ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 5

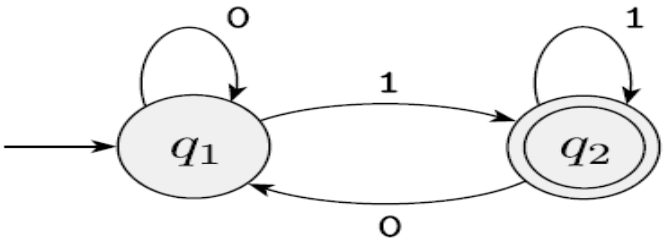
Equivalência entre AFNs e AFDs

Profa. Ariane Machado Lima
ariane.machado@usp.br

Aula passada

Aulas passadas

AFD

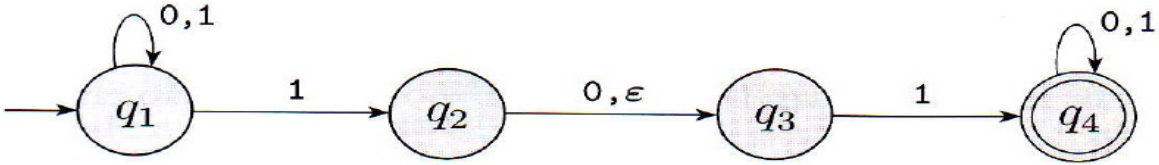


Para cada par (estado atual, próximo símbolo) está DETERMINADO qual é o próximo estado

Um ***autômato finito*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito conhecido como os ***estados***,
2. Σ é um conjunto finito chamado o ***alfabeto***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ é a ***função de transição***,[1]
4. $q_0 \in Q$ é o ***estado inicial***, e
5. $F \subseteq Q$ é o ***conjunto de estados de aceitação***.[2]

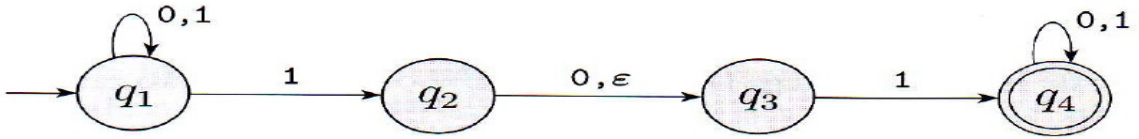
AFN



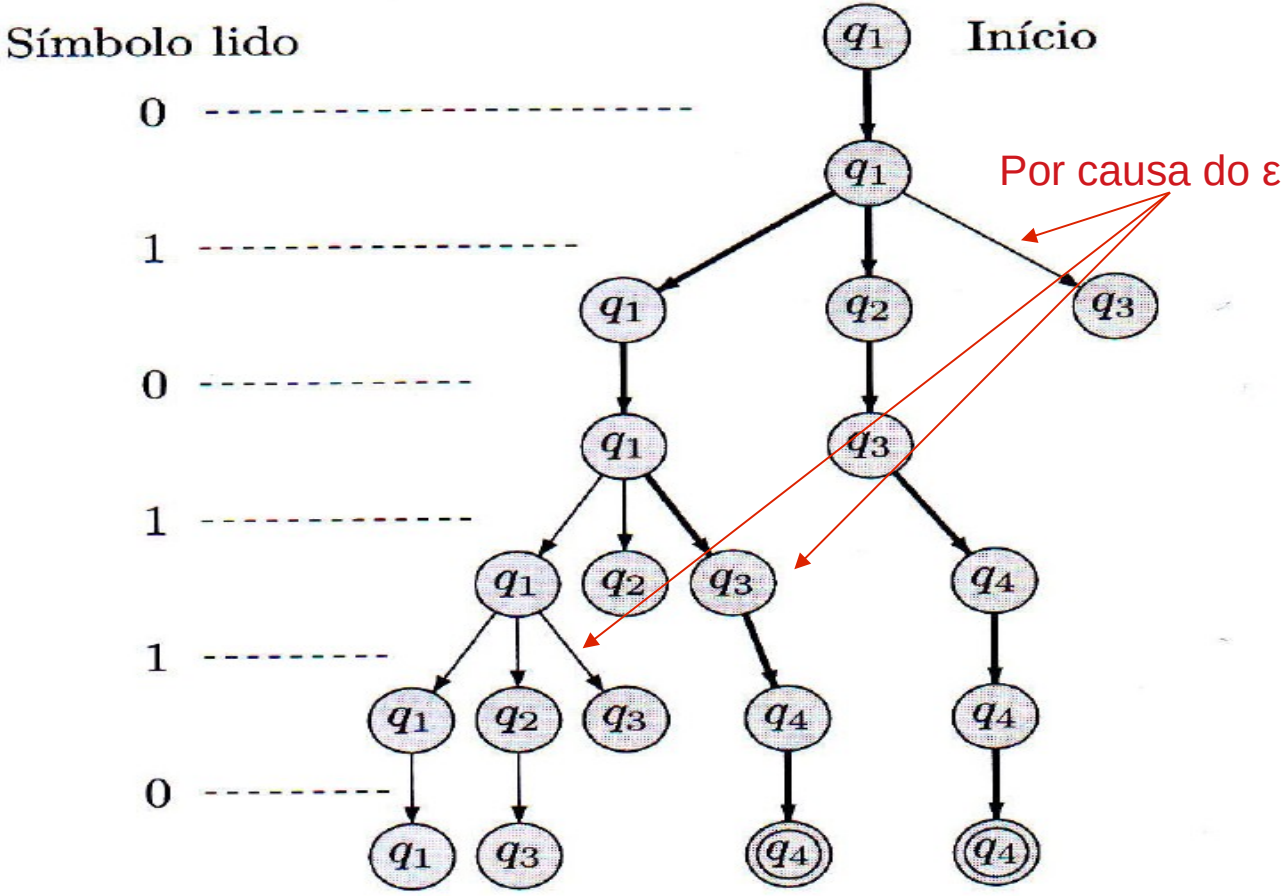
Um ***autômato finito não-determinístico*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito de estados,
2. Σ é um alfabeto finito,
3. $\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ 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.

Para cada par (estado atual, próximo símbolo – incluindo ε) há um conjunto de estados possíveis



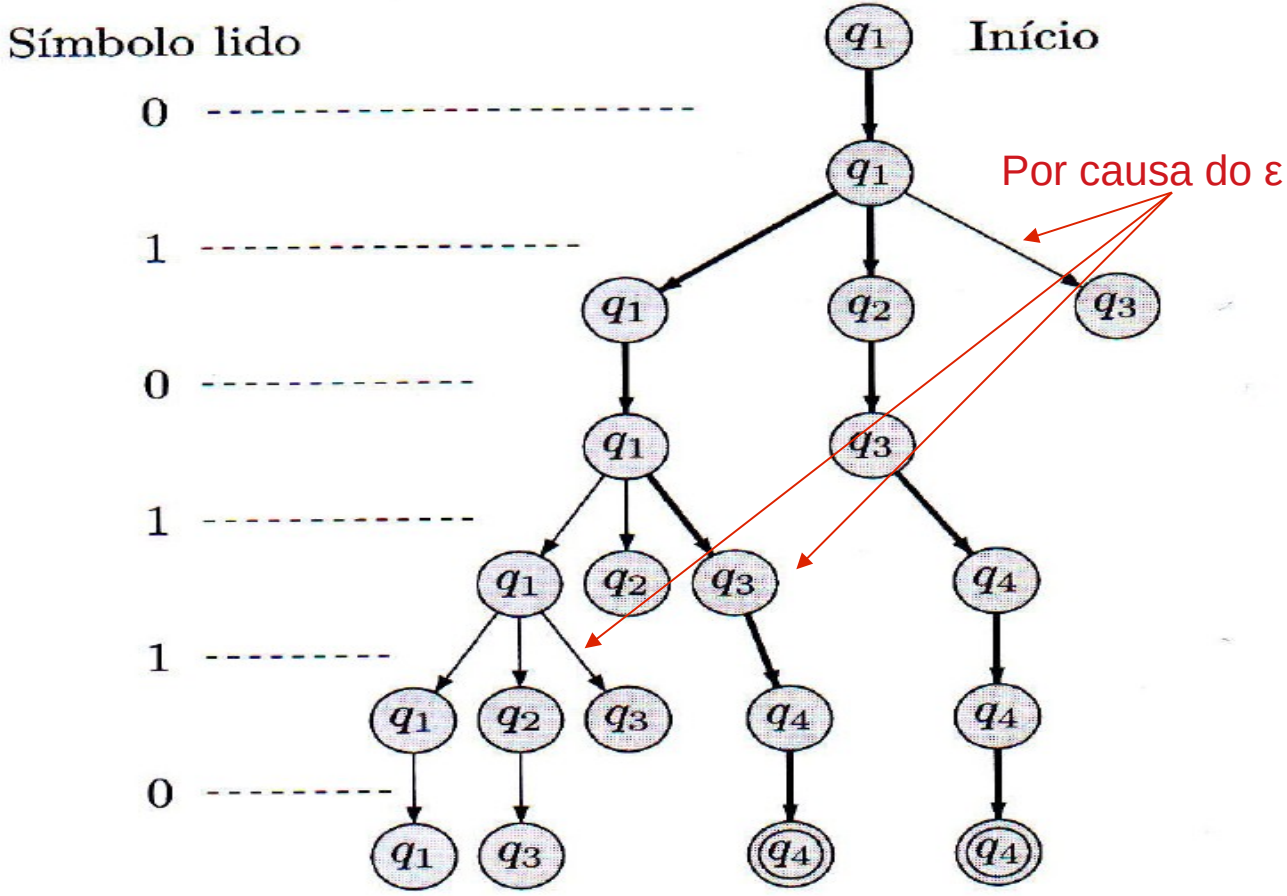
Essa cadeia 010110 é aceita por esse AFN ?



Qual a complexidade (tempo) de análise de uma cadeia por um **AFN**?

