Aspectos Formais da Computação

Prof. Sergio D. Zorzo
Departamento de Computação - UFSCar
1º semestre / 2017
05

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?

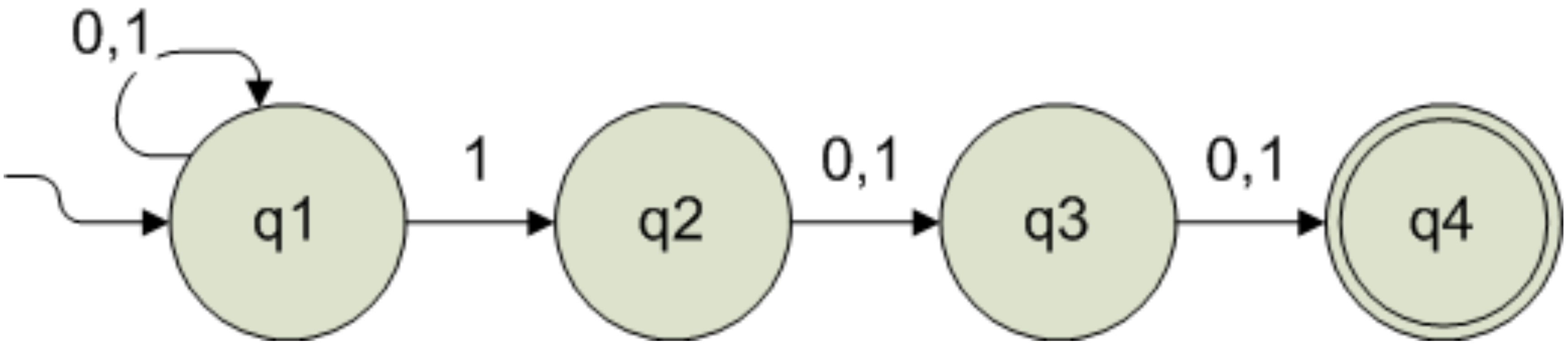
- Uma visão: um NFA pode “adivinhar” algumas coisas sobre a entrada
- Outra visão: em transições para mais de um estado, é o mesmo que dividir o autômato em dois e seguir cada execução em paralelo
- Outra: uma árvore de possibilidades – tentativa e erro

Não-Determinismo

Mais fáceis de projetar e entender

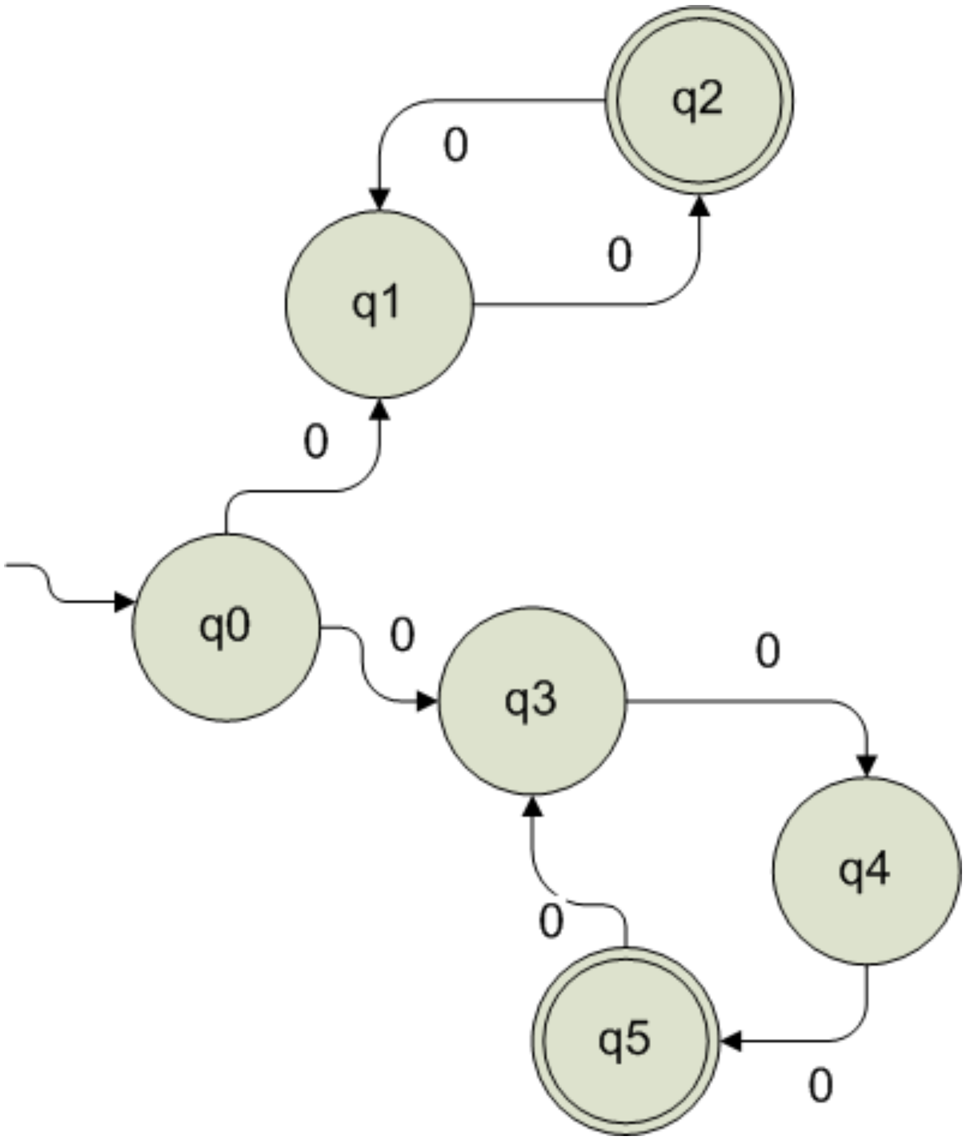
Ex: $\Sigma = \{0,1\}$

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
- Σ=Conjunto finito de símbolos de entrada
- δ=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

