Não determinismo

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

Aula anterior: LR é fechada pela união

$A_1 = L(M_1)$, onde $M_1 = <Q_1, \Sigma, \delta_1, q_1, F_1>$

$A_2 = L(M_2)$, onde $M_2 = <Q_2, \Sigma, \delta_2, q_2, F_2>$

Construa $M = <Q, \Sigma, \delta, q_0, F>$, tal que $A_1 \cup A_2 = L(M)$, da seguinte forma:

1. $Q = Q_1 \times Q_2 = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ e } r_2 \in Q_2\}$;
2. Σ é o mesmo;
3. $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$;
4. $q_0 = (q_1, q_2)$;
5. $F = \{(r_1, r_2) : r_1 \in F_1 \text{ ou } r_2 \in F_2\}$.

IMPORTANTE: usar o nome dos estados neste formato!

Entendimento desejável

- Além da união, como esse procedimento poderia ser adaptado para gerar um autômato que tem como linguagem a intersecção das linguagens de dois outros? E como seria para a diferença (L1 – L2)?

- Um pouquinho mais complicado: e se os alfabetos fossem diferentes?
  - Em especial, se um dos autômatos tivesse um símbolo a mais? Como seria a geração do terceiro autômato?
  - Primeiro, pense na forma intuitiva, depois na formal.

Implicações do fechamento?

- Sempre é possível construir a máquina...
  - Para o complemento de uma linguagem regular
  - Para a união/intersecção/diferença de duas linguagens regulares

Implicações do fechamento?

- Sempre é possível construir a máquina...
  - Para o complemento de uma linguagem regular
  - Para a união/intersecção/diferença de duas linguagens regulares

- Será que a classe das linguagens regulares é fechada pela operação de concatenação?
  - Um caso específico, é possível fazer um AF para $L = L_1.L_2$?
    - $L_1 = \{ w \in \sum^* \mid w$ contém exatamente 3 símbolos$\}$
    - $L_2 = \{ w \in \sum^* \mid w$ tem um número ímpar de 0s$\}$

Implicações do fechamento?

- Sempre é possível construir a máquina...
  - Para o complemento de uma linguagem regular
  - Para a união/intersecção/diferença de duas linguagens regulares

- Será que a classe das linguagens regulares é fechada pela operação de concatenação?
  - Um caso específico, é possível fazer um AF para $L = L_1.L_2$?
    - $L_1 = \{ w \in \sum^* \mid w$ contém exatamente 3 símbolos$\}$
    - $L_2 = \{ w \in \sum^* \mid w$ tem um número ímpar de 0s$\}$
  - Mas isso vale para quaisquer duas linguagens? E se tivéssemos $L = L_2.L_1$?

Uma nova funcionalidade...

Não determinismo

- Um poder a mais para autômatos finitos
- AF determinístico: os estados e as transições são bem determinadas (apenas uma opção para o que deve ser feito para cada símbolo que pode aparecer)
- AF não-determinístico
  - Podem existir múltiplas possibilidades para o próximo estado, dado o mesmo símbolo
  - Ou seja, ele pode explorar diversas possibilidades ao mesmo tempo

Autômatos Finitos Não Determinísticos (1)

- Um estado pode ter zero, um ou mais arcos “saindo” para o mesmo símbolo do alfabeto

- Todo autômato determinístico é um autômato não determinístico

- ...

Exemplo

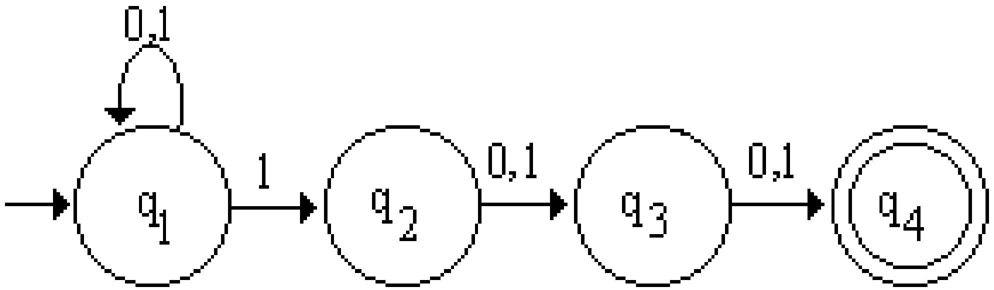
- Como seria o AF determinístico que reconhece palavras que terminam em 00?

