[Aula 09] Propriedades das LRs – Minimização de AFD

Prof. João F. Mari
joaof.mari@ufv.br

ROTEIRO

Minimização de um AFD

Igualdade de LRs

- AFD Mínimo ou Autômato Finito Mínimo
  - AFD equivalente, com o menor número de estados **possível**.
- Minimização em algumas aplicações especiais:
  - Não necessariamente o menor custo de implementação.
  - **[EX]** circuitos lógicos ou redes lógicas:
    - Pode ser desejável introduzir estados intermediários de forma a melhorar eficiência ou facilitar ligações físicas.
  - Prever variáveis específicas da aplicação.
- Autômato finito mínimo é único
  - A menos de isomorfismo;
  - Diferenciando-se, eventualmente, na identificação dos estados.

Minimização de um Autômato Finito

- Algoritmo de minimização:
  - Unifica os estados equivalentes.
- Estados equivalentes:
  - Processamento de uma entrada qualquer;
  - A partir de estados equivalentes;
  - Resulta na mesma condição de aceitação.
- **[DEFINIÇÃO]** Estados Equivalentes:
  - $M = (\Sigma, Q, \delta, q_0, F)$ é um AFD qualquer:
  - q e p de Q são **Estados Equivalentes** sse, para qualquer $w \in \Sigma^*$

$$\delta(q, w) \quad \text{e} \quad \delta(p, w)$$

  - resultam simultaneamente em estados finais, ou não-finais.

Autômato Finito Mínimo

- Seja L uma linguagem regular.
  - O Autômato Finito Mínimo é um AFD Mm:

$$M_m = (\Sigma, Q_m, \delta_m, q_{0m}, F_m)$$

  - Tal que $ACEITA(M_m) = L$.
- Para qualquer AFD $M = (\Sigma, Q, \delta, q_0, F)$ tal que $ACEITA(M) = L$

$$\#Q \geq \#Q_m$$

Pré-requisitos do algoritmo de minimização

- Autômato Finito Determinístico.
  - Todos os estados alcançáveis a partir do estado inicial.
  - Função programa total.
- Caso não satisfaça algum dos pré-requisitos:
  - Gerar um autômato determinístico equivalente;
    - Algoritmos de tradução apresentados nos teoremas.
- Eliminar estados inacessíveis (e transições)
- Função programa total:
  - Introduzir um estado não-final **d**;
  - Incluir transições não-previstas, tendo **d** como estado destino;
  - Incluir um ciclo em **d** para todos os símbolos do alfabeto.

Algoritmo de Minimização

- Identifica os estados equivalentes por exclusão.
- Montar uma **Tabela de Estados**:
  - Marcar os estados não-equivalentes.
  - Entradas não-marcadas:
    - São estados equivalentes.

Algoritmo de Minimização

Passo 1 – Construção da tabela

- Seja M = (Σ, Q, δ, q_0, F)
  - AFD que satisfaz aos pré-requisitos.
- Construção da tabela:

M = {q_0, q_1, q_2, ..., q_n)

q_1					
q_2					
...					
q_n					
d					
	q_0	q_1	...	q_{n-1}	q_n

Algoritmo de Minimização

Passo 2: Estados trivialmente não-equivalentes

- Marcação dos Estados Trivialmente Não-Equivalentes:
  - Pares do tipo { estado final, estado não-final }

Passo 3: Estados não-equivalentes

- Para { q_u, q_v } não-marcado e **a** ∈ Σ, suponha que

$$\delta(q_u, a) = p_u \quad \text{e} \quad \delta(q_v, a) = p_v$$

- **pu = pv**
  - qu é equivalente a qv para a: não marcar.
- **pu ≠ pv e { pu, pv } não está marcado**
  - { qu, qv } incluído na lista encabeçada por { pu, pv }.
- **pu ≠ pv e { pu, pv } está marcado**
  - { qu, qv } não é equivalente: marcar.
  - Se { qu, qv } encabeça uma lista:
    - Marcar todos os pares da lista;
    - E, recursivamente, se algum par da lista encabeça outra lista.

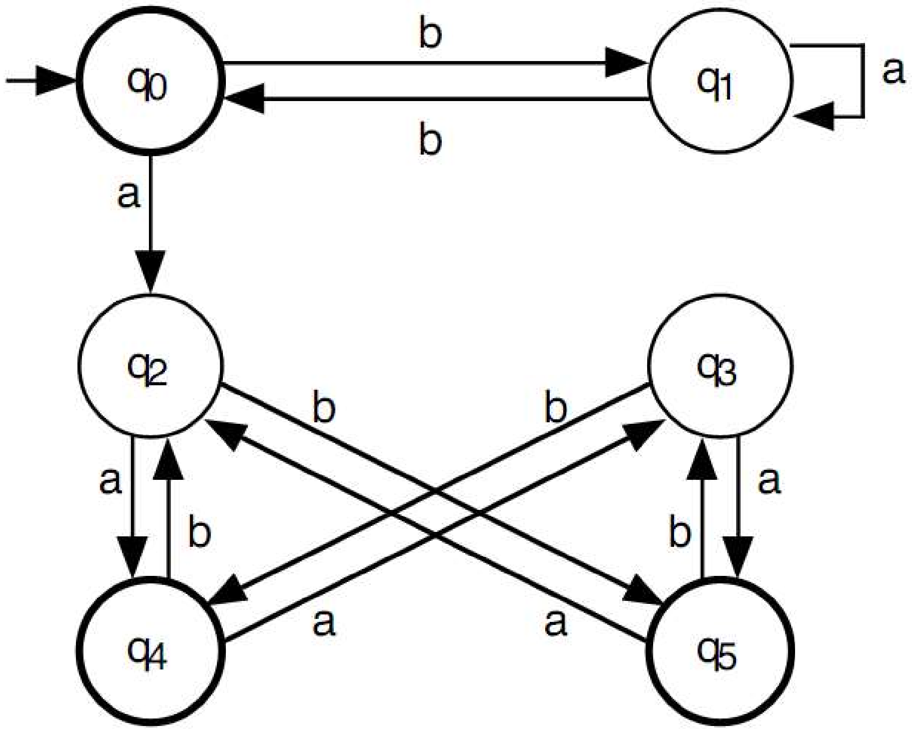
Passo 4: Unificação dos Estados Equivalentes