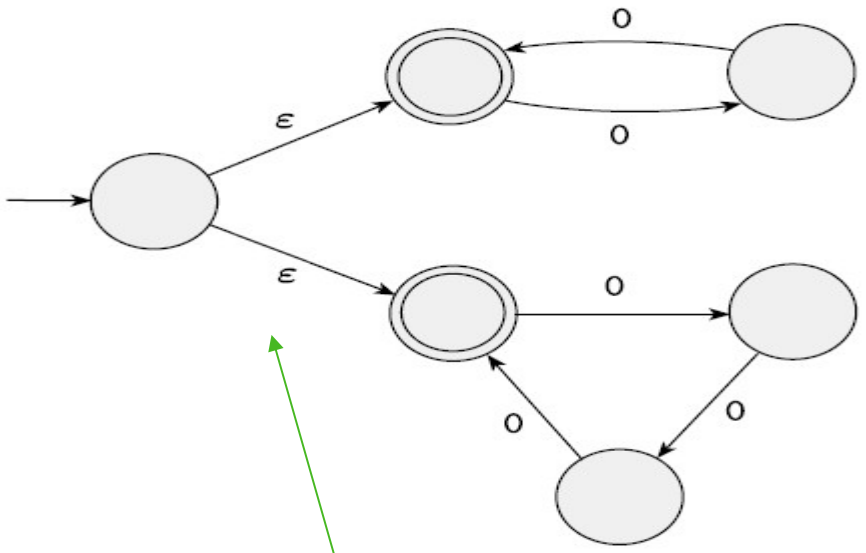
O(n * |Q|)

Cada “nó” dessa árvore poderia ter no máximo |Q| filhos, e poderíamos descartar nós repetidos em um mesmo nível



Exercício

- Desenhe um AFN para a linguagem formada por sequências formadas apenas por zeros, mas que contenham o nr de zeros sendo um múltiplo de 2 **ou** múltiplo de 3



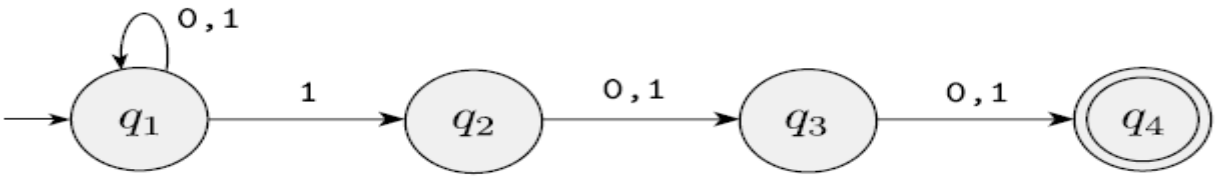
L = { w | w ϵ 0* e |w| = 2*i, i = 0, 1, …} U { w | w ϵ 0* e |w| = 3*i, i = 0, 1, …}

Exercícios

1) Desenhe um AFN para a linguagem formada por sequências binárias que contenham 1 na antepenúltima posição

Exercícios

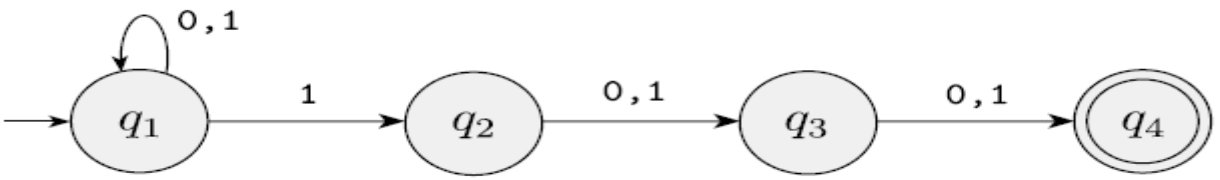
1) Desenhe um AFN para a linguagem formada por sequências binárias que contenham 1 na antepenúltima posição



Onde está o não determinismo?

Exercícios

1) Desenhe um AFN para a linguagem formada por sequências binárias que contenham 1 na antepenúltima posição



Onde está o não determinismo?

q1 tem duas opções de próximo estado se ler o símbolo “1”

* : 0 ou mais vezes
+ : 1 ou mais vezes
Ex: $1^*(001^+)^*$

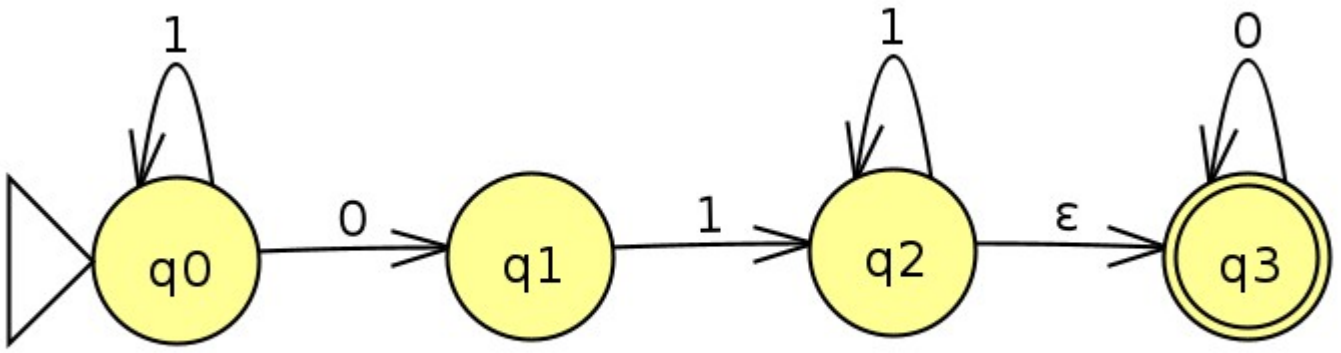
Exercícios

2) Desenhe o diagrama de estados de um AFN que reconheça a linguagem $1^*01^+0^*$

* : 0 ou mais vezes
+ : 1 ou mais vezes
Ex: $1^*(001^+)^*$

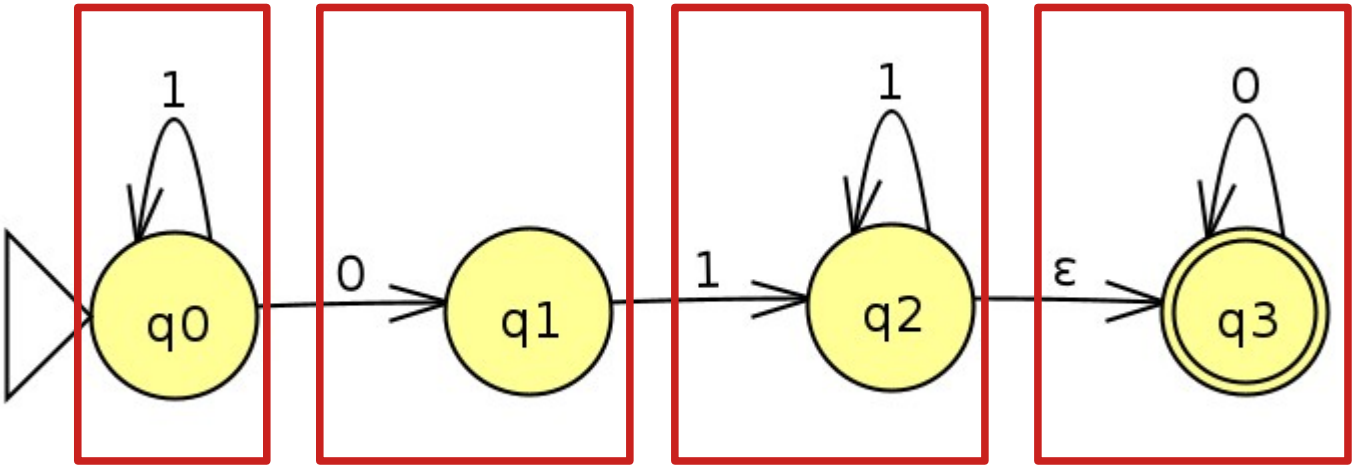
Exercícios

2) Desenhe o diagrama de estados de um AFN que reconheça a linguagem $1^*01^+0^*$



Exercícios

2) Desenhe o diagrama de estados de um AFN que reconheça a linguagem $1^*01^+0^*$



