Capítulo 2

Autómatos Finitos Deterministas (DFA)

Este capítulo aborda Autómatos Finitos Deterministas (ou DFAs, do inglês *Deterministic Finite Automata*). Os DFAs são uma das possibilidades de especificação de linguagens regulares (as outras possibilidades, Autómatos Finitos Não Deterministas e Expressões Regulares são abordadas nos próximos capítulos).

As linguagens regulares, e os autómatos, são muito úteis nas mais diversas áreas da informática, seja para ajudar na especificação de protocolos de comunicação (especificação do estados dos intervenientes e das possíveis mensagens que cada interveniente pode enviar a cada momento; isto é algo abordado nas disciplinas de Redes de Computadores ou Sistemas Distribuídos); em Diagramas de Estados, usados para especificação de sistemas (isto é algo abordado na disciplina de Engenharia de Software, entre outras); ...

São também muito usados (normalmente por geração automática a partir de expressões regulares, algo que vamos ver também mais à frente) para pesquisa de strings (por exemplo, com o comando *grep* em Linux); para validação de entrada em formulários (por exemplo para validar o formato de endereços de correio eletrónico, ou de um URL; isto é algo abordado nas disciplinas de Linguagens e Tecnologias Web e também em Laboratório de Bases de Dados e Aplicações Web, entre outras); ou na fase de análise lexical de um compilador (isto é abordado na disciplina de Compiladores), entre muitos outros usos.

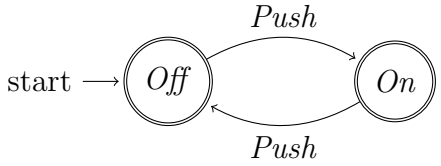
Definição

Um DFA representa um sistema que a qualquer momento pode estar num estado de vários possíveis. Vamos ver o exemplo muito simples (e clássico) de um interruptor, que pode estar ligado (On) ou desligado (Off), e que alterna consoante o interruptor é pressionado ($Push$).

Um DFA é definido formalmente como um tuplo $DFA = (Q, \Sigma, \delta, q_0, F)$, em que Q é o conjunto de estados do DFA, Σ é o conjunto de símbolos de entrada (alfabeto), δ representa a função de transição de estados e símbolos para estados ($\delta(q, a) = p$ denota uma transição do estado q para o estado p com o símbolo a), $q_0 \in Q$ representa o estado inicial, e $F \subseteq Q$ é o conjunto de estados finais.

$$Interruptor = (\{Off, On\}, \{Push\}, \{\delta(Off, Push) = On;$$
$$\delta(On, Push) = Off\}, Off, \{Off, On\})$$

Um DFA pode ser representado graficamente por um diagrama em que os nós representam os estados e as arestas as transições entre estados ($\delta(q, a) = p$ representado por um arco de q para p com etiqueta a); o estado inicial é denotado por uma seta de entrada sem origem etiquetada *Start*, e os nós reprentativos de estados finais possuem um duplo círculo.

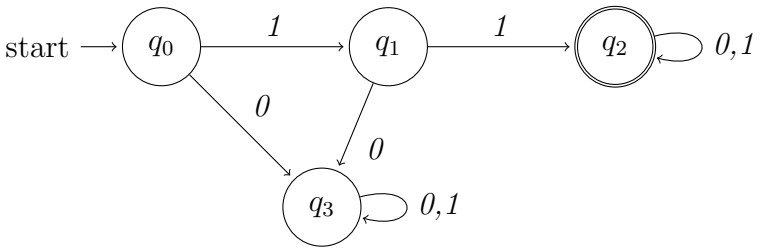


Um DFA pode ainda ser representado por uma tabela de transições, em que as linhas representam os estados, e as colunas os símbolos de entrada. Nas células de interseção entre estes encontra-se o estado de destino da transição respetiva. O estado inicial é indicado com uma seta e os estados finais com um asterisco.

	Push
→ * Off	On
* On	Off

Os DFAs são também usados no reconhecimento de linguagens, ou seja, permitem verificar se uma certa cadeia de caracteres pertence ou não a

uma determinada linguagem. Vejamos o exemplo da linguagem das cadeias binárias que começam por '11'. Ao construir o DFA é preciso garantir que conseguimos saber em que estado de reconhecimento este se encontra a cada momento: se ainda não reconheceu nenhum bit; se já reconheceu o primeiro 1, mas não o segundo; se já reconheceu os dois 1s (e portanto a cadeia é aceite); ou se reconheceu algo que não deveria estar ali, e portanto faz com que a cadeia não pertença à linguagem. Vamos ver a resolução em formato gráfico, para facilitar a interpretação.

