3 – Linguagens Regulares

3.3 Autômato Finito

- **Autômato Finito: sistema de estados finitos**
  - número finito e *predefinido* de estados
  - modelo computacional comum em diversos estudos teórico-formais
    - Linguagens Formais
    - Compiladores
    - Semântica Formal
    - Modelos para Concorrência

◆ Formalismo operacional/reconhecedor - pode ser

- determinístico
  - dependendo do estado corrente e do símbolo lido
  - pode assumir um *único* estado
- não-determinístico
  - dependendo do estado corrente e do símbolo lido
  - pode assumir um *conjunto* de estados alternativos
- com movimentos vazios
  - dependendo do estado corrente e *sem ler* qualquer símbolo
  - pode assumir um *conjunto* de estados (portanto é não-determinístico)

◆ Movimento vazio

- pode ser visto como transições encapsuladas
  - excetuando-se por uma eventual mudança de estado
  - nada mais pode ser observado
- análogo à encapsulação das linguagens orientadas a objetos

◆ Três tipos de autômatos: equivalentes

- em termos de poder computacional

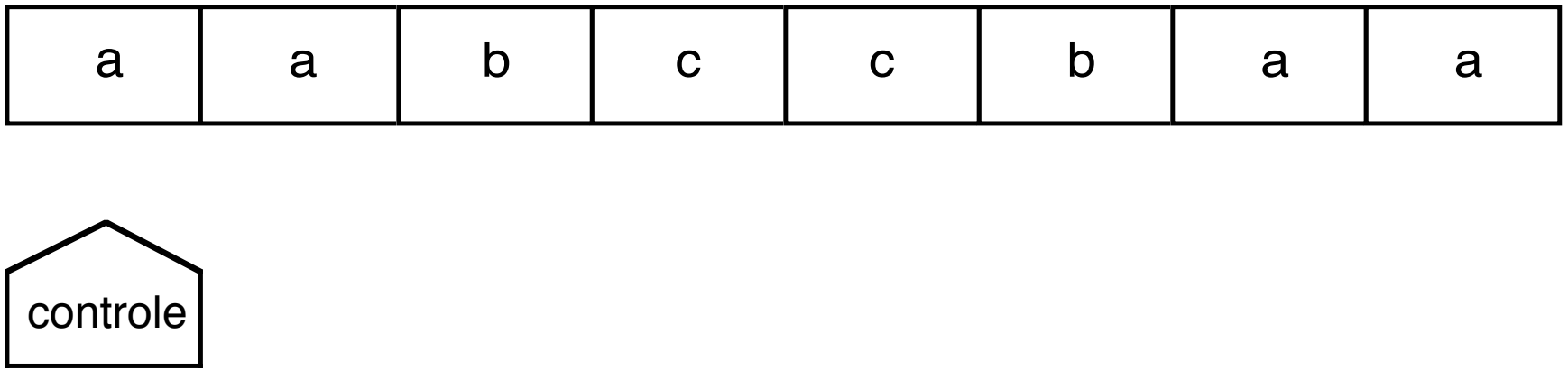
- **Autômato finito (determinístico): máquina constituída por**
  - Fita: dispositivo de entrada
    - contém informação a ser processada
  - Unidade de Controle: reflete o estado corrente da máquina
    - possui unidade de leitura (cabeça da fita)
    - acessa uma célula da fita de cada vez
    - movimenta-se exclusivamente para a direita
  - Programa, Função Programa ou Função de Transição
    - comanda as leituras
    - define o estado da máquina

◆ Fita é finita

- dividida em células
- cada célula armazena um símbolo
- símbolos pertencem a um alfabeto de entrada
- *não* é possível gravar sobre a fita (não existe memória auxiliar)
- palavra a ser processada ocupa toda a fita

Unidade de controle

- número finito e predefinido de estados
  - origem do termo controle finito
- leitura
  - lê o símbolo de uma célula de cada vez
  - move a cabeça da fita uma célula para a direita
  - posição inicial da cabeça célula mais à esquerda da fita



◆ Programa: função parcial

- dependendo do estado corrente e do símbolo lido
- determina o novo estado do autômato

