

# Teoria da Computação

Prof. Sergio D. Zorzo

Departamento de Computação – UFSCar

02

# **Teoria da Computação**

## **Linguagens Regulares**

**Sistema de Estados Finitos**

**Autômato Finito**

**Autômato Finito Não Determinístico**

**Autômato Finito com Transições Espontâneas  
(ou com movimentos vazios)**

**Equivalência entre as representações**

# Linguagens Regulares ou Tipo 3

## Formalismos de Estudo:

### **Autômato Finito**

formalismo operacional (reconhecedor de sentenças)  
um sistema de estados finitos

### **Expressão Regular**

formalismo denotacional (gerador de sentenças)  
conjuntos (linguagens) básicos + concatenação e união

### **Gramática Regular**

formalismo axiomático (gerador de sentenças)  
gramática com restrições nas regras de produção

## Linguagens Regulares ou Tipo 3

- Pela Hierarquia de Chomsky é a classe de linguagens mais simples
- Há três formalismos (autômatos, expressões regulares e gramáticas regulares), com algoritmos de reconhecimento, geração ou conversão entre formalismos
- Há pouca complexidade para a sua representação, mas com grande eficiência e de fácil implementação
- Há fortes limitações de expressividade, sendo que as linguagens de programação em geral não são linguagens regulares

## Linguagens Regulares ou Tipo 3

Há importantes propriedades que podem ser usadas para:

- construir novas linguagens regulares a partir de linguagens regulares conhecidas (definindo uma álgebra)
- provar propriedades de determinado conjunto (linguagem) regular
- construir algoritmos

Se um problema tiver uma solução regular deve-se considerar preferencialmente a qualquer outra não-regular, pelas propriedades da Classe e eficiência e simplicidade da implementação

O universo de aplicações das linguagens regulares é muito grande e constantemente ampliado

**Exemplos :** análise léxica dos compiladores, sistemas de animação, hipertextos e hipermídias

# Sistema de Estados Finitos

# Sistema de Estados Finitos

modelo matemático de sistema com entradas e saídas discretas

número *finito e predefinido de estados*

(podem ser definidos antes de iniciar o processamento)

## **Estado**

somente informações do passado

necessárias para determinar as ações para a próxima entrada

# Autômatos finitos

## Definição informal

Conjunto de estados: cada um “lembra” o que já foi feito na história de um sistema

Conjunto de transições: movimentos possíveis de um estado para outro

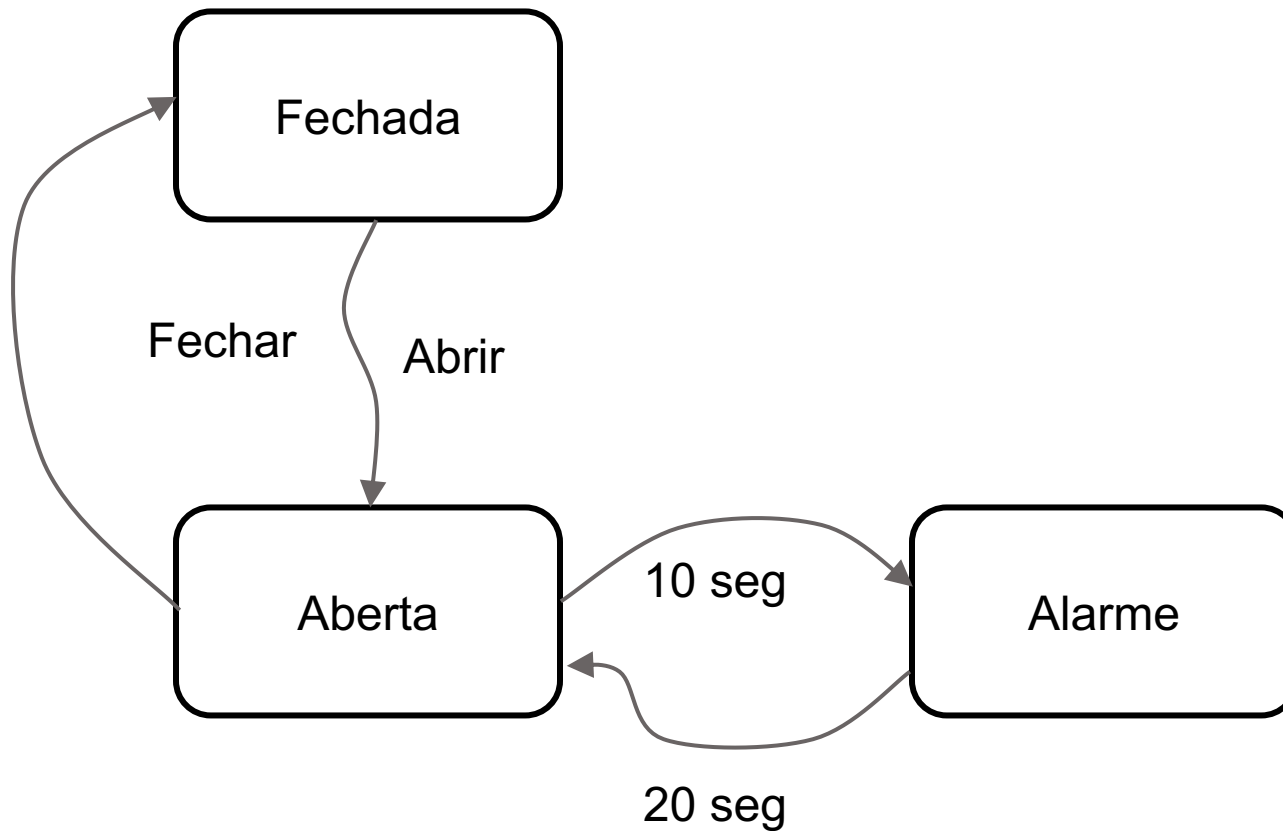
Controle: dispositivo hipotético que lê uma entrada externa e move de um estado para outro

Existem transições que não mudam o estado



# Autômatos finitos

## Geladeira



# Autômatos finitos

## Aplicações

Avaliar um determinado processo/protocolo em busca de erros/falhas

Ex: e se eu fechar a porta com o alarme tocando?  
Dependendo da implementação, pode resultar em erros

# Autômatos finitos determinísticos - DFA

Definição Formal: DFA M ou AF-d M

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

$Q$ =Conjunto finito de estados

$\Sigma$ =Conjunto finito de símbolos de entrada

$\delta$ =Função de transição

$q_0$ =Um estado inicial ( $q_0 \in Q$ )

$F$ =Um conjunto de estados finais ou de aceitação ( $F \subseteq Q$ )

# Autômatos finitos determinísticos

Função de transição

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

Define o funcionamento do autômato

Ex. da geladeira:

Precisa ter estado inicial e de aceitação

$Q = \{\text{Fechada}, \text{Aberta}, \text{Alarme}\}$

$\Sigma = \{\text{abrir}, \text{fechar}, 10\text{seg}, 20\text{seg}\}$

$q_0 = \text{Fechada}$

$F = \{\text{Fechada}\}$

## Autômatos finitos determinísticos

$\delta(\text{Fechada}, \text{abrir}) = \text{Aberta}$

$\delta(\text{Aberta}, \text{fechar}) = \text{Fechada}$

$\delta(\text{Aberta}, 10\text{seg}) = \text{Alarme}$

$\delta(\text{Alarme}, 20\text{seg}) = \text{Aberta}$

# Autômatos finitos determinísticos

Em um Autômato Finito Determinístico ( DFA )

As transições estão completas

Para todo estado e todo símbolo de entrada

Sempre sabe o que fazer

Definição de determinismo

# Autômatos finitos determinísticos

## Notações

- Diagramas (Exemplo mais comum)

- Tabelas de transição (Versão tabular do diagrama)

- Representam completamente a 5-upla do autômato

## Diagrama de estado

- Cada estado tem um nó correspondente

- Estado inicial (seta apontando para o estado a partir do nada)

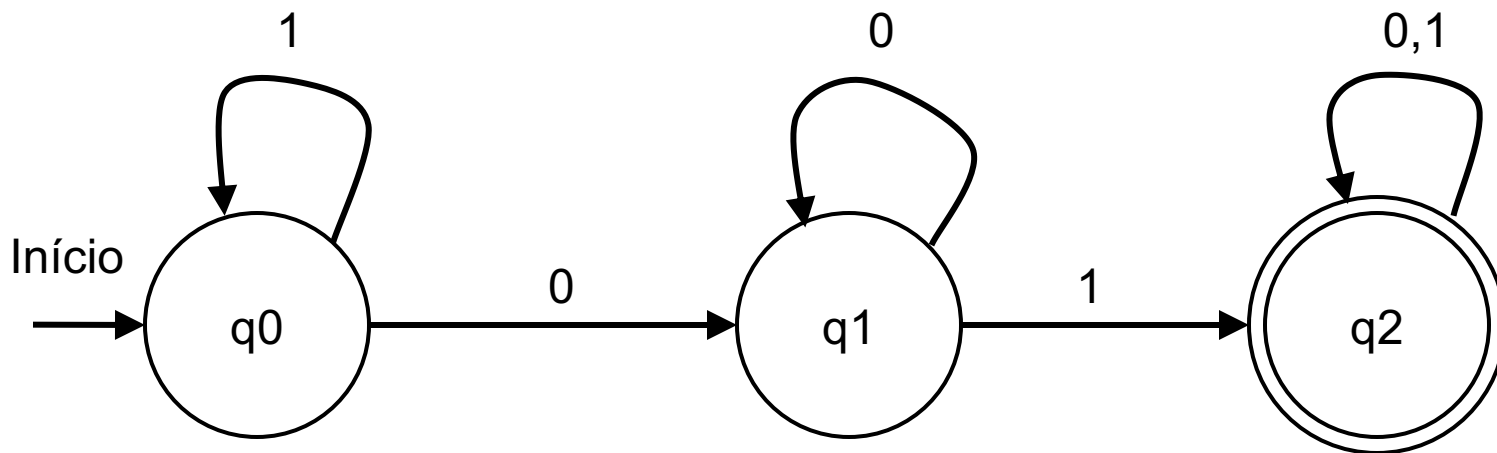
- Estado de aceitação (círculo duplo)

- Setas saindo de um estado para outro são transições

- Representação visual da função  $\delta$

## Autômatos finitos determinísticos

Ex:  $\{1...100...01y \mid y \text{ é qualquer cadeia de 0's e 1's}\}$





# Autômatos finitos determinísticos

## Versão tabular

Linhas correspondem aos estados

Colunas correspondem às entradas

Estado inicial é marcado com uma seta

Estados de aceitação são marcados com asterisco

	0	1
→ q0	q1	q0
q1	q1	q2
* q2	q2	q2

# Autômatos finitos determinísticos

Um DFA  $M$  denota uma linguagem

Conjunto de todas as cadeias que aceita

$$L(M) = A$$

$M$  reconhece  $A$

$M$  aceita  $A$

Mesmo que um  $M$  não aceite nenhuma cadeia

Ele aceita a linguagem vazia  $\emptyset$

# Autômatos finitos determinísticos

Quando o autômato recebe uma cadeia de entrada

Processa a cadeia e produz uma saída: aceita ou rejeita

Começa no estado inicial

Lê símbolos da esquerda para a direita

Após ler um símbolo, move-se de um estado para outro,  
de acordo com a função de transição