- $\boldsymbol{\delta: Q \times \Sigma \rightarrow 2^Q}$

Autômatos finitos determinísticos

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

Definição formal de linguagem (indutiva)

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

Definição:

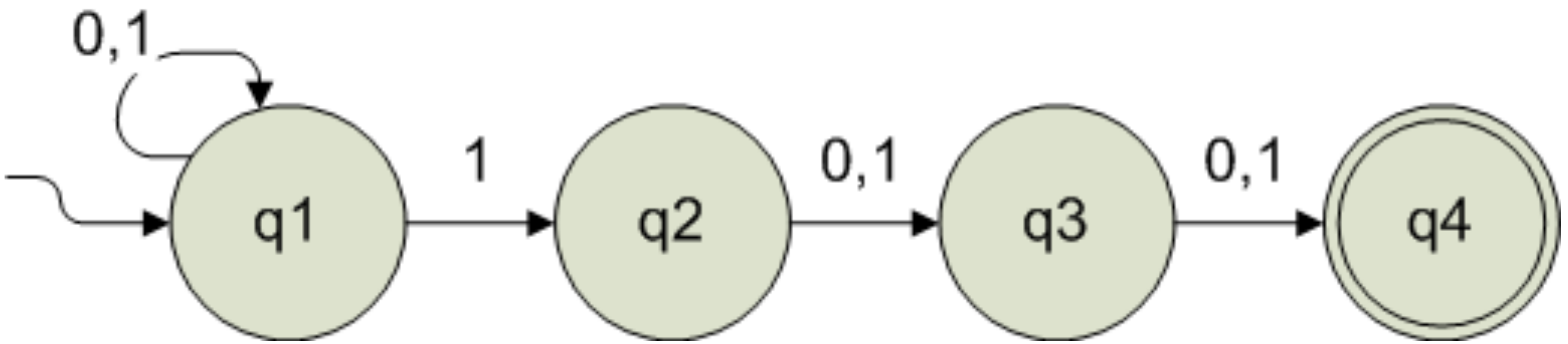
- Se L é L(M) para algum DFA M
- L é regular

Autômatos finitos não-determinísticos

- Definição formal de linguagem
  - $\delta\hat{}(q,\varepsilon)=\{q\}$
  - $\delta\hat{}(q,x)= \{p_1,p_2,\ldots,p_k\}$
  - $\delta\hat{}(q,w)$=união de todos $\delta(p_i,a)$, onde $w=xa$
  - $L(M)=\{w \mid \delta\hat{}(q_o,w) \cap F \neq \varnothing\}$