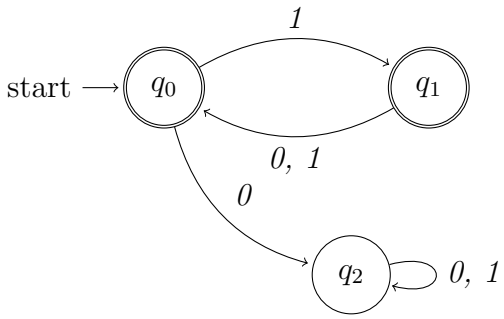


Temos então o DFA acima como solução para o problema de reconhecer cadeias binárias que começam por '11'. Se a cadeia de entrada for '1101', o DFA transita de q_0 para q_1 com o primeiro 1, e daí para q_2 com o segundo 1; daí para a frente, permanece em q_2 tanto com 0 como com 1, uma vez que a condição de aceitação (começar com 11) já foi cumprida. O estado q_3 serve para que o DFA fique completo (isto é, possuir transição para todos os pares *(estado, símbolo de entrada)*), sendo considerado um 'estado morto': há transições para ele de todos os estados com símbolos não usados até então, e do estado morto para ele próprio com todos os símbolos de entrada (isto garante que o DFA não morre até terminar de processar a cadeia de entrada).

Exercício 1

Apresente um DFA que permita reconhecer cadeias sobre o alfabeto {0,1} em que todas as posições ímpares têm um 1.

A solução para este problema passa por reconhecer que as transições representam posições ímpares ou pares, e garantir que nas transições ímpares temos um 1 (nas transições pares, tanto 0 como 1 são aceites).

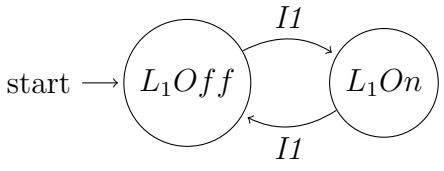


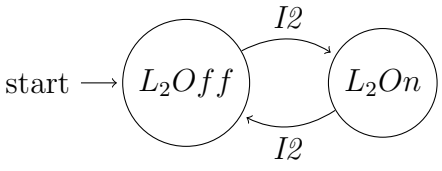
Se observarmos a solução acima, temos a transição de q_0 para q_1 a representar as posições ímpares e a de retorno as posições pares. Tanto q_0 como q_1 são estados finais, uma vez que em nenhum dos casos a regra (de posições ímpares terem 1) foi infrigida (note que a cadeia vazia, ε, faz também parte da linguagem). O estado q_2 é o estado morto que completa o DFA e trata dos dos casos em que uma posição ímpar é preenchida com 0.

Exercício 2

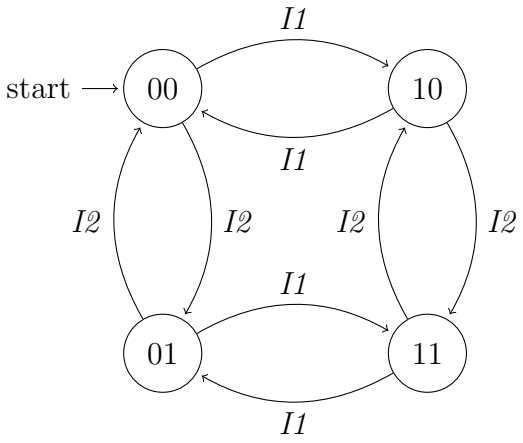
Apresente um DFA para modelar o estado das lâmpadas de uma sala com duas lâmpadas e respetivos interruptores.

Vamos considerar que cada lâmpada (L1 e L2) tem dois possíveis estados: ligada e desligada (*On* e *Off*); e que os dois interruptores (I1 e I2) afetam o estado das lâmpadas respetivas. Podemos modelar cada uma das lâmpadas de forma independente e depois fazer o produto dos dois DFAs daí resultantes, obtendo assim o DFA completo para as duas lâmpadas. Ficam assim os dois DFAs:





Para obter um único DFA é então necessário fazer o produto dos dois DFAs acima. Para isso, o novo DFA terá um estado para cada tuplo de estados dos DFAs anteriores (S_{DFA_1}, S_{DFA_2}) . Vamos representar cada novo estado como um número binário em que o primeiro bit representa o estado de L_1 (0 para Off e 1 para On), e o segundo bit representa o estado de L_2.

