

# Não determinismo (parte 2)

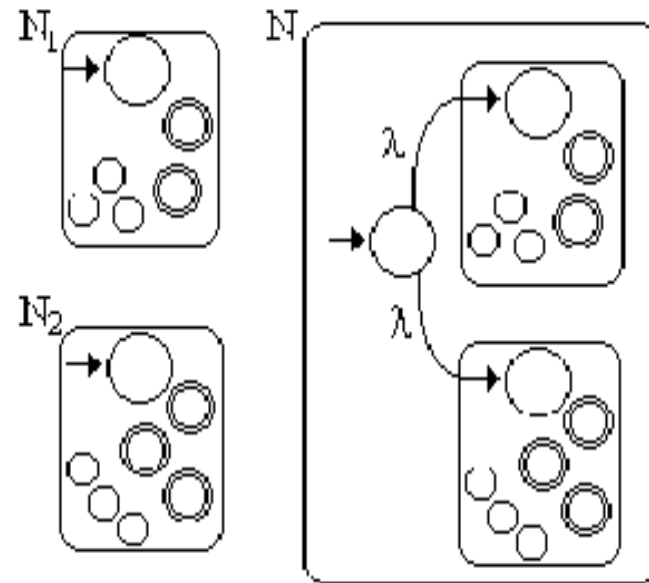
Universidade Federal de Campina Grande – UFCG  
Centro de Engenharia Elétrica e Informática – CEEI  
Departamento de Sistemas e Computação – DSC  
Professor: Andrey Brito                      Período: 2023.2



# Revisitando o teorema sobre a união

- Se  $A_1$  e  $A_2$  são linguagens regulares,  $A_1 \cup A_2$  é regular
- Será que o mesmo vale para AFNDs?
  - $L(N_1)=A_1$  e  $L(N_2)=A_2$
  - Deve existir um AFND  $N$  que reconheça, quando  $N_1$  ou  $N_2$  reconheceria

# Revisitando o teorema sobre a união



# Revisitando o teorema sobre a união

- Seja

- $N_1 = \langle Q_1, \Sigma, \delta_1, q_1, F_1 \rangle$
- $N_2 = \langle Q_2, \Sigma, \delta_2, q_2, F_2 \rangle$

- $N = \langle Q, \Sigma, \delta, q_0, F \rangle$ , onde

- $Q = \{q_0\} \cup Q_1 \cup Q_2$
- $F = F_1 \cup F_2$

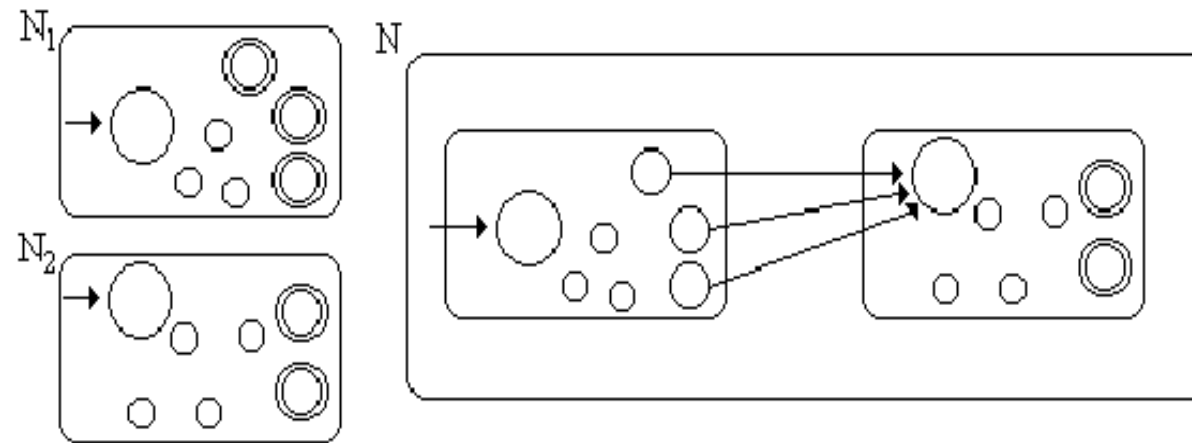
$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ e } a = \lambda \end{cases}$$

E a concatenação?