Def: Autômato Finito (Determinístico) ou AFD

$$M = (\Sigma, Q, \delta, q_0, F)$$

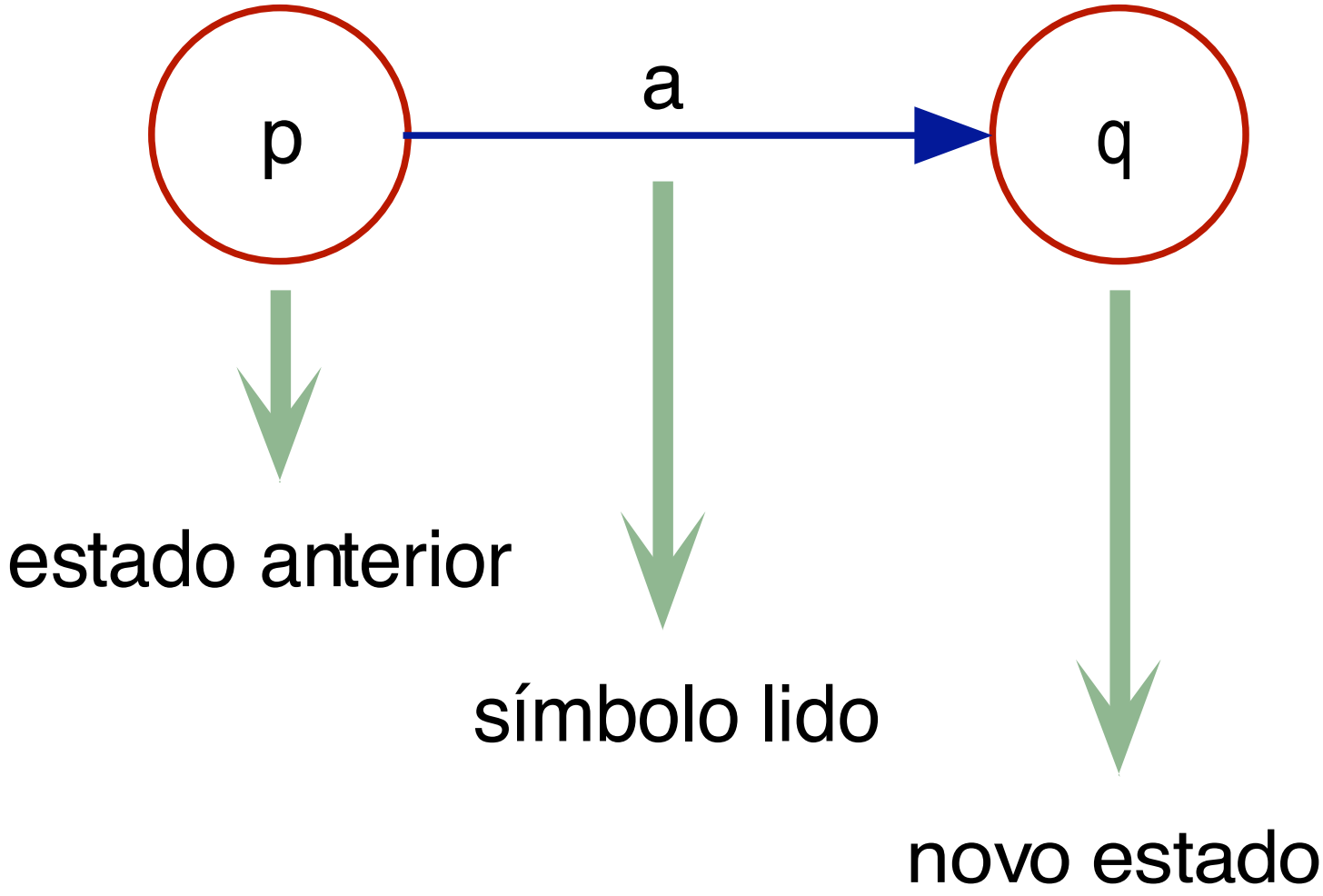
- Σ é um alfabeto (de símbolos) de entrada
- Q é um conjunto de estados possíveis do autômato (finito)
- δ é uma (função) programa ou função de transição (função parcial)

$$\delta: Q \times \Sigma \rightarrow Q$$

  - transição do autômato: $\delta(p, a) = q$

- q_0 é um elemento distinguido de Q: estado inicial
- F é um subconjunto de Q: conjunto de estados finais

- **Autômato finito como um diagrama: $\delta(p, a) = q$**



◆ Estados iniciais e finais

◆ Transições paralelas: $\delta(q, a) = p$ e $\delta(q, b) = p$

◆ Função programa como uma tabela de dupla entrada

$$\delta(p, a) = q$$

δ	a	...
p	q	...
q	...	...