- 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

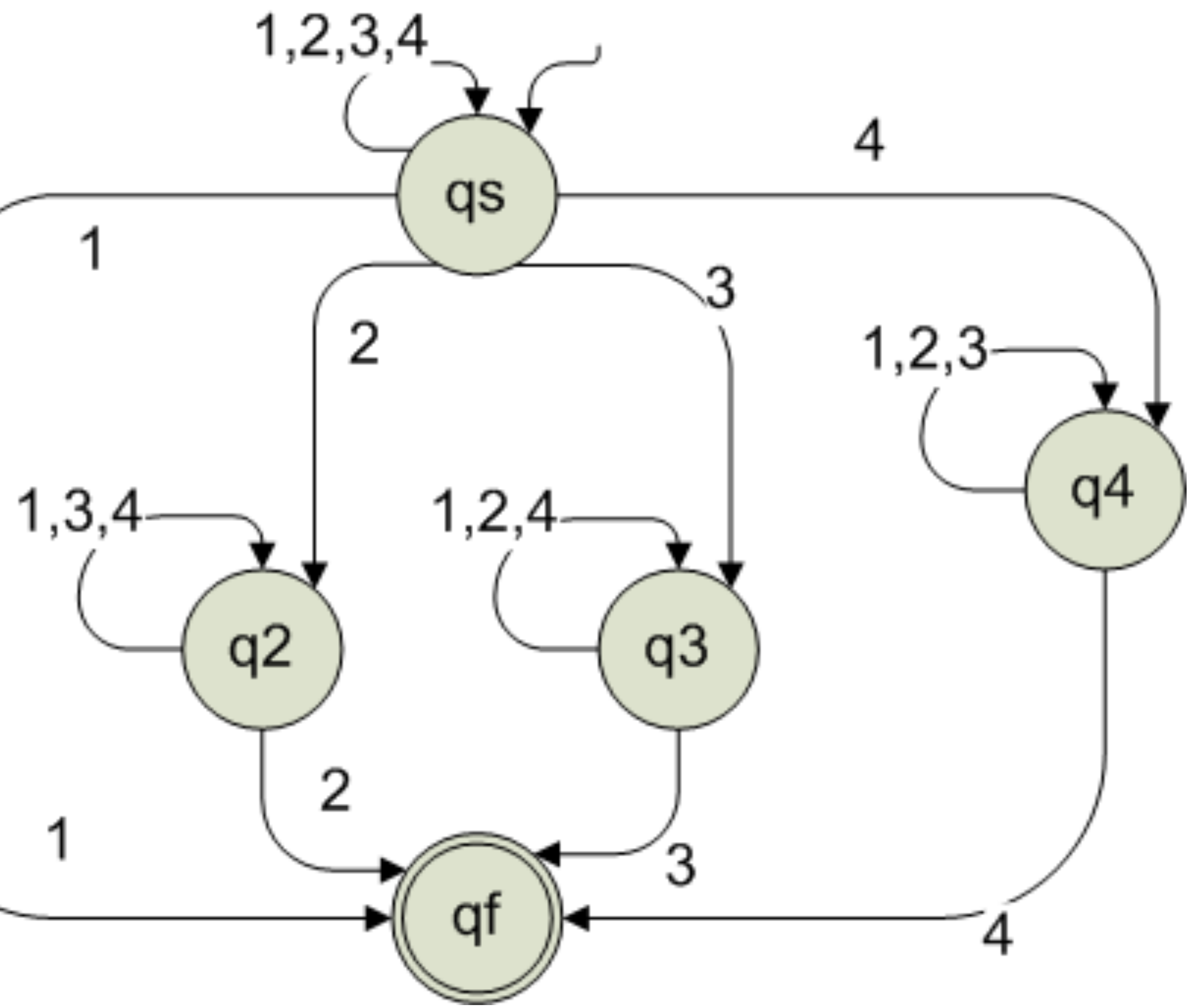
0100

(Use notação de conjuntos)

Descreva a linguagem reconhecida por este autômato

Interpretando NFAs

Σ = {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

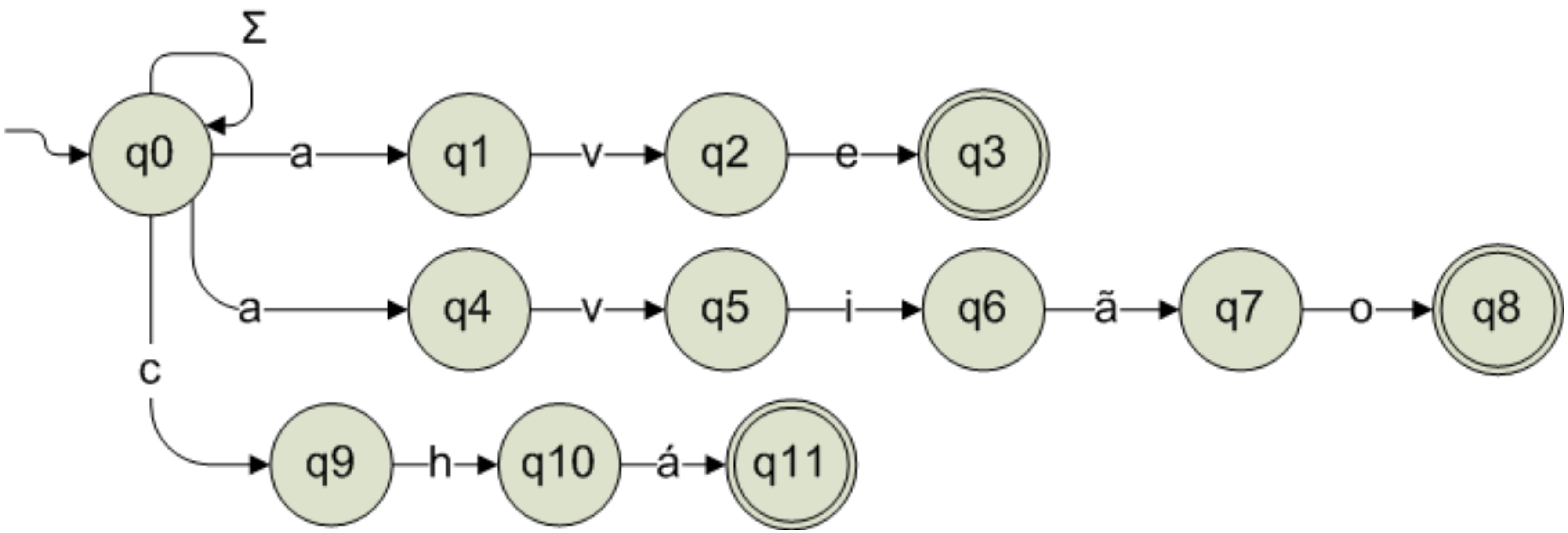
(Use notação de conjuntos)

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:

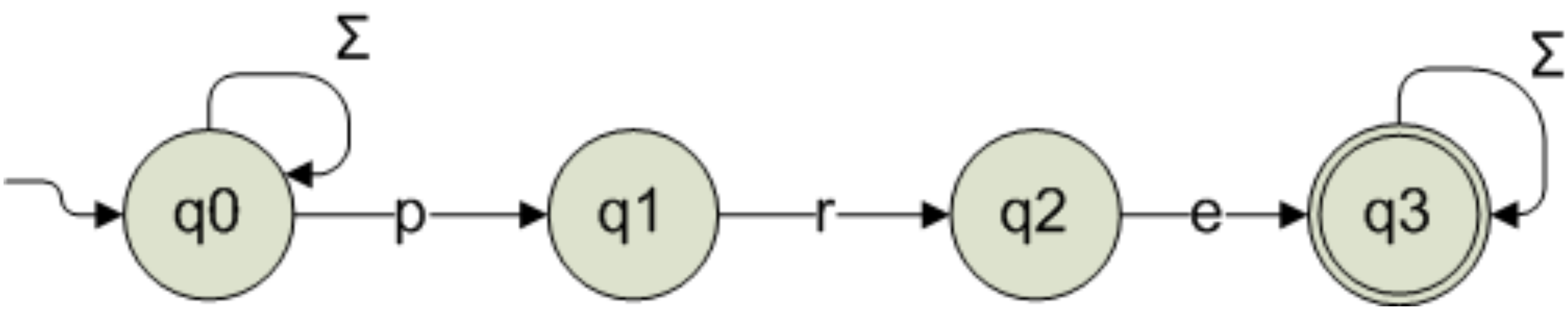
	0	**1**
→ q0	{q0,q1}	{q0}
q1	$\varnothing$	{q2}
***** q2**	$\varnothing$	$\varnothing$

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

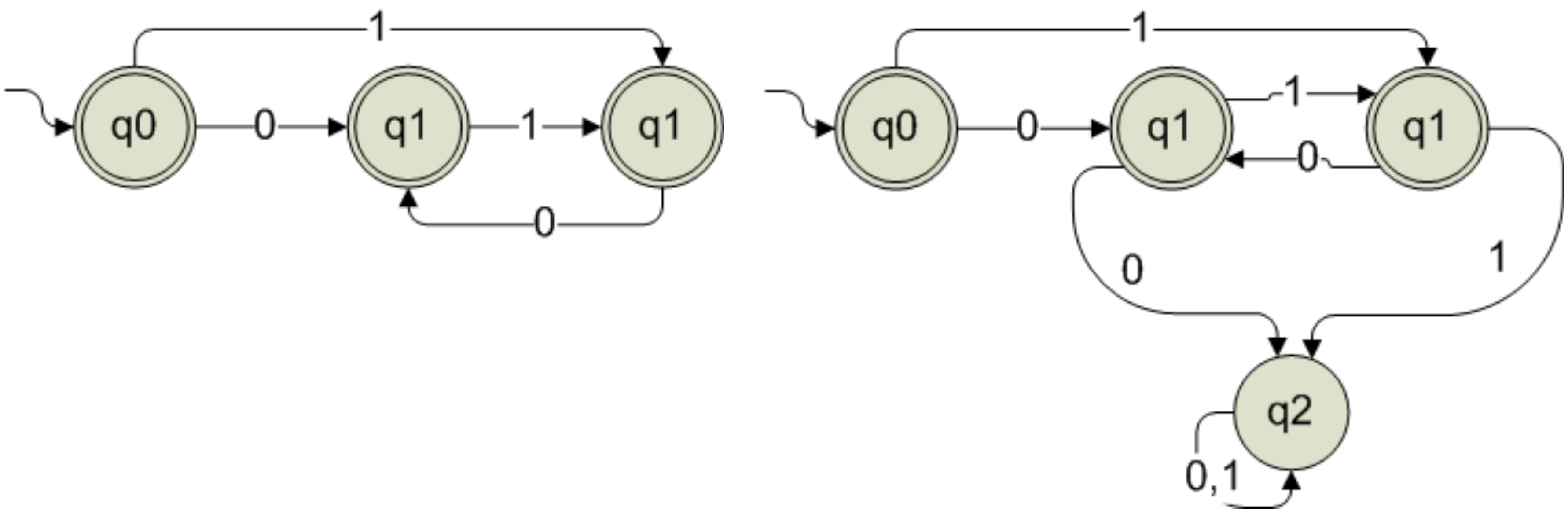
Projetando NFAs

- Ex:
  - $\Sigma = \{a,b,c,\ldots,z\}$
  - Linguagem = cadeias que contém a cadeia "pre" como uma subcadeia



Projetando NFAs

- Ex:
  - Σ = {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 → 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

NFA N = $(Q_N, \Sigma, \delta_N, q0, F_N)$

DFA D = $(Q_D, \Sigma, \delta_D, \{q0\}, F_D)$

$L(D) = L(N)$

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

F_D é o conjunto S de subconjuntos de Q_N,

tal que $S \cap F_N \neq \varnothing$

Para cada conjunto $S \subseteq Q_N$

e para cada $a \in \Sigma$

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

Conversão NFA → 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 → DFA

Passo a passo com exemplo

Dado o NFA (cadeias que terminam com 01):

	0	**1**
→ q0	{q0,q1}	{q0}
q1	∅	{q2}
*** q2**	∅	∅

Conversão NFA → DFA

Passo 1:

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

	0	**1**

Conversão NFA → DFA

Passo 2:

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

	0	**1**
→ {q0}		

Conversão NFA → 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**
→ **{q0}**	{q0,q1}	{q0}

Conversão NFA → 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**
→ **{q0}**	{q0,q1}	{q0}
{q0,q1}	{q0,q1}	{q0,q2}

Conversão NFA → DFA

Passo 4 (novamente):