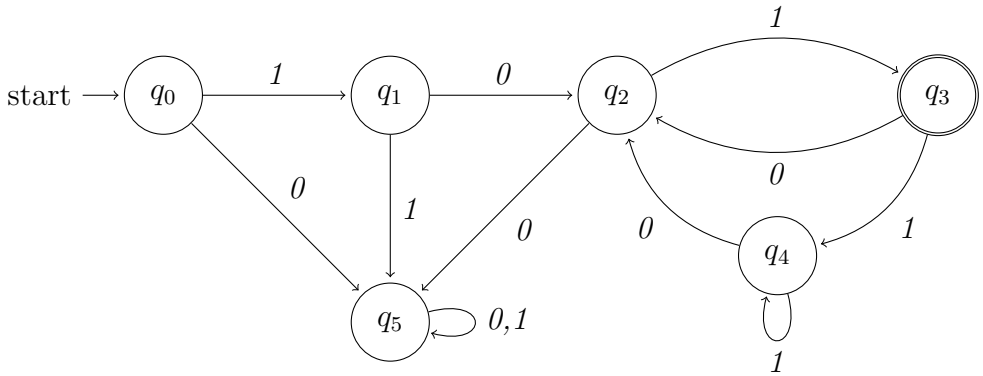


As transições devem ter em atenção quais as alterações que provocam nos estados dos dois DFAs. Neste caso, transições com $I1$ afetam apenas L_1, pelo que transições com $I1$ no autómato produto levam a estados sem alteração no 'sub-estado' de L_2.

Exercício 3 (Desafio 2014/15)

Modele um DFA que permita reconhecer cadeias binárias que iniciem em 10, terminem em 01, e não possuam nenhuma sequência com dois 0's. Ex.: 1001 não pertence à linguagem; 10101 pertence à linguagem.

Para resolver este exercício, podemos começar por tentar dividir o problema em 3 sub-problemas: início das cadeias, não possuir sequência com dois 0's consecutivos, e término das cadeias. Temos depois, para obter a solução final, de ver como integrar cada uma dessas partes, de forma a evitar estados desnecessários, e garantir que todas as cadeias pertencentes à linguagem (mesmo as mais curtas, como 101) são aceites.

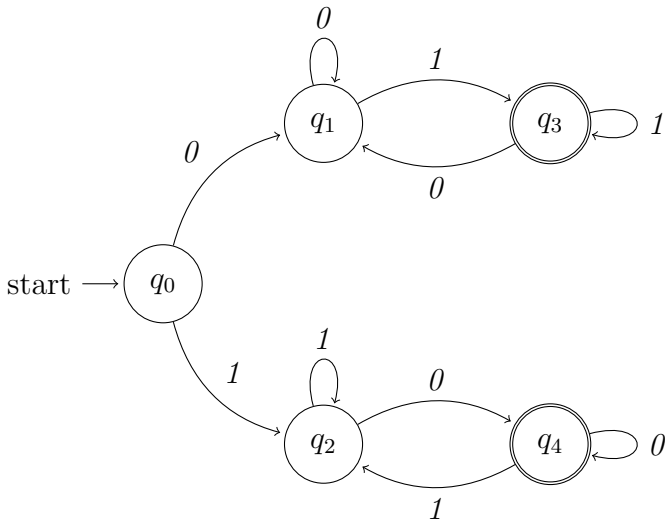


Analisando o autómato acima, podemos ver a parte inicial nos estados q_0, q_1 e q_2, em que temos a sequência '10'. Os estados q_1, q_2 e q_3 representam a parte final, com a sequência '01' para término, sendo o estado q_3 o estado de aceitação; Os estados q_1 e q_2 são partilhados entre as partes de início e de término de forma a garantir que a cadeia '101', que faz parte da linguagem, é também aceite pelo DFA. O estado q_4 e as transições com '0' de q_4 e de q_3 para q_2 garantem que não existem dois zeros consecutivos e também que podem existir vários 1's consecutivos. O estado q_5 é o estado morto, e existe apenas para tornar o DFA completo, garantindo que todos os estados possuem transições para todos os símbolos de entrada.

Exercício 4

Modele um DFA que permita reconhecer cadeias no alfabeto {0,1} com símbolos inicial e final diferentes.