# Provando o teorema 2

- A classe das linguagens regulares é fechada pela operação de concatenação
  - Se  $A_1$  e  $A_2$  são linguagem regulares, então  $A_1 \cdot A_2$  também é regular...
  - E existem dois AFs,  $M_1$  e  $M_2$ , que reconhecem  $A_1$  e  $A_2$ , respectivamente
- O problema
  - A palavra de entrada é composta de duas partes
  - A primeira tem que pertencer a  $A_1$  e, portanto, reconhecida por  $M_1$
  - A segunda tem que pertencer a  $A_2$  e, portanto, reconhecida por  $M_2$
- A dificuldade: onde quebrar a palavra?

# Revendo o teorema 2





# Observações

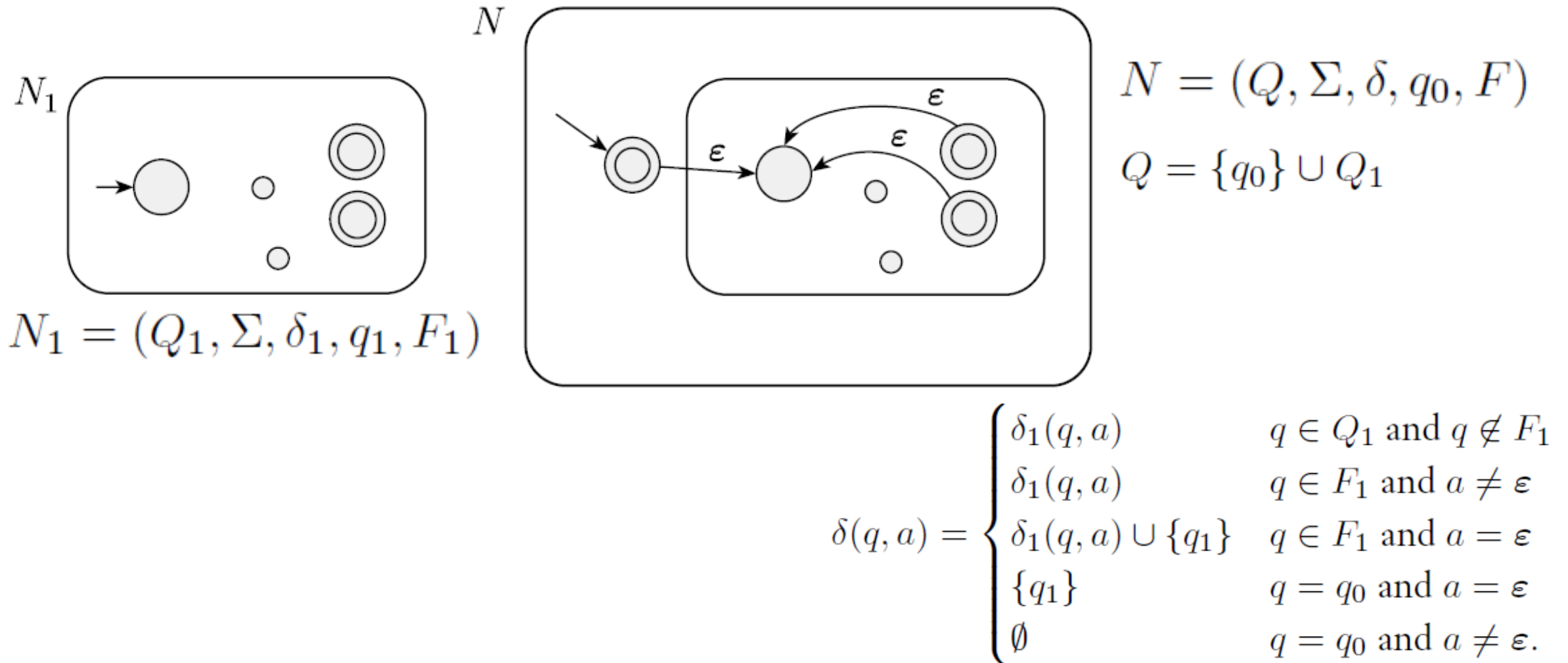
- Uma palavra da nova linguagem é uma concatenação de duas palavras, uma vinda de cada linguagem
- O novo autômato precisa aceitar palavras que foram construídas desse jeito
- Mas algumas palavras poderiam ter sido criadas de diferentes formas, isso não é importa para o autômato, basta saber se ela pertence ou não à nova linguagem