	0	**1**
→ **{q0}**	{q0,q1}	{q0}
{q0,q1}	{q0,q1}	{q0,q2}
{q0,q2}	{q0,q1}	{q0}

Conversão NFA → 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**
→ **{q0}**	{q0,q1}	{q0}
{q0,q1}	{q0,q1}	{q0,q2}
* **{q0,q2}**	{q0,q1}	{q0}

Conversão NFA → DFA

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

	0	**1**
→ A	B	A
B	B	C
*** C**	B	A

Conversão NFA → DFA

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

	0	**1**
→ **p**	{p,q}	{p}
q	{r}	{r}
r	{s}	∅
* **s**	{s}	{s}

Conversão NFA → DFA

	0	**1**
→ {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 → DFA

Dado o seguinte NFA:

Construa um DFA que aceite a mesma linguagem

	0	**1**
→ * q0	{q1}	{q2}
*** q1**	∅	{q0}
*** q2**	{q0}	∅

Conversão NFA → DFA

	0	**1**
→ * {q0} A	{q1} B	{q2} C
*** {q1} B**	{} D	{q0} A
*** {q2} C**	{q0} A	{} D
{} D (morto)	{} D	{} D

Conversão DFA → 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 → NFA

	0	**1**
→ q1	q1	q2
*** q2**	q1	q2

	0	**1**
→ q1	{q1}	{q2}
*** q2**	{q1}	{q2}

Conversão DFA → NFA

Formalmente:

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

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

Onde δ_N é definido pela regra:

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

Como consequência

Se $\delta\text{^}_D(q_0,w)=p$, então $\delta\text{^}_N(q_0,w) = \{p\}$

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

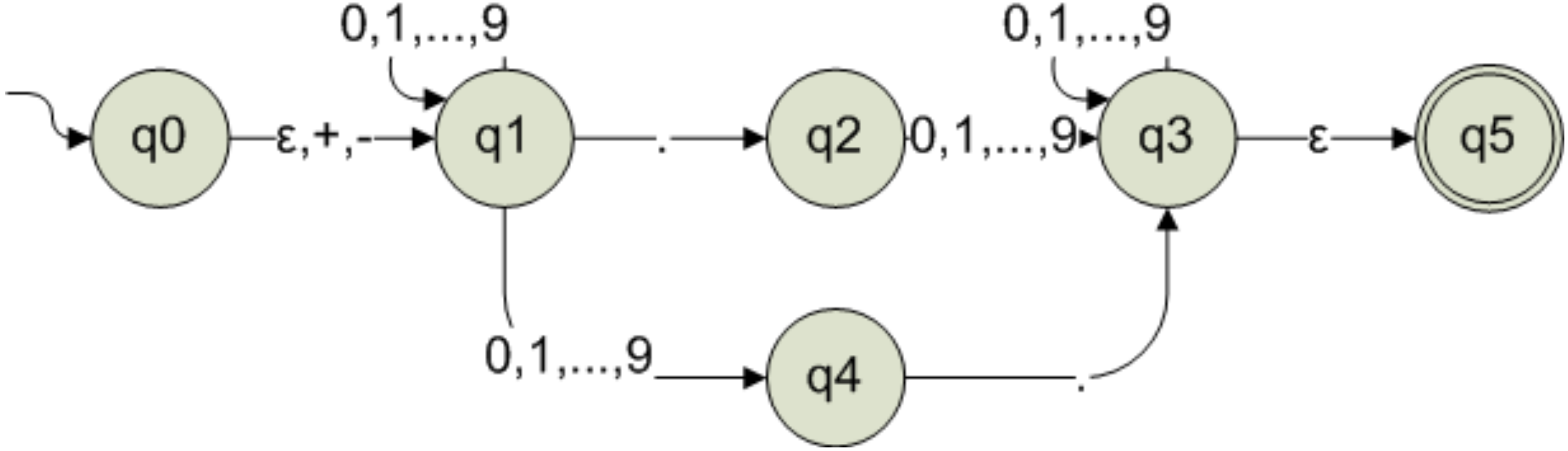
Movimentos ou Transições espontâneas

- Isto é, sem nenhuma entrada

É uma forma de não-determinismo

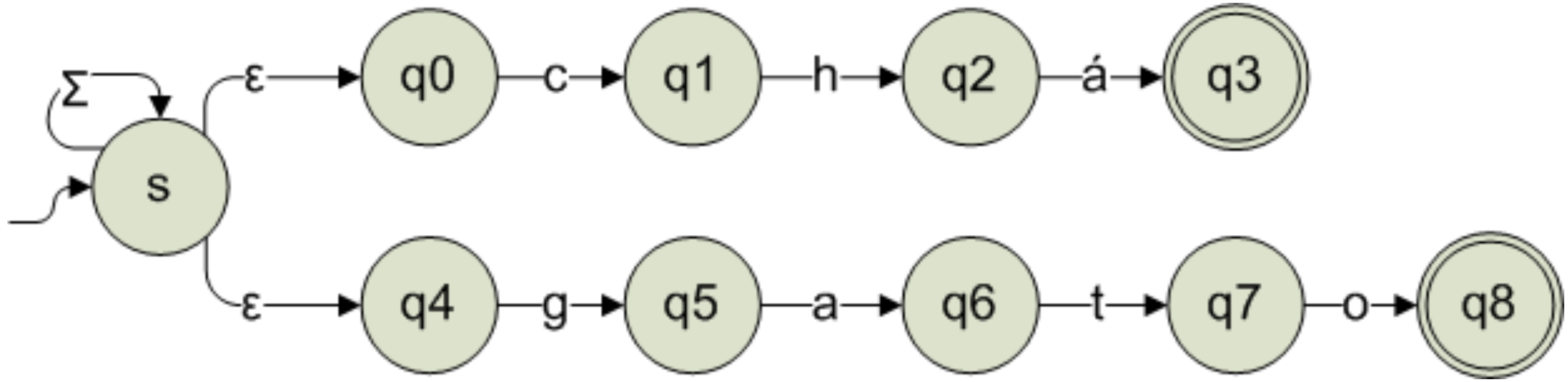
- Facilita a "programação"