- **Computação de um autômato finito**
  - sucessiva aplicação da função programa
  - para cada símbolo da entrada (da esquerda para a direita)
  - até ocorrer uma condição de parada
- **Lembre-se que um autômato finito**
  - não possui memória de trabalho
  - para armazenar as informações passadas
  - deve-se usar o conceito de estado

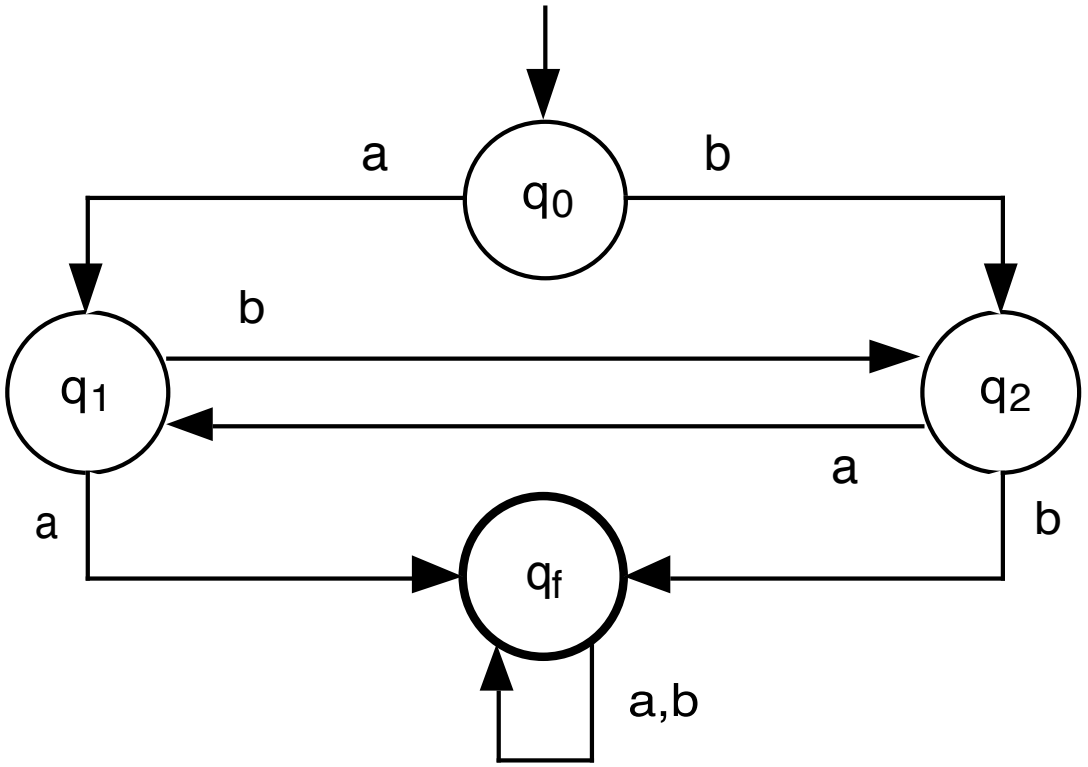
Exp: Autômato Finito: aa ou bb como subpalavra

