Autômatos Finitos e Não-determinismo

Área de Teoria DCC/UFMG

Fundamentos de Teoria da Computação

2020/01

Autômatos Finitos e Não-determinismo: Introdução

- Como já discutimos, uma pergunta central da teoria da computação é:

  "O que é um computador?"

- Mas computadores são complexos demais para que desenvolvamos uma teoria matemática para eles diretamente.
- A solução é utilizar um **modelo computacional**, que é um computador idealizado em que apenas detalhes relevantes são representados.
- Vamos começar agora o estudo de um modelo computacional extremamente simples, a **máquina de estados finitos**, ou **autômato finito**.
- Em particular, vamos ver como incorporar a estes autômatos **não-determinismo**, uma poderosa ferramenta da teoria da computação.
- Os autômatos finitos definem a classe das **linguagens regulares**.

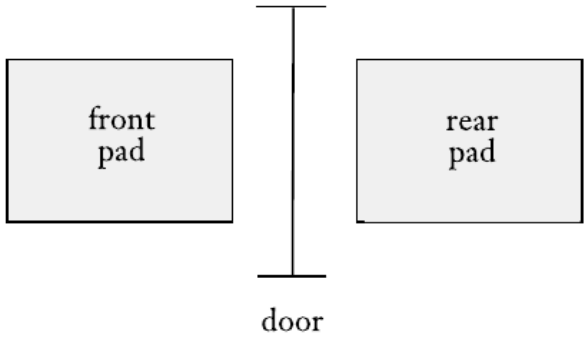
Autômatos Finitos

Autômatos Finitos: Motivação

- **Autômatos finitos** são bons modelos para computadores com uma quantidade muito limitada de memória.

  Apesar de extremamente simples, tais computadores são muito relevantes.

- Exemplo 1 Considere um controlador de porta automática, que tem dois tapetes (frontal e posterior) para detectar a presença de pessoas.



O controlador está em um de dois estados:

- OPEN: porta aberta.
- CLOSED: porta fechada.

Há quatro condições de entrada possíveis:

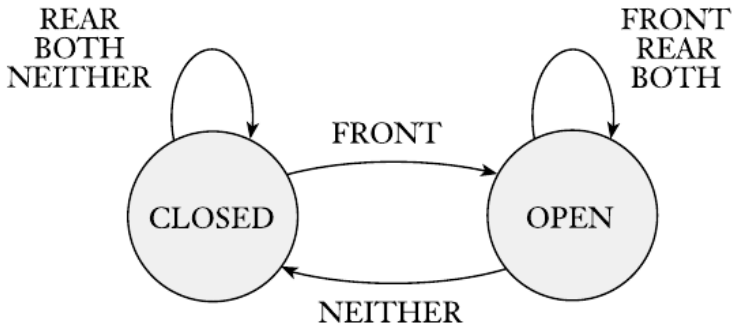
- FRONT: alguém está pisando no tapete da frente.
- REAR: alguém está pisando no tapete traseiro.
- BOTH: pessoas estão pisando em ambos os tapetes.
- NEITHER: ninguém está pisando em nenhum tapete.

Autômatos Finitos: Motivação

- Exemplo 1 (Continuação)

Podemos representar o funcionamento do controlador de várias formas:

- **Diagrama de estados**:

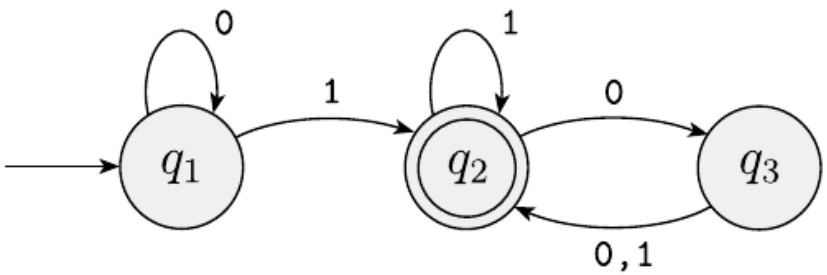


- **Tabela de transição**:

	input signal			
state	NEITHER	FRONT	REAR	BOTH
CLOSED	CLOSED	OPEN	CLOSED	CLOSED
OPEN	CLOSED	OPEN	OPEN	OPEN

Autômatos Finitos

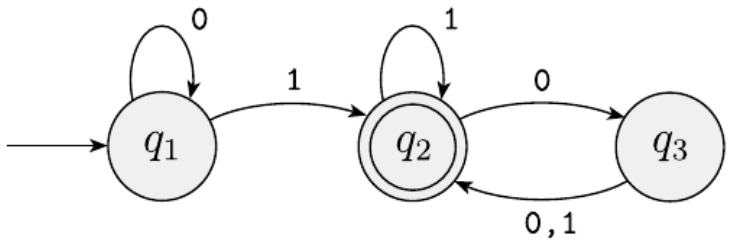
- O controlador do exemplo anterior foi modelado como um autômato finito.
- Vamos introduzir os componentes de um autômato finito com um exemplo.
- Exemplo 2 Considere o **diagrama de estados** do autômato finito M_1 abaixo, que tem três estados.



  - O autômato tem três **estados**: q_1, q_2 e q_3.
  - O **estado inicial** é q_1, indicado pela seta apontando para ele vinda do nada.
  - O **estado de aceitação** (ou **estado final**) é q_2, indicado por um círculo duplo.
  - As setas entre estados são **transições**.

- Exemplo 2 (Continuação)

Considere ainda o mesmo autômato finito M_1:



- Quando o autômato recebe uma cadeia de entrada (como 1101), ele a processa e produz uma de duas saídas possíveis: **aceita** ou **rejeita**.

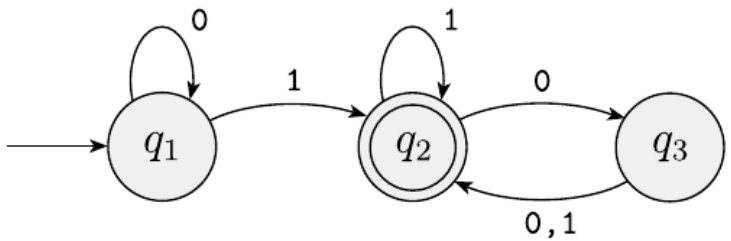
  Se o autômato finito M_1 recebe a entrada 1101, o processamento é:

  1. Começa no estado inicial q_1.
  2. Lê o símbolo 1 da entrada, e segue de q_1 para q_2.
  3. Lê o símbolo 1 da entrada, e segue de q_2 para q_2.
  4. Lê o símbolo 0 da entrada, e segue de q_2 para q_3.
  5. Lê o símbolo 1 da entrada, e segue de q_3 para q_2.
  6. Aceita porque M_1 está no estado de aceitação q_2 no final da entrada.

Autômatos Finitos

- Exemplo 2 (Continuação)

Considere ainda o mesmo autômato finito M_1:



Experimentando com M_1, esta máquina aceita ou rejeita as cadeias:

- 1: *aceita*
- 01: *aceita*
- 11: *aceita*
- 0101010101: *aceita*

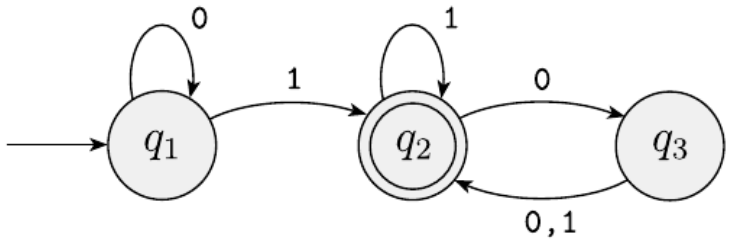
Logo, podemos concluir que:

M_1 aceita qualquer cadeia terminada com 1, porque a máquina sempre se move para o estado de aceitação q_2 ao ler o símbolo 1.

Autômatos Finitos

- Exemplo 2 (Continuação)

Considere ainda o mesmo autômato finito M_1:



Experimentando com M_1, esta máquina aceita ou rejeita as cadeias:

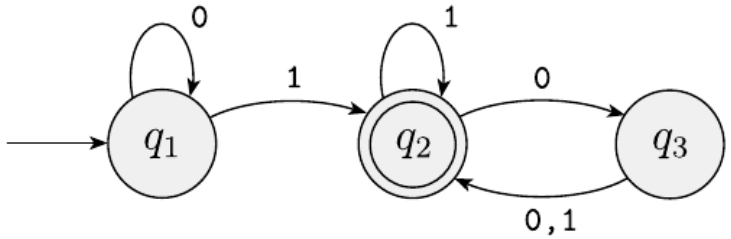
- 100: *aceita*
- 0100: *aceita*
- 110000: *aceita*
- 0101000000: *aceita*

Logo, podemos concluir que:

M_1 também aceita qualquer cadeia que tem pelo menos um 1 e que termina com um número par de 0s seguindo o último 1.

- Exemplo 2 (Continuação)

Considere ainda o mesmo autômato finito M_1:



Experimentando com M_1, esta máquina aceita ou rejeita as cadeias:

- 0: *rejeita*
- 10: *rejeita*
- 101000: *rejeita*
- 1000: *rejeita*

Logo, podemos concluir que:

M_1 rejeita qualquer cadeia que tem pelo menos um 1 e que termina com um número ímpar de 0s seguindo o último 1.

Qual a linguagem reconhecida por M_1?

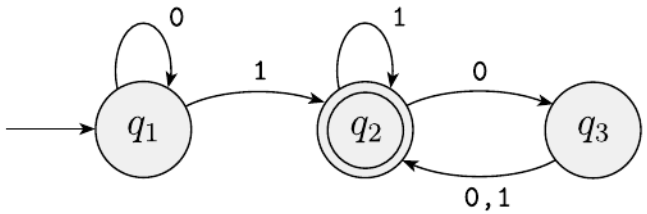
Veremos a resposta em breve...

Autômatos Finitos: Definição formal

- Um **autômato finito (determinístico) (AFD)** é uma 5-tupla $(Q, \Sigma, \delta, q_0, F)$, onde:
  1. Q é um conjunto finito de **estados**,
  2. Σ é um conjunto finito chamado de **alfabeto**,
  3. $\delta : Q \times \Sigma \to Q$ é uma **função de transição**,
  4. $q_0 \in Q$ é o **estado inicial**, e
  5. $F \subseteq Q$ é o conjunto de **estados de aceitação** (ou **estados finais**).

Autômatos Finitos: Definição formal

- **Exemplo 3** Considere o diagrama de estados do autômato finito M_1:



Formalmente, este autômato é uma 5-tupla $M_1 = (Q, \Sigma, \delta, q_0, F)$, onde:

1. o conjunto de estados é $Q = \{q_1, q_2, q_3\}$,
2. o alfabeto é $\Sigma = \{0, 1\}$,
3. A função de transição δ é descrita como

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

,

4. O estado inicial é $q_1 \in Q$, e
5. O conjunto de estados de aceitação é $F = \{q_2\}$.

◁

Autômatos Finitos: Linguagem reconhecida

- Se A é o conjunto de todas as cadeias que uma máquina M aceita, dizemos que A **é a linguagem reconhecida pela máquina** M.

  Denotamos a linguagem de uma máquina por $L(M) = A$.

- Uma máquina pode aceitar várias cadeias, mas ela só reconhece uma única linguagem (i.e., o conjunto de todas as cadeias aceitas).