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

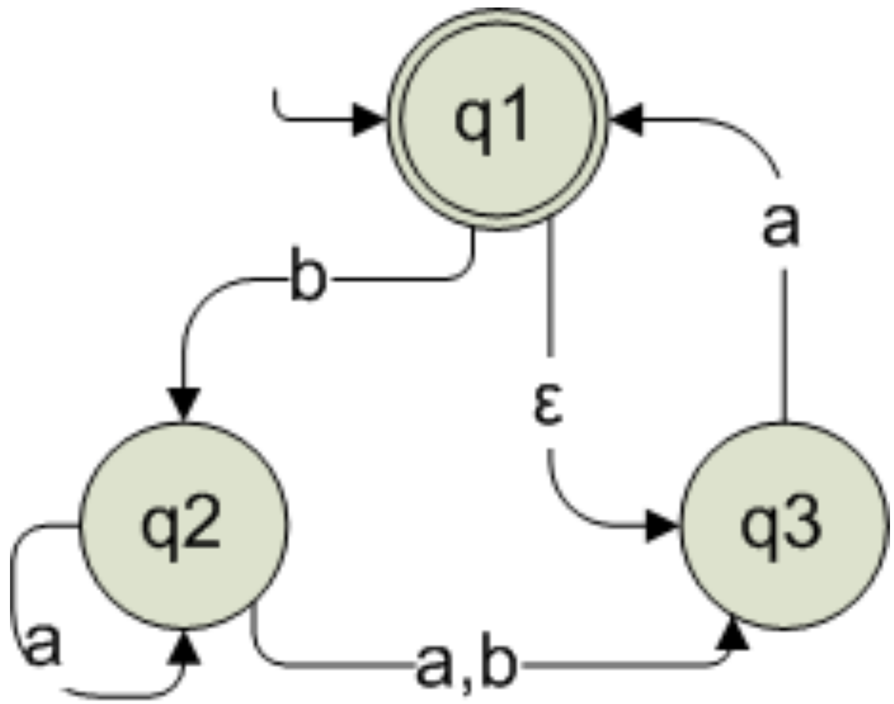
	ε	**+,-**	**.**	**0,1,...,9**
→ q0	{q1}	{q1}	∅	∅
q1	∅	∅	{q2}	{q1,q4}
q2	∅	∅	∅	{q3}
q3	{q5}	∅	∅	{q3}
q4	∅	∅	{q3}	∅
*** q5**	∅	∅	∅	∅

AF com movimentos vazios

- Essa nova característica não aumenta o poder do NFA
  - Ainda reconhece linguagens regulares
- Conceito de épsilon-fechamento
  - 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 ε
- Função de transição estendida
  - Deve considerar sempre o ECLOSE