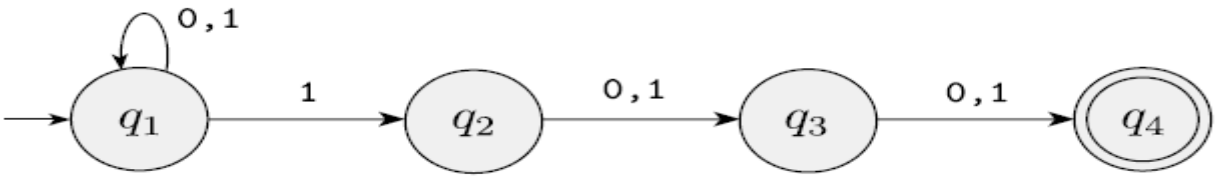
1* (no estado inicial) | 0 | 1^+ | 0*

Aula de Hoje

- Equivalência entre AFDs e AFNs
- Fechamentos de linguagens regulares

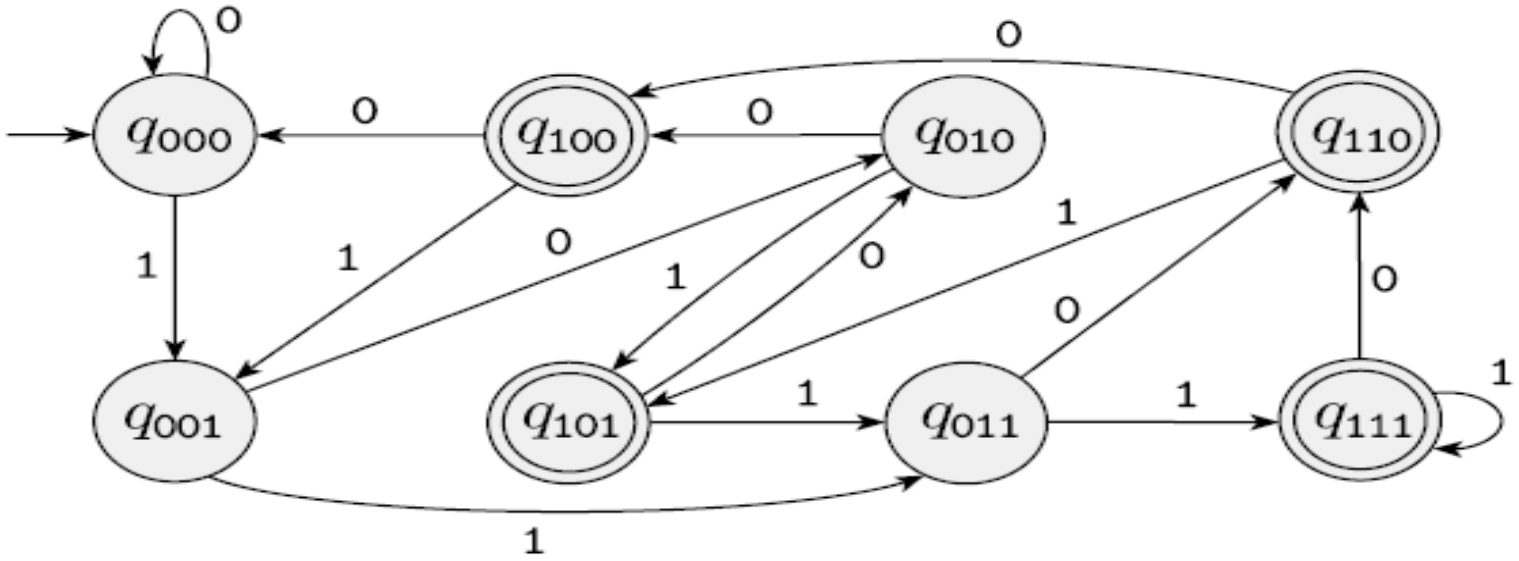
Exercício da aula passada

- Desenhe um AFN para a linguagem formada por sequências binárias que contenham 1 na antepenúltima posição



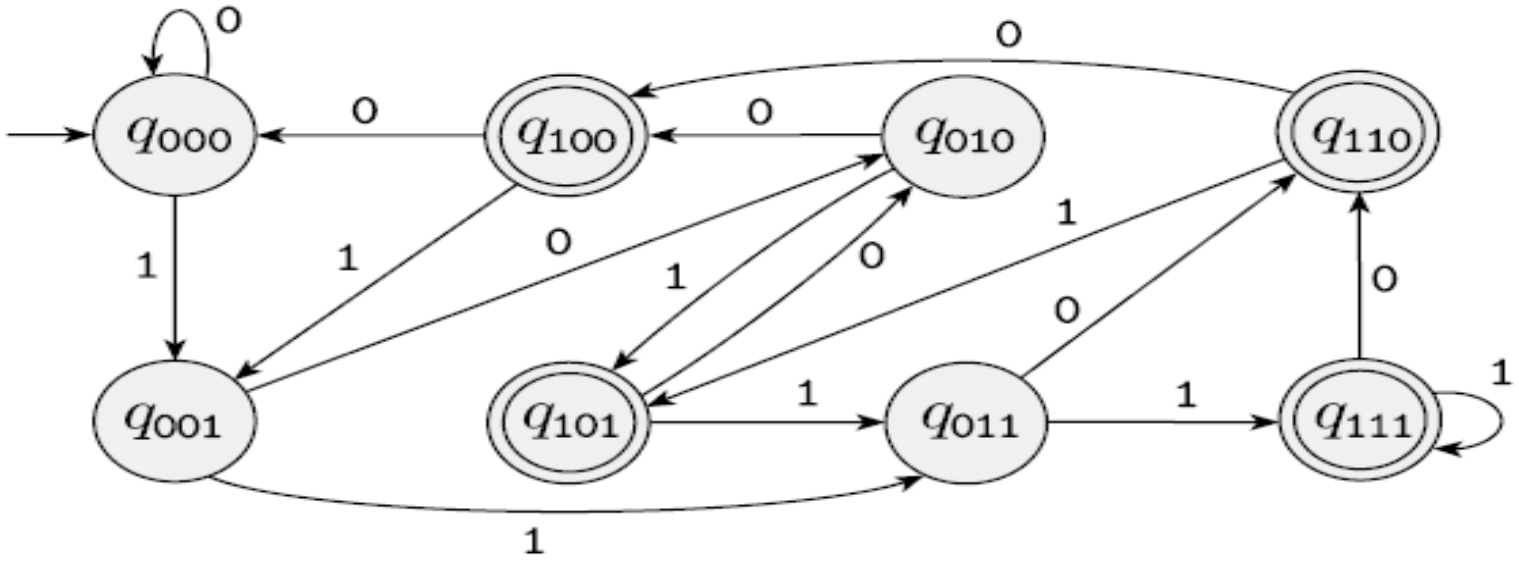
Daria para escrever um AFD para essa linguagem? (sequências binárias que contenham 1 na antepenúltima posição)

Daria para escrever um AFD para essa linguagem? (sequências binárias que contenham 1 na antepenúltima posição)



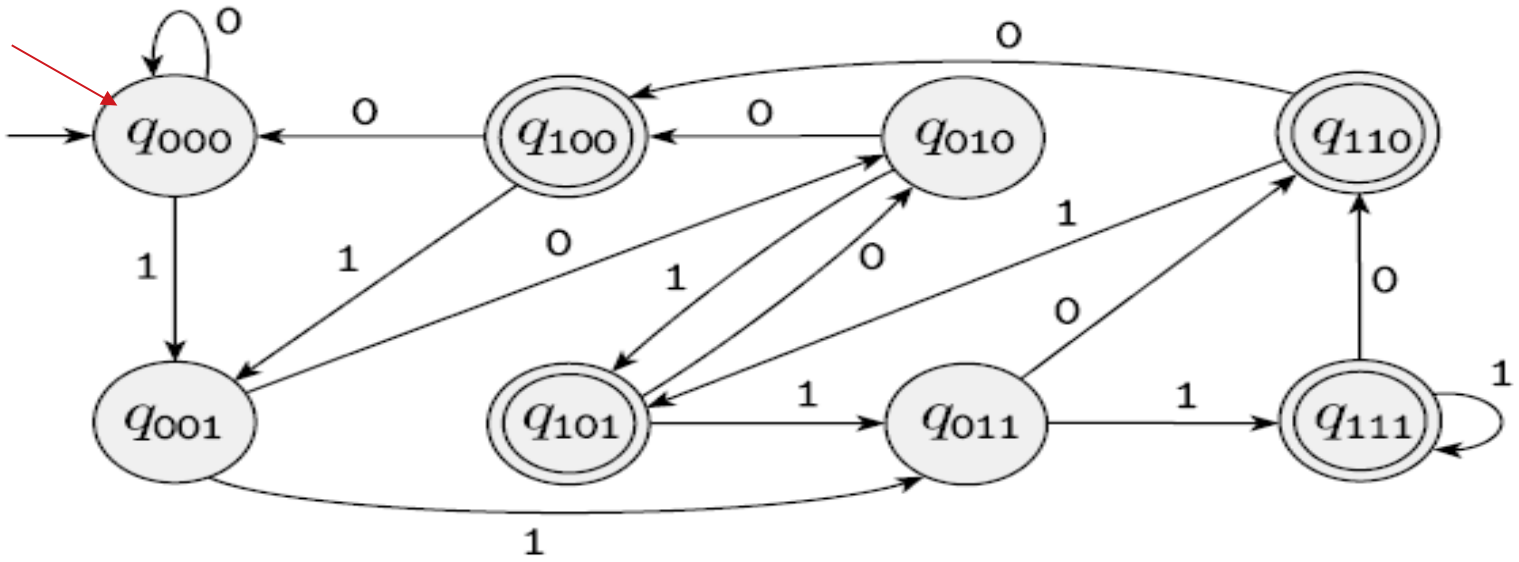
Daria para escrever um AFD para essa linguagem? (sequências binárias que contenham 1 na antepenúltima posição)

Ex: 010111



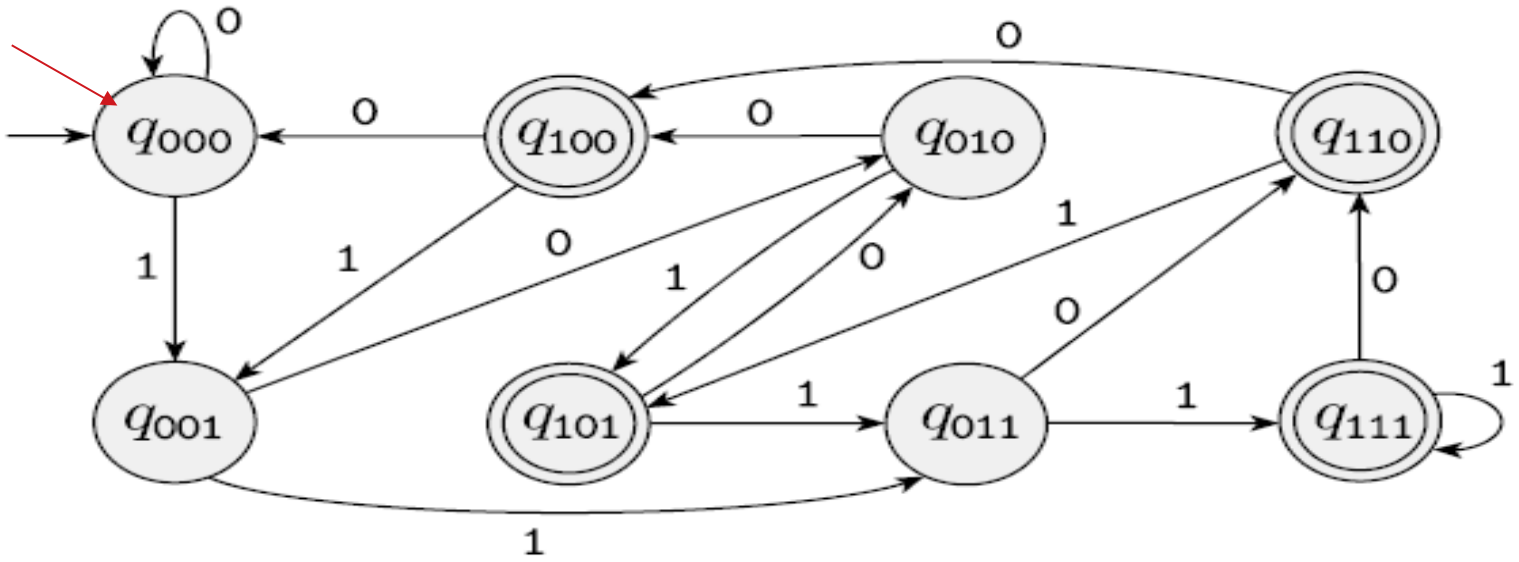
Daria para escrever um AFD para essa linguagem? (sequências binárias que contenham 1 na antepenúltima posição)