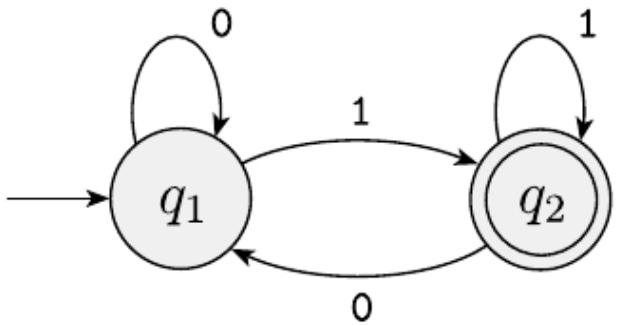
  Se uma máquina não aceita nenhuma cadeia, ela reconhece a linguagem vazia $\emptyset$.

- Exemplo 4 Retomando novamente o autômato M_1, podemos dizer que a linguagem reconhecida por esta máquina é:

$$L(M_1) = \{w \in \{0,1\}^* \mid w \text{ contém pelo menos um 1 e um número par de zeros segue o último 1}\}.$$

Autômatos Finitos: Exemplos

- **Exemplo 5** Considere o diagrama de estados do autômato finito M_2:



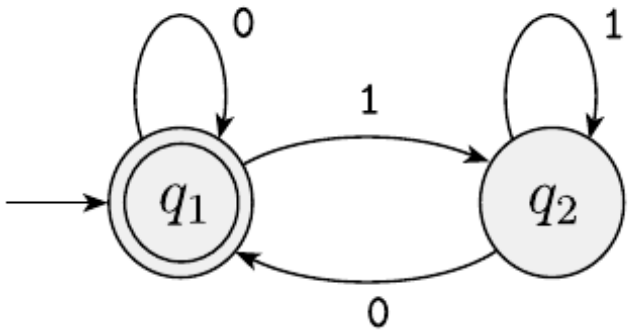
Podemos dizer que a linguagem reconhecida por este autômato é

$$L(M_2) = \{w \in \{0,1\}^* \mid |w| > 0 \text{ e } w \text{ termina com } 1\}.$$

◁

Autômatos Finitos: Exemplos

- **Exemplo 6** Considere o diagrama de estados do autômato finito M_3:



A máquina M_3 é semelhante á máquina M_2, exceto pelo estado de aceitação.

Podemos dizer que a linguagem reconhecida por este autômato é

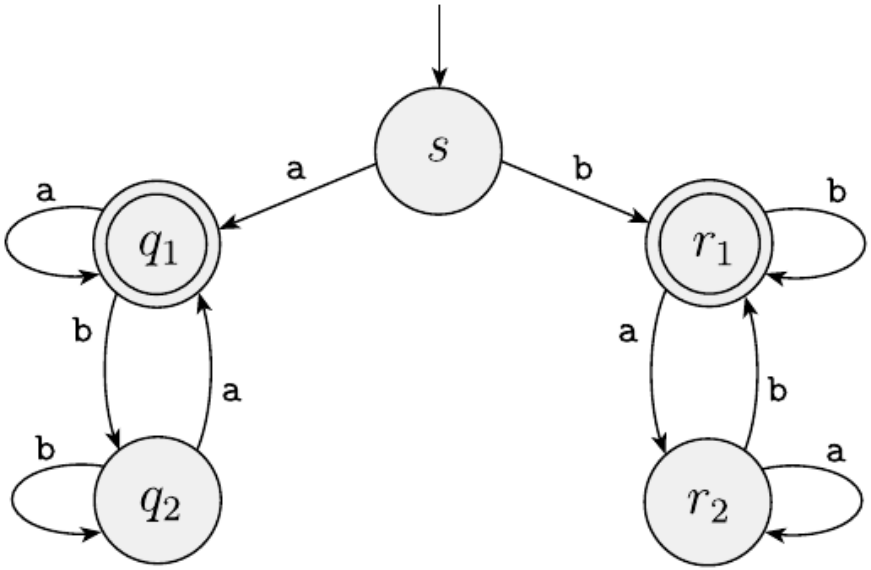
$$L(M_3) = \{w \in \{0,1\}^* \mid w \text{ é a cadeia vazia, ou } w \text{ termina com } 0\}.$$

Note que a linguagem reconhecida por M_3 pode também ser descrita como:

$$L(M_3) = \{w \in \{0,1\}^* \mid w \text{ não termina com } 1\}.$$

Autômatos Finitos: Exemplos

- **Exemplo 7** Considere o diagrama de estados do autômato finito M_4:

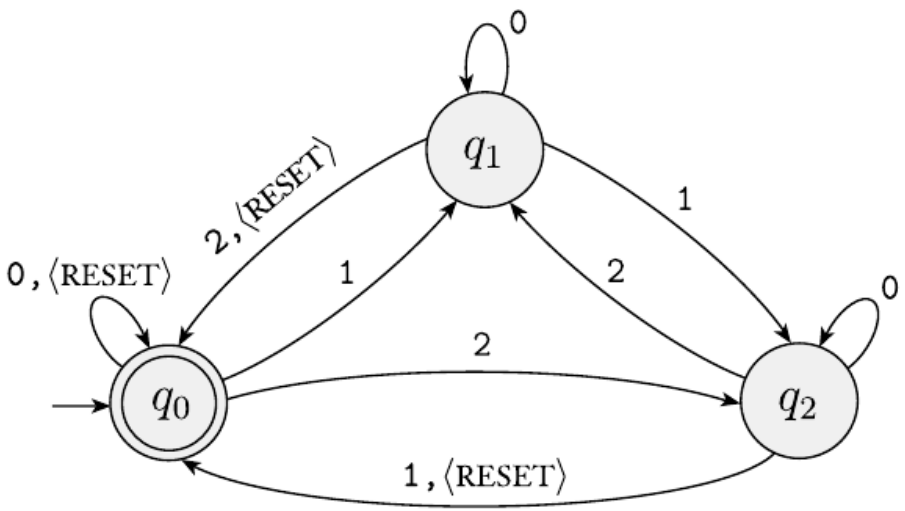


Podemos dizer que a linguagem reconhecida por este autômato é

$$L(M_4) = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w| > 0 \text{ e } w \text{ começa e termina com o mesmo símbolo}\}.$$

Autômatos Finitos: Exemplos

- **Exemplo 8** Considere o diagrama de estados do autômato finito M_5:



Podemos dizer que a linguagem reconhecida por este autômato é

$$L(M_5) = \{w \in \{\langle \text{RESET} \rangle, 0, 1, 2\}^* \mid \text{a soma dos dígitos de } w \text{ após o último } \langle \text{RESET} \rangle \text{ é múltiplo de } 3\}.$$

Autômatos Finitos: Exemplos

- **Exemplo 9** Considere uma generalização do autômato finito M_5 do exemplo anterior, usando o mesmo alfabeto de entrada $\Sigma = \{\langle \text{RESET} \rangle, 0, 1, 2\}$.

  Para cada $i \geq 1$, seja A_i a linguagem de todas as cadeias em que a soma dos números é um múltiplo de i, exceto que a soma é reiniciada para 0 sempre que o símbolo $\langle \text{RESET} \rangle$ aparece.

  Para cada linguagem A_i, damos um autômato finito B_i reconhecendo A_i da seguinte forma:

$$B_i = (Q_i, \Sigma, \delta_i, q_0, \{q_0\}), \qquad \text{onde}$$

  - $Q_i = \{q_0, q_1, \ldots, q_{i-1}\}$ é o conjunto de estados, e
  - a função de transição δ_i é tal que para todo $q_j \in Q_i$:
    - $\delta_i(q_j, 0) = q_j$,
    - $\delta_i(q_j, 1) = q_k$, onde $k = j + 1$ módulo i,
    - $\delta_i(q_j, 2) = q_k$, onde $k = j + 2$ módulo i, e
    - $\delta_i(q_j, \langle \text{RESET} \rangle) = q_0$.

Autômatos Finitos: Definição formal de computação

- Até agora vimos, intuitivamente, como ocorre a computação de autômato finito. Agora vamos formalizar esta intuição do conceito de computação.
- Seja $M = (Q, \Sigma, \delta, q_0, F)$ um autômato finito, e suponha que $w = w_1 w_2 \ldots w_n$ seja uma cadeia em que cada $w_i \in \Sigma$.

  Então o autômato finito M aceita a cadeia w se existe uma sequência de estados $r_0, r_1, \ldots, r_n \in Q$ satisfazendo as três condições abaixo:

  1. $r_0 = q_0$,

     (Esta condição diz que a máquina começa no estado inicial.)

  2. $\delta(r_i, w_{i+1}) = r_{i+1}$, para $i = 0, 1, \ldots, n-1$, e

     (Esta condição diz que a máquina transita conforme a função de transição.)

  3. $r_n \in F$.

     (Esta condição diz que a máquina pára em estado de aceitação.)

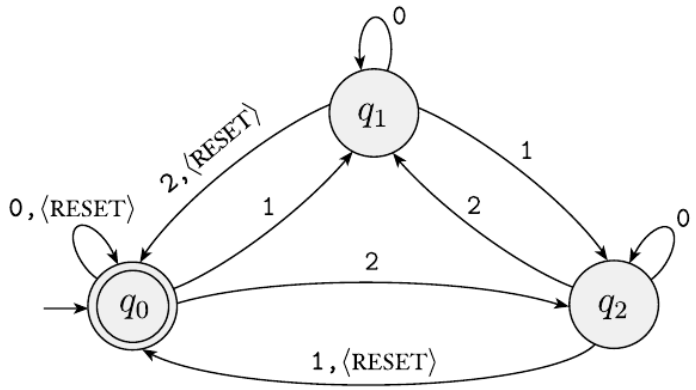
Linguagens Regulares

- Dizemos que a máquina M **reconhece a linguagem** A se

$$A = \{w \mid M \text{ aceita } w\}.$$

- Uma linguagem é chamada de **linguagem regular** se algum autômato finito a reconhece.

- **Exemplo 10** Consideremos novamente o autômato M_5.



Considere a cadeia:

$$w = 10\langle\text{RESET}\rangle 22\langle\text{RESET}\rangle 012$$

M_5 aceita w, pois a sequência de estados visita pela máquina ao computar w é:

$$q_0, q_1, q_1, q_0, q_2, q_1, q_0, q_1, q_0,$$

que satisfaz as 3 condições.

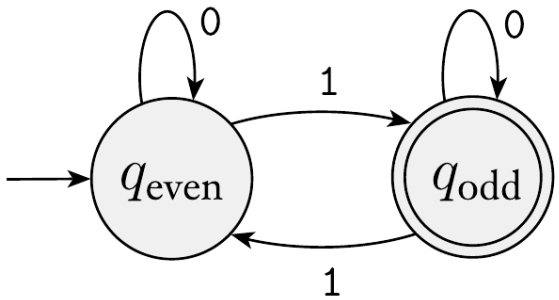
Além disso, a linguagem $L(M_5)$ é uma linguagem regular, pois é reconhecida por um autômato finito. ◁

Projetando Autômatos Finitos

- Projetar um autômato finito, assim como projetar uma obra de arte, é um processo criativo.
- Apenas com prática, atenção e empenho podemos ser capazes de projetar autômatos finitos com facilidade.
- Vamos praticar isto agora!
- Exemplo 11 Projete um autômato finito que reconheça a linguagem

$$L = \{w \in \{0,1\}^* \mid w \text{ tem um número ímpar de 1s}\}.$$

Solução.