Interpretando ε-NFAs

- Dado o seguinte autômato

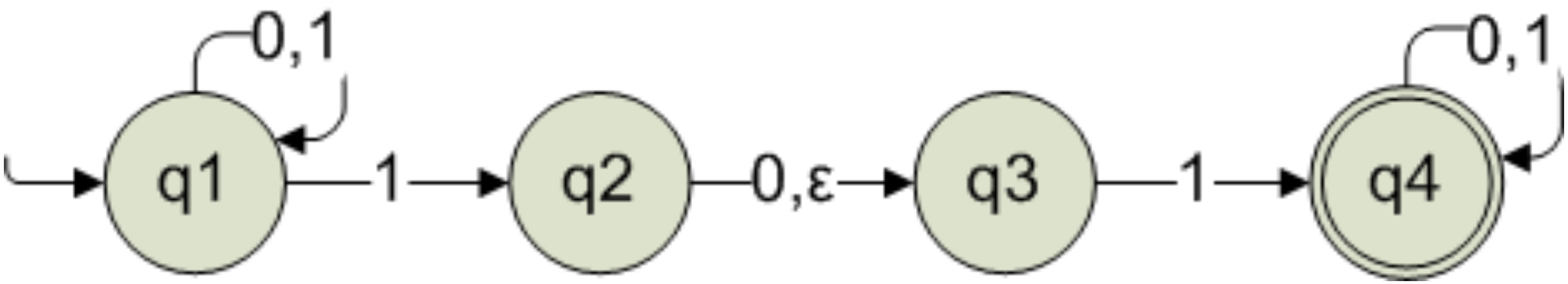


Interpretando ε-NFAs

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

Interpretando ε-NFAs

- Dado o seguinte autômato

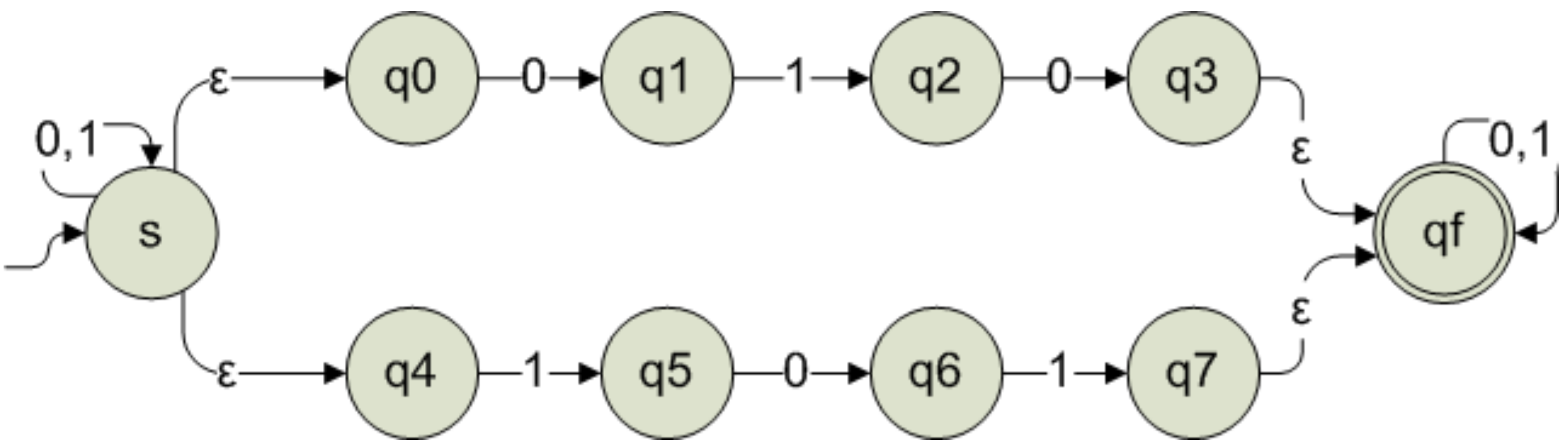


Interpretando ε-NFAs

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

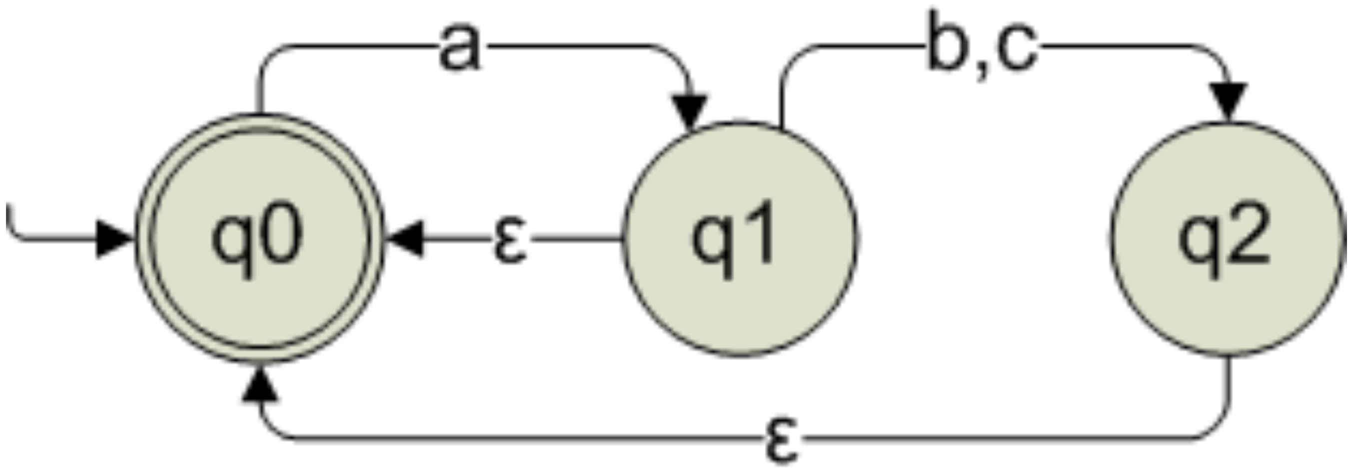
Projetando ε-NFAs

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



Projetando ε-NFAs

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



Implementando ε-NFAs

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

Equivalência ε-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 ε-NFA
  - Prova por construção dos dois lados:
    - “Se”: um processo que constrói um DFA a partir de um ε-NFA
    - “Somente se”: um processo que constrói um ε-NFA a partir de um DFA

Conversão ε-NFA → DFA

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

Conversão ε-NFA → DFA

- Passo a passo com exemplo
- Dado o NFA:

	ε	**a**	**b**	**c**
→ **p**	∅	{p}	{q}	{r}
q	{p}	{q}	{r}	∅
* **r**	{q}	{r}	∅	{p}

Conversão ε-NFA → DFA

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

	ε	**a**	**b**	**c**
→ p	∅	{p}	{q}	{r}
q	{p}	{q}	{r}	∅
*** r**	{q}	{r}	∅	{p}

Conversão ε-NFA → DFA

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

	a	b	c

Conversão ε-NFA → DFA

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

	a	**b**	**c**
→ **{p}**			

Conversão ε-NFA → 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**
→ **{p}**	{p}	{p,q}	{p,q,r}