- E uma versão mais enxuta dele usando não-determinismo?

Exemplo: N_1

- Como N_1 computa? Qual a linguagem dele?

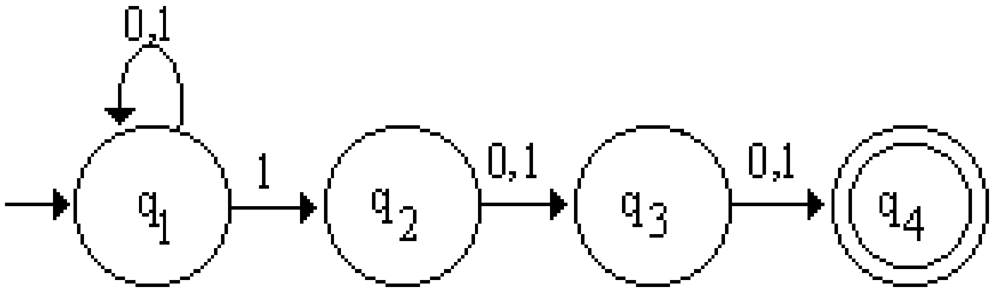
N_1:



Exemplo: N_1

- Como N_1 computa? Qual a linguagem dele?

N_1:



$L(N_2) = \{ w \in \{0,1\}^* \mid w = u1v, \; u \in \Sigma^* , v \in \Sigma^2\}$

Autômatos Finitos Não Determinísticos

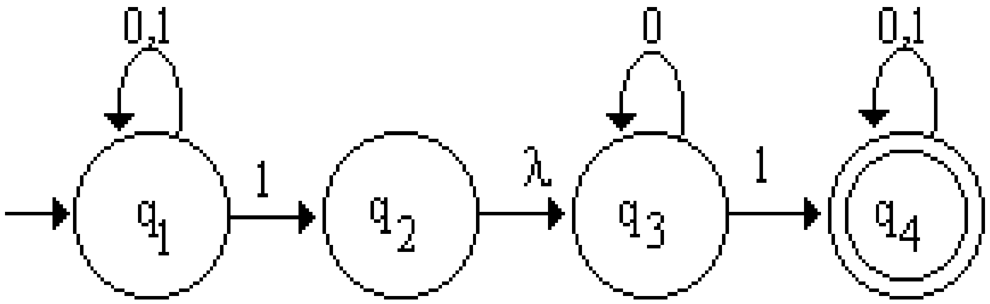
- Um estado pode ter zero, um ou mais arcos “saindo” para o mesmo símbolo do alfabeto

- Todo autômato determinístico é um autômato não determinístico

- Zero, um ou mais arcos rotulados com λ podem sair de cada estado

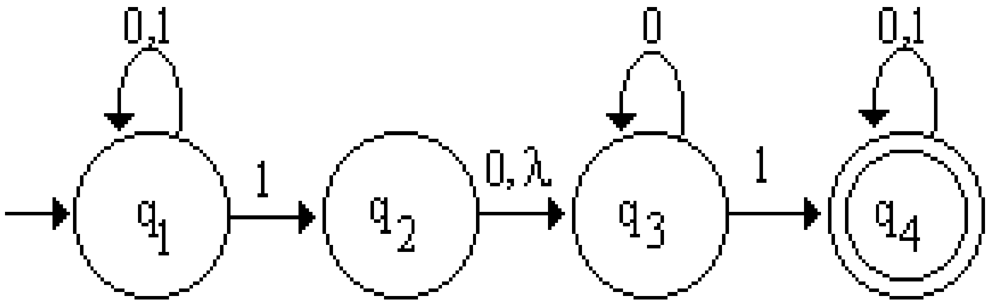
Exemplo: N_2

- E como N_2 executa?



Exemplo: N_2

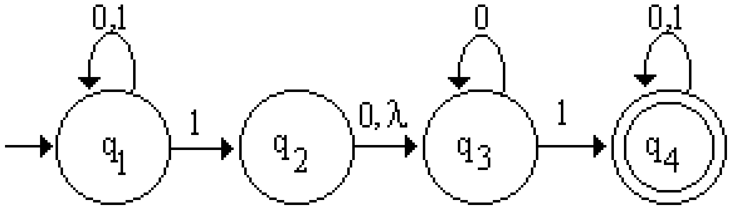
- E como N_3, um pouco mais complicado que N_2, executa?