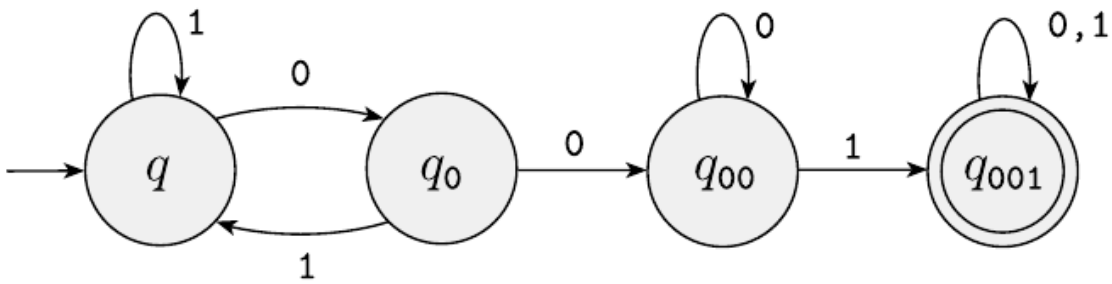


Projetando Autômatos Finitos

- **Exemplo 12** Projete um autômato finito que reconheça a linguagem

$$L = \{w \in \{0,1\}^* \mid w \text{ contém a subcadeia } 001\}.$$

Solução.



◁

Projetando Autômatos Finitos

- Exemplo 13 Projete um autômato finito que reconheça cadeias binárias que, quando interpretadas como números, são divisíveis por 6. Ou seja,

$$L = \{w \in \{0,1\}^* \mid w \text{ representa um binário divisível por } 6\}.$$

Por exemplo:

1. $110 \in L$,
2. $000000 \in L$,
3. $1 \notin L$,
4. $00111 \notin L$.

Solução.

Para construir um autômato para L, vamos usar 6 estados correspondendo aos restos possíveis de um número na divisão por 6: 0, 1, 2, 3, 4, ou 5.

O estado inicial do autômato é 0, pois a cadeia vazia ϵ representa 0 e, portanto, tem resto 0 na divisão por 6.

O (único) estado final do autômato também é 0, pois a cadeia deve ser aceita se, e somente se, representa um binário divisível por 6.

Projetando Autômatos Finitos

- Exemplo 13 (Continuação)

  Assuma agora que o AFD já processou um prefixo x de w, determinando que o resto da divisão de x por 6 é r.

  Então, ao ler um novo símbolo de entrada:

  - Se o símbolo for 0, o prefixo consumido passa a ser $x0$, que, em binário, é o dobro do número representado por x.

    Logo o resto de $x0$ por 6 é o dobro do resto de x por 6 (mod 6).

    Isto quer dizer que, ao ler um 0, o AFD deve transitar do estado r para o estado $2r \pmod 6$.

  - Se o símbolo for 1, o prefixo consumido passa a ser $x1$, que, em binário, é o dobro do número representado por x mais 1.

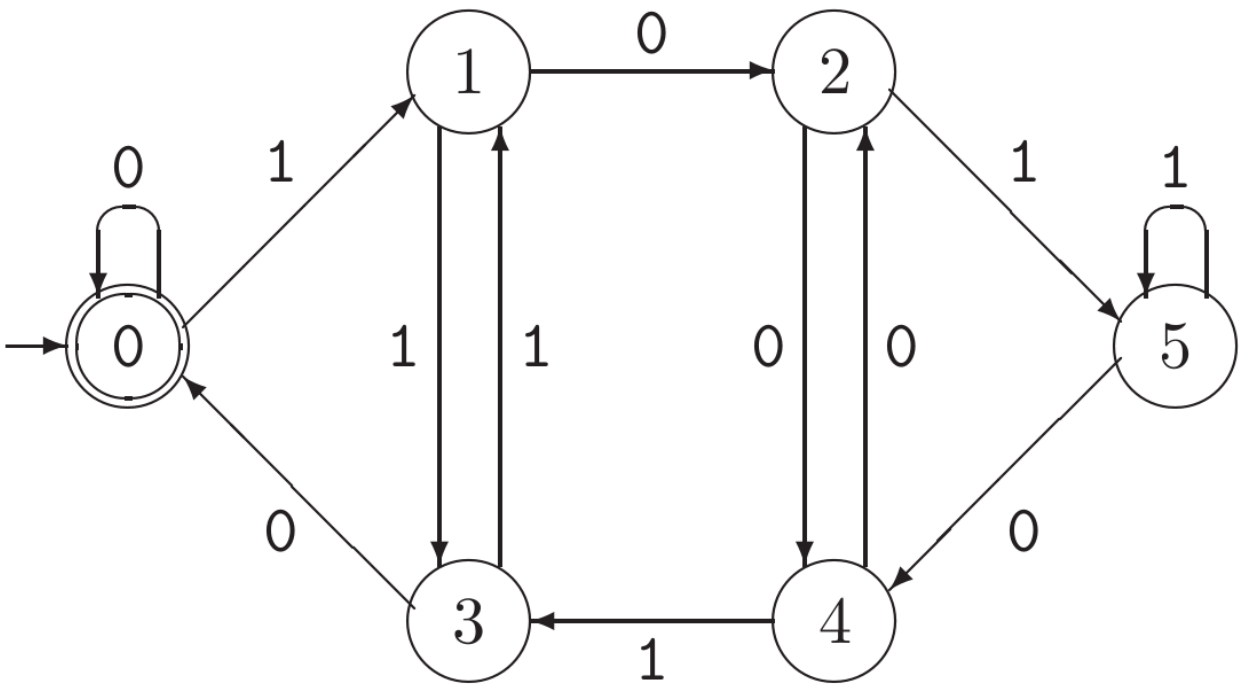
    Logo o resto de $x0$ por 6 é o dobro mais 1 do resto de x por 6 (mod 6).

    Isto quer dizer que, ao ler um 0, o AFD deve transitar do estado r para o estado $2r + 1 \pmod 6$.

Projetando Autômatos Finitos

- Exemplo 13 (Continuação)

Assim, o AFD que reconhece a linguagem L é o seguinte:



Minimização de AFDs

Minimização de AFDs

- Quando projetamos um AFD, uma pergunta relevante é se construímos aquele com o menor número de estados possível.
- Dois AFDs M e M' são **equivalentes** se eles reconhecem a mesma linguagem, ou seja,

$$L(M) = L(M').$$

- Um **AFD mínimo** para uma linguagem é aquele que, dentre todos os AFDs equivalentes para a mesma linguagem, possui o menor número de estados.
- Para construir um AFD mínimo para uma linguagem L, nós podemos:
  1. Construir um AFD M que reconheça L.
  2. Aplicar um algoritmo de minimização a M, obtendo um AFD equivalente mínimo M' para reconhecer L.

Minimização de AFDs

- O algoritmo de minimização de AFDs é baseado no conceito de estados equivalentes, ou seja, que se comportam identicamente.
- Dois estados e e e' são **equivalentes** se o resultado final (aceitação ou rejeição) do processamento de qualquer cadeia $w \in \Sigma^*$ é o mesmo se iniciado a partir e ou a partir de e'.

  Mais formalmente, dois estados e e e' são equivalentes se

  $$\forall w \in \Sigma^* : \quad w \text{ é aceita a partir de } e \iff w \text{ é aceita a partir de } e'.$$

- O algoritmo de minimização funciona ao unir todos os estados equivalentes entre si (o que não altera o funcionamento do autômato, uma vez que estados equivalentes se comportam de maneira idêntica para toda cadeia).

  Já estados não-equivalentes não podem ser unidos, e permanecem separados.

Minimização de AFDs

- A questão central do algoritmo de minimização é, então, determinar quais estados de um AFD são equivalentes.
- Note que, pela definição, se dois estados e e e' não são equivalentes, então existe pelo menos uma cadeia $w \in \Sigma^*$ tal que w é aceita ao ser processada a partir de e, mas rejeitada ao ser processada a partir de e' (ou vice-versa).
- Para determinar quais estados não são equivalentes, o algoritmo:

  1. Inicialmente assume que todos os estados são equivalentes entre si, e os agrupa.
  2. Iterativamente identifica cadeias $w \in \Sigma^*$ que atestam que alguns estados não são equivalentes, e separa estes estados.
  3. Quando nenhum estado mais pode ser separado, sabe-se que somente os estados equivalentes permanecem juntos, e com eles se constrói o autômato equivalente minimizado.

Minimização de AFDs

- Mais precisamente, inicialmente o algoritmo particiona o conjunto de estados Q do AFD em um bloco de estados finais F, e um bloco de estados não-finais $Q - F$.

  Note que cada estado em F não pode ser equivalente a nenhum estado em $Q - F$, porque a cadeia $\epsilon \in \Sigma^*$ é aceita a partir de qualquer estado em F, mas rejeitada a partir de qualquer estado em $Q - F$.

- Em seguida, o algoritmo particiona mais os estados, iterativamente.

- Dois estados e e e' são n-**distinguíveis** se existe uma cadeia $s \in \Sigma^*$ de tamanho n tal que s é aceita a partir de e, mas rejeitada a partir de e' (ou vice-versa).

- O algoritmo procura uma partição em que estados 1,2,3,...-distinguíveis ficam em blocos diferentes e que usa o número mínimo de blocos.

Minimização de AFDs

- O algoritmo de minimização funciona, então, assim:

  1. Produza uma partição $\mathcal{P}_0 = \{F, Q - F\}$ do conjunto de estados Q, em que estados finais são separados dos não-finais.
  2. Para cada bloco de estados B na partição $\mathcal{P}_i$, cada símbolo s do alfabeto Σ, e cada par de estados e, e' contidos no bloco B:

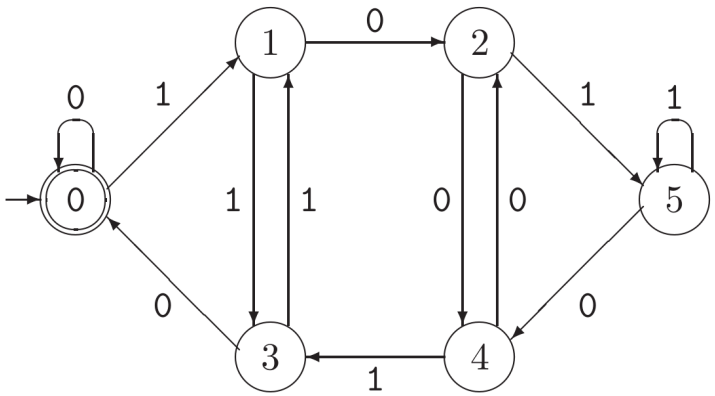
     a) Sejam $d = \delta(e, s)$ e $d' = \delta(e', s)$ os estados para os quais o AFD transita quando lê o símbolo s a partir dos estados e e e', respectivamente.

     Se d e d' pertencem a blocos diferentes na partição $\mathcal{P}_i$, então os estados e e e' não são equivalentes e devem ser separados na partição $\mathcal{P}_{i+1}$.

  3. Se a partição $\mathcal{P}_{i+1}$ for diferente da partição $\mathcal{P}_i$, repita o passo (2).
  4. O autômato mínimo é construído de tal forma que seus estados são os blocos da última partição $\mathcal{P}_{i+1}$ produzida.

Minimização de AFDs

- Exemplo 14 Usando o algoritmo de minimização explicado acima, minimize o AFD para a linguagem dos binários divisíveis por 6.



Solução.

Vamos aplicar o algoritmo de minimização para identificar os estados equivalentes no AFD e, com isso, construir o AFD equivalente mínimo.

1. Primeiro criamos a partição inicial dos estados, separando estados finais de não finais:

$$\mathcal{P}_0 = \{\{0\}, \{1, 2, 3, 4, 5\}\}.$$

Minimização de AFDs

- **Exemplo 14** (Continuação)

2. Agora, iterativamente, procuramos por estados não equivalentes.

   a) Na partição $\mathcal{P}_0 = \{\{0\}, \{1,2,3,4,5\}\}$, notamos que o bloco $\{0\}$ não pode mais ser separado, logo só precisamos considerar o bloco $\{1,2,3,4,5\}$.