$$L_1 = \{ w \mid w \text{ possui aa ou bb como subpalavra} \}$$

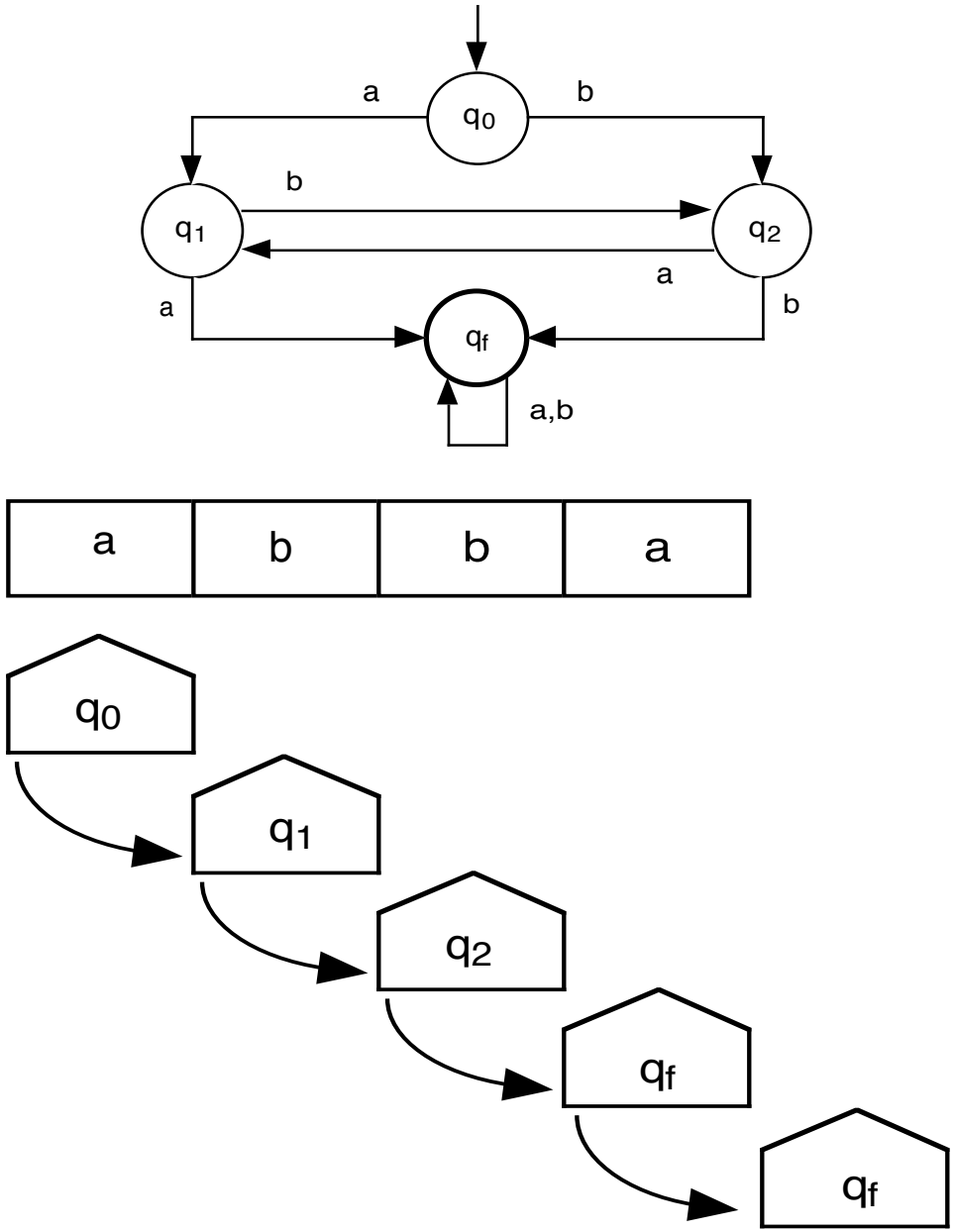
Autômato finito

$$M_1 = (\{ a, b \}, \{ q_0, q_1, q_2, q_f \}, \delta_1, q_0, \{ q_f \})$$

δ_1	a	b
q_0	q_1	q_2
q_1	q_f	q_2
q_2	q_1	q_f
q_f	q_f	q_f



- q_1: "símbolo anterior é a"
- q_2: "símbolo anterior é b"
- qual a informação memorizada por q_0 e q_f
- após identificar aa ou bb
  - q_f (final): varre o sufixo da entrada - terminar o processamento


a
b
q0
b
q1
q2
a
a
b
qf
a,b
a
b
b
a
q0
q1
q2
qf
qf

Obs: Autômato Finito Sempre Pára

Como

- qualquer palavra é finita
- novo símbolo é lido a cada aplicação da função programa

não existe a possibilidade de ciclo (*loop*) infinito

◆ Parada do processamento

- Aceita a entrada
  - ∗ após processar o último símbolo, assume um estado final
- Rejeita a entrada. Duas possibilidades
  - ∗ após processar o último símbolo, assume um estado não-final
  - ∗ programa indefinido para argumento (estado e símbolo)

Obs: Autômato Finito × Grafo Finito Direto

Qual a diferença entre um autômato finito e um grafo finito direto?