Ex: ___010111



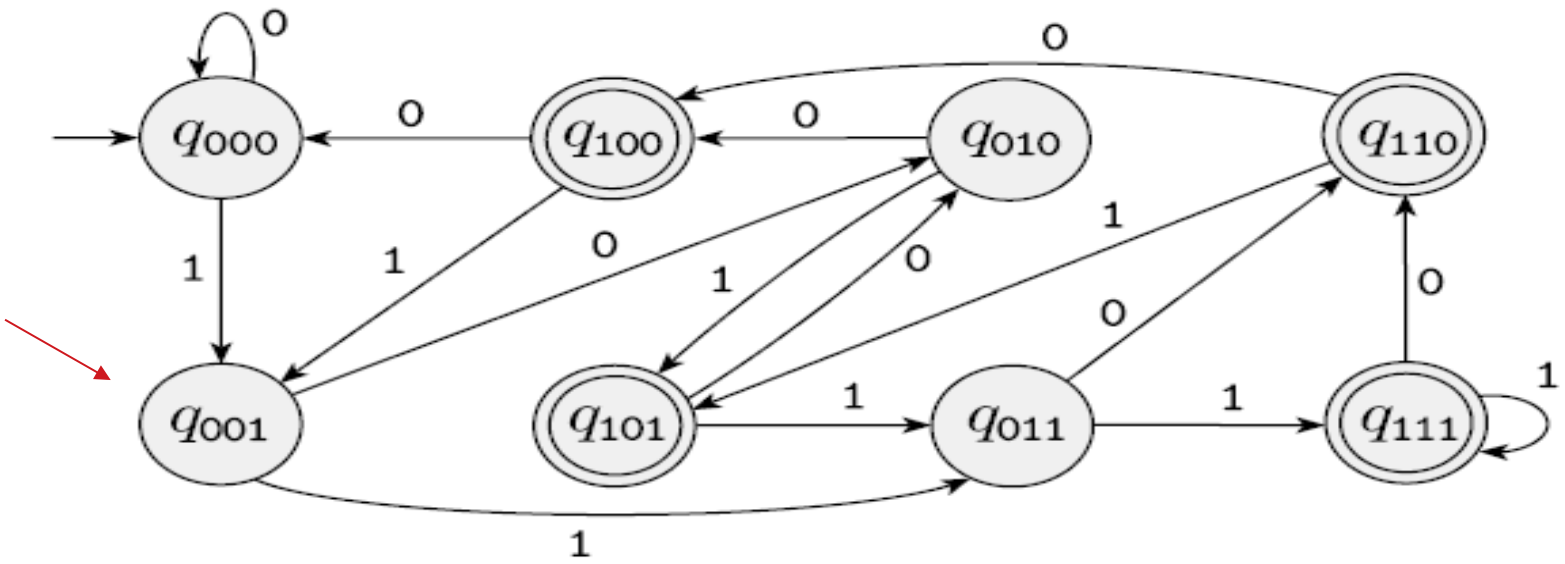
Daria para escrever um AFD para essa linguagem? (sequências binárias que contenham 1 na antepenúltima posição)

Ex: 010111



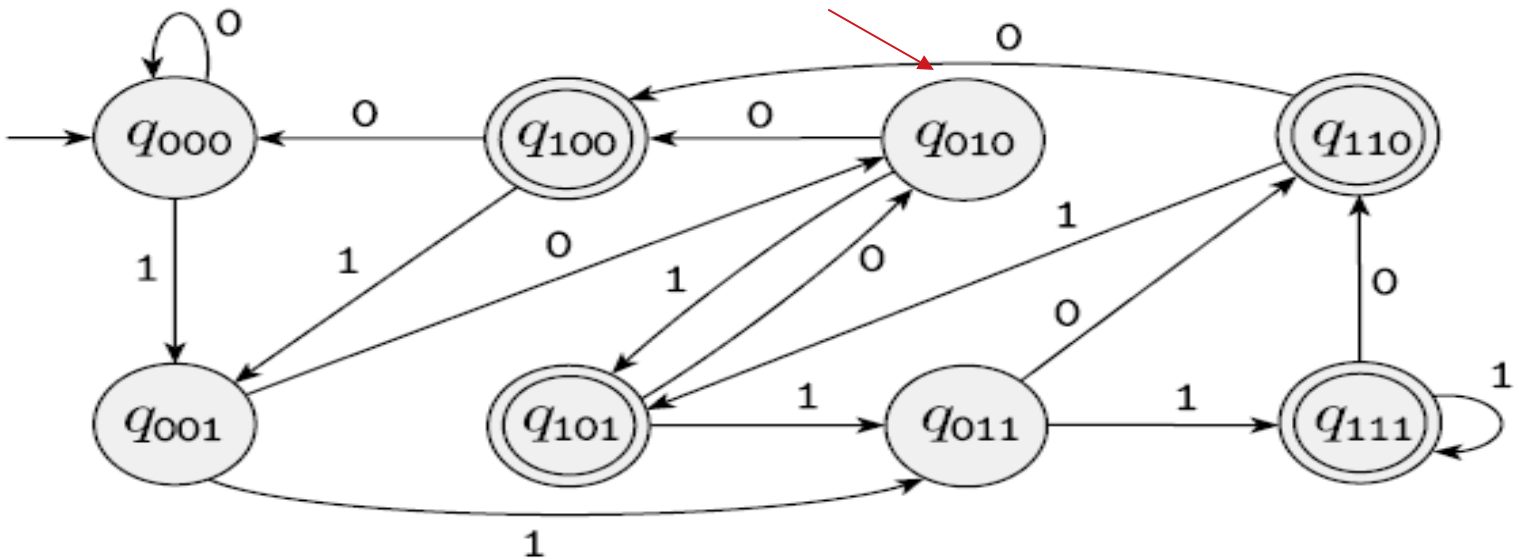
Daria para escrever um AFD para essa linguagem? (sequências binárias que contenham 1 na antepenúltima posição)

Ex: 010111



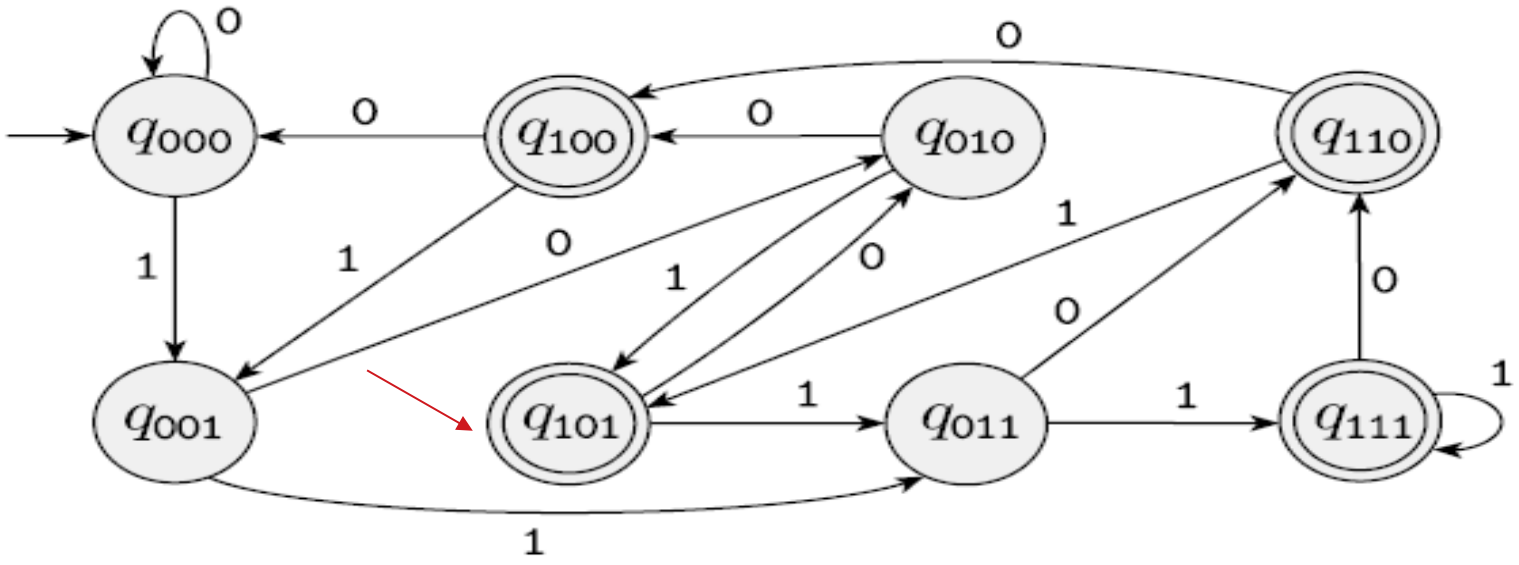
Daria para escrever um AFD para essa linguagem? (sequências binárias que contenham 1 na antepenúltima posição)