   Aplicamos os símbolos 0 e 1 a cada estado do bloco e verificamos a qual bloco da partição $\mathcal{P}_0$ o resultado pertence:

0	1	2	3	4	5
	$\downarrow 0$ $2 \in \{1,2,3,4,5\}$	$\downarrow 0$ $4 \in \{1,2,3,4,5\}$	$\downarrow 0$ $0 \in \{0\}$	$\downarrow 0$ $2 \in \{1,2,3,4,5\}$	$\downarrow 0$ $4 \in \{1,2,3,4,5\}$
	$\downarrow 1$ $3 \in \{1,2,3,4,5\}$	$\downarrow 1$ $5 \in \{1,2,3,4,5\}$	$\downarrow 1$ $1 \in \{1,2,3,4,5\}$	$\downarrow 1$ $3 \in \{1,2,3,4,5\}$	$\downarrow 1$ $5 \in \{1,2,3,4,5\}$

   Note que o estado 3 é distinguível dos estados $1, 2, 4$ e 5 por palavras de tamanho 1, pois sob o símbolo 0 o estado 3 transita para o bloco $\{0\}$, enquanto sob o símbolo 0 os estados $1, 2, 4$ e 5 transitam para o bloco $\{1,2,3,4,5\}$.

   Logo precisamos separar o estado 3 dos demais no bloco $\{1,2,3,4,5\}$, fazendo com que a próxima partição seja

$$\mathcal{P}_1 = \{\{0\}, \{1,2,4,5\}, \{3\}\}.$$

Minimização de AFDs

- Exemplo 14 (Continuação)

2. Continuamos a, iterativamente, procurar por estados não equivalentes.

   b) Na partição $\mathcal{P}_1 = \{\{0\}, \{1,2,4,5\}, \{3\}\}$, notamos que os blocos $\{0\}$ e $\{3\}$ não podem mais ser separados, logo só precisamos considerar o bloco $\{1,2,4,5\}$.

   Aplicamos os símbolos 0 e 1 a cada estado do bloco e verificamos a qual bloco da partição $\mathcal{P}_1$ o resultado pertence:

0	1	2	4	5	3
	$\downarrow 0$ $2 \in \{1,2,4,5\}$	$\downarrow 0$ $4 \in \{1,2,4,5\}$	$\downarrow 0$ $2 \in \{1,2,4,5\}$	$\downarrow 0$ $4 \in \{1,2,4,5\}$	
	$\downarrow 1$ $3 \in \{3\}$	$\downarrow 1$ $5 \in \{1,2,4,5\}$	$\downarrow 1$ $3 \in \{3\}$	$\downarrow 1$ $5 \in \{1,2,4,5\}$	

   Note que os estados 1 e 4 são distinguíveis dos estados 2 e 5 por palavras de tamanho 2, pois sob o símbolo 1 os estados 1 e 4 transitam para o bloco $\{3\}$, enquanto sob o símbolo 1 os estados 2 e 5 transitam para o bloco $\{1,2,4,5\}$.

   Logo precisamos separar os estado 1 e 4 dos estados 2 e 5 no bloco $\{1,2,4,5\}$, fazendo com que a próxima partição seja

$$\mathcal{P}_2 = \{\{0\}, \{1,4\}, \{2,5\}, \{3\}\}.$$

- Exemplo 14 (Continuação)

2. Continuamos a, iterativamente, procurar por estados não equivalentes.

   c) Na partição $\mathcal{P}_2 = \{\{0\}, \{1,4\}, \{2,5\}, \{3\}\}$, notamos que os blocos $\{0\}$ e $\{3\}$ não podem mais ser separados, logo só precisamos considerar os blocos $\{1,4\}$ e $\{2,5\}$.

   Aplicamos os símbolos 0 e 1 a cada estado de cada bloco e verificamos a qual bloco da partição $\mathcal{P}_2$ o resultado pertence:

0	1	4	2	5	3
	$\downarrow 0$ $2 \in \{2,5\}$	$\downarrow 0$ $2 \in \{2,5\}$	$\downarrow 0$ $4 \in \{1,4\}$	$\downarrow 0$ $4 \in \{1,4\}$	
	$\downarrow 1$ $3 \in \{3\}$	$\downarrow 1$ $3 \in \{3\}$	$\downarrow 1$ $5 \in \{2,5\}$	$\downarrow 1$ $5 \in \{2,5\}$	

   Note que os estados 1 e 4 não são distinguíveis entre si, nem os estados estados 2 e 5 são distinguíveis entre si.

   Logo não precisamos separar nenhum estado de nenhum bloco, fazendo com que a próxima partição seja

$$\mathcal{P}_3 = \{\{0\}, \{1,4\}, \{2,5\}, \{3\}\}.$$

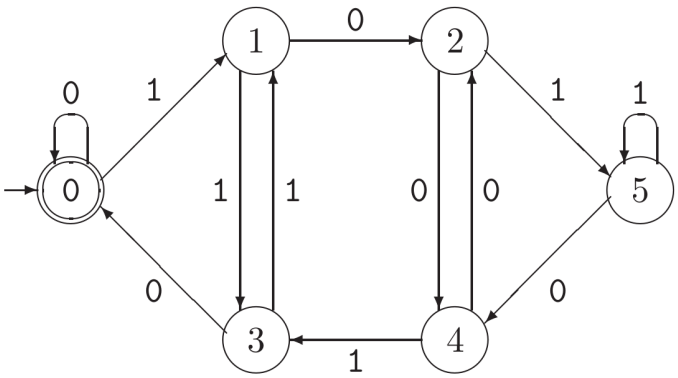
Minimização de AFDs

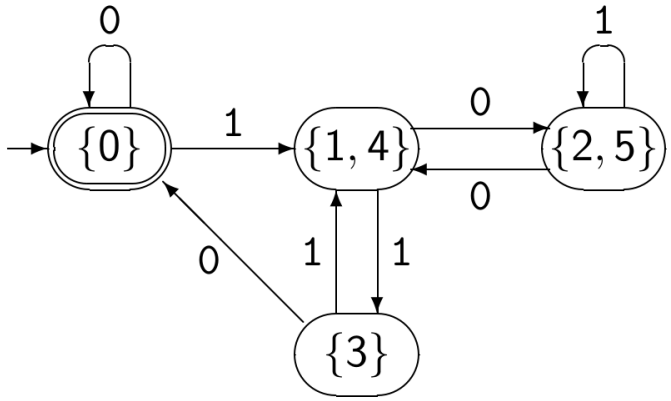
- **Exemplo 14** (Continuação)

3. Como a partição
$$\mathcal{P}_3 = \{\{0\}, \{1,4\}, \{2,5\}, \{3\}\},$$
é igual à partição anterior $\mathcal{P}_2$, sabemos que o processo de procurar estados equivalentes acabou.

   Logo, no AFD em questão o estado 1 é equivalente ao 4, e o estado 2 é equivalente ao 5.

4. Por fim, construímos o autômato mínimo de tal forma que seus estados sejam os blocos da última partição $\mathcal{P}_3$ produzida.





Operações Regulares

Operações regulares

- Até agora introduzimos autômatos finitos e linguagens regulares.

  Agora vamos estudar suas propriedades que funcionarão como uma "caixa de ferramentas" para mais tarde construir autômatos e reconhecer linguagens mais complicadas usando elementos mais simples.

- Na aritmética, os objetos básicos são os números e as ferramentas são operações para manipulá-los, como $+$ e $\times$.

  Na teoria da computação, os objetos básicos são as linguagens e as ferramentas incluem operações projetadas para manipulá-las.

- Vamos definir agora três operações chamadas de **operações regulares**.

Operações regulares

- Sejam A e B linguagens. Definimos as operações regulares **união**, **concatenação**, e **fecho de Kleene** (ou **estrela**) da seguinte forma:
  - **União**: $A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$.
  - **Concatenação**: $A \circ B = \{xy \mid x \in A \text{ e } y \in B\}$.
  - **Estrela (ou fecho de Kleene)**: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ e cada } x_i \in A\}$.

- Exemplo 15 Suponha que o alfabeto $\Sigma = \{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$ seja o alfabeto latino padrão de 26 letras.

  Se $A = \{\text{garoto}, \text{garota}\}$ e $B = \{\text{legal}, \text{ruim}\}$, então:
  - $A \cup B = \{\text{garoto}, \text{garota}, \text{legal}, \text{ruim}\}$.
  - $A \circ B = \{\text{garotolegal}, \text{garotoruim}, \text{garotalegal}, \text{garotaruim}\}$.
  - $A^* = \{\epsilon, \text{garoto}, \text{garota}, \text{garotogaroto}, \text{garotogarota}, \text{garotagaroto}, \text{garotagarota}, \text{garotogarotogaroto}, \text{garotogarotogarota}, \text{garotogarotagaroto}, \ldots\}$.

Fechamento de um Conjunto sob uma Operação

- Dizemos que um **conjunto A é fechado sob uma operação** op se ao aplicarmos a operação op a elementos de A, o resultado é também um elemento de A.

- Exemplo 16 Exemplos de fechamento de operações:

  1. O conjunto dos naturais $\mathbb{N}$ é fechado sob a operação de soma? SIM.
  2. O conjunto dos naturais $\mathbb{N}$ é fechado sob a operação de subtração? NÃO.
  3. O conjunto dos reais $\mathbb{R}$ é fechado sob a operação de multiplicação? SIM.
  4. O conjunto dos reais $\mathbb{R}$ é fechado sob a operação de divisão? NÃO.

$\triangleleft$

- Vamos mostrar que o conjunto de todas as linguagens regulares é fechado sob as três operações regulares.

  Isto vai ser extremamente útil num futuro próximo, porque vai nos possibilitar construir autômatos complexos a partir de autômatos mais simples.

Fechamento de Linguagens Regulares sob Operações Regulares

- **Teorema** A classe de linguagens regulares é fechada sob união.

  Demonstração (Intuição). Suponha que A_1 e A_2 sejam duas linguagens regulares. Então, pela definição de linguagem regular, sabemos que existem um autômato finito $M_1 = \{Q_1, \Sigma, \delta_1, q_1, F_1\}$ que reconhece A_1, e um autômato finito $M_2 = \{Q_2, \Sigma, \delta_2, q_2, F_2\}$ que reconhece A_2.

  Para mostrarmos que $A_1 \cup A_2$ é regular, basta mostrar que existe um autômato finito que reconhece $A_1 \cup A_2$.

  Vamos mostrar que tal autômato existe construindo a partir de M_1 e M_2 um novo autômato $M = \{Q, \Sigma, \delta, q_0, F\}$ para reconhecer $A_1 \cup A_2$ da seguinte forma:

  1. O conjunto de estados é $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ e } r_2 \in Q_2\}$, ou seja, o produto cartesiano $Q_1 \times Q_2$ dos estados de M_1 e M_2.
  2. O alfabeto Σ é o mesmo de M_1 e M_2. (Se tivéssemos alfabetos diferentes, bastaria tomar a união dos dois).

Fechamento de Linguagens Regulares sob Operações Regulares

- **Demonstração.** (Continuação)

  3. A função de transição δ é definida da seguinte forma. Para cada estado $(r_1, r_2) \in Q$ do novo autômato, e cada símbolo $a \in \Sigma$, faça

  $$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)).$$

  4. O estado inicial é o par (q_1, q_2) dos estados iniciais de M_1 e M_2.

  5. O conjunto de estados finais é

  $$F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ ou } r_2 \in F_2\},$$

  em que um dos elementos do par de estado é um estado de aceitação, seja em M_1 ou em M_2.