Qualquer autômato finito pode ser visto como um grafo finito direto

- podem existir arcos paralelos (mesmos nodos origem e destino)
- dois ou mais arcos podem ser identificados com a mesma etiqueta (símbolo do alfabeto)
- existe um nodo distinguido: estado inicial
- existe um conjunto de nodos distinguidos: estados finais

Usual considerar um autômato finito como grafo finito direto especial

- herda resultados da Teoria dos Grafos

◆ Definição formal do comportamento de um autômato finito

- dar semântica à sintaxe
- necessário estender a função programa
- argumento: estado e *palavra*

Def: Função Programa Estendida, Computação

$M = (\Sigma, Q, \delta, q_0, F)$ autômato finito determinístico

$$\delta^*: Q \times \Sigma^* \rightarrow Q$$

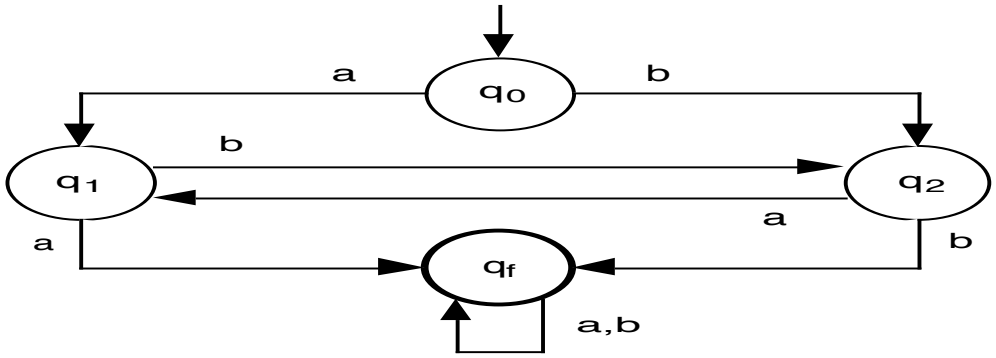
é $\delta: Q \times \Sigma \rightarrow Q$ estendida para palavras - indutivamente definida

- $\delta^*(q, \varepsilon) = q$
- $\delta^*(q, aw) = \delta^*(\delta(q, a), w)$

◆ Observe

- sucessiva aplicação da função programa
  - para cada símbolo da palavra
  - a partir de um dado estado
- se a entrada for vazia, fica parado
- aceita/rejeita: função programa estendida a partir do estado inicial

Exp: Função Programa Estendida



- $\underline{\delta}^*(q_0, abaa) =$ função estendida sobre *abaa*
- $\underline{\delta}^*(\delta(q_0, a), baa) =$ processa *a*baa
- $\underline{\delta}^*(q_1, baa) =$ função estendida sobre *baa*
- $\underline{\delta}^*(\delta(q_1, b), aa) =$ processa *b*aa
- $\underline{\delta}^*(q_2, aa) =$ função estendida sobre *aa*
- $\underline{\delta}^*(\delta(q_2, a), a) =$ processa *a*a
- $\underline{\delta}^*(q_1, a) =$ função estendida sobre *a*
- $\underline{\delta}^*(\delta(q_1, a), \varepsilon) =$ processa *a*
- $\underline{\delta}^*(q_f, \varepsilon) = q_f$ função estendida sobre ε: fim da indução; ACEITA

Def: Linguagem Aceita, Linguagem Rejeitada

$M = (\Sigma, Q, \delta, q_0, F)$ autômato finito determinístico.

