como podemos verificar, todas as condições de equivalência são verdadeiras, pelo que todas as células ficam por cruzar, indicando que todos os pares de estados são equivalentes (ou seja, $q_0 \equiv q_1 \equiv q_2$).

Podemos então desenhar o seguinte autómato, tendo em conta as equivalências encontradas:



Em relação à expressão regular para a linguagem, dado que temos apenas um estado, o qual é final, a expressão será simplesmente (a+b)*.

Exercício 6

Verifique se os seguintes dois autómatos representam a mesma linguagem.

	a	b
$\rightarrow q_w$	q_x	q_y
q_x	q_y	q_z
q_y	q_z	q_w
$*\ q_z$	q_w	q_x

	a	b
$\rightarrow q_0$	q_2	q_6
$*\ q_1$	q_0	q_4
q_2	q_3	q_7
q_3	q_1	q_5
q_4	q_6	q_1
q_5	q_4	q_3
q_6	q_7	q_0
$*\ q_7$	q_5	q_2

Para resolver o problema podemos optar por duas estratégias:

- Minimizar cada um dos autómatos e comparar depois os resultados obtidos, verificando se os dois autómatos minimizados são iguais (a menos de nomes dos estados).

- Juntar os estados dos dois autómatos na mesma tabela de minimização, e verificar se os estados iniciais dos dois autómatos são equivalentes.

Vamos usar este segundo método, começando então por apresentar a tabela de equivalência de estados contendo os estados dos dois autómatos (dado o tamanho da tabela, apresenta-se já preenchida com os estados não equivalentes marcados, e com as condições de equivalência para as restantes células).

	q_w	q_x	q_y	q_z	q_0	q_1	q_2	q_3	q_4	q_5	q_6
q_x	$q_x = q_y$ $q_y = q_z$										
q_y	$q_x = q_z$ $q_y = q_w$	$q_y = q_z$ $q_z = q_w$									
* q_z	X	X	X								
q_0	$q_x = q_2$ $q_y = q_6$	$q_y = q_2$ $q_z = q_6$	$q_z = q_2$ $q_w = q_6$	X							
* q_1	X	X	X	$q_w = q_0$ $q_x = q_4$	X						
q_2	$q_x = q_3$ $q_y = q_7$	$q_y = q_3$ $q_z = q_7$	$q_z = q_3$ $q_w = q_7$	X	$q_2 = q_3$ $q_6 = q_7$	X					
q_3	$q_x = q_1$ $q_y = q_5$	$q_y = q_1$ $q_z = q_5$	$q_z = q_1$ $q_w = q_5$	X	$q_2 = q_1$ $q_6 = q_5$	X	$q_3 = q_1$ $q_7 = q_5$				
q_4	$q_x = q_6$ $q_y = q_1$	$q_y = q_6$ $q_z = q_1$	$q_z = q_6$ $q_w = q_1$	X	$q_2 = q_6$ $q_6 = q_1$	X	$q_3 = q_6$ $q_7 = q_1$	$q_1 = q_6$ $q_5 = q_1$			
q_5	$q_x = q_4$ $q_y = q_3$	$q_y = q_4$ $q_z = q_3$	$q_z = q_4$ $q_w = q_3$	X	$q_2 = q_4$ $q_6 = q_3$	X	$q_3 = q_4$ $q_7 = q_3$	$q_1 = q_4$ $q_5 = q_3$	$q_6 = q_4$ $q_1 = q_3$		
q_6	$q_x = q_7$ $q_y = q_0$	$q_y = q_7$ $q_z = q_0$	$q_z = q_7$ $q_w = q_0$	X	$q_2 = q_7$ $q_6 = q_0$	X	$q_3 = q_7$ $q_7 = q_0$	$q_1 = q_7$ $q_5 = q_0$	$q_6 = q_7$ $q_1 = q_0$	$q_4 = q_7$ $q_3 = q_0$	
* q_7	X	X	X	$q_w = q_5$ $q_x = q_2$	X	$q_0 = q_5$ $q_4 = q_2$	X	X	X	X	X