  Esta definição é equivalente a

  $$F = (F_1 \times Q_2) \cup (Q_1 \times F_2).$$

Fechamento de Linguagens Regulares sob Operações Regulares

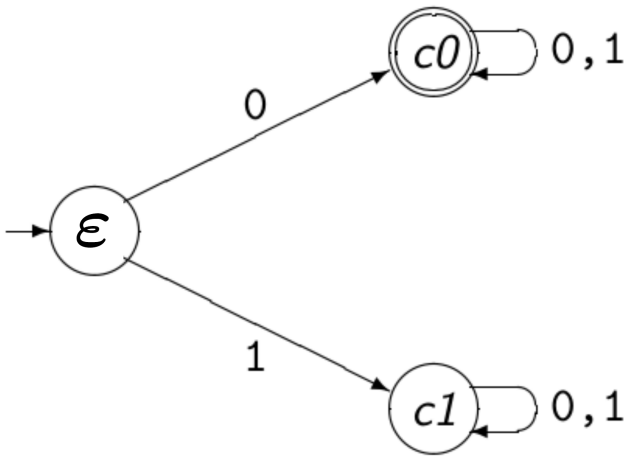
- **Exemplo 17** Sejam as linguagens

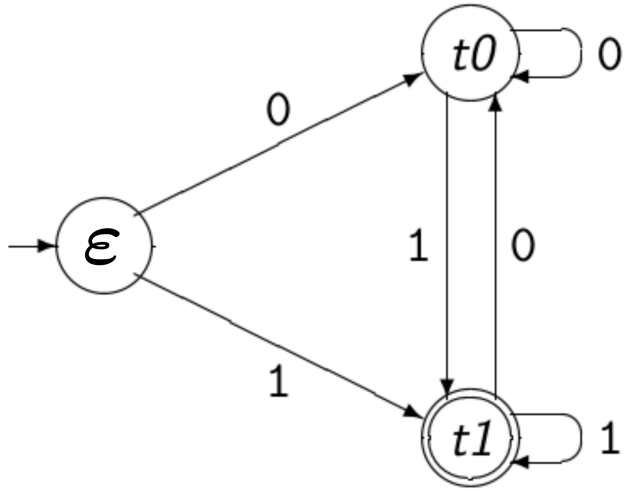
$$A = \{w \in \{0,1\}^* \mid |w| > 0 \text{ e } w \text{ começa com } 0\}, \quad \text{e}$$

$$B = \{w \in \{0,1\}^* \mid |w| > 0 \text{ e } w \text{ termina com } 1\}.$$

Usando o método do teorema anterior, construa um autômato finito para $A \cup B$.

Solução. Podemos construir autômatos finitos para as linguagens A e B como a seguir:

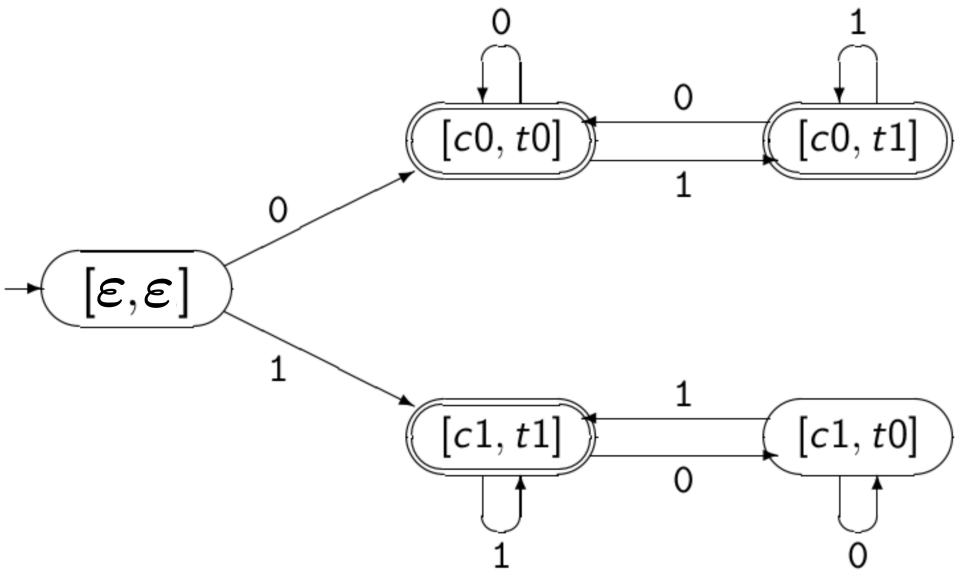




Fechamento de Linguagens Regulares sob Operações Regulares

- Exemplo 17 (Continuação)

E podemos combinar os autômatos usando o método do teorema anterior, para obter o seguinte autômato.



Note que a linguagem reconhecida por este autômato é

$$A \cup B = \{w \in \{0,1\}^* \mid w \text{ começa com 0 ou termina com 1}\}.$$

Fechamento de Linguagens Regulares sob Operações Regulares

- **Teorema** A classe de linguagens regulares é fechada sob a operação de concatenação.

  Demonstração. Para fazer esta demonstração, vamos precisar introduzir o conceito de não-determinismo.

  Vamos voltar para completar esta demonstração mais à frente. □

- **Teorema** A classe de linguagens regulares é fechada sob a operação de fecho de Kleene.

  Demonstração. Para fazer esta demonstração, vamos precisar introduzir o conceito de não-determinismo.

  Vamos voltar para completar esta demonstração mais à frente. □

Não-determinismo

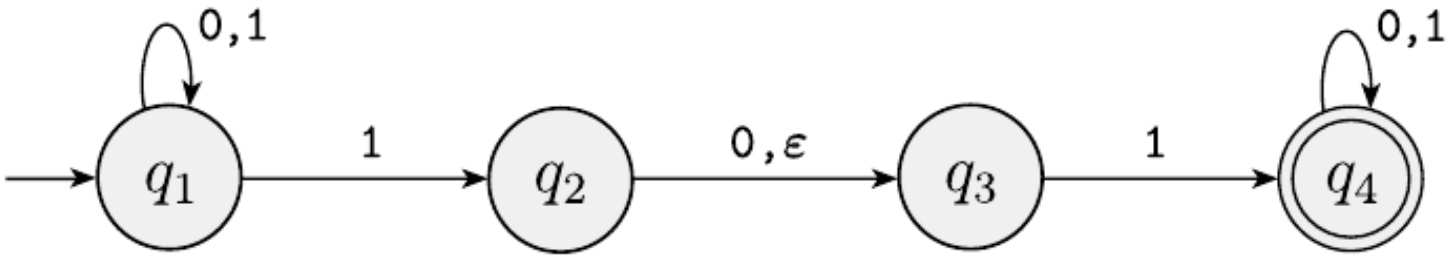
Não-determinismo

- Não-determinismo é um conceito extremamente útil que tem grande impacto sobre a teoria da computação.
- Até agora lidamos com computação **determinística**: quando a máquina está num estado e lê um símbolo, só existe uma única opção para o próximo estado.
- Em uma computação **não-determinística**, quando a máquina está num dado estado e lê um símbolo, podem existir várias escolhas para o próximo estado.
- O não-determinismo é uma generalização do determinismo.

  Todo autômato finito determinístico é automaticamente um autômato finito não-determinístico também.
- Daqui por diante, poderemos abreviar autômatos não-determinísticos como **AFNs**, para diferenciar dos autômatos finitos determinísticos (**AFDs**).

Não-determinismo

- Exemplo 18 Considere o diagrama de estados do autômato finito não-determinístico N_1 abaixo.



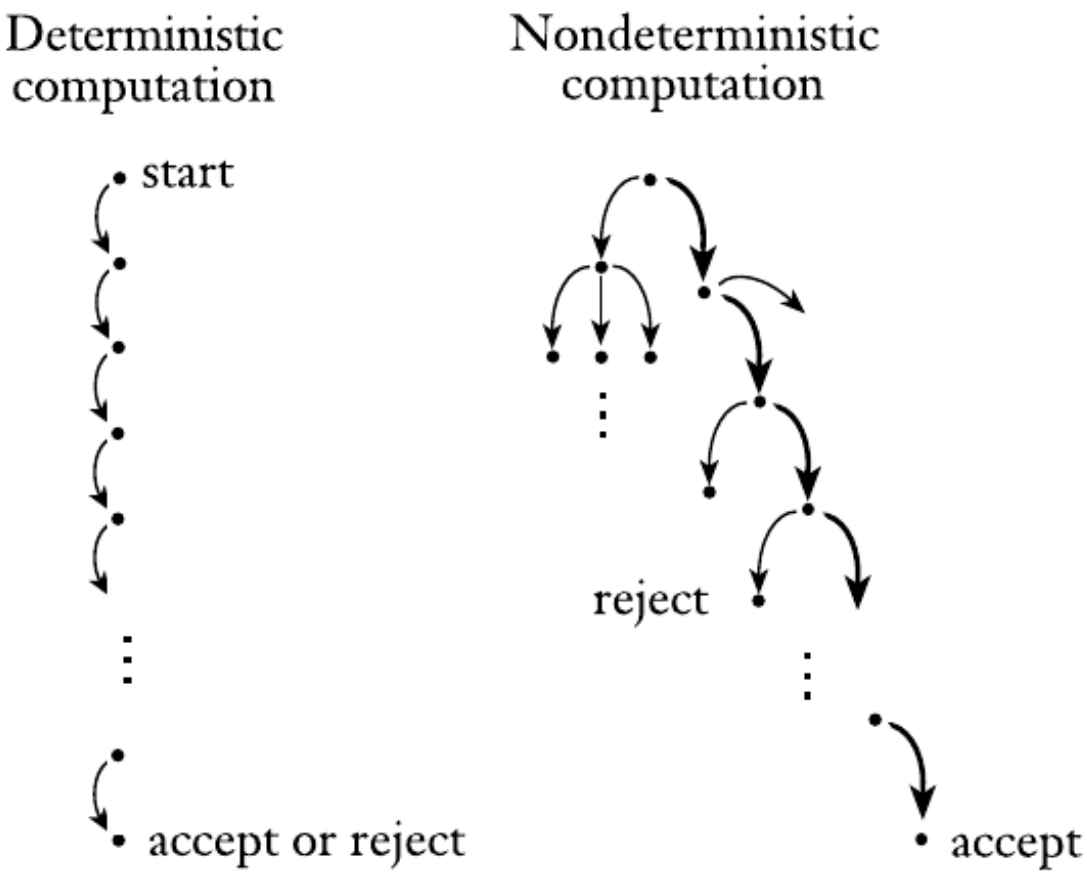
Este AFN tem algumas fontes de não-determinismo:

- Estando no estado q_1 e lendo o símbolo 1, o próximo estado pode ser q_1 ou q_2.
- Estando no estado q_2, o próximo estado pode ser q_3 ao ler o símbolo 0, mas o AFN pode pular para de q_2 para q_3 sem ler símbolo algum (pois há uma transição sob ϵ).

◁

Não-determinismo: Interpretação

- Comparação entre computações determinísticas e não-determinísticas com um ramo de aceitação:



Não-determinismo: Interpretação

- Existem ao menos três interpretações para uma computação não-determinista de uma cadeia de entrada:
  - **Oráculo**: a cada ponto em que há mais de uma possibilidade, a máquina "adivinha" qual escolha leva ao reconhecimento da cadeia (se tal escolha existe) e segue esta escolha.

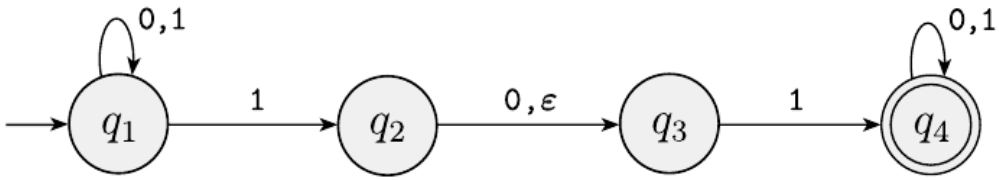
    Ou seja: se existe uma maneira de aceitar a cadeia, a máquina "adivinha" a maneira e aceita a cadeia.
  - **Paralelismo**: a cada ponto em que há mais de uma possibilidade, a máquina se divide em múltiplas cópias, e cada uma continua computando normalmente.