Linguagem Aceita ou Linguagem Reconhecida por M

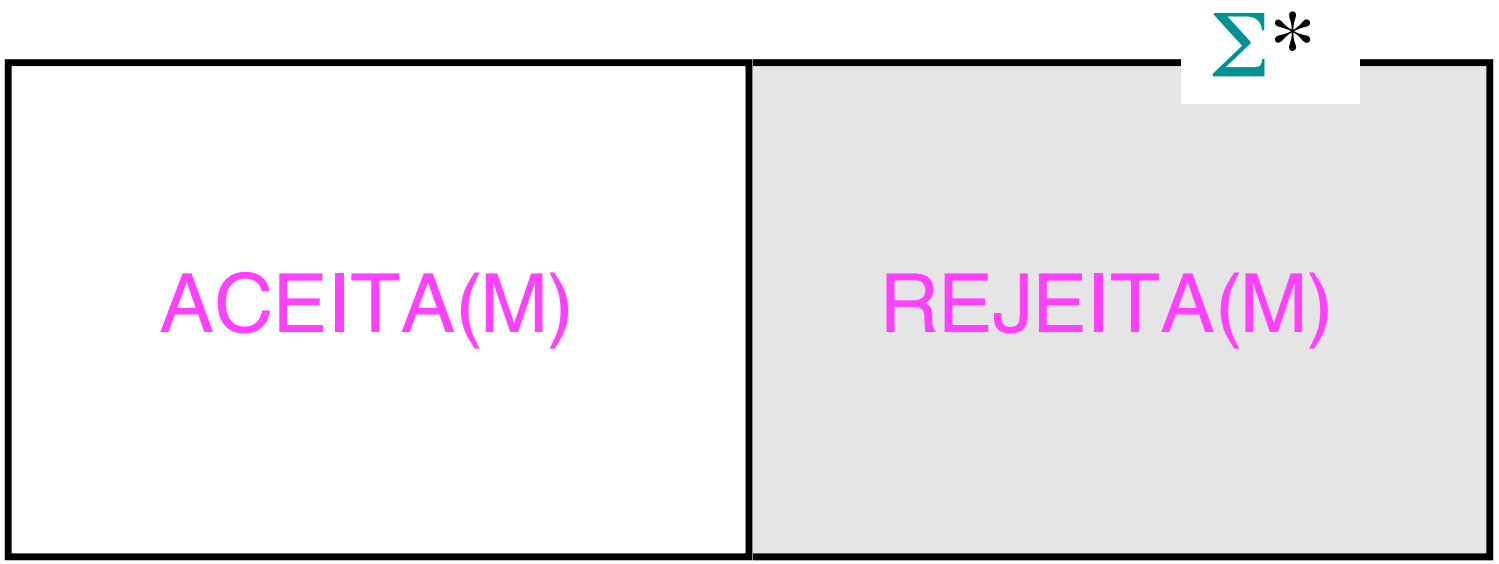
$$L(M) = ACEITA(M) = \{ w \mid \delta^*(q_0, w) \in F \}$$

Linguagem Rejeitada por M:

$$REJEITA(M) = \{ w \mid \delta^*(q_0, w) \notin F \text{ ou } \delta^*(q_0, w) \text{ é indefinida} \}$$

- **Supondo que Σ^* é o conjunto universo**
  - $ACEITA(M) \cap REJEITA(M) = \varnothing$
  - $ACEITA(M) \cup REJEITA(M) = \Sigma^*$
  - $\sim ACEITA(M) = REJEITA(M)$
  - $\sim REJEITA(M) = ACEITA(M)$

- **Cada autômato finito M sobre Σ**
  - induz uma partição de Σ* em duas classes de equivalência
  - e se um dos dois conjuntos for vazio?