A próxima etapa passa então por percorrer a tabela iterativamente, marcando com um X as células em que pelo menos uma das condições de equi-

valência seja falsa (isto é, esteja já marcada). A tabela abaixo apresenta já o resultado final deste processo, não havendo já mais alterações que possam ser realizadas à tabela.

	q_w	q_x	q_y	q_z	q_0	q_1	q_2	q_3	q_4	q_5	q_6
q_x	X										
q_y	X	X									
* q_z	X	X	X								
q_0	$q_x = q_2$ $q_y = q_6$	X	X	X							
* q_1	X	X	X	$q_w = q_0$ $q_x = q_4$	X						
q_2	X	$q_y = q_3$ $q_z = q_7$	X	X	X	X					
q_3	X	X	$q_z = q_1$ $q_w = q_5$	X	X	X	X				
q_4	X	$q_y = q_6$ $q_z = q_1$	X	X	X	X	$q_3 = q_6$ $q_7 = q_1$	X			
q_5	$q_x = q_4$ $q_y = q_3$	X	X	X	$q_2 = q_4$ $q_6 = q_3$	X	X	X	X		
q_6	X	X	$q_z = q_7$ $q_w = q_0$	X	X	X	X	$q_1 = q_7$ $q_5 = q_0$	X	X	
* q_7	X	X	X	$q_w = q_5$ $q_x = q_2$	X	$q_0 = q_5$ $q_4 = q_2$	X	X	X	X	X

Podemos então ver que existem vários pares de estados equivalentes, incluindo os estados iniciais de cada um dos autómatos, q_w e q_0, o que significa que os dois autómatos são equivalentes.

Lema da Bombagem

O Lema da Bombagem diz-nos que se uma linguagem é regular, então segue o lema, ou seja, que para cadeias '*grandes*' existe uma parte da cadeia que se pode repetir, dando origem a mais cadeias da linguagem. Isto permite-nos usar autómatos finitos para representar linguagens infinitas, usando *loops* no autómato para repetir parte da cadeia.

Como o que nos interessa é a prova da regularidade ou não de uma linguagem, temos de usar a prova por contra-positiva, ou seja, se uma linguagem não segue o Lema da Bombagem, então não é regular (repare-se que uma linguagem pode seguir o lema, mas mesmo assim não ser regular, como vamos ver mais à frente). Para provar que uma linguagem é regular, podemos apresentar um autómato ou uma expressão regular que a represente (se formos capazes de o fazer, é porque a linguagem é regular).

Então, e de uma forma mais formal, o Lema diz-nos que se L é uma linguagem regular, existe uma constante m (um comprimento mínimo, que será dependente da linguagem) tal que para todas as cadeias w pertencentes a L com comprimento maior ou igual a m existe uma decomposição de w em três partes ($w = xyz$, com $y \neq \varepsilon$ e $|xy| \leq m$), tal que para qualquer $n \geq 0$, as cadeias geradas por n repetições de y também pertencem à linguagem (note-se que não estamos a dizer que são geradas todas as cadeias da linguagem, mas sim que todas as cadeias geradas pertencem à linguagem). De uma forma simplificada, será algo como:

$$\exists m : \forall w \in L, |w| \geq m : \exists w = xyz, y \neq \varepsilon, |xy| \leq m : \forall n : xy^n z \in L$$

.

Para fazer a prova, temos de negar o lema, o que significa que será necessário inverter os quantificadores. Então, temos que para uma linguagem L não seguir o lema, qualquer que seja o valor de m (por muito grande que seja), existe pelo menos uma cadeia w pertencente a L com comprimento maior ou igual a m, tal que qualquer que seja a divisão dessa cadeia em três partes, existem pelo menos um n tal que $xy^n z$ não pertence a L. Novamente, de uma forma simplificada, será algo como:

$$\forall m : \exists w \in L, |w| \geq m : \forall w = xyz, y \neq \varepsilon, |xy| \leq m : \exists n : xy^n z \notin L$$

Vamos ver um exemplo, para L = $\{vv^R : v \in \{0,1\}^* \}$, ou seja, cadeias binárias (de comprimento par) em que a segunda metade da cadeia é igual ao reverso da primeira metade da cadeia.