    Ou seja: uma cadeia é aceita se pelo menos uma cópia da máquina aceita a cadeia.
  - **Backtracking**: a cada ponto em que há mais de uma possibilidade, a máquina escolhe uma que ainda não foi testada e prossegue. Caso o escolha leve à rejeição da cadeia, a máquina retorna ao ponto da última escolha em aberto e faz uma nova opção.

    Ou seja: se existe uma maneira de aceitar a cadeia, a máquina vai encontrá-la pois ela tenta todas as computações alternativas possíveis.

Não-determinismo

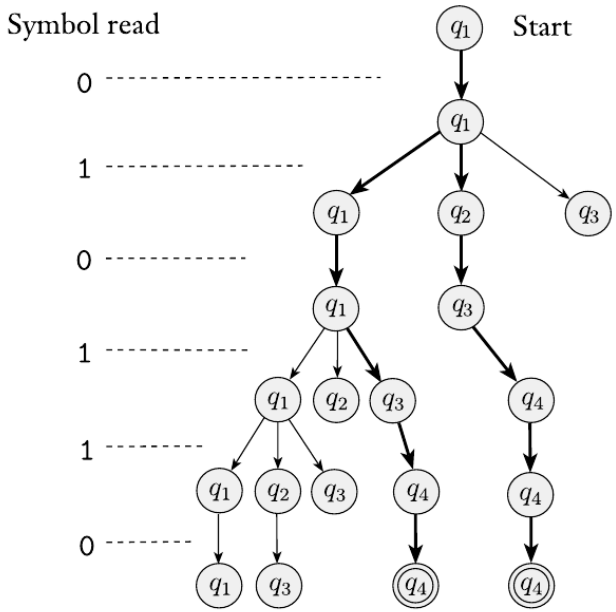
- **Exemplo 19** Seja novamente o AFN N_1 abaixo.



Qual a computação de N_1 sobre a cadeia de entrada 010110?

Solução.

- Caminhos em negrito indicam computações de aceitação, os demais indicam computações de rejeição.
- Como há no mínimo uma computação de aceitação, a cadeia é aceita.

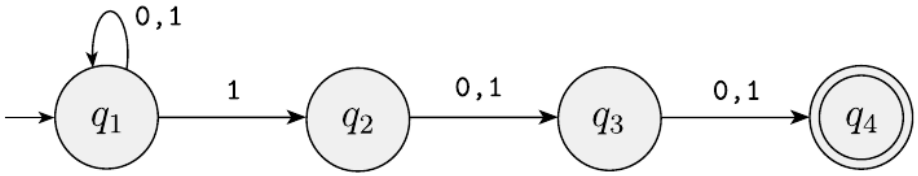


Não-determinismo

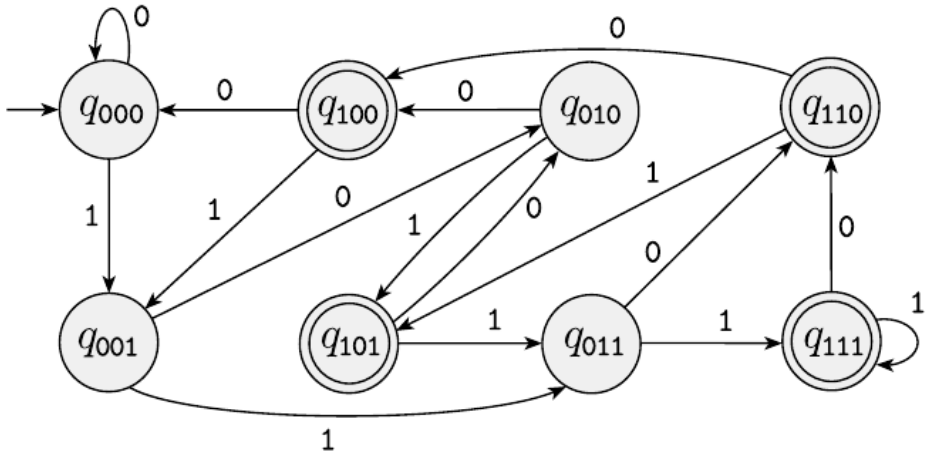
- **Exemplo 20** Construa um AFN que reconheça a linguagem

$$L = \{w \in \{0,1\}^* \mid |w| \geq 3 \text{ e o antepenúltimo símbolo de } w \text{ é } 1\}.$$

Solução. Um AFN para L é o seguinte:



Para efeito de comparação, um AFD para a mesma linguagem seria:

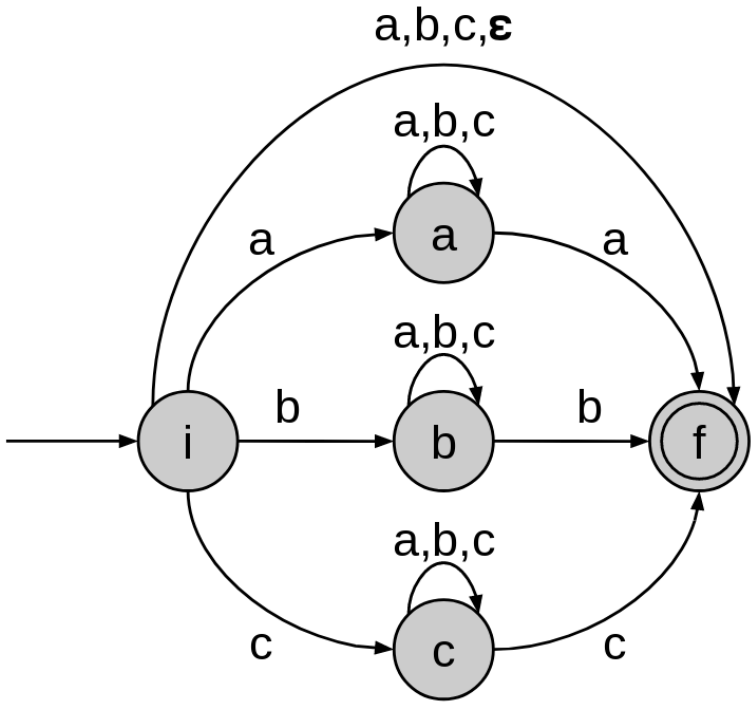


Não-determinismo

- **Exemplo 21** Construa um AFN que reconheça a linguagem

$$L = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid \text{o último símbolo de } w \text{ é igual ao primeiro}\}.$$

Solução. Um AFN para L é o seguinte:

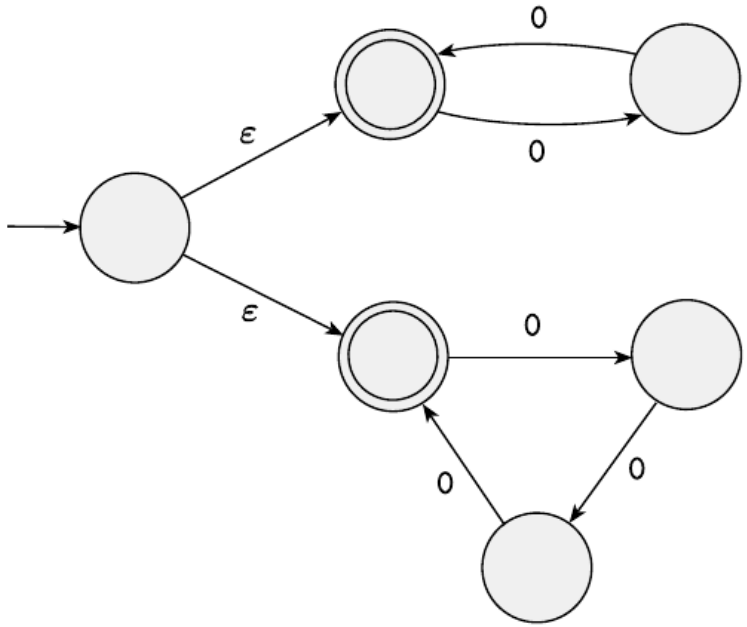


Para efeito de comparação, construa um AFD para a mesma linguagem.

Não-determinismo

- Exemplo 22 Seja o AFN N_3 a seguir, que possui transições ϵ (ou seja, transições que ocorrem sem que haja consumo de um símbolo da cadeia de entrada).

  Qual a linguagem reconhecida por N_3?



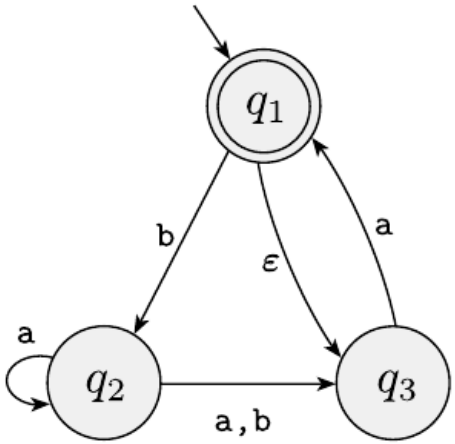
Solução.

$$L(N_3) = \{0^k \mid k \text{ é múltiplo de 2 ou de 3}\}.$$

◁

Não-determinismo

- Exemplo 23 Seja o AFN N_4 abaixo.



Diga se ele aceita ou rejeita as cadeias abaixo:

- ϵ: aceita.
- a: aceita.
- baba: aceita.
- baa: aceita.
- b: rejeita.
- bb: rejeita.
- babba: rejeita.

Autômatos Finitos Não-determinísticos: Definição formal

- Para definir um autômato finito não-determinístico, vamos introduzir a notação

$$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$$

para indicar a extensão do alfabeto Σ com um símbolo correspondendo à cadeia vazia ϵ.

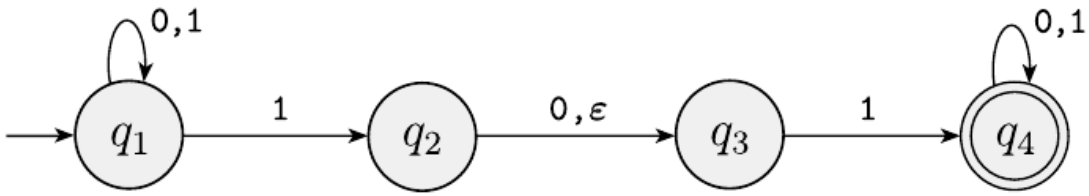
Lembre-se de que $\mathcal{P}(A)$ representa o conjunto potência de A.

- Um **autômato finito não-determinístico (AFN)** é uma 5-tupla $(Q, \Sigma, \delta, q_0, F)$, onde:

1. Q é um conjunto finito de **estados**,
2. Σ é o **alfabeto de entrada**,
3. $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ é a **função de transição**,
4. $q_0 \in Q$ é o **estado inicial**, e
5. $F \subseteq Q$ é o conjunto de **estados de aceitação** (ou **estados finais**).

Autômatos Finitos Não-determinísticos: Definição formal

- **Exemplo 24** Considere o diagrama de estados do autômato finito N_1:



Formalmente, este autômato é uma 5-tupla $N_1 = (Q, \Sigma, \delta, q_0, F)$, onde:

1. o conjunto de estados é $Q = \{q_1, q_2, q_3, q_4\}$,
2. o alfabeto é $\Sigma = \{0, 1\}$,
3. a função de transição δ é descrita como

δ	0	1	ϵ
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

,

4. o estado inicial é $q_1 \in Q$, e
5. o conjunto estados de aceitação é $F = \{q_4\}$.