Quando lê o último símbolo, produz a saída

Se o autômato estiver em estado de aceitação, a saída  
será aceita

Caso contrário, será rejeitada

Ex: abrir-fechar-abrir-fechar

Ex: 0010111

# Autômatos finitos determinísticos

Definição formal de linguagem (indutiva)

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

$$\delta^*(q,\epsilon)=q$$

$$\delta^*(q,w)=\delta(\delta^*(q,x),a) \text{ onde } w=xa$$

$$L(M)=\{w \mid \delta^*(q_0,w) \text{ está em } F\}$$

Definição:

Se  $L$  é  $L(M)$  para algum DFA  $M$

$L$  é regular

# Autômatos finitos determinísticos

Configuração instantânea

$w_1w_2w_3\dots w_kqw_{k+1}w_{k+2}\dots w_{n-2}w_{n-1}w_n$

Ex:

Entrada: 001010011

Configuração: 001[ $q_3$ ]010011

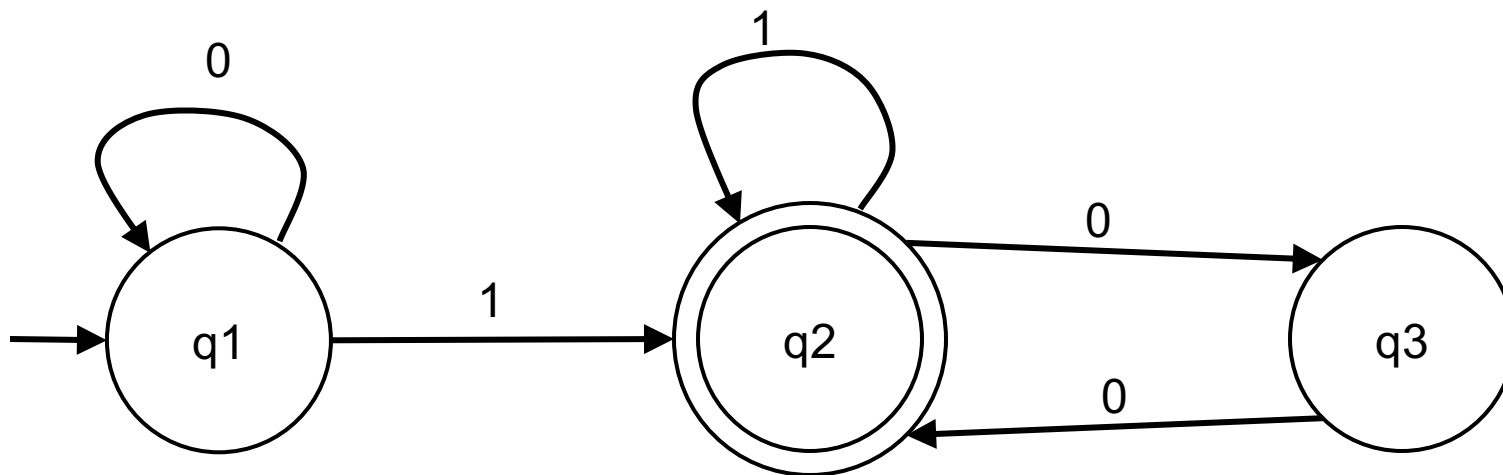
Já leu 001, falta ler 010011

Encontra-se no estado  $q_3$

Próxima entrada é 0

## Interpretando autômatos finitos

Dado o seguinte autômato finito:



## Interpretando autômatos finitos

Calcule a função estendida e mostre, passo a passo, as configurações instantâneas para as seguintes cadeias:

00000

0100

01000

010000

010101

11111

Quais dessas cadeias fazem parte da linguagem do autômato?

Descreva informalmente a linguagem do autômato

Descreva formalmente a linguagem do autômato

## Respostas

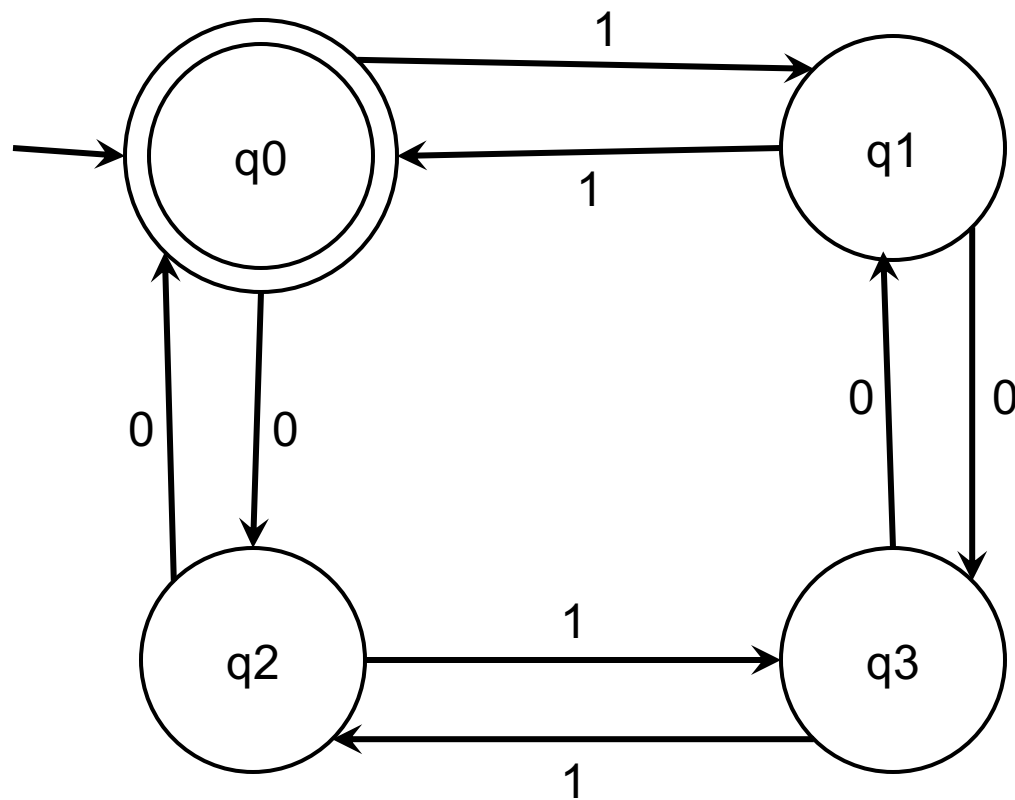
Linguagens com um 1 pelo menos e um número par de zeros depois do último 1 (zero é par)

$A = \{w \mid w \text{ contém pelo menos um } 1 \text{ e um número par de } 0\text{s} \text{ segue o último } 1\}$



## Interpretando autômatos finitos

Dado o seguinte autômato finito:



## Interpretando autômatos finitos

Calcule a função estendida e mostre, passo a passo, as configurações instantâneas para as seguintes cadeias:

0101

0110

1001

1111

1

11100

Quais dessas cadeias fazem parte da linguagem do autômato?

Descreva informalmente a linguagem do autômato

Descreva formalmente a linguagem do autômato

## Respostas

Cadeias com um número par de 0s e 1s

$L = \{w \mid w \text{ tem ao mesmo tempo um número par de 0s e um número par de 1s}\}$

## Interpretando Autômatos Finitos

Dado o seguinte autômato finito:

	0	1
$\rightarrow q1$	q1	q2
* q2	q1	q2

## Interpretando autômatos finitos

Calcule a função estendida e mostre, passo a passo, as configurações instantâneas para as seguintes cadeias:

010	0111110
-----	---------

0000001	1101
---------	------

001	11100
-----	-------

Quais dessas cadeias fazem parte da linguagem do autômato?

Descreva informalmente a linguagem do autômato

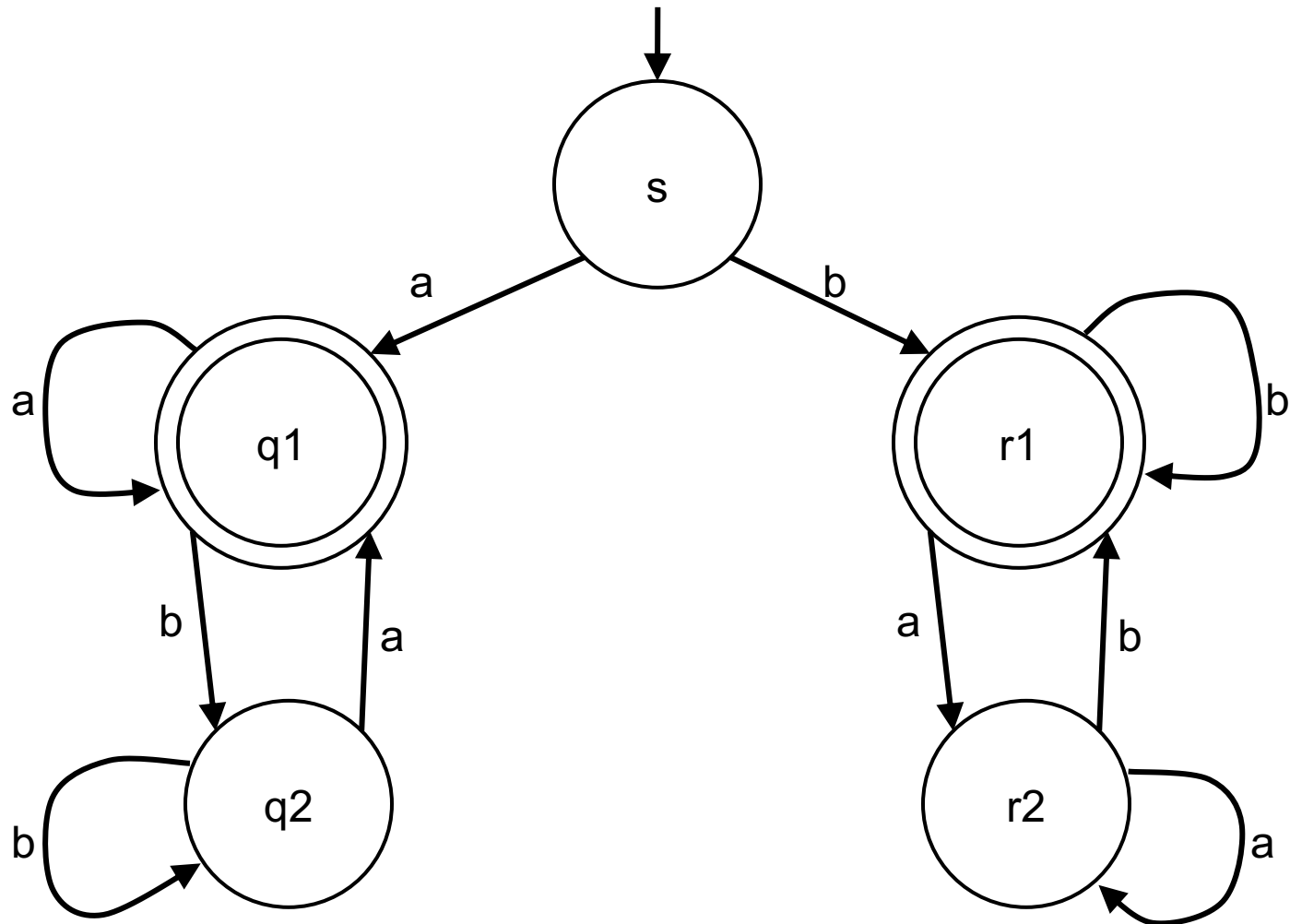
Descreva formalmente a linguagem do autômato

## Respostas

Todas as cadeias que terminam com 1

$$L = \{w \mid w \text{ termina com um } 1\}$$

## Interpretando autômatos finitos



## Interpretando autômatos finitos

Calcule a função estendida e mostre, passo a passo, as configurações instantâneas para as seguintes cadeias:

a	b	
aa		bb
bab		ab
ba		bbba

Quais dessas cadeias fazem parte da linguagem do autômato?