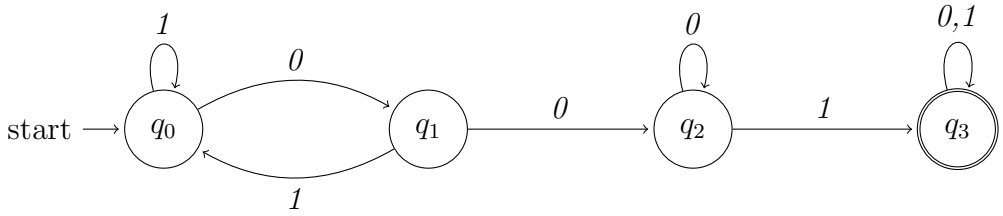
Precisamos aqui de distinguir os dois casos iniciais, e assim dividir o DFA em dois 'ramos', um para cada um dos possíveis símbolos iniciais, e que terá de garantir símbolo final diferente.



Os estados q_3 e q_4 são os estados de aceitação, uma vez que é para eles que transitam os símbolos contrários dos símbolos iniciais de cada um dos respetivos ramos. Quando temos uma transição com o mesmo símbolo que o símbolo inicial, os estados de destino (q_1 e q_2) deixam de ser estados finais.

Exercício 5

Considere o seguinte DFA. Apresente a definição formal e a representação em tabela para o DFA, e indique por palavras qual a linguagem representada. Indique ainda quais das cadeias '010101', '0100101' e '1010010' são aceites pelo DFA.



Para obter a definição formal e a representação tabular do DFA é apenas necessário interpretar o DFA e mudar o formato da representação.

DFA A = ($\{q_0, q_1, q_2, q_3\}$, $\{0, 1\}$, δ, q_0, $\{q_3\}$), em que δ representa as seguintes transições: $\delta(q_0, 0) = q_1$; $\delta(q_0, 1) = q_0$; $\delta(q_1, 0) = q_2$; $\delta(q_1, 1) = q_0$; $\delta(q_2, 0) = q_2$; $\delta(q_2, 1) = q_3$; $\delta(q_3, 0) = q_3$; $\delta(q_3, 1) = q_3$.

E a representação em tabela:

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

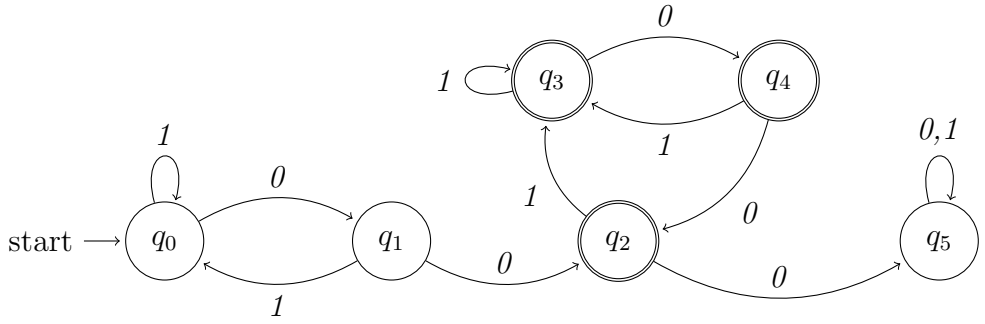
Em relação à linguagem aceite pelo autómato, temos de tentar ver as transições que levam ao estado final. Podemos ver que uma vez atingido o estado final não voltamos a um estado não-final, o que significa que queremos reconhecer a sequência de símbolos que levam até ele. Ao olhar para as transições, vemos que a sequência é 001: q_0 é o estado em que ainda não se reconheceu qualquer símbolo desta sequência; q_1 é o estado em que já se reconheceu o primeiro 0; q_2 é o estado em que já se reconheceram os dois 0s e finalmente o estado final quando toda a sequência foi reconhecida.

Em relação às cadeias aceites, podemos calcular a função de transição estendida $\delta\hat{}$ para cada uma das cadeias e verificar se é atingido o estado final. $\delta\hat{}(q_0, 010101) = q_0$; $\delta\hat{}(q_0, 0100101) = q_3$; $\delta\hat{}(q_0, 010010) = q_3$. Assim, apenas as duas últimas cadeias são aceites pelo DFA.

Exercício 6

Modele um DFA que permita reconhecer cadeias binárias que contenham a sequência 00, mas não a sequência 000.