Execução de N_3

- Considere a entrada 01011
  - Essa entrada é aceita?

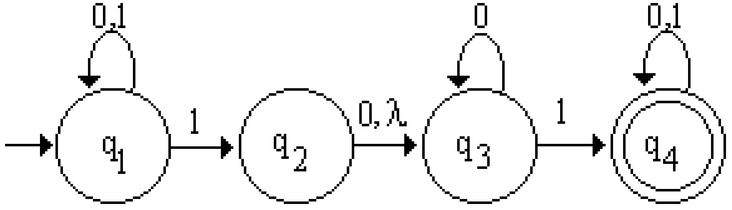
N_1:

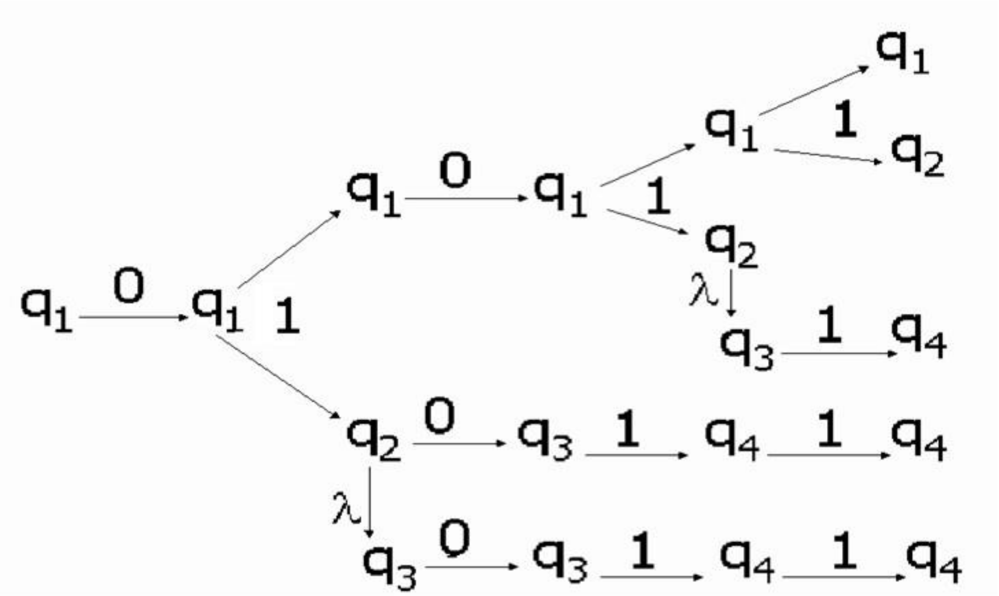


Execução de N_2

- Considere a entrada 01011
    - Essa entrada é aceita?

N_1:

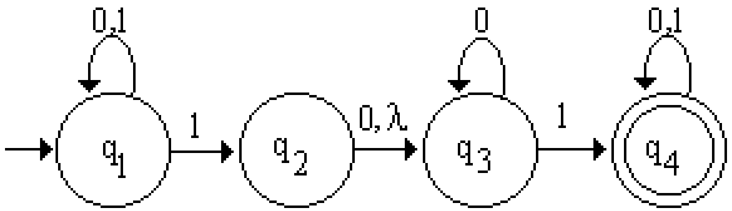




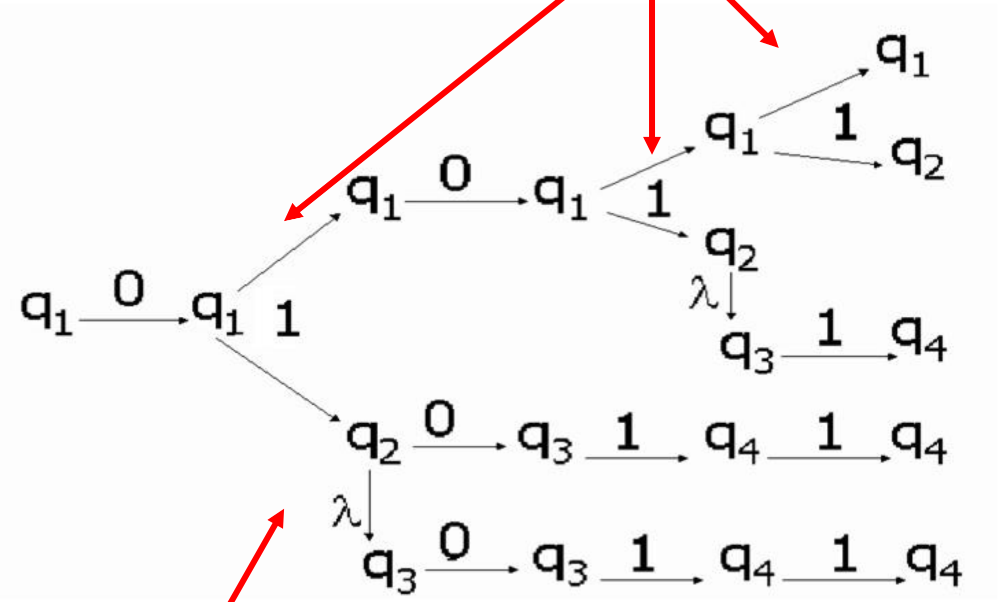
Execução de N_2

- Considere a entrada 01011
  - Essa entrada é aceita?
  - Como é a execução de N_2?

N_1:



Consome um símbolo e avança (dividindo a execução).



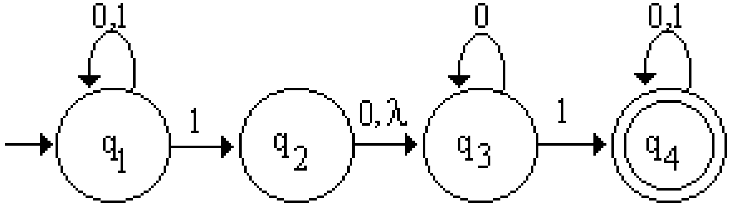
Avança sem consumir (dividindo a execução).