Descreva informalmente a linguagem do autômato

Descreva formalmente a linguagem do autômato



## Respostas

Cadeias que começam e terminam com o mesmo símbolo

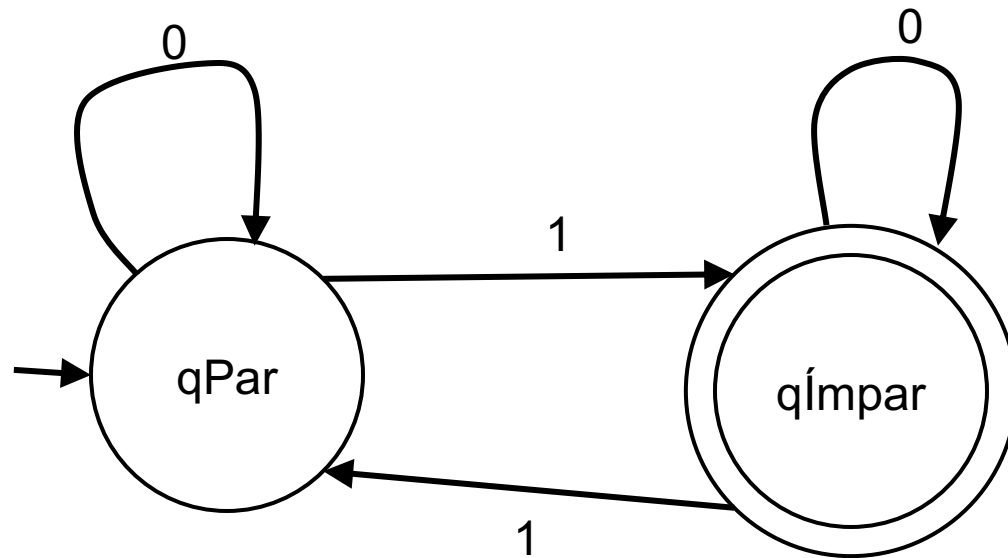
$L = \{w \mid w \text{ começa e termina com } a \text{ ou } w \text{ começa e termina com } b\}$

# Projetando autômatos finitos

Ex:

$\Sigma = \{0,1\}$

Linguagem = cadeias com número ímpar de 1s

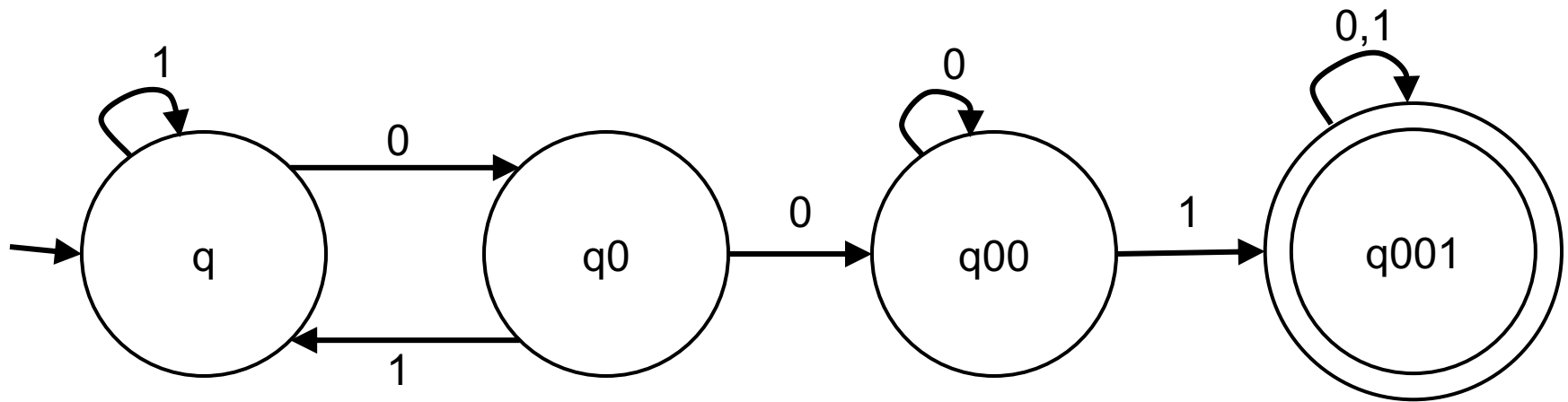


# Projetando autômatos finitos

Ex:

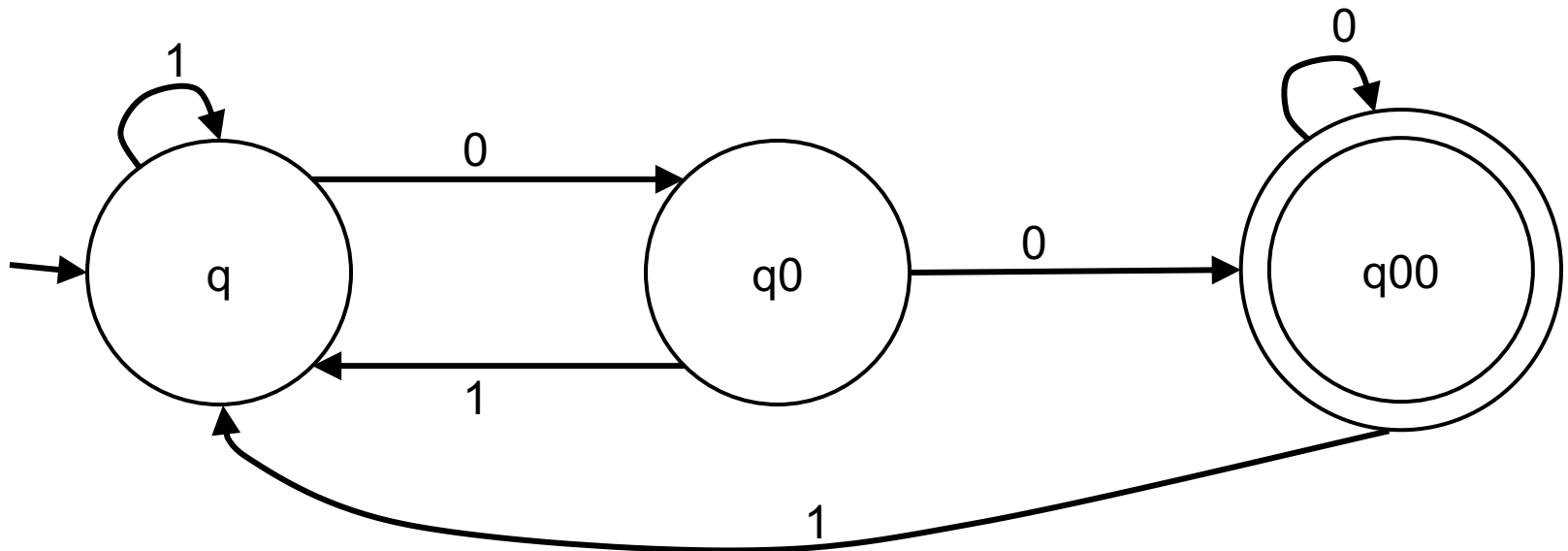
$\Sigma = \{0,1\}$

Linguagem = cadeias que contém a cadeia 001 como uma subcadeia



# Projetando autômatos finitos

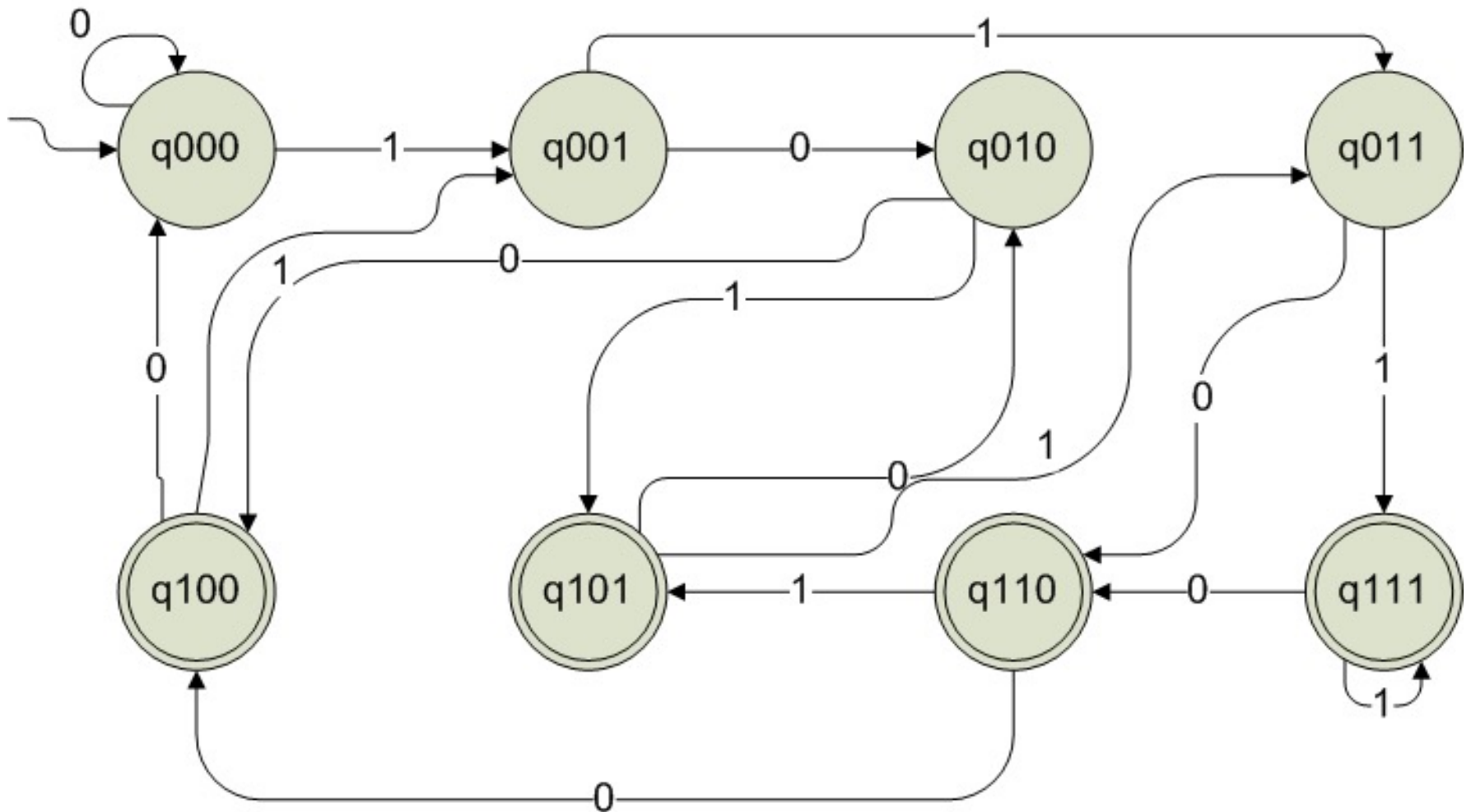
- Ex:
  - $\Sigma = \{0,1\}$
  - Linguagem = cadeias que terminam em 00