◁

Autômatos Finitos Não-determinísticos: Definição formal de computação

- A formalização do conceito de computação para AFNs é similar à dos AFDs.
- Seja $N = (Q, \Sigma, \delta, q_0, F)$ um AFN, e suponha que $w = w_1 w_2 \ldots w_n$ seja uma cadeia em que cada $w_i \in \Sigma$.

  Então o AFN N aceita a cadeia w se podemos escrever w como $w = y_1 y_2 \ldots y_m$, onde cada $y_i \in \Sigma_\epsilon$, e existe uma sequência de estados $r_0, r_1, \ldots, r_m \in Q$ satisfazendo as três condições abaixo:

  1. $r_0 = q_0$,

     (Esta condição diz que a máquina começa no estado inicial.)

  2. $r_{i+1} \in \delta(r_i, y_{i+1})$, para $i = 0, 1, \ldots, m-1$, e

     (Esta condição diz que o estado r_{i+1} é um dos próximos estados permissíveis quando N está no estado r_i e lendo y_{i+1}.)

  3. $r_m \in F$.

     (Esta condição diz que a máquina pára em estado de aceitação.)

Equivalência entre AFNs e AFDs

- Os AFNs podem fazer tudo o que os AFDs fazem, e parecem poder fazer ainda mais (e.g., transitar sob ϵ, ter mais de um estado ativo ao mesmo tempo).

  Pode parecer, a princípio, que os AFNs têm a capacidade de reconhecer linguagens que os AFDs não podem.

- Mas este não é o caso: os autômatos finitos determinísticos e não-determinísticos reconhecem a mesma classe de linguagens.

- Parar mostrar isso, vamos usar o conceito de máquinas equivalentes.

- Duas máquinas são **equivalentes** se elas reconhecem a mesma linguagem.

Equivalência entre AFNs e AFDs

- **Teorema** Todo autômato finito não-determinístico tem um autômato finito determinístico equivalente.

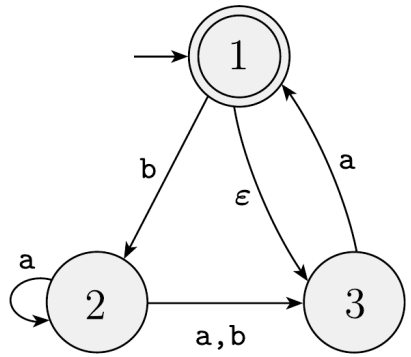
  Demonstração. Dada no livro-texto. □

- A equivalência de AFDs e AFNs permite a seguinte reformulação da definição de linguagens regulares.

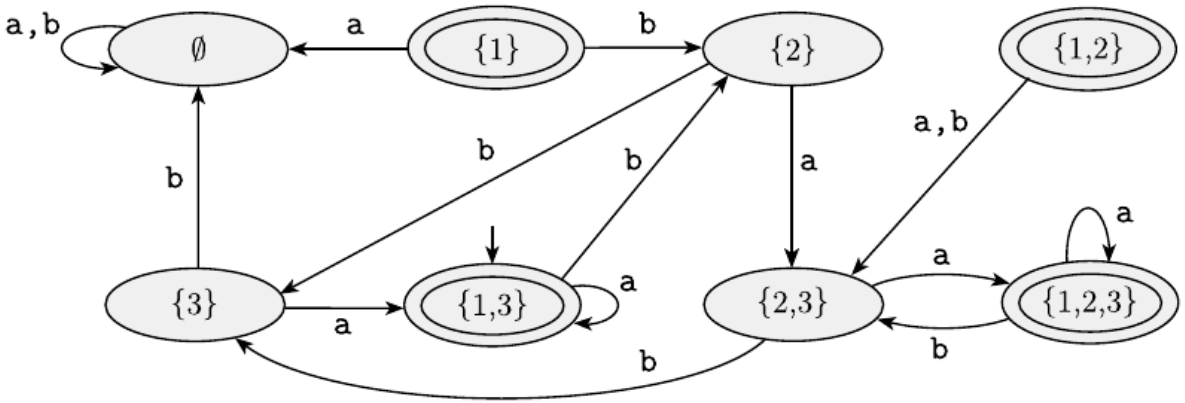
- **Corolário** Uma linguagem é regular se, e somente se, algum autômato finito não-determinístico a reconhece.

Equivalência entre AFNs e AFDs

- Exemplo 25 Usando o método da demonstração do teorema anterior, encontre um AFD equivalente para o AFN N_4 a seguir.



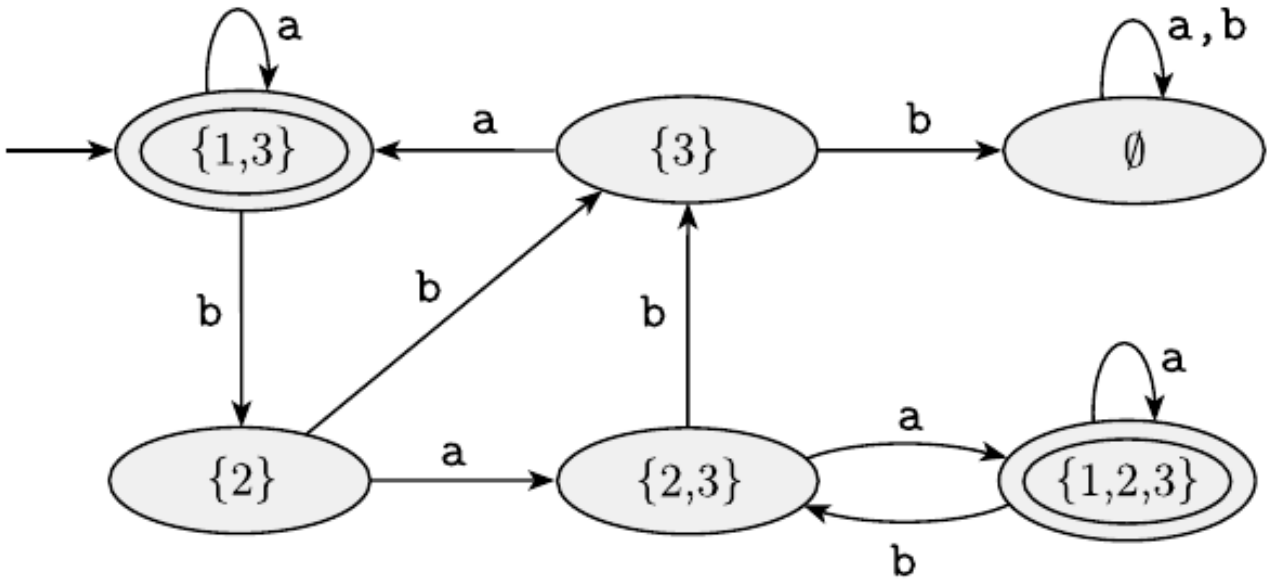
Solução. Aplicando o método da demonstração do teorema, obtemos o AFD abaixo:



Equivalência entre AFNs e AFDs

- Exemplo 25 (Continuação)

Notando que alguns estados do AFD encontrado são inúteis, podemos eliminá-los e obter o seguinte AFD simplificado, que é equivalente ao AFN original N_4:



Fechamento de Linguagens Regulares sob Operações Regulares

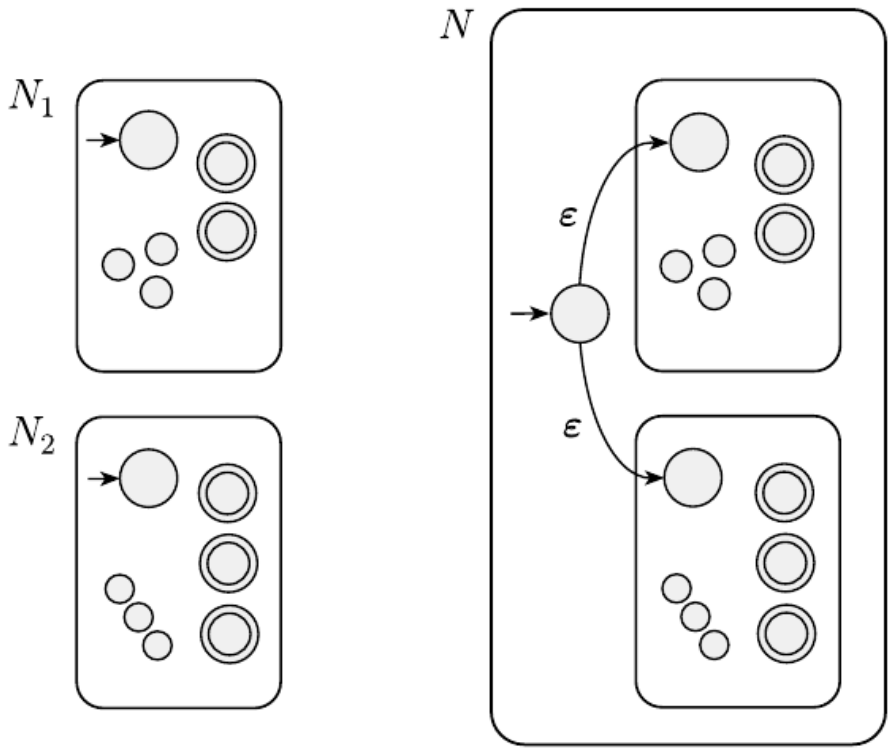
- Quanto estávamos tentando provar que a classe das linguagens regulares é fechada sob as operações regulares, nós:
  - Tivemos sucesso em provar o fechamento sob a operação de união.
  - Vimos que provar o fechamento sob as operações de concatenação e fecho de Kleene era muito complicado para fazer diretamente.
- Mas agora nossa tarefa vai ficar mais fácil, pois aprendemos a:
  1. Usar não-determinismo para construir AFNs.
  2. Transformar qualquer AFN em um AFD equivalente.

  Com as habilidades acima, podemos encontrar demonstrações simples para todos os fechamentos acima.

Fechamento de Linguagens Regulares sob Operações Regulares

- **Teorema** A classe de linguagens regulares é fechada sob a operação de união.

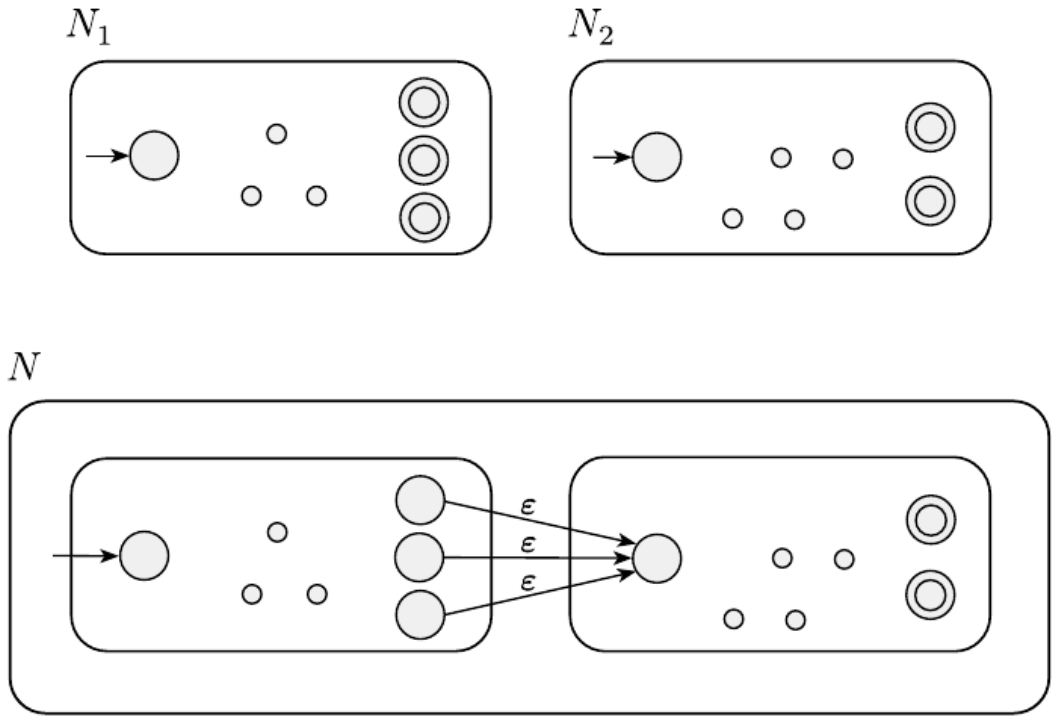
  Demonstração. A demonstração formal encontra-se no livro texto. Aqui daremos uma ideia da demonstração.