Ex: 010111



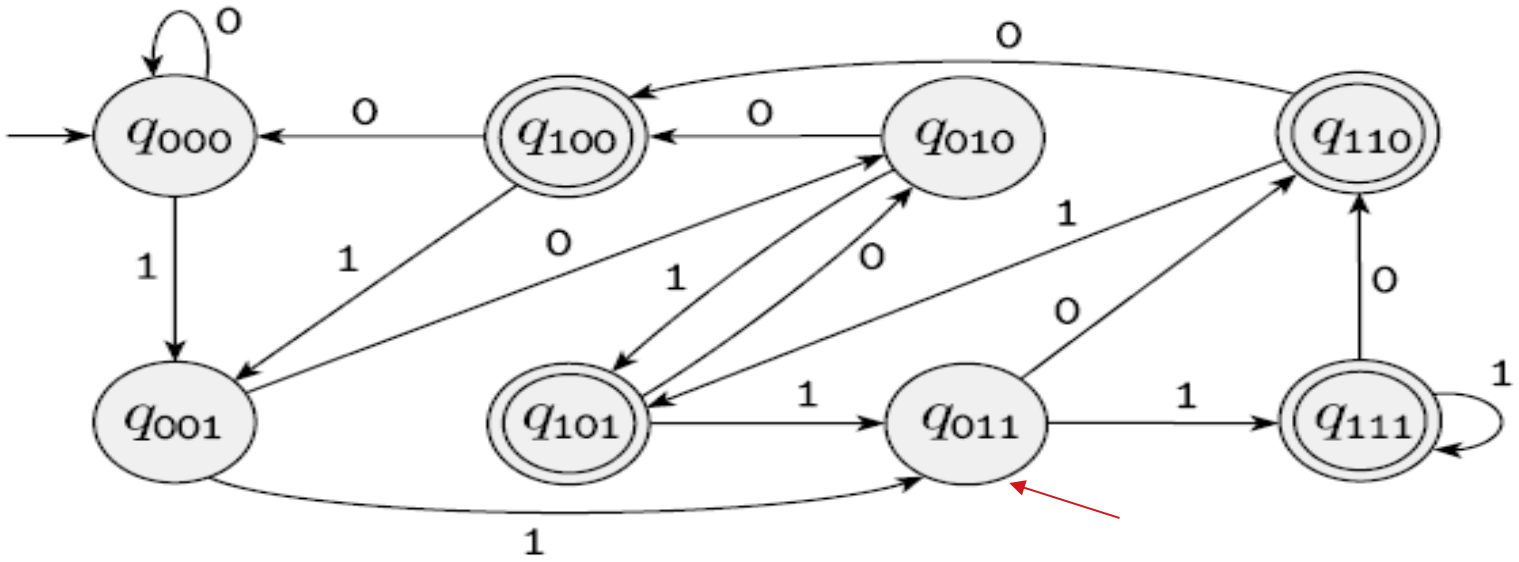
Daria para escrever um AFD para essa linguagem? (sequências binárias que contenham 1 na antepenúltima posição)

Ex: 010111



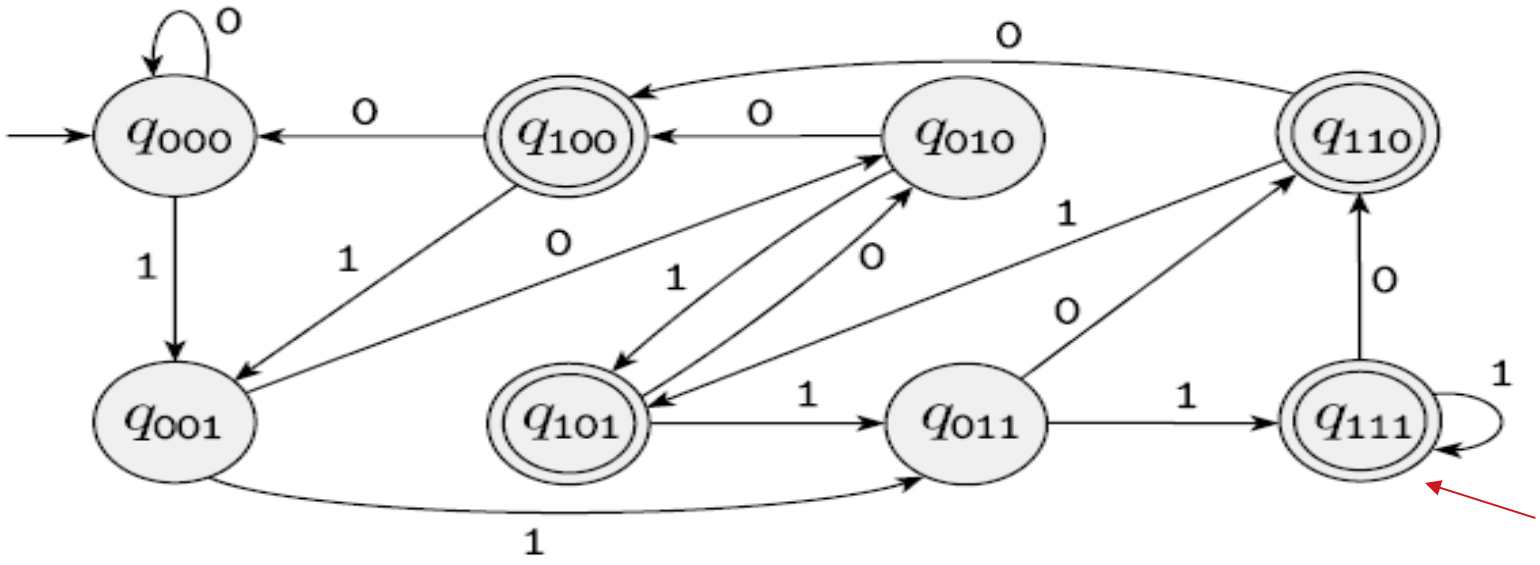
Daria para escrever um AFD para essa linguagem? (sequências binárias que contenham 1 na antepenúltima posição)

Ex: 010111



Daria para escrever um AFD para essa linguagem? (sequências binárias que contenham 1 na antepenúltima posição)

Ex: 010111



Moral da estória: muitas vezes é bem mais fácil projetar um AFN do que um AFD…

… embora um AFD seja mais eficiente...

Equivalência entre

AFDs e AFNs

AFDs e AFNs

- Quem reconhece mais linguagens?

AFDs e AFNs

- Quem reconhece mais linguagens?

- Os dois reconhecem a mesma classe de linguagens

Equivalência entre AFDs e AFNs

- Duas máquinas são equivalentes se elas reconhecem a mesma linguagem

TEOREMA 1.39

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

AFDs e AFNs

- Por que o teorema de equivalência é importante?

AFDs e AFNs

- Por que o teorema de equivalência é importante?
- Pode-se optar por um ou outro dependendo do objetivo
- AFDs são mais eficientes
- AFNs podem:
  - ser mais fáceis de serem projetados
  - facilitar demonstração de teoremas
  - ser úteis em versões probabilísticas

PROVA do Teorema da Equivalência

A prova é por construção

(aliás, estudem o cap 0 do livro do Sipser...)

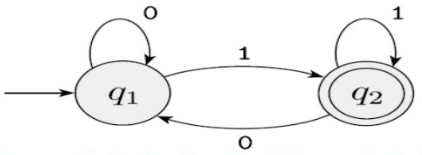
AFD <=> AFN

(tenho que prova a ida e a volta)

AFD => AFN

Como transformo um AFD em um AFN?

AFD

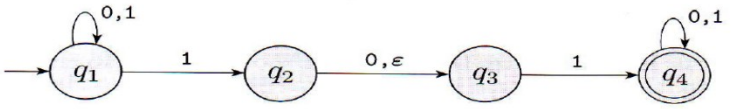


Para cada par (estado atual, próximo símbolo) está DETERMINADO qual é o próximo estado

Um ***autômato finito*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito conhecido como os ***estados***,
2. Σ é um conjunto finito chamado o ***alfabeto***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ é a ***função de transição***,[1]
4. $q_0 \in Q$ é o ***estado inicial***, e
5. $F \subseteq Q$ é o ***conjunto de estados de aceitação***.[2]

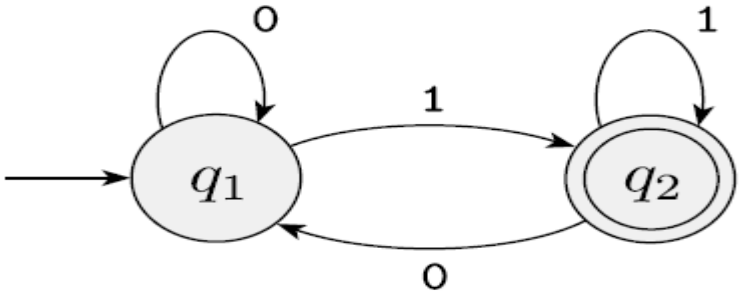
AFN



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

1. Q é um conjunto finito de estados,
2. Σ é um alfabeto finito,
3. $\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ 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.

Para cada par (estado atual, próximo símbolo – incluindo ε) há um conjunto de estados possíveis

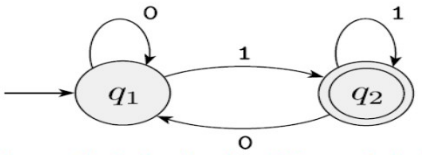


		0	1
→	**q1**	q1	q2
←	**q2**	q1	q2

AFD => AFN

Como transformo um AFD em um AFN?

AFD

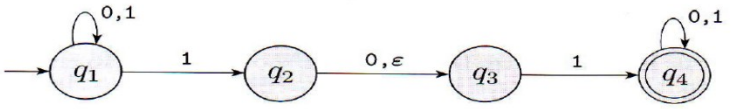


Para cada par (estado atual, próximo símbolo) está DETERMINADO qual é o próximo estado

Um ***autômato finito*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito conhecido como os ***estados***,
2. Σ é um conjunto finito chamado o ***alfabeto***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ é a ***função de transição***,[1]
4. $q_0 \in Q$ é o ***estado inicial***, e
5. $F \subseteq Q$ é o ***conjunto de estados de aceitação***.[2]

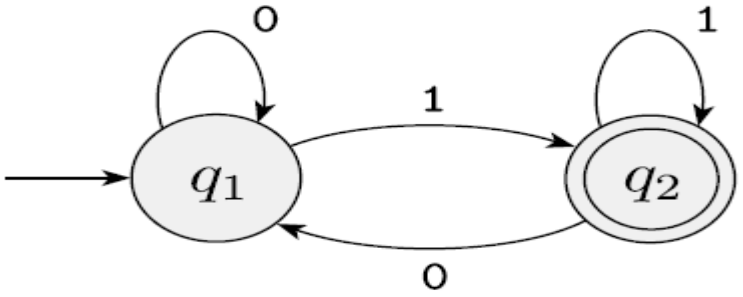
AFN



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

1. Q é um conjunto finito de estados,
2. Σ é um alfabeto finito,
3. $\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ 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.