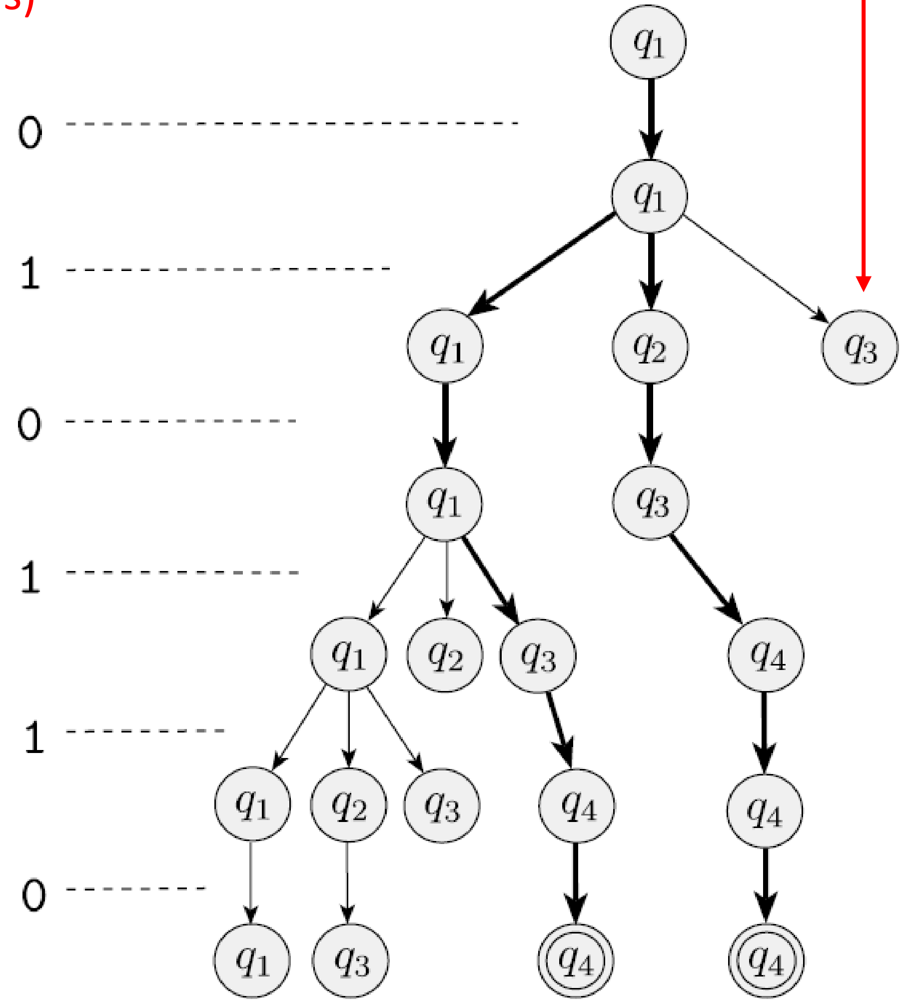
Execução de N_2

Versão do livro dá os dois passos de uma vez! (note que, nesta execução, a palavra tem um símbolo a mais)

- Considere a entrada 01011**0**
  - Essa entrada é aceita?
  - Como é a execução de N_2?

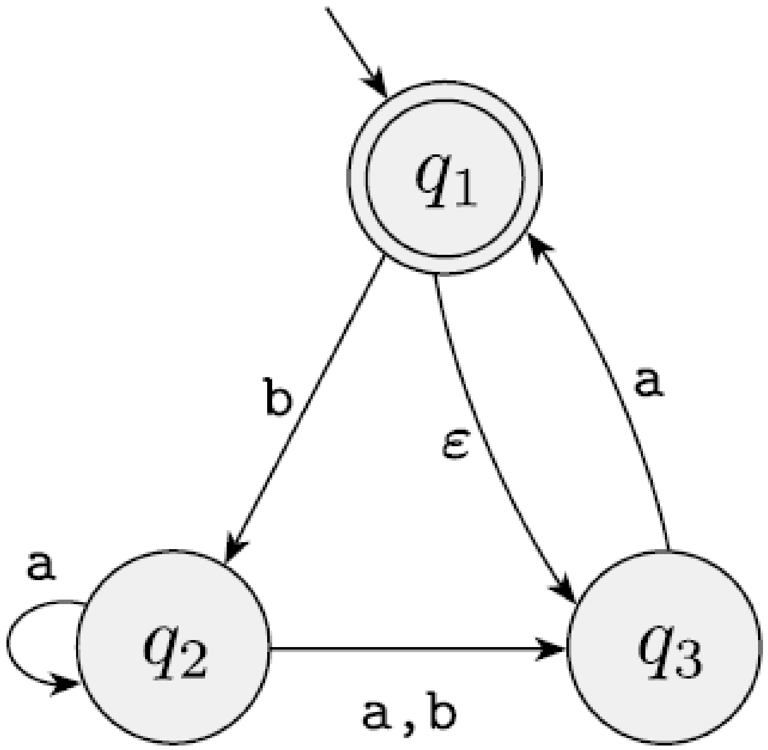
N_1:





Execução de N_3

- Como esta máquina executa?



Execução de autômatos finitos não-determinísticos

- Comparações possíveis
  - Uma árvore de possibilidades
  - Um conjunto de processos paralelos

Nossos AFs: autômatos finitos determinísticos

- Definição formal precisa ser completa e precisa

- Um AFD M é uma 5-tupla $<Q, \Sigma, \delta, q_0, F>$
  - Q é um conjunto finito e não-vazio chamado de conjunto de estados
  - Σ é um conjunto finito e não-vazio chamado de alfabeto
  - $\delta : Q \times \Sigma \to Q$ é a função de transição do autômato
  - $q_0 \in Q$ é o estado inicial
  - $F \subseteq Q$ é o conjunto de estados finais

AFND: definição formal

- $P(Q)$: conjunto dos subconjuntos de Q (conjunto das partes)

- Um Autômato Finito Não-Determinístico (AFND) é uma 5-tupla $<Q, \Sigma, \delta, q_0, F>$, onde

  λ

  - Q é um conjunto finito de estados
  - Σ é um alfabeto finito
  - $\delta: Q \times \Sigma_{\varepsilon} \rightarrow P(Q)$ é a função de transição
  - $q_0 \in Q$ é o estado inicial
  - $F \subseteq Q$ é o conjunto de estados de aceitação

Qual a definição formal do AFND abaixo?

- Use uma tabela para representar a função de transição