Fechamento de Linguagens Regulares sob Operações Regulares

- **Teorema** A classe de linguagens regulares é fechada sob a operação de concatenação.

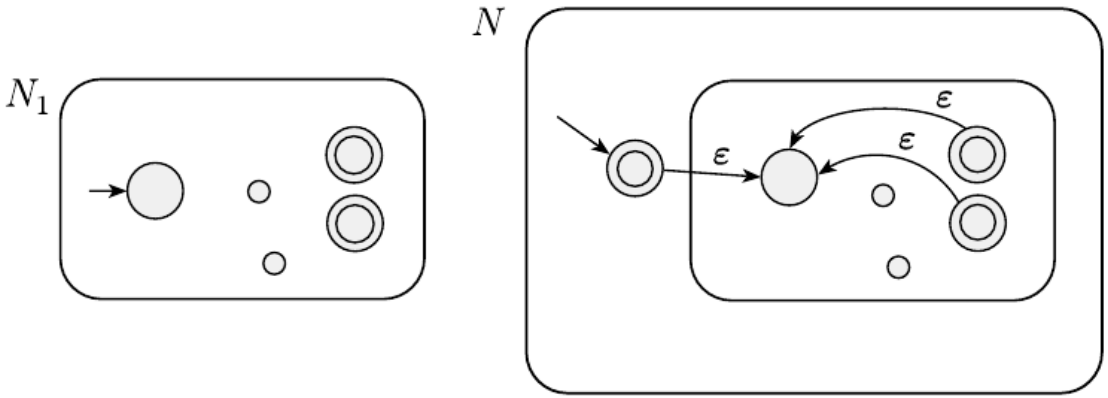
  Demonstração. A demonstração formal encontra-se no livro texto. Aqui daremos uma ideia da demonstração.



Fechamento de Linguagens Regulares sob Operações Regulares

- **Teorema** A classe de linguagens regulares é fechada sob a operação de fecho de Kleene.

  Demonstração. A demonstração formal encontra-se no livro texto. Aqui daremos uma ideia da demonstração.



□

Fechamento de Linguagens Regulares sob Outras Operações

- Mostramos até agora que a classe das linguagens regulares é fechada sob as operações regulares: união, concatenação e fecho de Kleene.
- Outras operações também são interessantes:
  - **Complemento**: $\overline{A} = \{w \in \Sigma^* \mid w \notin A\}$, , onde Σ é o alfabeto de A.
  - **Interseção**: $A \cap B = \{x \mid x \in A \text{ e } x \in B\}$.

- Exemplo 26 Sejam $A = \{w \in \{0,1\}^* \mid w \text{ tem tamanho par}\}$ e $B = \{w \in \{0,1\}^* \mid w \text{ contém a subcadeia } 101\}$ linguagens sob o alfabeto $\Sigma = \{0,1\}$.
  - $A \cap B = \{w \in \{0,1\}^* \mid w \text{ tem tamanho par e contém a subcadeia } 101\}$.
  - $\overline{A} = \{w \in \{0,1\}^* \mid w \text{ tem tamanho ímpar}\}$.
  - $\overline{B} = \{w \in \{0,1\}^* \mid w \text{ não contém a subcadeia } 101\}$. ◁

Fechamento de Linguagens Regulares sob Outras Operações

- **Teorema** A classe das linguagens regulares é fechada sob complemento.

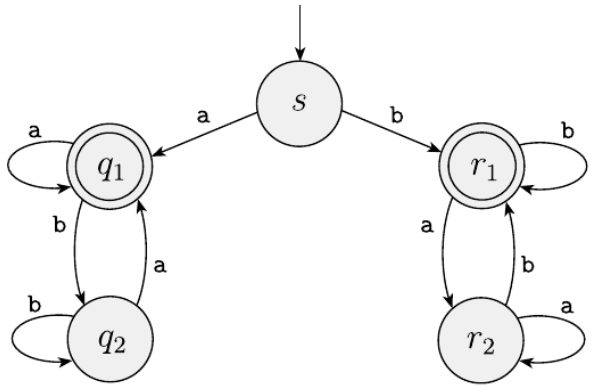
  Demonstração. Seja A uma linguagem regular com um AFD M que a reconheça. Um AFD M' que reconhece $\overline{A}$ é o próprio AFD M, mas em que os estados de aceitação e de não-aceitação são invertidos.

  Formalmente: se $M = (Q, \Sigma, \delta, q_0, F)$, então $M' = (Q, \Sigma, \delta, q_0, Q - F)$. □

Fechamento de Linguagens Regulares sob Outras Operações

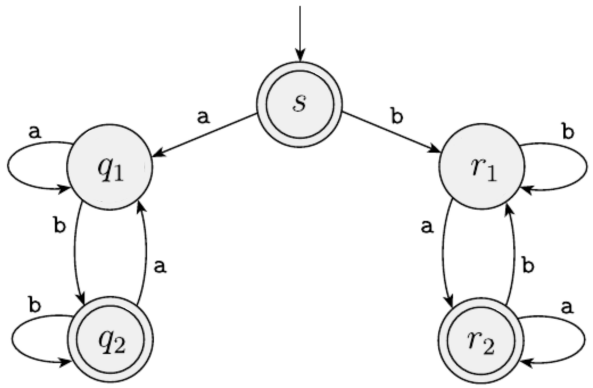
- **Exemplo 27** Em um exercício anterior construímos um AFD M_4 para:

$L = \{w \in \{\mathtt{a},\mathtt{b}\}^* \mid |w| > 0$ e
w começa e termina
com o mesmo símbolo$\}$



Podemos transformar M_4 em um AFD para reconhecer:

$\overline{L} = \{w \in \{\mathtt{a},\mathtt{b}\}^* \mid |w| = 0$ ou
w não começa e termina
com o mesmo símbolo$\}$



Fechamento de Linguagens Regulares sob Outras Operações

- **Teorema** A classe das linguagens regulares é fechada sob interseção.

  Demonstração. Sejam A e B duas linguagens regulares. Note que

  $$A \cap B = \overline{\overline{A} \cup \overline{B}}.$$

  Como sabemos que as linguagens regulares são fechadas sob complemento e união, deduzimos que o lado direito da igualdade acima é uma linguagem regular. Logo $A \cap B$ também é regular. □

- Alternativamente, para criar um AFD para a interseção de duas linguagens regulares, podemos usar um método muito similar àquele para criar um AFD para a união de duas linguagens.

  Basta fazer como estados de aceitação aqueles em que ambos os estados são finais nos autômatos originais.

Fechamento de Linguagens Regulares sob Operações Regulares

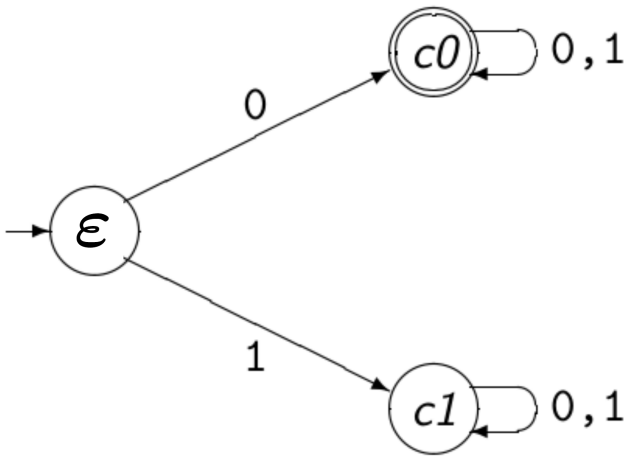
- **Exemplo 28** Sejam as linguagens

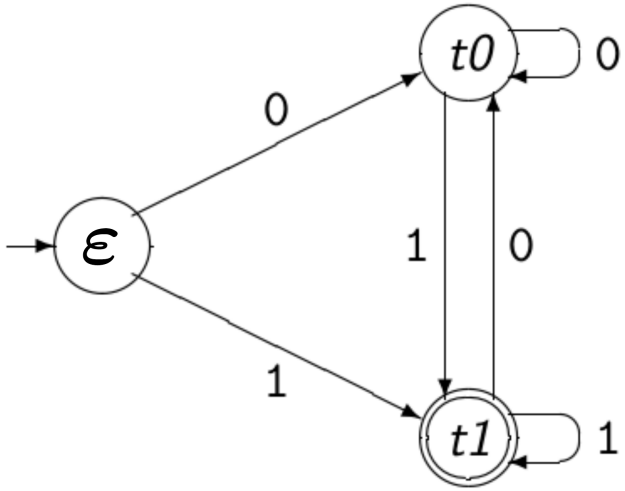
$$A = \{w \in \{0,1\}^* \mid |w| > 0 \text{ e } w \text{ começa com } 0\}, \quad \text{e}$$

$$B = \{w \in \{0,1\}^* \mid |w| > 0 \text{ e } w \text{ termina com } 1\}.$$

Adaptando o método do teorema sobre união de linguagens regulares, construa um autômato finito para $A \cap B$.

Solução. Já construímos autômatos finitos para as linguagens A e B como a seguir:

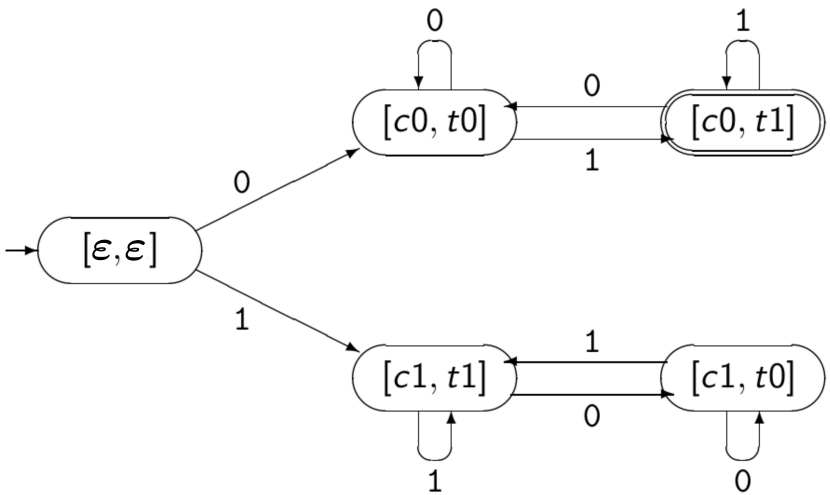




Fechamento de Linguagens Regulares sob Operações Regulares

- Exemplo 28 (Continuação)

E podemos combinar os autômatos usando o método do teorema sobre união de linguagens regulares, com o cuidado de tornar estados finais aqueles em que ambos os componentes são finais nos autômatos originais.



Note que a linguagem reconhecida por este autômato é

$$A \cap B = \{w \in \{0,1\}^* \mid |w| > 0 \text{ e } w \text{ começa com 0 e termina com 1}\}.$$