## Mais um exercício

- Linguagem A consistindo de todas as cadeias sobre  $\{0,1\}$  contendo um 1 na terceira posição a partir do final
  - Ex: 000100, 010110 estão em A, mas 0011 não

## Mais um exercício



# Autômatos Finitos Equivalentes

Dois Autômatos Finitos  $M1$  e  $M2$  são equivalentes

se e somente se

$$L(M1) = L(M2)$$

# Linguagem Regular e Autômato Finito

Uma Linguagem  $L$  é linguagem regular

se e somente se

Existe um autômato finito  $M$  tal que  $L=L(M)$



# Não-Determinismo

## Não-Determinismo

Um autômato pode estar em muitos estados ao mesmo tempo

### **No diagrama:**

DFA tem exatamente 1 seta com 1 símbolo para todo símbolo e estado

NFA pode ter zero ou mais setas para um símbolo/estado

### **Na tabela:**

DFA é completamente preenchida, com exatamente um estado em cada célula

NFA pode ter células com zero ou mais estados

# Não-Determinismo

O que isto significa na prática?

Uma visão: um NFA pode “adivinhar” algumas coisas sobre a entrada

Outra visão: em transições para mais de um estado, é o mesmo que dividir o autômato em dois e seguir cada execução em paralelo

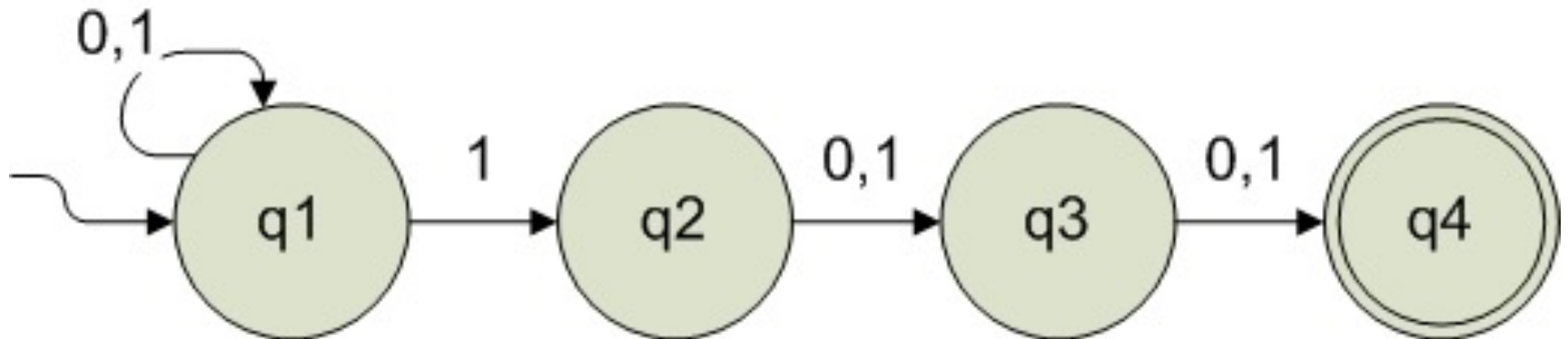
Outra: uma árvore de possibilidades – tentativa e erro

## Não-Determinismo

Mais fáceis de projetar e entender

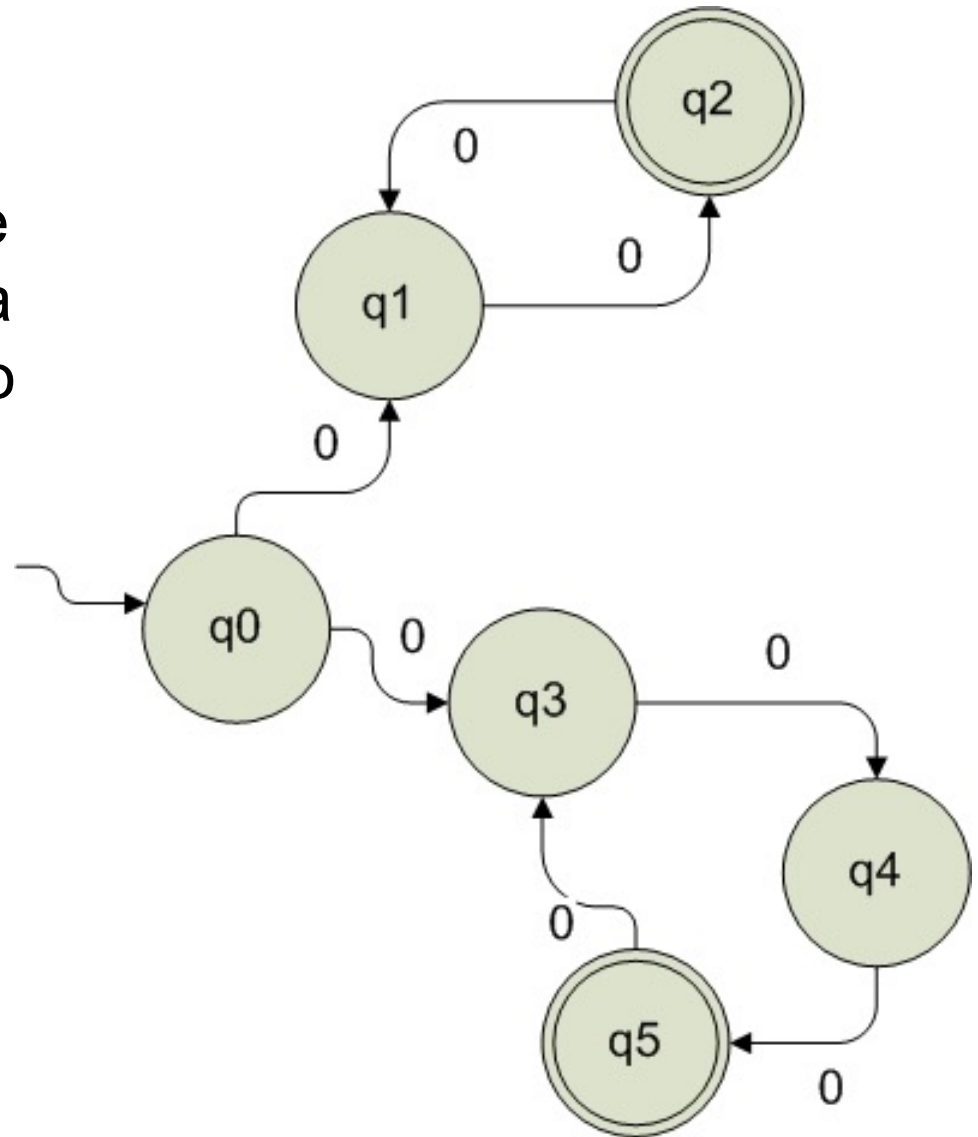
Ex:  $\Sigma = \{0,1\}$

A é uma linguagem consistindo de todas as cadeias contendo um 1 na terceira posição a partir do final (exemplo: 000100)



# Não-Determinismo

- Ex:  $\Sigma = \{0\}$ 
  - A é uma linguagem que aceita cadeias da forma  $0^k$ , onde  $k$  é um múltiplo de 2 ou 3



# Autômatos finitos não-determinísticos (NFA)

Definição formal: NFA  $M=(Q,\Sigma,\delta,q_0,F)$

$Q$ =Conjunto finito de estados

$\Sigma$ =Conjunto finito de símbolos de entrada

$\delta$ =Função de transição

$q_0$ =Um estado inicial ( $q_0 \in Q$ )

$F$ =Um conjunto de estados finais ou de aceitação ( $F \subseteq Q$ )

**Diferença está na função de transição**

$$\delta:Q \times \Sigma \rightarrow 2^Q$$

# Autômatos finitos determinísticos

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

Definição formal de linguagem (indutiva)

- $\delta(q,a)=p$
- $\delta^{\wedge}(q,\varepsilon)=q$
- $\delta^{\wedge}(q,w)=\delta(\delta^{\wedge}(q,x),a)$  onde  $w=xa$  ,  $x \in \Sigma^*$  e  $a \in \Sigma$
- $L(M)=\{w \mid \delta^{\wedge}(q_0,w) \text{ está em } F\}$

Definição:

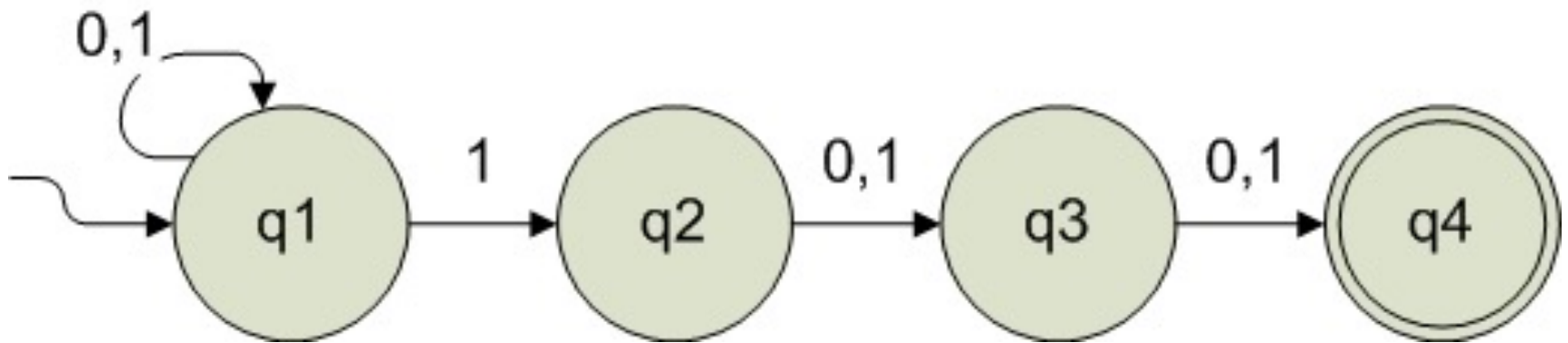
- Se  $L$  é  $L(M)$  para algum DFA  $M$
- $L$  é regular