Podemos ver desde já que a linguagem não é regular, dado que um DFA para reconhecer estas cadeias teria de ser infinito, de forma a memorizar toda a primeira metade da cadeia, para garantir que a segunda metade da cadeia seria o reverso da primeira metade. Vamos fazer a prova usando o Lema da Bombagem. Começamos por assumir que existe um m, provando então que o Lema falha qualquer que seja o valor escolhido para m.

Agora, temos de escolher uma cadeia na qual não seja possível realizar a divisão e bombear a parte do meio. Vamos escolher uma cadeia 'grande', por exemplo a cadeia no formato $(01)^m(10)^m$, com comprimento $4m$. Numa cadeia com comprimento $4m$, e como sabemos que $y \neq \varepsilon$ e $|xy| \leq m$, então tanto x como y estão necessariamente na primeira metade da cadeia, e z abarca ainda parte da primeira metade, para além da segunda metade da cadeia. Agora, temos de provar que para cada possibilidade de divisão da cadeia, existe um n tal que $xy^nz \notin L$. Com este formato de cadeia, temos sempre 0's e 1's intercalados, com exceção do ponto central da cadeia, em que existem dois 1's consecutivos. Ora, como y não pode ser vazio, y vai conter pelo menos um símbolo, o que significa que para $n = 0$, a cadeia $xy^0z = xz$ vai ficar mais curta à esquerda do par de uns do que à direita, o que significa que este deixa de ser o ponto central da cadeia, o que significa que a cadeia deixa de estar no formato vv^R, o que significa que não pertence à linguagem.

Note-se que existem cadeias para as quais pode existir uma divisão que permite bombear a parte central, como por exemplo as cadeias apenas com 0's ou apenas com 1's (ao bombear por exemplo os dois primeiros símbolos da cadeia é possível gerar cadeias pertencentes à linguagem), e por isso é essencial que a escolha da cadeia seja feita com cuidado, para garantir que não escapa nenhuma cadeia que permita fazer a prova.

Podemos concluir que, uma vez que provamos que a linguagem não segue o Lema da Bombagem, então a linguagem não é regular.

Exercício 7

Determine se a linguagem $L = \{a^nb^p : n \leq p \leq 2n\}$ é ou não regular usando o Lema da Bombagem.

Vamos então começar por assumir a existência de um m, escolhendo depois uma cadeia de comprimento maior ou igual a m, como por exemplo $w = a^mb^m$, que pertence a L (repare-se que esta escolha vai diminuir o número de divisões possíves mais à frente).

No próximo passo, ao fazer a divisão de w em três partes, e como $|xy| \leq m$, sabemos que y estará na primeira parte da cadeia, e vai conter apenas a's (pelo menos 1, dado que $y \neq \varepsilon$). Ao bombear y com n maior ou igual a 2, a cadeia xy^nz deixa de pertencer a L, dado que vamos ficar com mais a's do que b's.

De uma forma mais formal, as divisões possíveis são com $x = a^p$, $p \geq 0$; $y = a^q$, $q \geq 1$; e $z = a^{m-p-q}b^m$. Para $n = 2$, a cadeia resultante será $xy^2z = xyyz = a^pa^qa^qa^{m-p-q}b^m = a^{m+q}b^m$. Como sabemos que $q > 0$, então vamos ter mais a's do que b's, o que significa que a cadeia não pertence à linguagem.

Como o valor de m não foi instanciado, significa que por muito grande que este seja, a cadeia w com o dobro desse comprimento no formato a^mb^m (que pertence a L) não pode ser dividida em três partes e ter a parte do meio bombeada, pois isso originaria cadeias com mais a's do que b's.

Fica então provado que a linguagem não é regular, uma vez que não segue o Lema da Bombagem.