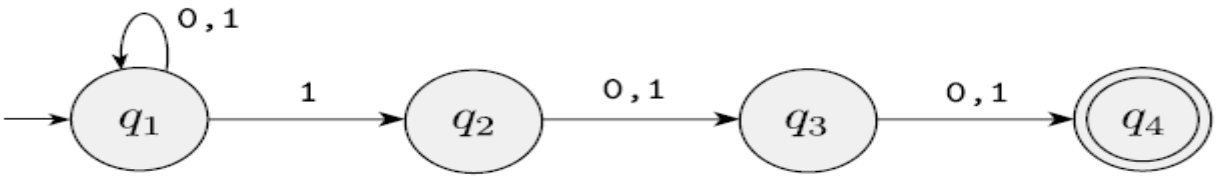
Para cada par (estado atual, próximo símbolo – incluindo ε) há um conjunto de estados possíveis



		0	1	ε
→	**q1**	{q1}	{q2}	Ø
←	**q2**	{q1}	{q2}	Ø

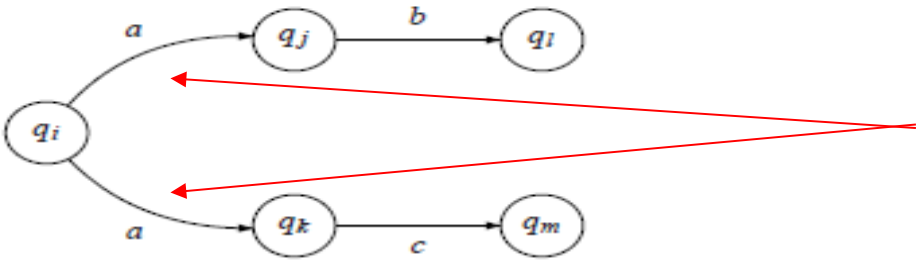
AFD <= AFN

Como transformo um AFN em um AFD?



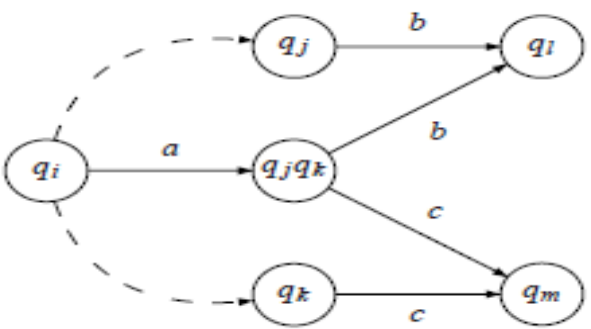
Equivalência entre AFDs e AFNs

Primeiro um racional...



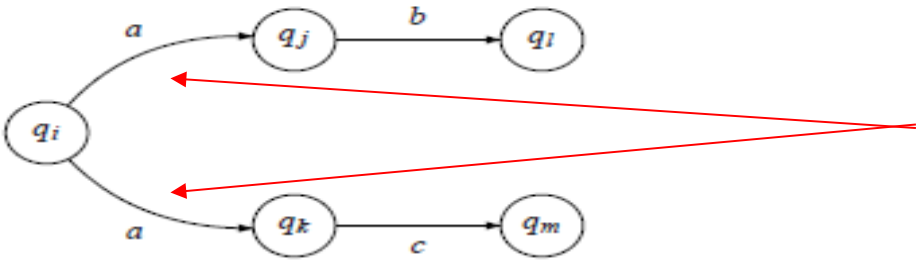
Eliminando este determinismo...

Situação não-determinística original



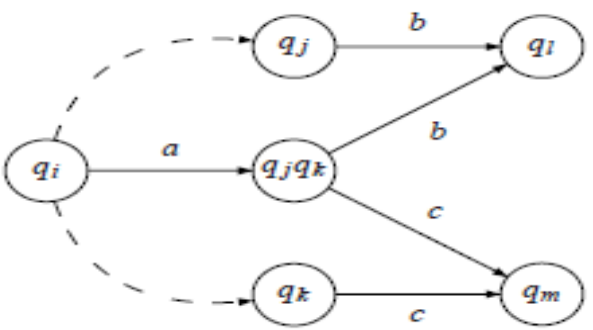
Situação determinística equivalente

Equivalência entre AFDs e AFNs
Primeiro um racional...



Eliminando este determinismo...

Situação não-determinística original

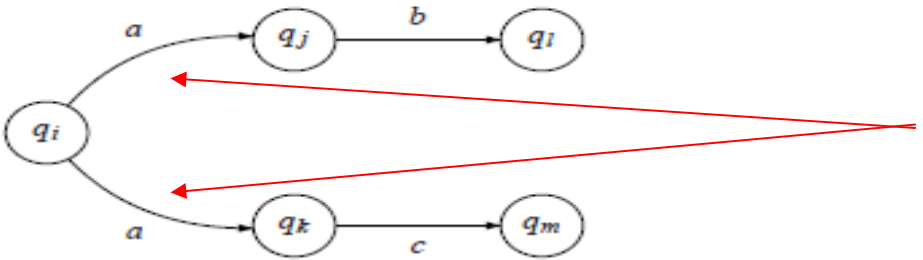


E se esse “b” e “c” fossem o mesmo símbolo?

Situação determinística equivalente

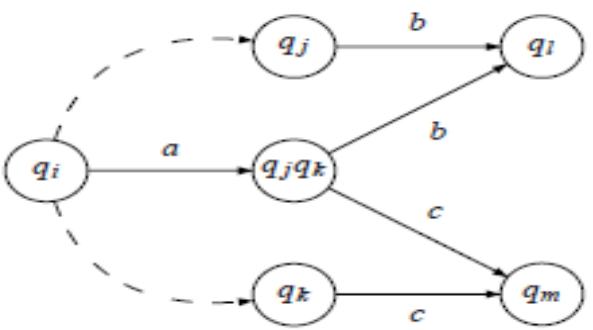
Equivalência entre AFDs e AFNs

Primeiro um racional...



Situação não-determinística original

Eliminando este determinismo...



Situação determinística equivalente

E se esse “b” e “c” fossem o mesmo símbolo?

Deveríamos repetir o procedimento...

Equivalência entre AFDs e AFNs

- Ideia da Prova: um novo estado (no AFD) para cada subconjunto de estados originais (do AFN)

Não queremos construir um AFD elegante nem vamos nos preocupar com eficiência → **não vamos economizar estados** …

Queremos apenas provar que, no caso geral, existe tal AFD (e para isso vamos propor um algoritmo de transformação AFN → AFD)

Equivalência entre AFDs e AFNs

- Ideia da Prova: um novo estado (no AFD) para cada subconjunto de estados originais (do AFN)

- Primeiro vamos desconsiderar setas ε
  - Ou seja, vamos considerar AFNs sem transições no vazio

PROVA Seja $N = (Q, \Sigma, \delta, q_0, F)$ o AFN que reconhece alguma linguagem A. Construímos um AFD $M = (Q', \Sigma, \delta', q_0', F')$ que reconhece A. Antes de realizar a construção completa, vamos primeiro considerar o caso mais fácil no qual N não tem setas ε. Mais adiante levamos as setas ε em consideração.

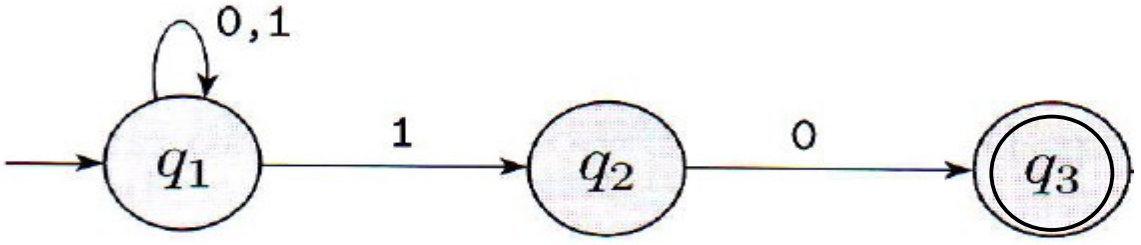
PROVA Seja $N = (Q, \Sigma, \delta, q_0, F)$ o AFN que reconhece alguma linguagem A. Construímos um AFD $M = (Q', \Sigma, \delta', q_0', F')$ que reconhece A. Antes de realizar a construção completa, vamos primeiro considerar o caso mais fácil no qual N não tem setas ε. Mais adiante levamos as setas ε em consideração.

1. $Q' = \mathcal{P}(Q)$.
   Todo estado de M é um conjunto de estados de N. Lembre-se de que $\mathcal{P}(Q)$ é o conjunto de subconjuntos de Q.

PROVA Seja $N = (Q, \Sigma, \delta, q_0, F)$ o AFN que reconhece alguma linguagem A. Construímos um AFD $M = (Q', \Sigma, \delta', q_0', F')$ que reconhece A. Antes de realizar a construção completa, vamos primeiro considerar o caso mais fácil no qual N não tem setas ε. Mais adiante levamos as setas ε em consideração.

1. $Q' = \mathcal{P}(Q)$.
Todo estado de M é um conjunto de estados de N. Lembre-se de que $\mathcal{P}(Q)$ é o conjunto de subconjuntos de Q.

Ex:



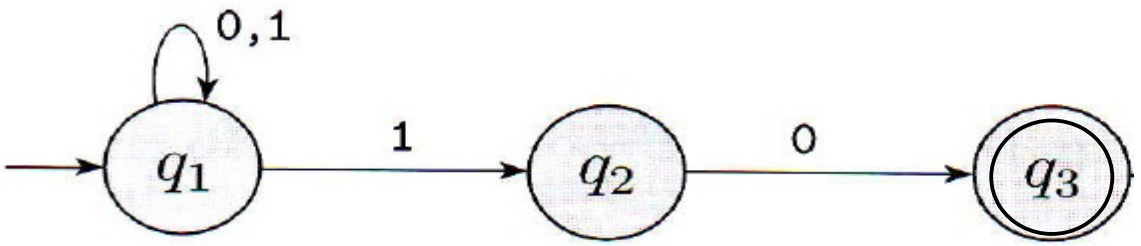
PROVA Seja $N = (Q, \Sigma, \delta, q_0, F)$ o AFN que reconhece alguma linguagem A. Construímos um AFD $M = (Q', \Sigma, \delta', q_0', F')$ que reconhece A. Antes de realizar a construção completa, vamos primeiro considerar o caso mais fácil no qual N não tem setas ε. Mais adiante levamos as setas ε em consideração.