# Autômatos finitos não-determinísticos

- Definição formal de linguagem
  - $\delta^*(q, \epsilon) = \{q\}$
  - $\delta^*(q, x) = \{p_1, p_2, \dots, p_k\}$
  - $\delta^*(q, w) = \text{união de todos } \delta(p_i, a), \text{ onde } w = xa$
  - $L(M) = \{w \mid \delta^*(q_0, w) \cap F \neq \emptyset\}$
- Definição:
  - Se  $L$  é  $L(M)$  para algum NFA  $M$
  - $L$  é regular



## Interpretando NFAs



## Interpretando NFAs

Calcule a função estendida e mostre, passo a passo, as configurações instantâneas para as seguintes cadeias:

111

010

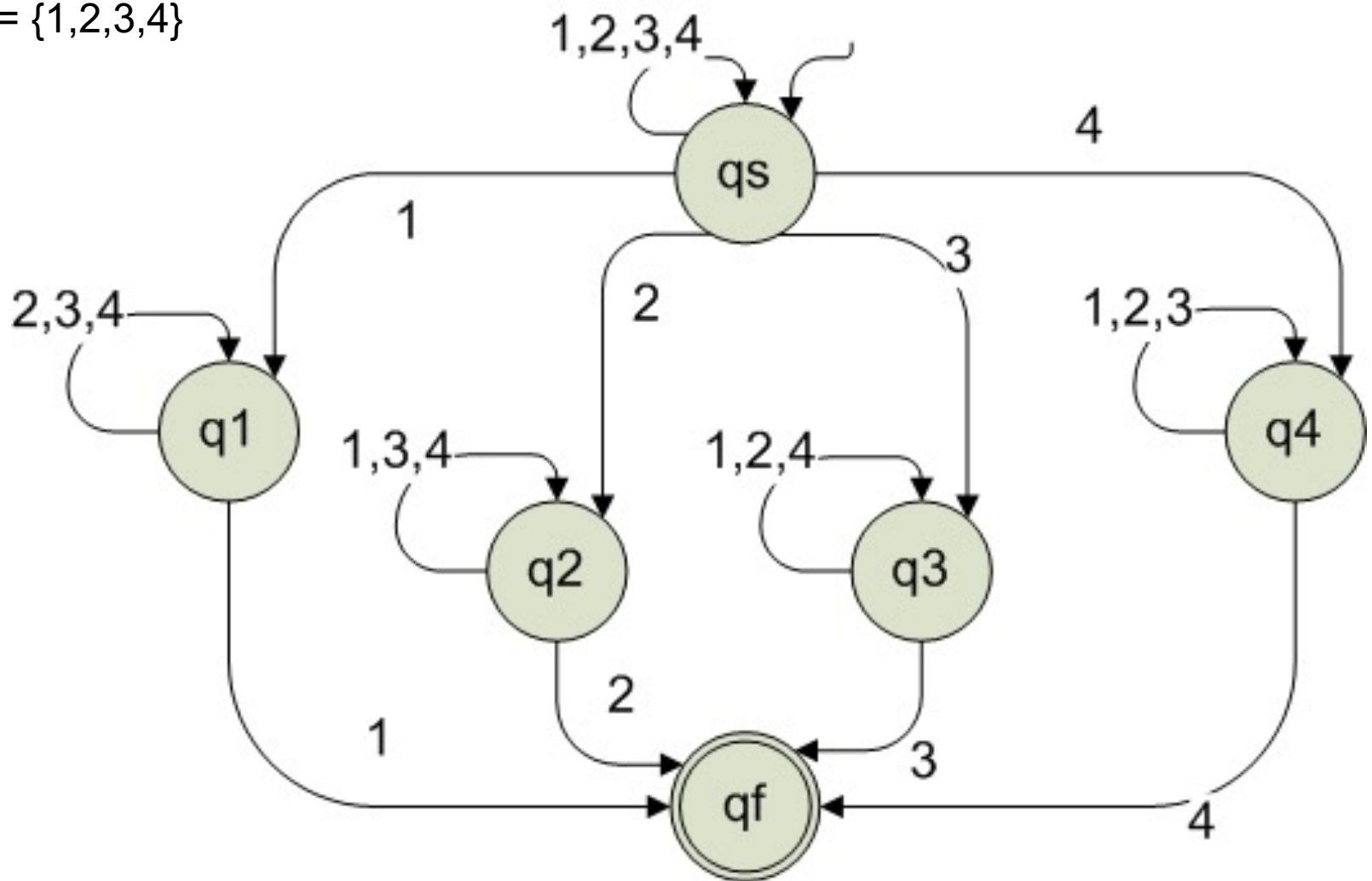
0100

(Use notação de conjuntos)

Descreva a linguagem reconhecida por este autômato

# Interpretando NFAs

$\Sigma = \{1,2,3,4\}$



## Interpretando NFAs

Calcule a função estendida e mostre, passo a passo, as configurações instantâneas para as seguintes cadeias:

1234

123

1231

433

412

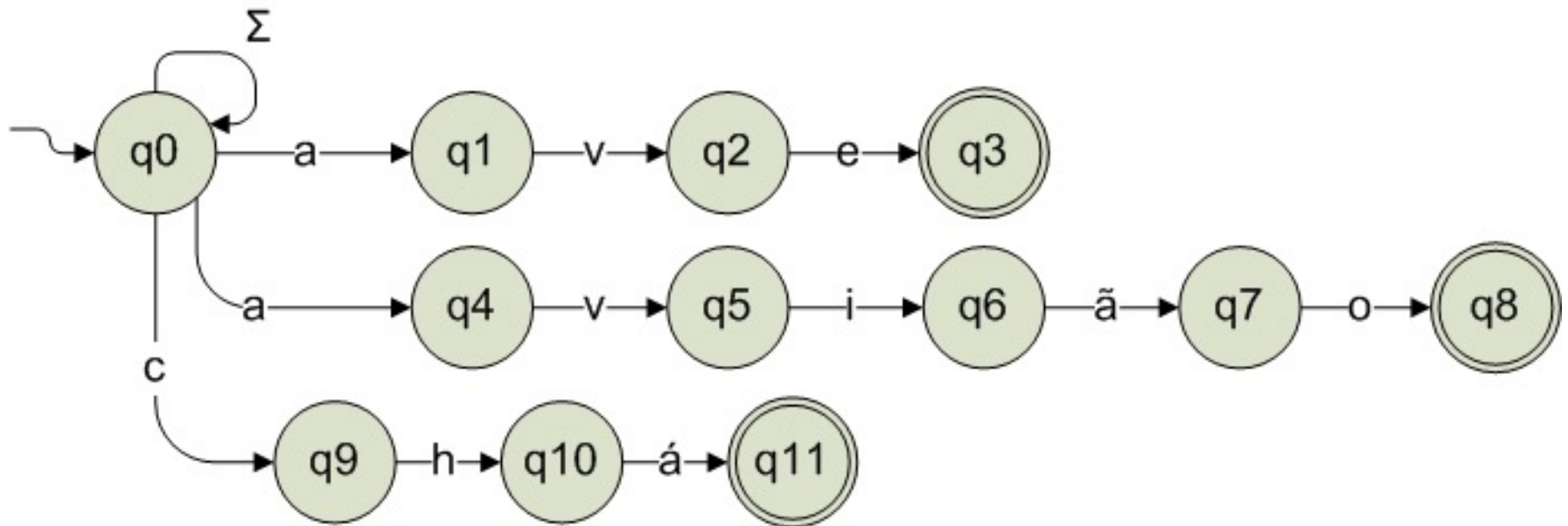
(Use notação de conjuntos)

Descreva a linguagem reconhecida por este autômato

## Interpretando NFAs

- Resposta
  - Aceita cadeias cujo símbolo final já apareceu antes

## Interpretando NFAs



## Interpretando NFAs

- Calcule a função estendida e mostre, passo a passo, as configurações instantâneas para as seguintes cadeias:
  - ave
  - avião
  - aves
  - chave
  - (Use notação de árvore ou conjuntos)
- Descreva a linguagem reconhecida por este autômato

## Interpretando NFAs

Dado o seguinte autômato finito:

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
$q_1$	$\emptyset$	$\{q_2\}$
$* q_2$	$\emptyset$	$\emptyset$

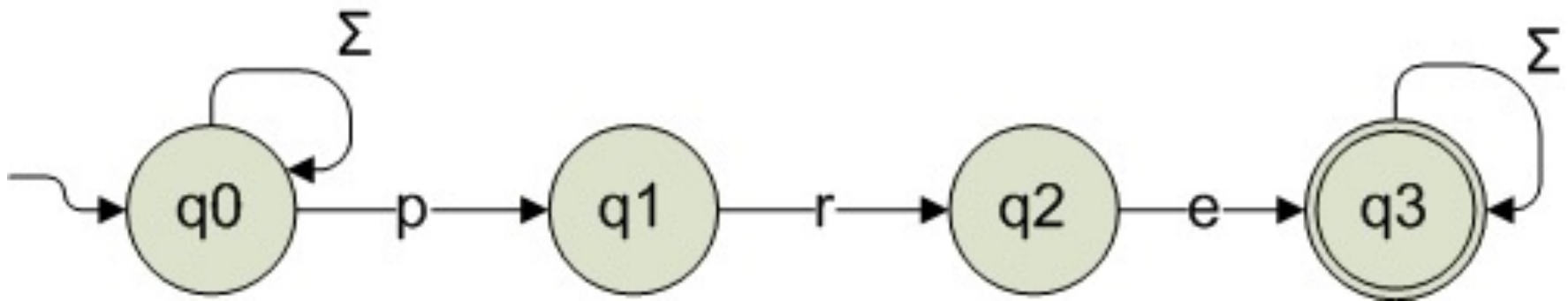


## Interpretando NFAs

- Calcule a função estendida e mostre, passo a passo, as configurações instantâneas para as seguintes cadeias:
  - 0101010
  - 11111
  - 001
  - 1101
  - (Use notação de árvore ou conjuntos)
- Descreva a linguagem aceita por este autômato
  - Resp: cadeias que terminam em 01