- **Diferentes autômatos finitos podem aceitar uma mesma linguagem**

Def: Autômatos Finitos Equivalentes

M_1 e M_2 são Autômatos Finitos Equivalentes se e somente se

$$ACEITA(M_1) = ACEITA(M_2)$$

Def: Linguagem Regular, Linguagem Tipo 3

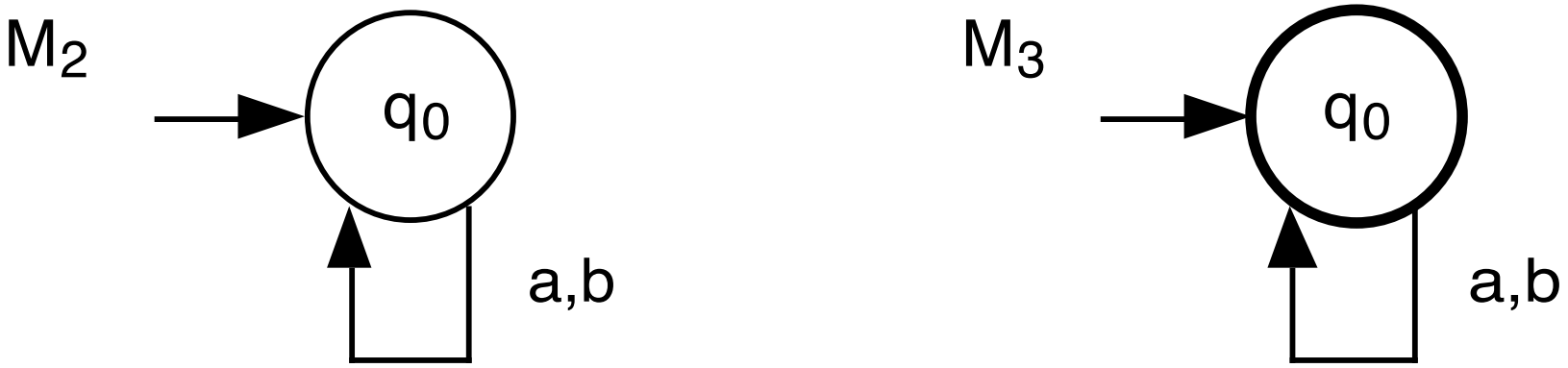
L é uma Linguagem Regular ou Linguagem Tipo 3

- existe pelo menos um autômato finito determinístico que aceita L

Exp: …Autômato Finito: Vazia, Todas as Palavras

Linguagens sobre o alfabeto { a, b }

$L_2 = \varnothing$ e $L_3 = \Sigma^*$



Exp: Autômato Finito: Vazia, Todas as Palavras

$L_2 = \varnothing$ e $L_3 = \Sigma^*$

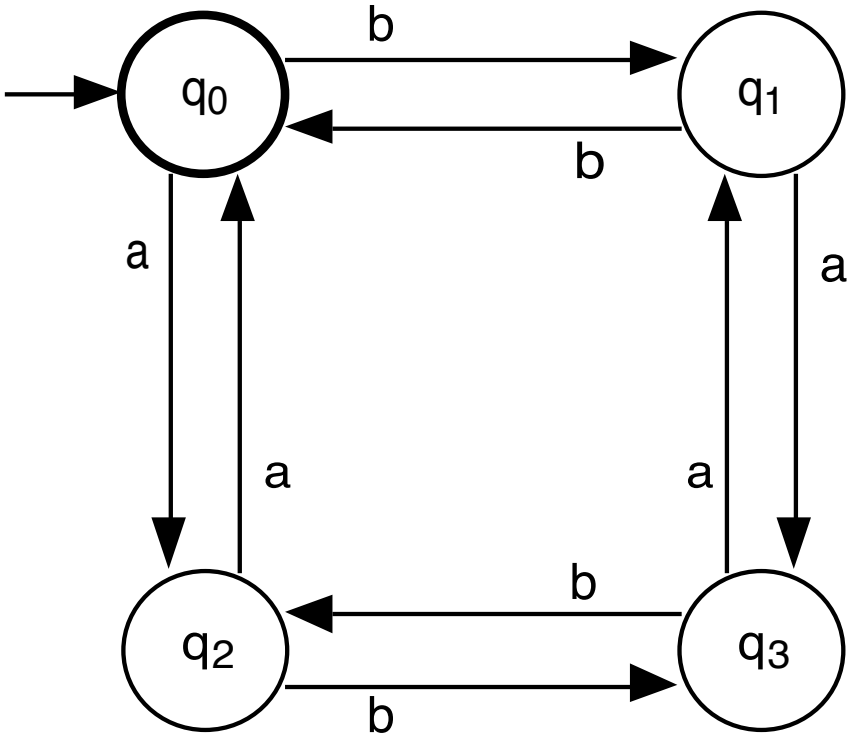
δ_2	a	b
q_0	q_0	q_0

δ_3	a	b
q_0	q_0	q_0

- diferença entre δ_2 e δ_3?
- o que, exatamente, diferencia M_2 de M_3?