1. $Q' = \mathcal{P}(Q)$.
   Todo estado de M é um conjunto de estados de N. Lembre-se de que $\mathcal{P}(Q)$ é o conjunto de subconjuntos de Q.

2. Para $R \in Q'$ e $a \in \Sigma$ seja $\delta'(R, a) = \{q \in Q | q \in \delta(r, a)$ para algum $r \in R\}$.
   Se R é um estado de M, é também um conjunto de estados de N. Quando M lê um símbolo a no estado R, ele mostra para onde a leva cada estado em R. Dado que cada estado pode ir para um conjunto de estados, tomamos a união de todos esses conjuntos. Outra maneira de escrever essa expressão é

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a).$$

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$



Neste ponto estamos ignorando a coluna **ε**

Q \ Σ	0	1
q1	q1	q1q2
q2	q3	vazio
q3	vazio	vazio

PROVA Seja $N = (Q, \Sigma, \delta, q_0, F)$ o AFN que reconhece alguma linguagem A. Construímos um AFD $M = (Q', \Sigma, \delta', q_0', F')$ que reconhece A. Antes de realizar a construção completa, vamos primeiro considerar o caso mais fácil no qual N não tem setas ε. Mais adiante levamos as setas ε em consideração.

1. $Q' = \mathcal{P}(Q)$.
   Todo estado de M é um conjunto de estados de N. Lembre-se de que $\mathcal{P}(Q)$ é o conjunto de subconjuntos de Q.

2. Para $R \in Q'$ e $a \in \Sigma$ seja $\delta'(R, a) = \{q \in Q | q \in \delta(r, a)$ para algum $r \in R\}$.
   Se R é um estado de M, é também um conjunto de estados de N. Quando M lê um símbolo a no estado R, ele mostra para onde a leva cada estado em R. Dado que cada estado pode ir para um conjunto de estados, tomamos a união de todos esses conjuntos. Outra maneira de escrever essa expressão é

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a).$$

3. $q_0' = \{q_0\}$.
   M começa no estado correspondente à coleção contendo somente o estado inicial de N.

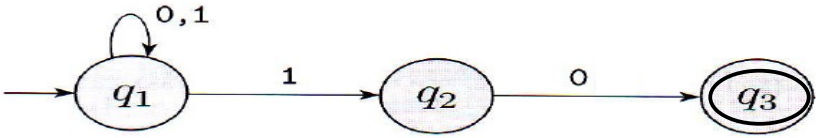
4. $F' = \{R \in Q' \mid R$ contém um estado de aceitação de $N\}$.
A máquina M aceita se um dos possíveis estados nos quais N poderia estar nesse ponto é um estado de aceitação.

Isto quer dizer que um dos ramos de computação do AFN alcançou o estado de aceitação

- Sem considerar transições no vazio:

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$

Como fica δ’(q1, 1) considerando este AFN?

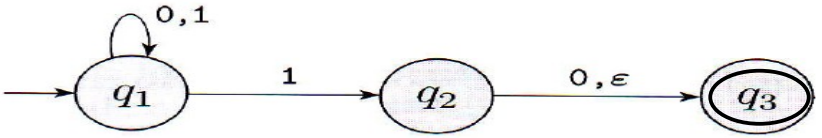


δ’(q1, 1) = q1q2

- Sem considerar transições no vazio:

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$

Como fica δ'(q1, 1) considerando este AFN?



δ'(q1, 1) = q1q2

- Agora considerando setas ε (transições no vazio):

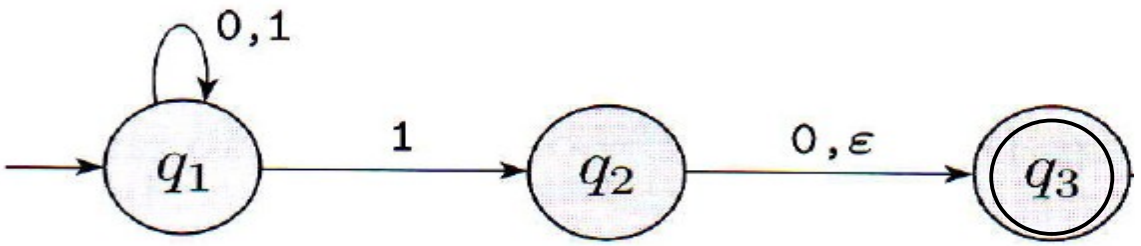
$E(R) = \{q |\ q$ pode ser atingido a partir de R viajando-se ao longo de 0 ou mais setas $\varepsilon\}$.

$$\delta'(R, a) = \{q \in Q |\ q \in E(\delta(r, a)) \text{ para algum } r \in R\}.$$

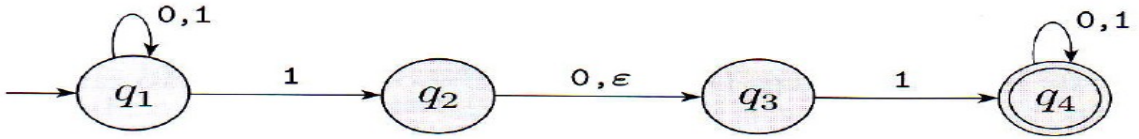
δ'(q1, 1) = q1q2q3

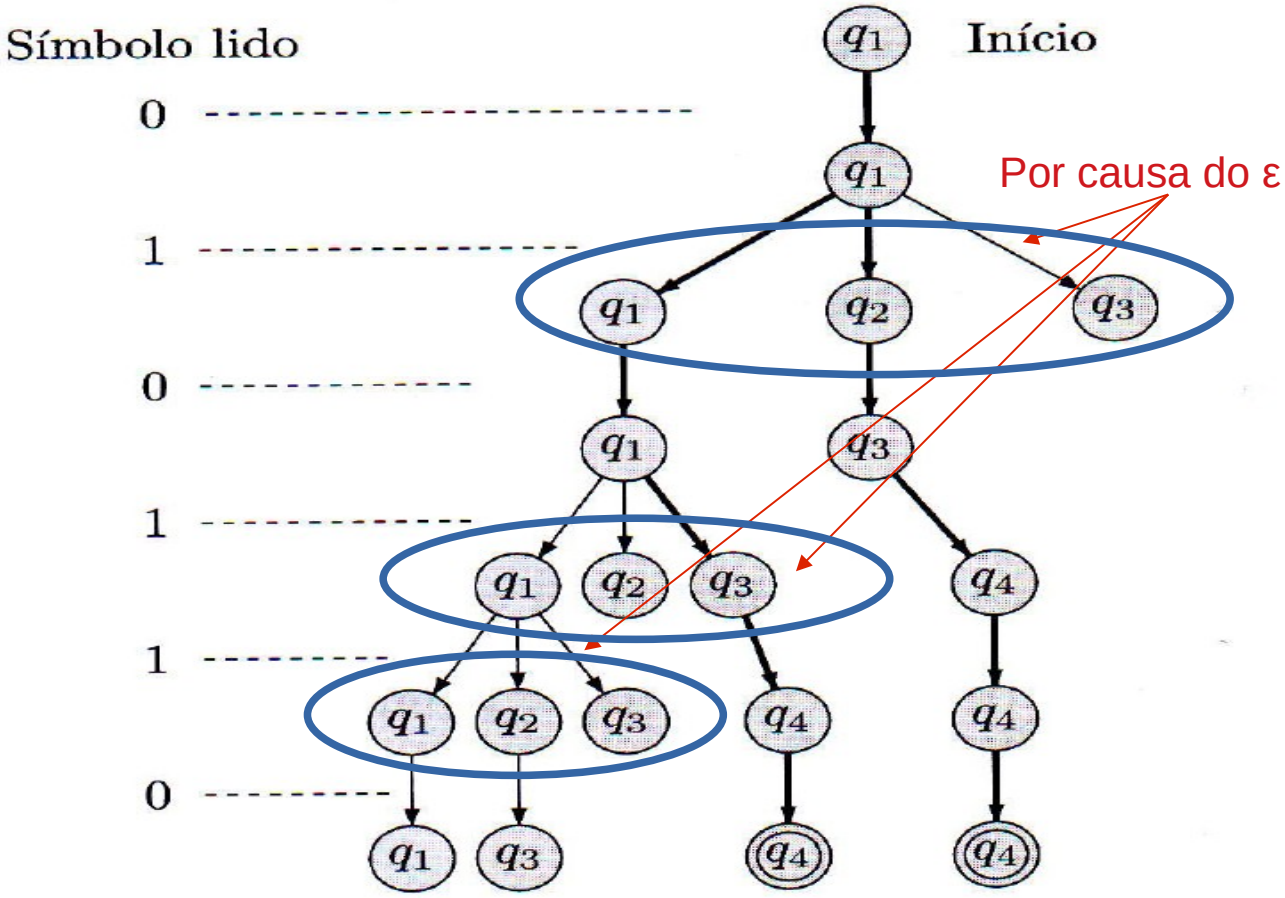
$$q_0' = E(\{q_0\})$$

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$



Q \ Σ	0	1	ε
q1	q1	q1q2	vazio
q2	q3	vazio	q3
q3	vazio	vazio	vazio


0,1
1
0,ε
1
0,1
q1
q2
q3
q4


Símbolo lido
Início
0
1
0
1
1
0
Por causa do ε
q1
q2
q3
q4

Equivalência entre

AFDs e AFNs

(últimos comentários)