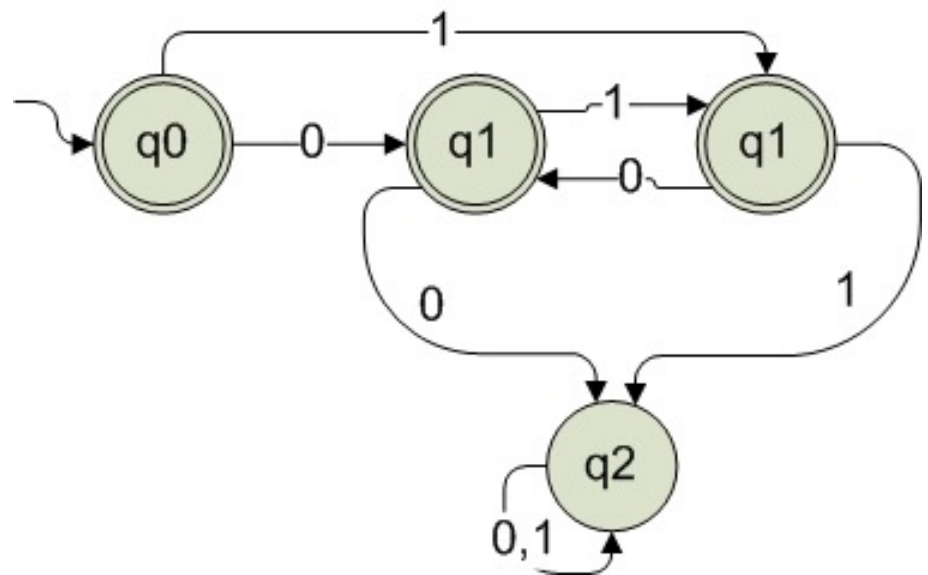
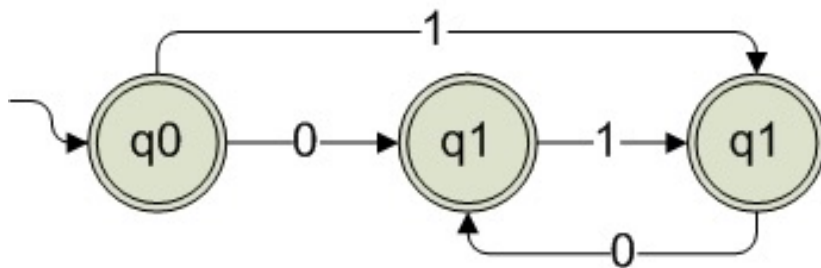
## Projetando NFAs

- Ex:
  - $\Sigma = \{a,b,c,\dots,z\}$
  - Linguagem = cadeias que contêm a cadeia “pre” como uma subcadeia



# Projetando NFAs

- Ex:
  - $\Sigma = \{0,1\}$
  - Linguagem = cadeias que não possuem símbolos repetidos em sequência



## Implementando NFAs

A implementação é mais complexa do que o DFA

Mas em essência é o mesmo mecanismo

Envolve duas estratégias

Processamento paralelo

“Backtracking”

## Equivalência DFA e NFA

Intuitivamente, NFA é mais poderoso

Mas as linguagens aceitas por um NFA são regulares

Ou seja, qualquer NFA pode ser convertido em um DFA que reconhece a mesma linguagem

### **Teorema:**

Uma Linguagem  $L$  é aceita por algum DFA se e somente se  $L$  é aceita por algum NFA

Prova por construção dos dois lados:

“Se”: um processo que constrói um DFA a partir de um NFA

“Somente se”: um processo que constrói um NFA a partir de um DFA

## Conversão NFA $\rightarrow$ DFA

Na maioria dos casos, um DFA equivalente tem o mesmo número de estados que o NFA, só que mais transições

No pior caso, tem  $2^n$  estados

$$\text{NFA } N = (Q_N, \Sigma, \delta_N, q_0, F_N)$$

$$\text{DFA } D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$$

$$L(D) = L(N)$$

$Q_D$  é o conjunto de subconjuntos de  $Q_N$  ( $2^{Q_N}$ )

$F_D$  é o conjunto  $S$  de subconjuntos de  $Q_N$ ,  
tal que  $S \cap F_N \neq \emptyset$

Para cada conjunto  $S \subseteq Q_N$

e para cada  $a \in \Sigma$

$$\delta_D(S, a) = \cup \text{ todos os } \delta_N(p, a) \text{ para } p \in S$$

## Conversão NFA $\rightarrow$ DFA

Consiste em pegar todas as combinações de estados e agregar as transições do NFA

Cada combinação de estados do NFA é um estado no DFA

Consiste basicamente na implementação “em paralelo”  
Mas pré-calculando as combinações de estados

## Conversão NFA $\rightarrow$ DFA

Passo a passo com exemplo

Dado o NFA (cadeias que terminam com 01):

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
$q_1$	$\emptyset$	$\{q_2\}$
$* q_2$	$\emptyset$	$\emptyset$



## Conversão NFA $\rightarrow$ DFA

Passo 1:

Faça uma tabela “vazia”, com as mesmas entradas como colunas (a tabela vai crescer para baixo)

	0	1

## Conversão NFA $\rightarrow$ DFA

Passo 2:

Crie um novo estado inicial no DFA, um conjunto que contém somente o estado inicial do NFA

	0	1
$\rightarrow \{q_0\}$		

## Conversão NFA $\rightarrow$ DFA

Passo 3:

Para cada entrada, insira no DFA um conjunto que contém a união de todos os resultados da transição NFA daquela entrada para todos os estados do conjunto à esquerda

	0	1
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$

## Conversão NFA $\rightarrow$ DFA

Passo 4:

Para cada novo conjunto de estados que aparecer, insira uma nova linha na tabela do DFA e volte para o passo 3

	0	1
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

## Conversão NFA $\rightarrow$ DFA

Passo 4 (novamente):

	0	1
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

## Conversão NFA $\rightarrow$ DFA

Passo 5: Quando não houver mais novos estados, marque como estado de aceitação os conjuntos que contém ao menos um estado de aceitação do NFA

	0	1
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$* \{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

## Conversão NFA $\rightarrow$ DFA

- Passo 6: “Renomeie” os conjuntos para estados, de forma a facilitar a leitura do DFA

	0	1
$\rightarrow A$	B	A
B	B	C
* C	B	A

## Conversão NFA $\rightarrow$ DFA

- Dado o seguinte NFA:
  - Construa um DFA que aceite a mesma linguagem

	0	1
$\rightarrow p$	{p,q}	{p}
q	{r}	{r}
r	{s}	$\emptyset$
* s	{s}	{s}



## Conversão NFA $\rightarrow$ DFA

	0	1
$\rightarrow \{p\}$ A	$\{p,q\}$ B	$\{p\}$ A
$\{p,q\}$ B	$\{p,q,r\}$ D	$\{p,r\}$ C
$\{p,r\}$ C	$\{p,q,s\}$ E	$\{p\}$ A
$\{p,q,r\}$ D	$\{p,q,r,s\}$ F	$\{p,r\}$ C
* $\{p,q,s\}$ E	$\{p,q,r,s\}$ F	$\{p,r,s\}$ G
* $\{p,q,r,s\}$ F	$\{p,q,r,s\}$ F	$\{p,r,s\}$ G
* $\{p,r,s\}$ G	$\{p,q,s\}$ E	$\{p,s\}$ H
* $\{p,s\}$ H	$\{p,q,s\}$ E	$\{p,s\}$ H

## Conversão NFA $\rightarrow$ DFA

Dado o seguinte NFA:

Construa um DFA que aceite a mesma linguagem

	0	1
$\rightarrow$ * q0	{q1}	{q2}
* q1	$\emptyset$	{q0}
* q2	{q0}	$\emptyset$

## Conversão NFA $\rightarrow$ DFA

	0	1
$\rightarrow$ * {q0} A	{q1} B	{q2} C
* {q1} B	{ } D	{q0} A
* {q2} C	{q0} A	{ } D
{ } D (morto)	{ } D	{ } D

## Conversão DFA $\rightarrow$ NFA

“Resto” da prova

Parte fácil

Construir um NFA a partir de um DFA

Basta “copiar” o diagrama (ou tabela), trocando estados por conjuntos de estados

Um DFA é um caso específico de NFA

NFA permite 0 ou mais transições em cada situação

DFA permite sempre 1 transição em cada situação

1 está entre 0 ou mais

## Conversão DFA $\rightarrow$ NFA

	0	1
$\rightarrow q1$	q1	q2
* q2	q1	q2



	0	1
$\rightarrow q1$	{q1}	{q2}
* q2	{q1}	{q2}

## Conversão DFA $\rightarrow$ NFA

Formalmente:

Seja  $D = (Q, \Sigma, \delta_D, q_0, F)$  um DFA

Defina  $N = (Q, \Sigma, \delta_N, q_0, F)$

Onde  $\delta_N$  é definido pela regra:

Se  $\delta_D(q, a) = p$ , então  $\delta_N(q, a) = \{p\}$

Como consequência

Se  $\delta_D^*(q_0, w) = p$ , então  $\delta_N^*(q_0, w) = \{p\}$

Portanto,  $w$  é aceito por  $D$  se e somente se é aceito por  $N$

Isto é:  $L(D) = L(N)$

# Autômatos Finitos com Movimentos Vazios

## AF com movimentos ou transições vazias

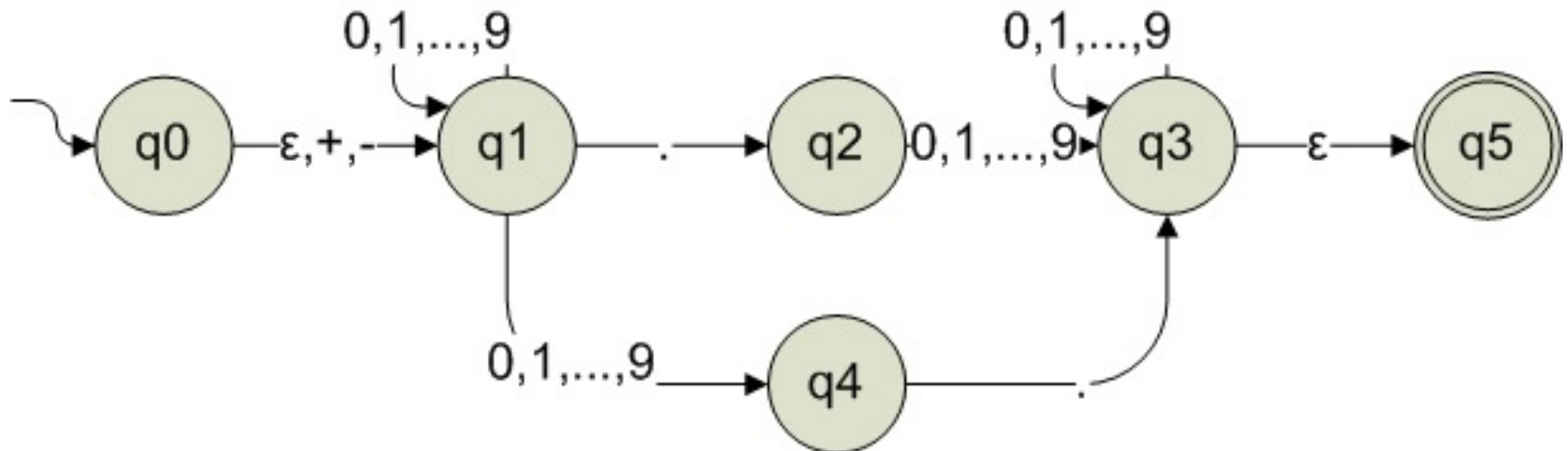
Movimentos ou Transições espontâneas

Isto é, sem nenhuma entrada

É uma forma de não-determinismo

Facilita a “programação”