# Formalmente...

- Seja
  - $N_1 = \langle Q_1, \Sigma, \delta_1, q_1, F_1 \rangle$
  - $N_2 = \langle Q_2, \Sigma, \delta_2, q_2, F_2 \rangle$
- $N = \langle Q, \Sigma, \delta, q_0, F \rangle$ , onde
  - $Q = Q_1 \cup Q_2$
  - $q_0 = q_1$
  - $F = F_2$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ e } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ e } a \neq \lambda \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ e } a = \lambda \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

# É a operação estrela?




Qual a alternativa correta:

- a) AFNDs conseguem resolver mais problemas que AFDs;
- b) AFDs conseguem resolver mais problemas que AFNDs;
- c) Nenhuma das anteriores.

Qual a alternativa correta:

a) AFNDs conseguem resolver mais problemas que AFDs;

b) AFDs conseguem resolver mais problemas que AFNDs;

 c) Nenhuma das anteriores.

Equivalência entre AFNDs e AFDs

# AFNDs vs. AFDs

- AFNDs são mais fáceis de construir
  - Não precisam ter todas as transições
  - Podem estar em dois estados (“isso ou isso pode acontecer”)
    - Ex.: um pacote válido de dados tem um prefixo “010101...” e conteúdo de tamanho  $N$ , com paridade par e um sufixo “000111”, ou tem um sufixo “010111” e neste caso, não tem comprimento  $N/2$  e não tem sufixo...
- AFNDs são menos eficientes para execução (manter vários estados, procurar redundâncias)
- Resumo: Faça o AFND e depois converta (mecanicamente) para um AFD

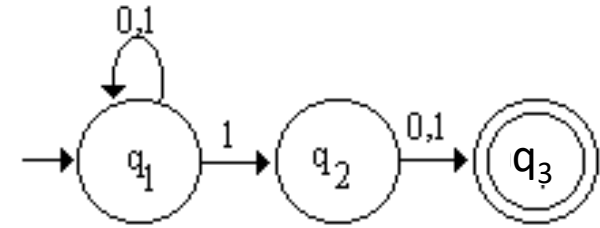
# Equivalência entre AFD e AFND

- Teorema 3: todo AFND tem um AFD equivalente
  - Equivalência: reconhecer a mesma linguagem
  - Como provar?
    - Cada estado do AFD será um registro de todos os possíveis estados do AFND para aquela entrada
    - Se um dos possíveis estados do AFND for final o estado do AFD também será
    - Prova: a partir de uma especificação de N construir um M que reconhece a mesma linguagem

Ponto de partida: lembrar do procedimento de união para dois autômatos determinísticos.



# Equivalência



- Duas formas de fazer (como na união de dois AFDs)
- Primeira: iterativa → como o AF começa e como ele reage a cada símbolo?
  - E quando o AFND não tem uma transição para um símbolo?
- Segunda: exaustiva
  - Em quantos estados “um” autômato ND poderia estar ao mesmo tempo?
  - Quais são as possibilidades?
  - Como ele muda de um para o outro?

E se tiver transições com  $\lambda$ ?

E se tiver transições com  $\lambda$ ?