Uma solução para este exercício passa por reconhecer que quando temos a sequência 000, há uma transição para o estado morto, e portanto a cadeia não será aceite. Por outro lado, um estado de aceitação só poderá ser atingido a partir do momento em que apareçam dois 0s consecutivos. Assim o DFA terá de manter nos seus estados informação de quantos 0s consecutivos foram lidos, e se já atingiu a condição de aceitação ou não. Podemos pensar nesta solução como uma espécie de contador de zeros, em que há um *reset* quando é lido um 1.



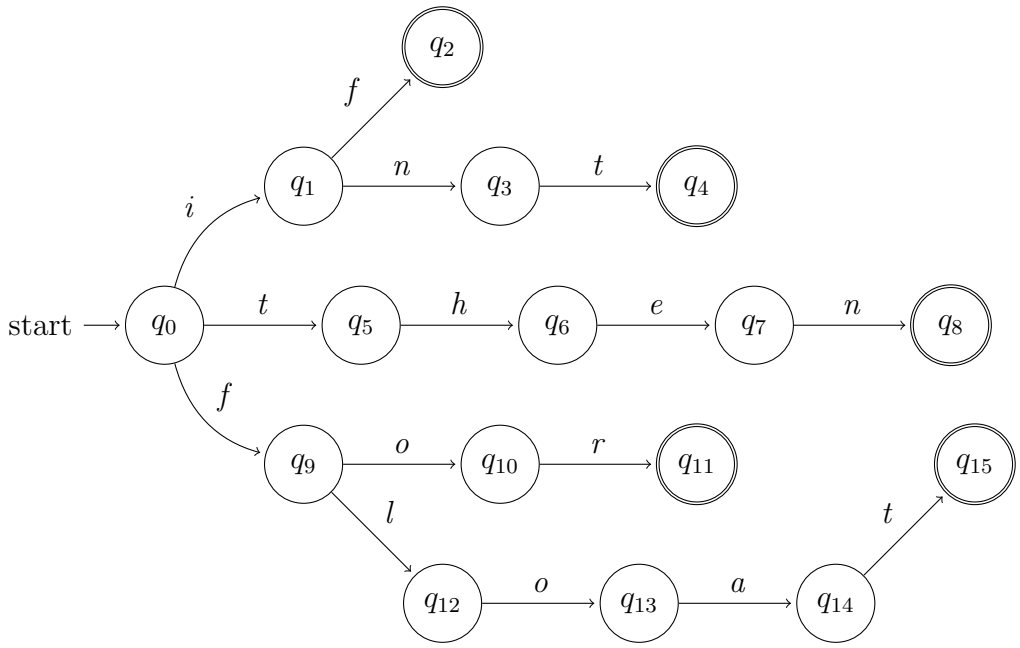
Olhando para o DFA acima, podemos ver que só atingimos um estado de aceitação quando há dois 0s consecutivos (passando de q_0 para q_1 e daí para q_2). Depois de atingir q_2, a condição de aceitação foi atingida, mas é necessário ainda garantir que a restrição dos três zeros consecutivos não é quebrada, e para isso necessitamos dos estados q_3 e q_4, que são uma 'cópia' dos estados q_0 e q_1, respetivamente, mas neste caso sendo estados de aceitação. O estado q_5 e a transição para ele a partir de q_2 (o estado que representa dois zeros consecutivos) garante que ao terceiro 0, o DFA não atinge nunca mais um estado de aceitação.

Exercício 7 (TPC 2010/11)

Desenhe um DFA que reconheça as cadeias de caracteres: *if*, *then*, *int*, *for*, e *float*. De seguida, represente o DFA usando a notação formal.

Partindo das palavras que se pretender ver reconhecidas, podemos definir o nosso alfabeto como contendo apenas esses caracteres. Assim, o alfabeto a

usar será $\Sigma = \{a, e, f, h, i, l, n, o, r, t\}$. Como temos palavras começadas pelo mesmo símbolo (por exemplo *if* e *int* começam ambos por i), temos de ter cuidado para que exista apenas uma transição possível em cada ponto. Uma possível solução será a seguinte (note que se apresenta o DFA incompleto, por questões de simplificação):



Temos então o reconhecimento de *if* em q_2, *int* em q_4, *then* em q_8, *for* em q_{11} e *float* em q_{15}. Em cada estado há apenas uma hipótese de transição para cada entrada, faltando apenas o estado morto e respetivas transições para que o DFA esteja completo.