Exp: Autômato Finito: número par de cada símbolo

$L_4 = \{ w \mid w$ possui um número par de a e um número par de b $\}$



Como seria para aceitar um número ímpar de cada símbolo?

Obs: Função Programa × Função Programa Estendida

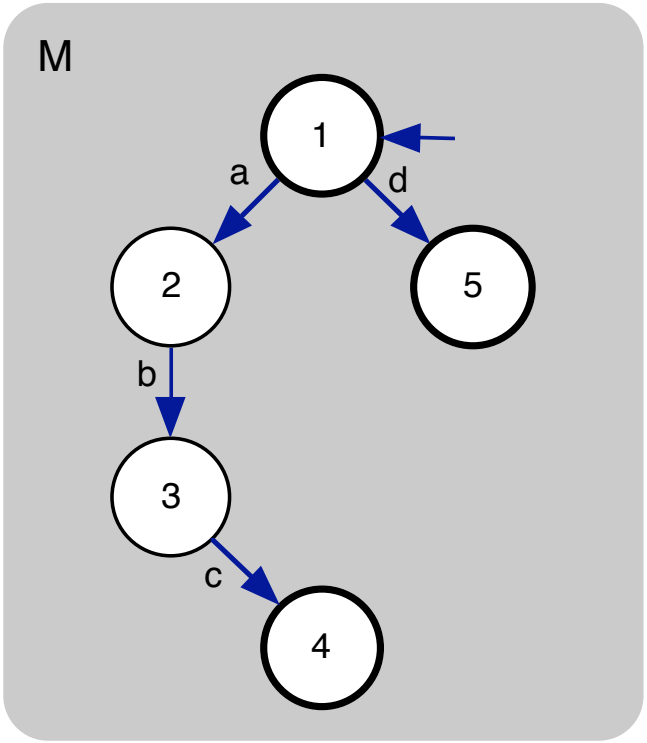
Objetivando simplificar a notação

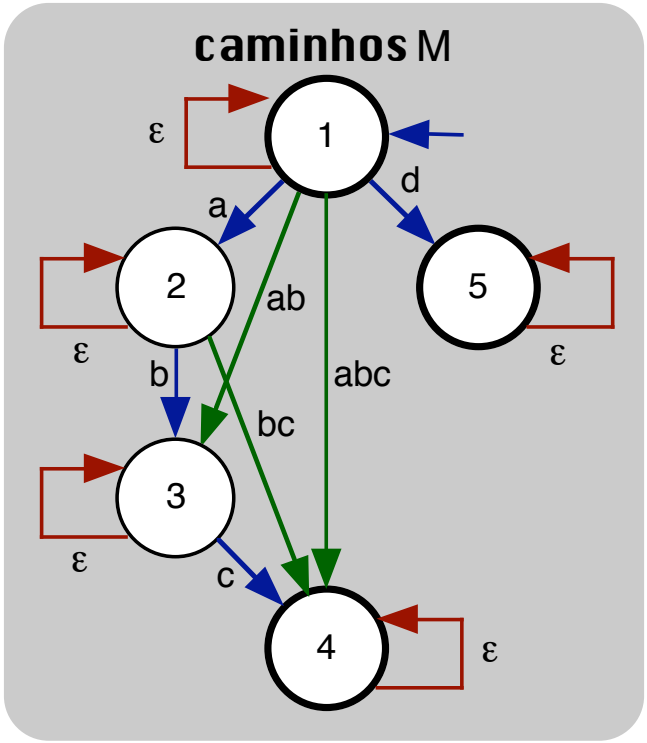
- δ e a sua correspondente extensão $\underline{\delta}^*$
- podem ser ambas denotadas por δ

Obs: Computações × Caminhos de um Grafo

- conjunto de arcos: computações possíveis
- subconjunto de arcos
  - com origem no estado inicial
  - destino em algum estado final
  - linguagem aceita

Obs: …Computações × Caminhos de um Grafo





Computações(M)
=
{ ε, a, b, c, d,
ab, bc, abc}

ACEITA (M)
=
{ε , d, abc}