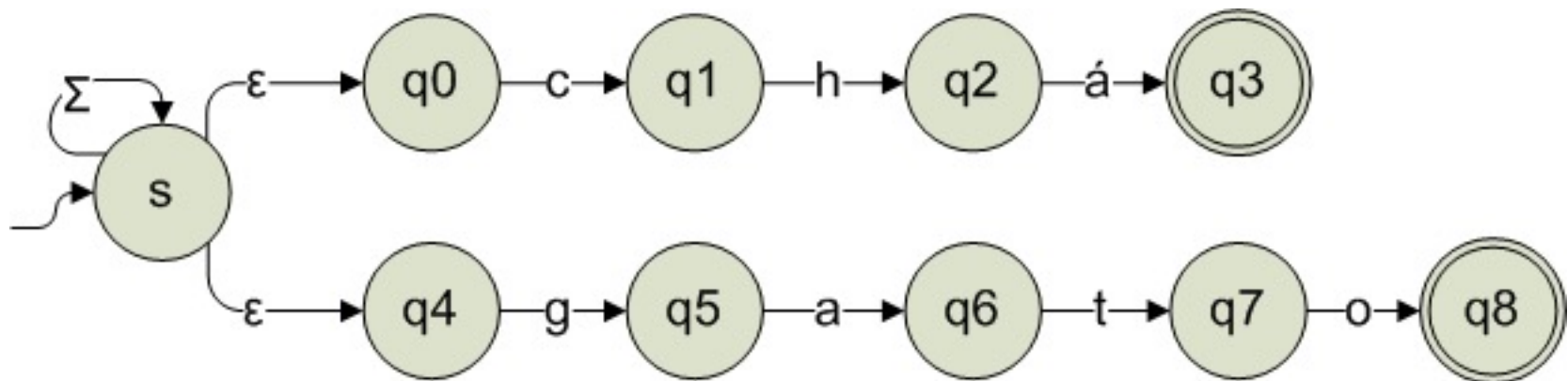
Ex: números decimais





## AF com movimentos vazios

Ex: busca por palavras-chave



## NFA com transições vazias

Definição formal

A mesma que NFA

Muda somente a função de transição

$$\delta: Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$$

Uma coluna extra na tabela, ou transições vazias no diagrama

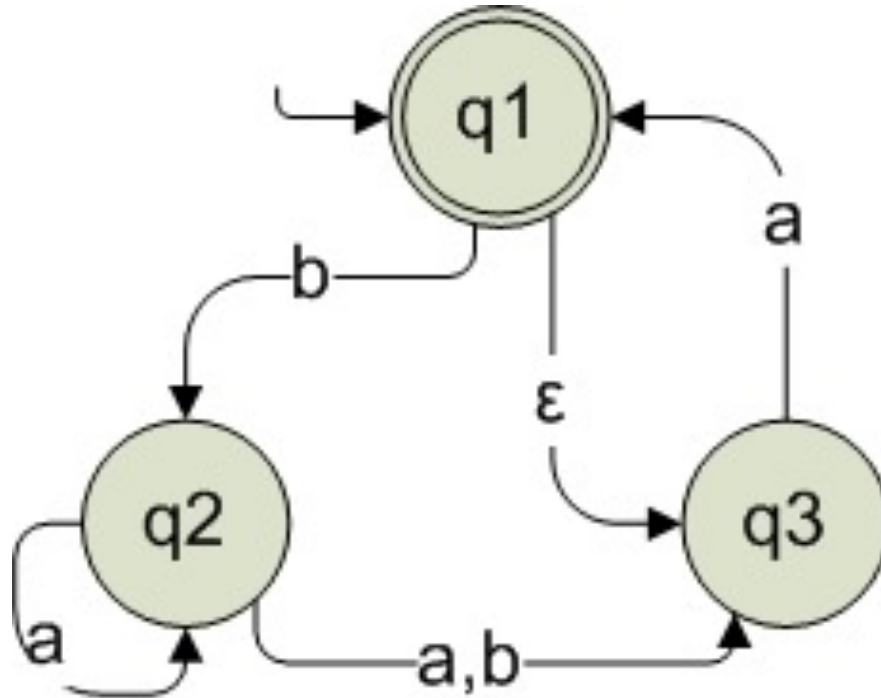
	$\varepsilon$	$+, -$	$.$	$0, 1, \dots, 9$
$\rightarrow q_0$	$\{q_1\}$	$\{q_1\}$	$\emptyset$	$\emptyset$
$q_1$	$\emptyset$	$\emptyset$	$\{q_2\}$	$\{q_1, q_4\}$
$q_2$	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_3\}$
$q_3$	$\{q_5\}$	$\emptyset$	$\emptyset$	$\{q_3\}$
$q_4$	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
$* q_5$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

## AF com movimentos vazios

- Essa nova característica não aumenta o poder do NFA
  - Ainda reconhece linguagens regulares
- Conceito de épsilon-fechamento
  - $\text{ECLOSE}(q)$
  - Conjunto de todos os estados alcançáveis espontaneamente a partir de  $q$ 
    - Incluindo os vizinhos diretos e indiretos
    - Analisando-se os arcos rotulados com  $\epsilon$
- Função de transição estendida
  - Deve considerar sempre o  $\text{ECLOSE}$

## Interpretando $\epsilon$ -NFAs

- Dado o seguinte autômato

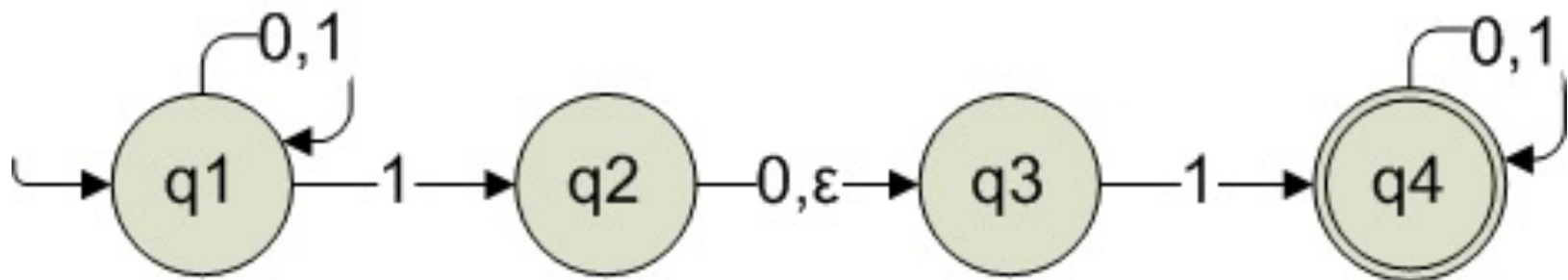


## Interpretando $\epsilon$ -NFAs

- Calcule a função estendida e mostre, passo a passo, as configurações instantâneas para as seguintes cadeias:
  - $\epsilon$
  - a
  - baba
  - baa
  - b
  - bb
  - babba
  - (Use notação de árvore ou conjuntos)

## Interpretando $\epsilon$ -NFAs

- Dado o seguinte autômato

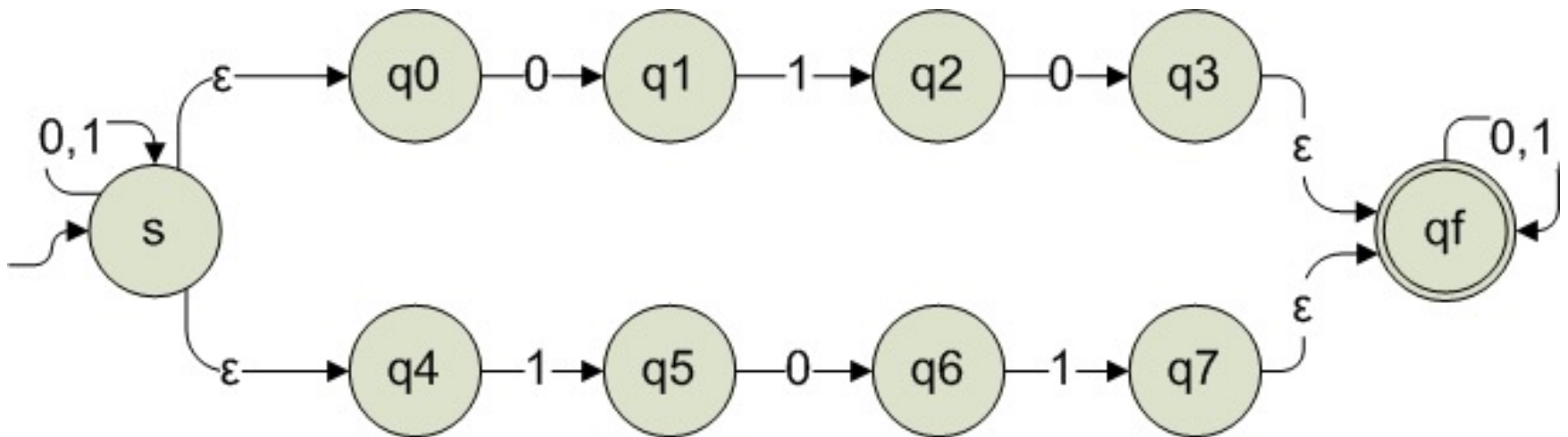


## Interpretando $\varepsilon$ -NFAs

- Calcule a função estendida e mostre, passo a passo, as configurações instantâneas para as seguintes cadeias:
  - 1111
  - 11
  - 101
  - 00010
  - (Use notação de conjuntos)

## Projetando $\epsilon$ -NFAs

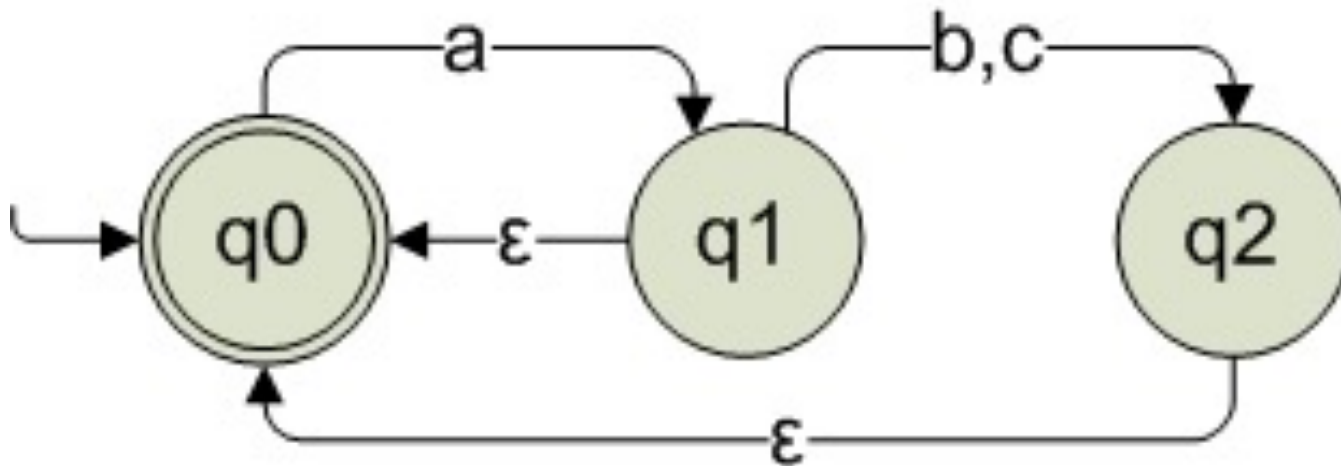
- Ex:
  - $\Sigma = \{0,1\}$
  - Linguagem = cadeias que contêm a sequência 010 ou 101 como subcadeia