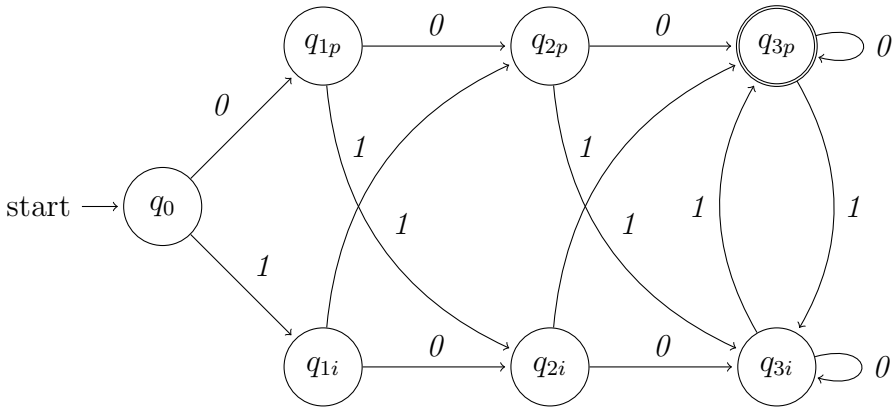
Para a representação formal do DFA, temos apenas de contruir o tuplo que o representa, ficando com

DFA A = ($\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}\}$, $\{$a,e,f,h,i,l,n,o,r,t$\}$, δ, q_0, $\{q_2, q_4, q_8, q_{11}, q_{15}\}$), em que δ representa as seguintes transições: $\delta(q_0, i) = q_1$; $\delta(q_1, f) = q_2$; $\delta(q_1, n) = q_3$; $\delta(q_3, t) = q_4$; $\delta(q_0, t) = q_5$; $\delta(q_5, h) = q_6$; $\delta(q_6, e) = q_7$; $\delta(q_7, n) = q_8$; $\delta(q_0, f) = q_9$; $\delta(q_9, o) = q_{10}$; $\delta(q_{10}, r) = q_{11}$; $\delta(q_9, l) = q_{12}$; $\delta(q_{12}, o) = q_{13}$; $\delta(q_{13}, a) = q_{14}$; $\delta(q_{14}, t) = q_{15}$.

Exercício 8 (TPC 2012/13)

Desenhe um DFA que reconheça cadeias sobre o alfabeto $\Sigma = \{0, 1\}$, de tamanho maior ou igual a 3, e com um número par de 1's.

Para manter informação de tamanho da cadeia e de paridade de 1's, precisamos de 6 estados (mais o estado inicial, para tamanho 0). Uma possível solução será então:

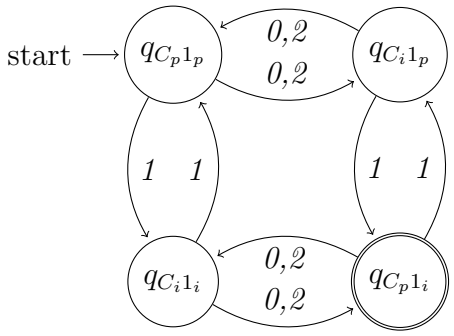


Os nomes dos estados indicam número de símbolos já lidos, e a paridade dos 1's. Como podemos ver, a parte de cima contém os estados com um número par de 1's, e os estados de baixo com um número ímpar de 1's, sendo que as transições com 0 mantém-se na metade respetiva do autómato, e as transições com 1 trocam entre a parte de cima e a de baixo do autómato. Olhando da esquerda para a direita, podemos ver o aumento do número de símbolos já lidos, sendo que o estado de aceitação será apenas quando já temos 3 ou mais símbolos lidos, e o número de 1's é par.

Exercício 9 (TPC 2013/14)

Desenhe um DFA completo que permita reconhecer cadeias no alfabeto $\Sigma = \{0, 1, 2\}$ de comprimento par e com um número ímpar de 1's.