Outro exemplo: AFN

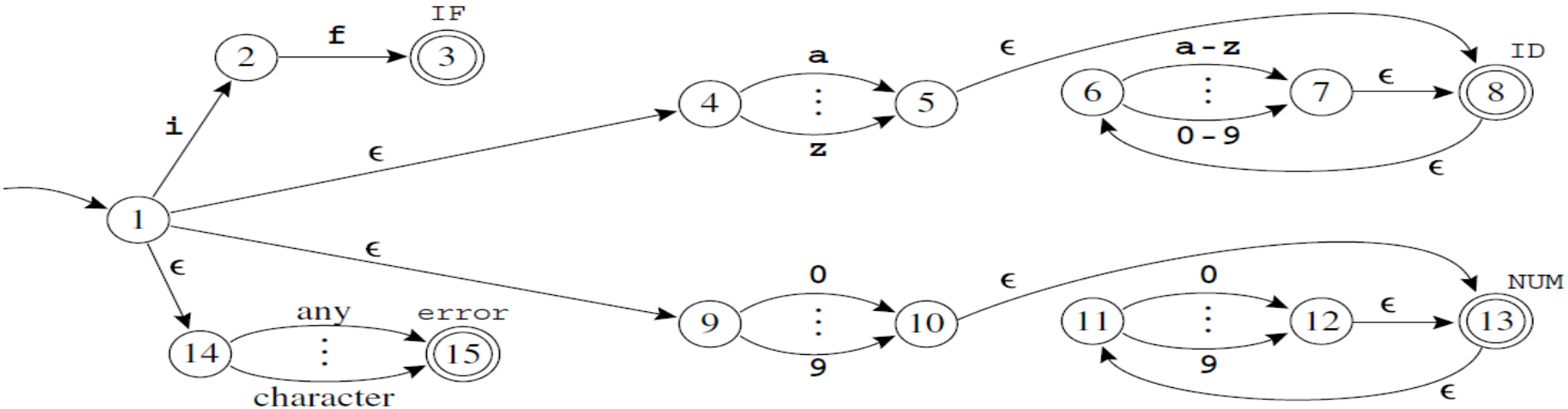


FIGURE 2.7. Four regular expressions translated to an NFA.

Exemplo: AFD equivalente

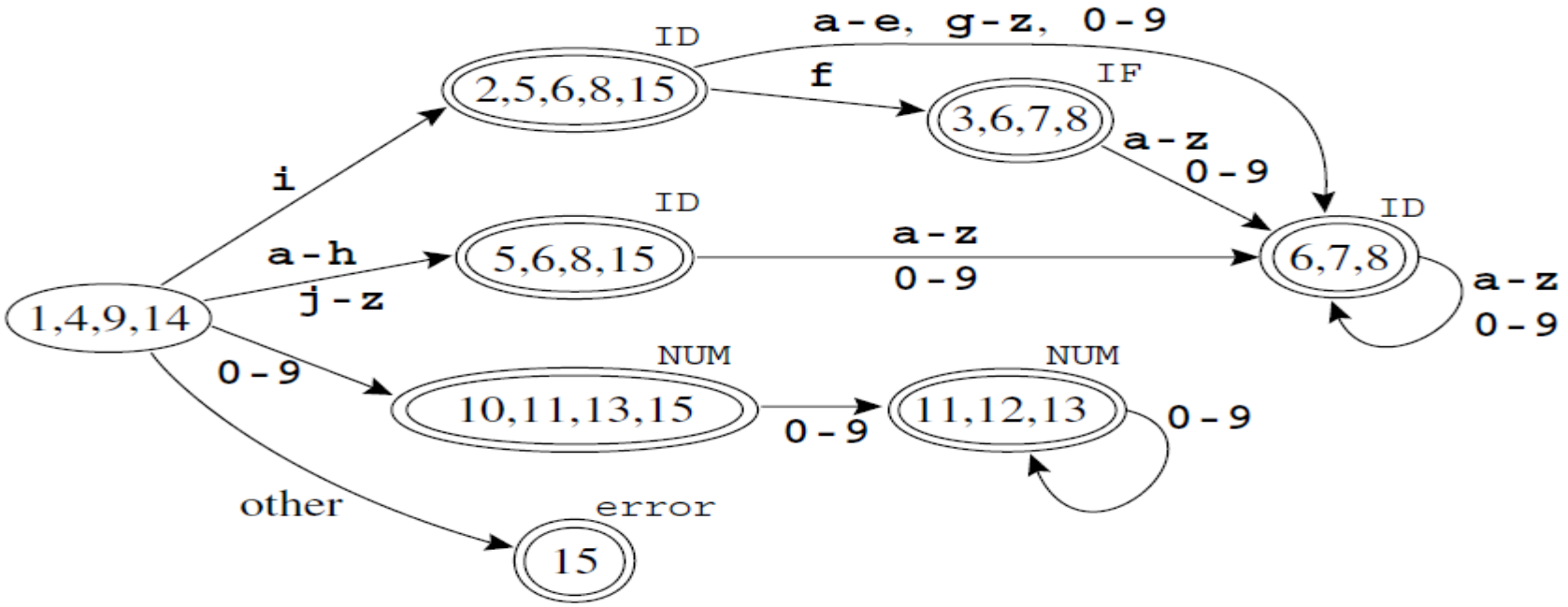


FIGURE 2.8. NFA converted to DFA.

Linguagens Regulares

DEFINIÇÃO 1.16

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

TEOREMA 1.39

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

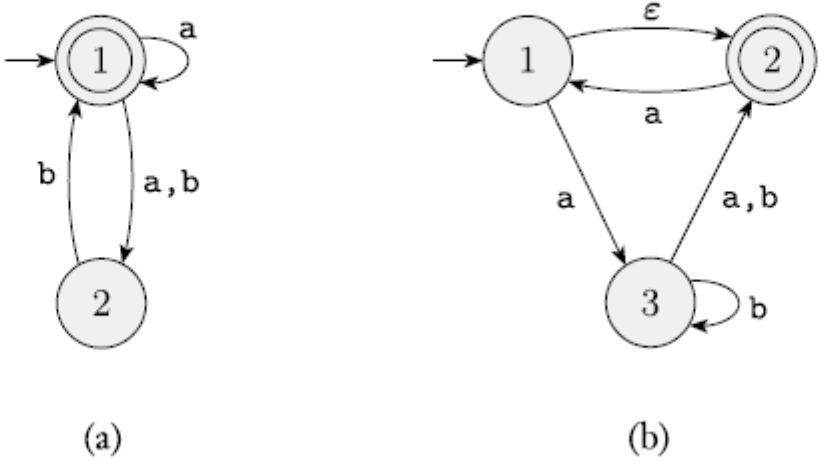
COROLÁRIO 1.40

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

Lista de Exercícios do Sipser (2ª ed)

Exercício 1.16

1.16 Use the construction given in Theorem 1.39 to convert the following two nondeterministic finite automata to equivalent deterministic finite automata.



Usem o JFlap para conferir!!!

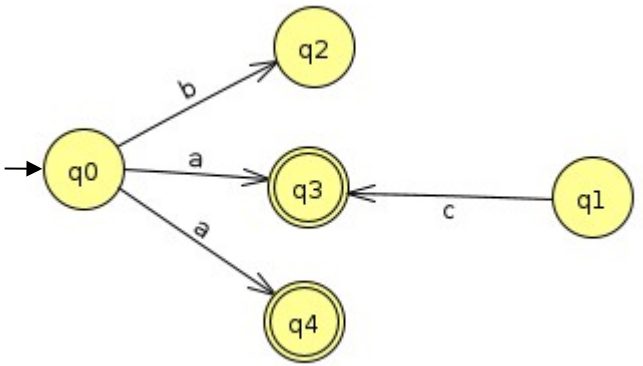
Observações gerais sobre autômatos

Minimização de autômatos finitos

- Vários autômatos podem gerar a mesma linguagem
- Cada linguagem regular é reconhecida por um autômato finito determinístico mínimo (com relação ao número de estados) e único
  - Utilidades:
    - Gerar um reconhecedor o mais compacto e eficiente possível
    - Comparar se duas linguagens são equivalentes

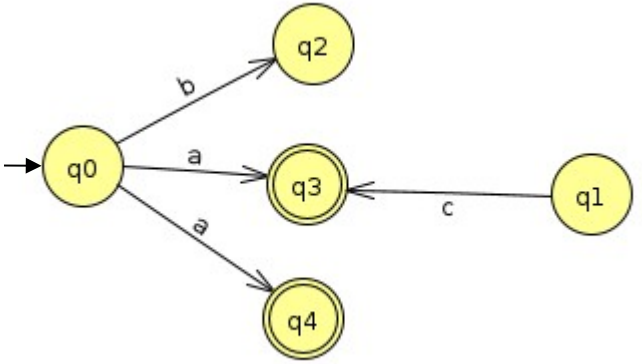
Minimização de autômatos finitos

- Processo:
  - Eliminação de estados inacessíveis
    - Não há caminho de q_0 até ele
  - Eliminação de estados inúteis
    - Não conduzem a um estado final
  - Agrupamento e fusão de estados equivalentes
- Ex:
  - $\delta = \{(q_0, a) \rightarrow q_3, (q_0, a) \rightarrow q_4, (q_0, b) \rightarrow q_2, (q_1, c) \rightarrow q_3\}$
  - $F = \{q_3, q_4\}$
  -



Minimização de autômatos finitos

- Processo:
  - Eliminação de estados inacessíveis
    - Não há caminho de q_0 até ele
  - Eliminação de estados inúteis
    - Não conduzem a um estado final
  - Agrupamento e fusão de estados equivalentes
- Ex:
  - $\delta = \{(q_0, a) \rightarrow q_3, (q_0, a) \rightarrow q_4, (q_0, b) \rightarrow q_2, (q_1, c) \rightarrow q_3\}$
  - $F = \{q_3, q_4\}$
  - q_1 é inacessível, q_2 é inútil, q_3 e q_4 são equivalentes



autômato minimizado

Juntando tudo

- Então você pode, dada uma linguagem:
  - Projetar um AFN que a reconheça (pois é mais fácil de implementar)
  - Transformá-lo em um AFD equivalente (usando o algoritmo mostrado na prova de equivalência) – assim terá eficiência de tempo de execução
  - Minimizar esse AFD, assim terá um modelo mais simples e que ocupa menos espaço