## Projetando $\epsilon$ -NFAs

- Ex:
  - $\Sigma = \{a,b,c\}$
  - Linguagem = cadeias onde b e c sempre aparecem depois de uma ocorrência de a



## Implementando $\epsilon$ -NFAs

- Envolve a implementação do épsilon-fechamento
  - E também precisa de uma coluna para transições vazias

## Equivalência $\epsilon$ -NFAs e DFAs

- Transições vazias não adicionam poder ao autômato
  - Ainda reconhece as mesmas linguagens
  - Linguagens regulares
- Teorema:
  - Uma Linguagem  $L$  é aceita por algum DFA se e somente se  $L$  é aceita por algum  $\epsilon$ -NFA
  - Prova por construção dos dois lados:
    - “Se”: um processo que constrói um DFA a partir de um  $\epsilon$ -NFA
    - “Somente se”: um processo que constrói um  $\epsilon$ -NFA a partir de um DFA

## Conversão $\epsilon$ -NFA $\rightarrow$ DFA

- É o mesmo procedimento da construção de subconjuntos dado anteriormente
  - Porém incorporando o cálculo de  $\epsilon$ -fechamento após cada passo
  - Similar à implementação do  $\epsilon$ -NFA

## Conversão $\varepsilon$ -NFA $\rightarrow$ DFA

- Passo a passo com exemplo
- Dado o NFA:

	$\varepsilon$	a	b	c
$\rightarrow p$	$\emptyset$	$\{p\}$	$\{q\}$	$\{r\}$
q	$\{p\}$	$\{q\}$	$\{r\}$	$\emptyset$
* r	$\{q\}$	$\{r\}$	$\emptyset$	$\{p\}$

## Conversão $\varepsilon$ -NFA $\rightarrow$ DFA

- Passo 1 (auxiliar): calcule o ECLOSE de todos os estados
- $\text{ECLOSE}(p) = \{p\}$
- $\text{ECLOSE}(q) = \{p, q\}$
- $\text{ECLOSE}(r) = \{p, q, r\}$

	$\varepsilon$	a	b	c
$\rightarrow p$	$\emptyset$	$\{p\}$	$\{q\}$	$\{r\}$
q	$\{p\}$	$\{q\}$	$\{r\}$	$\emptyset$
* r	$\{q\}$	$\{r\}$	$\emptyset$	$\{p\}$

## Conversão $\epsilon$ -NFA $\rightarrow$ DFA

- Passo 2: faça uma tabela “vazia” com as entradas (sem a coluna  $\epsilon$ )

	a	b	c

## Conversão $\epsilon$ -NFA $\rightarrow$ DFA

- Passo 3: Crie um novo estado inicial no DFA, um conjunto que contém o ECLOSE do estado inicial do NFA

	a	b	c
$\rightarrow \{p\}$			



## Conversão $\epsilon$ -NFA $\rightarrow$ DFA

- Passo 4:
  - Para cada entrada, insira no DFA um conjunto que contém o ECLOSE da união de todos os resultados da transição NFA daquela entrada para todos os estados do conjunto à esquerda

	a	b	c
$\rightarrow \{p\}$	$\{p\}$	$\{p,q\}$	$\{p,q,r\}$

## Conversão $\varepsilon$ -NFA $\rightarrow$ DFA

- Passo 5:
  - Para cada novo conjunto de estados que aparecer, insira uma nova linha na tabela do DFA e volte para o passo 4

	a	b	c
$\rightarrow \{p\}$	$\{p\}$	$\{p,q\}$	$\{p,q,r\}$
$\{p,q\}$	$\{p,q\}$	$\{p,q,r\}$	$\{p,q,r\}$
$\{p,q,r\}$	$\{p,q,r\}$	$\{p,q,r\}$	$\{p,q,r\}$

## Conversão $\varepsilon$ -NFA $\rightarrow$ DFA

- Passo 6: Quando não houver mais novos estados, marque como estado de aceitação os conjuntos que contém ao menos um estado de aceitação do NFA

	a	b	c
$\rightarrow \{p\}$	$\{p\}$	$\{p,q\}$	$\{p,q,r\}$
$\{p,q\}$	$\{p,q\}$	$\{p,q,r\}$	$\{p,q,r\}$
$* \{p,q,r\}$	$\{p,q,r\}$	$\{p,q,r\}$	$\{p,q,r\}$

## Conversão $\varepsilon$ -NFA $\rightarrow$ DFA

- Passo 7: “Renomeie” os conjuntos para estados, de forma a facilitar a leitura do DFA

	a	b	c
$\rightarrow$ A	A	B	C
B	B	C	C
* C	C	C	C

## Conversão $\varepsilon$ -NFA $\rightarrow$ DFA

- Dado o seguinte  $\varepsilon$ -NFA
  - Converta para um DFA que aceita a mesma linguagem

	$\varepsilon$	a	b	c
$\rightarrow p$	$\{q, r\}$	$\emptyset$	$\{q\}$	$\{r\}$
q	$\emptyset$	$\{p\}$	$\{r\}$	$\{p, q\}$
* r	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

## Conversão $\varepsilon$ -NFA $\rightarrow$ DFA

- Resposta

	a	b	c
$\rightarrow^* \{p, q, r\}$	$\{p, q, r\}$	$\{q, r\}$	$\{p, q, r\}$
$^* \{q, r\}$	$\{p, q, r\}$	$\{r\}$	$\{p, q, r\}$
$^* \{r\}$	$\{\}$	$\{\}$	$\{\}$
$\{\}$	$\{\}$	$\{\}$	$\{\}$

## Conversão $\varepsilon$ -NFA $\rightarrow$ DFA

- Dado o seguinte  $\varepsilon$ -NFA
  - Converta para um DFA que aceita a mesma linguagem

	$\varepsilon$	a	b
$\rightarrow^* 1$	$\{3\}$	$\emptyset$	$\{2\}$
2	$\emptyset$	$\{2,3\}$	$\{3\}$
3	$\emptyset$	$\{1\}$	$\emptyset$

## Conversão $\varepsilon$ -NFA $\rightarrow$ DFA

- Resposta

	a	b
$\rightarrow^* \{1,3\}$	$\{1,3\}$	$\{2\}$
$\{2\}$	$\{2,3\}$	$\{3\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\{3\}$	$\{1,3\}$	$\{\}$
$^* \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\{\}$	$\{\}$	$\{\}$



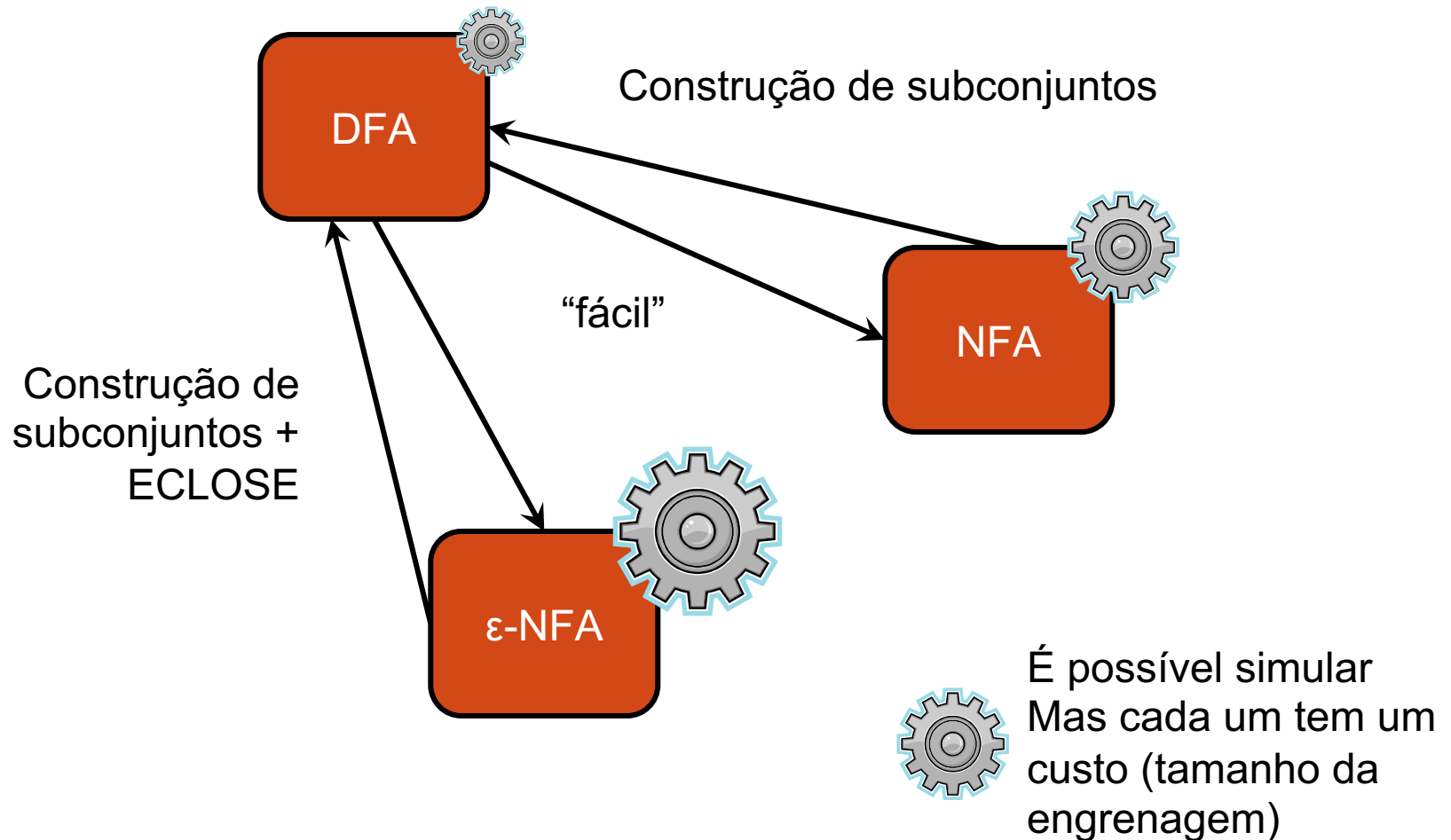
## Conversão DFA $\rightarrow$ $\epsilon$ -NFA

- “Resto” da prova
- Parte fácil
  - Mesmo caso da conversão de DFA para NFA
  - Mas fazendo com que  $\delta(q, \epsilon) = \emptyset$  para todo estado  $q$  do DFA
  - Ou seja, não existem transições espontâneas (mas poderia ter)

# Resumo

- Definições X Linguagens X Problemas
- Autômatos Finitos
  - DFA
  - NFA
  - $\epsilon$ -NFA
- Aceitam as mesmas linguagens (regulares)

# Resumo



Fim