Para este exercício, necessitamos novamente de verificar duas condições: paridade do comprimento da cadeia e paridade dos 1's. A introdução de um 0 ou um 2 altera apenas a paridade do comprimento da cadeia, mas mantém a paridade dos 1's. Já a introdução de um 1 altera não só a paridade dos 1's na cadeia como também a paridade do comprimento da cadeia. Uma solução possível será então:

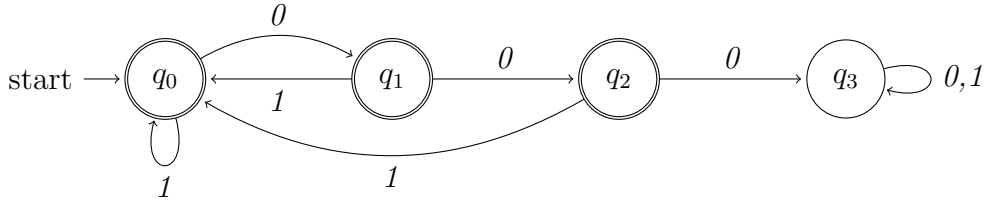


Os estado indicam a paridade do comprimento (C_p para comprimento par e C_i para comprimento ímpar) e do número de 1's (1_p para número par de 1's e 1_i para número ímpar de 1's). Como podemos ver, os estados de cima têm número par de 1's, e os estados de baixo têm um número ímpar de 1's. Assim, as transições com 0 e 2 mantém-se na sua parte (cima/baixo), alterando apenas a paridade do comprimento da cadeia. As transições com 1 alteram a paridade tanto do comprimento da cadeia como dos 1's.

Exercício 10 (Desafio 2015/16)

Modele um DFA completo que permita reconhecer cadeias binárias em que cada sub-cadeia de comprimento 3 não tenha mais de dois zeros.
Ex.: 1001 pertence à linguagem ; 10101 pertence à linguagem ; 10001 não pertence à linguagem

Uma forma simples de resolver este exercício é reformular o enunciado: dizer que para uma cadeia pertencer à linguagem, cada sub-cadeia de comprimento 3 não pode ter mais do que dois zeros, será equivalente a dizer que não podem existir sub-cadeias de comprimento 3 com 3 zeros, ou seja, não podem existir 3 zeros consecutivos. Apesar de ser a mesma linguagem, esta formulação aponta-nos logo para uma possível solução: um DFA que reconheça a sequência '000', e que aceite as cadeias até esse ponto, deixando de as aceitar caso a sequência seja reconhecida. Olhando então para uma possível resolução:

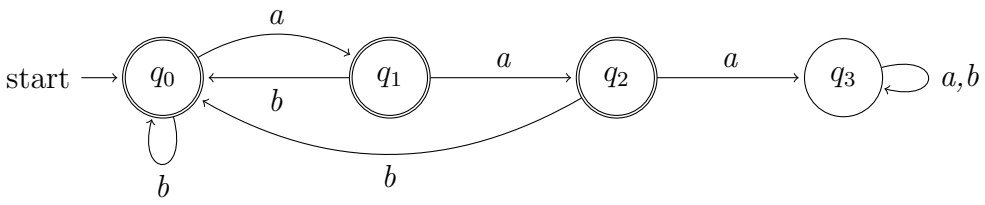


Como podemos ver, só não são aceites as cadeias que tenham chegado ao estado q_3, e para isso é necessário que existam pelo menos 3 zeros consecutivos. Assim, consegue-se rejeitar essas cadeias, e aceitar as restantes, ou seja, aquelas em que em cada sub-cadeia de 3 símbolos, não existem mais de dois zeros.

Exercício 11 (Exercício 2015/16)

Desenhe um DFA completo que reconheça cadeias no alfabeto a, b em que cada 'aa' é seguido imediatamente de um 'b'.

Para este exercício, uma forma mais simples de resolver será novamente reformular o enunciado, e perceber que a linguagem não aceita cadeias com 3 a's consecutivos, aceitando todas as outras cadeias. Uma possível solução será então:



Se repararmos, este autómato é estruturalmente igual ao DFA obtido no exercício anterior, sendo que apenas as transições apresentam símbolos diferentes. Conseguimos então verificar que os dois exercícios são formulações diferentes (e com alfabeto diferente) para o mesmo problema.

Exercício Proposto 1 Desenhe um DFA que aceite a linguagem das cadeias binárias com pelo menos dois 0s consecutivos e que não contenha dois 1s consecutivos.

Exercício Proposto 2 Desenhe um DFA que reconheça cadeias no alfabeto Σ={0, 1, 2, 3, 4, 5, 6, 7, 8, 9} em que a soma dos seus dígitos seja um múltiplo de 3. Exemplo de cadeias aceites: ε, 30, 45, 81.