Exercício 8

Determine se a linguagem $L = \{0^a1^b0^c : a > 4, b > 2, c \leq b\}$ é ou não regular usando o Lema da Bombagem.

Começamos então por assumir a existência de uma constante m, e escolhemos uma cadeia pertencente a L. Vamos escolher a cadeia no formato 000001^m0^m, que pertence à linguagem, desde que m seja maior do que 2 (o que se pode assumir, dado que o comprimento mínimo da cadeia é maior do que 2, e não faria sentido ter um valor de m menor do que o comprimento mínimo das cadeias da linguagem). Esta escolha da cadeia é pensada de forma a conseguirmos ao mesmo tempo diminuir o número de casos de partições a analisar e também encontrar um caso (um valor de n) em cada divisão tal que $xy^nz \notin L$.

Então, com a cadeia 0000001^m0^m, e considerando que $y \neq \varepsilon$ e $|xy| \leq m$, podemos ter três possíveis divisões da cadeia:

– y contém apenas 0's. Neste caso, como a cadeia escolhida tem apenas 5 zeros inicialmente, quando $n = 0$, a cadeia gerada (xz) não vai pertencer à linguagem, dado que irá ter no máximo quatro 0's (e a linguagem especifica $a > 4$).

– y contém 0's e 1's. Neste caso, para $n = 0$, a cadeia irá também ter no máximo quatro zeros, pelo que também não irá pertencer à linguagem.

– y contém apenas 1's. Neste caso, para $n = 0$, a cadeia irá ter um número de 1's inferior ao número de 0's do final da cadeia. Dado que uma das condições da linguagem é $c \leq b$, esta cadeia deixa de pertencer a L.

Prova-se então que não existe nenhuma divisão da cadeia que permita bombear a parte do meio da mesma, pelo que podemos afirmar que, não seguindo o Lema da Bombagem, a linguagem é não regular.

Exercício 9

Determine se a linguagem $L = \{pp^Rq : p, q \in \{0,1\}^+\}$ é ou não regular usando o Lema da Bombagem.

Esta linguagem já permite verificar o Lema da Bombagem. Escolhendo $m = 4$, por exemplo, conseguimos realizar a prova para qualquer cadeia da linguagem.

Para cadeias com $|p| = 1$, podemos fazer a divisão de forma a que $x = pp^R$ e y seja o primeiro símbolo de q, com z representando a restante cadeia. Seja qual for o valor de n, a cadeia xy^nz pertence a L, pois a primeira parte da cadeia (x) permanece inalterada, mantendo-se as cadeias resultantes no formato pp^Rq.

Para cadeias com $|p| > 1$, podemos fazer a divisão de forma a que $x = \varepsilon$, y seja o primeiro símbolo de p, e z represente a restante cadeia. Seja qual for o valor de n, a cadeia irá pertencer à linguagem: para $n = 0$, estamos basicamente a cortar o primeiro símbolo de p; basta assumir que o último símbolo de p^R passa agora a pertencer a q para que a cadeia continue no formato pp^Rq. Para $n > 1$, vamos ter repetições do primeiro símbolo; as duas primeiras repetições desse símbolo podem ser vistas como pp^R, em que p é agora constituído apenas por um símbolo, e toda a restante cadeia pertencerá a q, ficando assim a cadeia também no formato pp^Rq.

Então, acabamos de provar que o Lema da Bombagem se verifica para esta linguagem L: existe realmente um valor m (neste caso até relativamente pequeno, usamos 4), para o qual todas as cadeias $w \in L$ com $|w| \geq m$ podem ser divididas em três partes ($w = xyz$), tal que $xy^nz \in L$ para $n \geq 0$.

Na realidade, se tentarmos fazer o DFA para a linguagem, podemos constatar que teríamos de ter um DFA infinito para conseguir fazer o reconhecimento de pp^R, e sabemos que a linguagem não é regular.

No entanto, não conseguindo provar que a linguagem não segue o Lema da Bombagem, não podemos retirar qualquer conclusão sobre a regularidade da linguagem.