Conversão ε-NFA → 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**
→ {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 ε-NFA → 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**
→ **{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 ε-NFA → DFA

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

	a	**b**	**c**
→ **A**	A	B	C
B	B	C	C
* **C**	C	C	C

Conversão ε-NFA → DFA

- Dado o seguinte ε-NFA
  - Converta para um DFA que aceita a mesma linguagem

	ε	**a**	**b**	**c**
→ p	{q,r}	∅	{q}	{r}
q	∅	{p}	{r}	{p,q}
*** r**	∅	∅	∅	∅

Conversão ε-NFA → DFA

- Resposta

	a	b	c
→ * {p,q,r}	{p,q,r}	{q,r}	{p,q,r}
*** {q,r}**	{p,q,r}	{r}	{p,q,r}
*** {r}**	{}	{}	{}
{}	{}	{}	{}

Conversão ε-NFA → DFA

- Dado o seguinte ε-NFA
  - Converta para um DFA que aceita a mesma linguagem

	ε	**a**	**b**
→ * 1	{3}	∅	{2}
2	∅	{2,3}	{3}
3	∅	{1}	∅

Conversão ε-NFA → DFA

- Resposta

	a	b
→ * {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}
{}	{}	{}

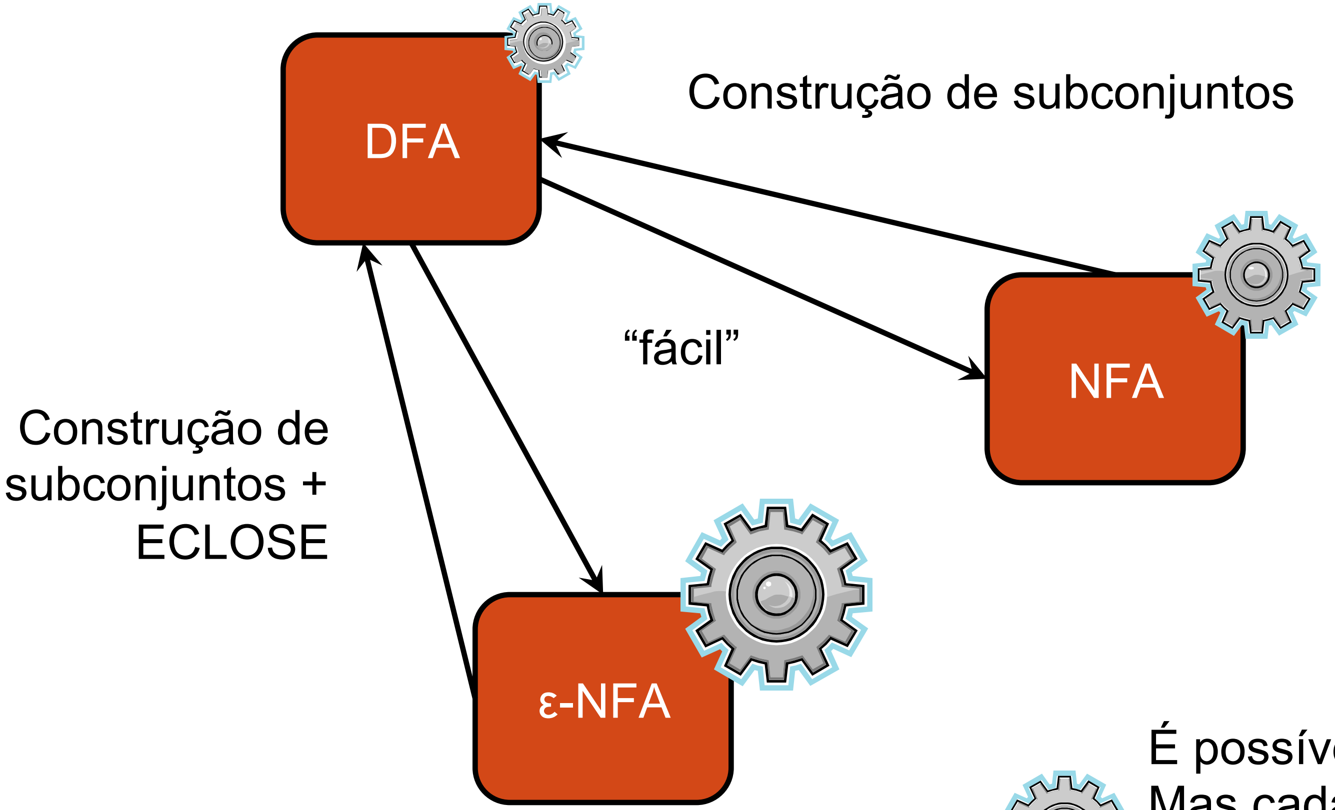
Conversão DFA → ε-NFA

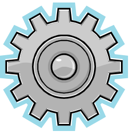
- “Resto” da prova
- Parte fácil
  - Mesmo caso da conversão de DFA para NFA
  - Mas fazendo com que $\delta(q, \varepsilon) = \varnothing$ para todo estado q do DFA
  - Ou seja, não existem transições espontâneas (mas poderia ter)

Resumo

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

Resumo





É possível simular
Mas cada um tem um custo (tamanho da engrenagem)

Fim