- Pares não-marcados são equivalentes:
  - Equivalência de estados é transitiva;
  - Os pares de estados não finais equivalentes.
    - geram um único estado não final.
  - Pares de estados finais equivalentes
    - geram um único estado final.
  - Se algum dos estados equivalentes é inicial
    - O estado unificado é inicial.
  - Transições com origem (destino) em um estado equivalente
    - origem (destino) no estado unificado.

Passo 5: Exclusão dos Estados Inúteis

- **q** é um estado inútil se:
  - É não-final;
  - A partir de **q** não é possível atingir um estado final.
- O estado **d** (se incluído) sempre é inútil.
- Excluir as transições com origem ou destino em estado inútil.

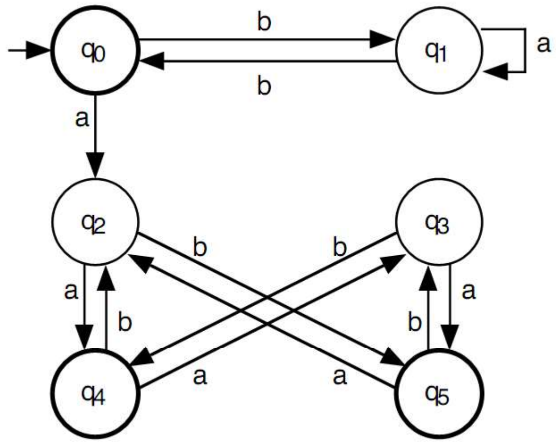
[EX] Algoritmo de minimização



[EX] Algoritmo de minimização

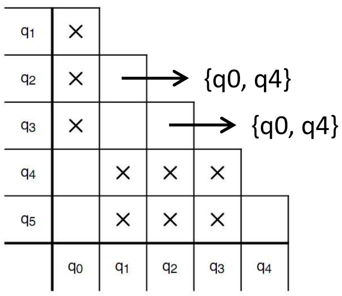
q1	×				
q2	×				
q3	×				
q4		×	×	×	
q5		×	×	×	
	q0	q1	q2	q3	q4

- Passo 1
  - Construção da tabela
- Passo 2
  - Marcação dos pares { estado final, estado não-final }

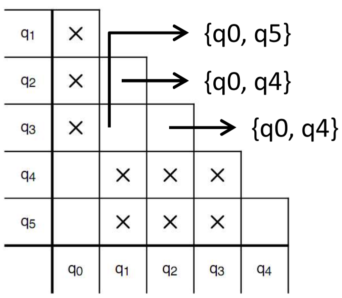


[EX] Algoritmo de minimização

- Analisando o par {q0, q4}:
  - Para a: δ(q0, a) = q2 e δ(q4, a) = q3, então {q2, q3}
    - {q2, q3} é não-marcado...
      - Incluir {q0, q4} nas listas de {q2, q3}.
  - Para b: δ(q0, b) = q1 e δ(q4, b) = q2, então {q1, q2}
    - {q1, q2} é não-marcado...
      - Incluir {q0, q4} na listas de {q1, q2} .

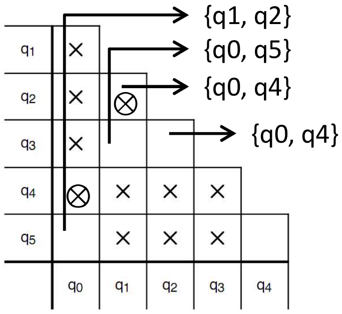


- Analisando o par {q0, q5}:
  - Para a: δ(q0, a) = q2 e δ(q5, a) = q2, então {q2, q2}
    - {q2, q2} é trivialmente equivalente...
      - Nada a fazer.
  - Para b: δ(q0, b) = q1 e δ(q5, b) = q3, então {q0, q5}
    - {q1, q3} é não-marcado...
      - Incluir {q0, q5} na lista de {q1, q3}.

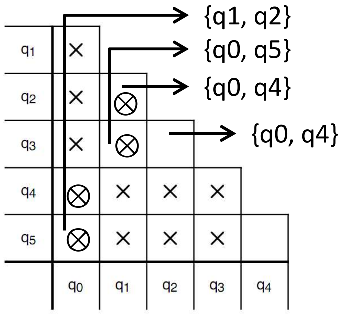


[EX] Algoritmo de minimização

- Analisando {q1, q2}:
  - Para a: δ(q1, a) = q1 e δ(q2, a) = q4, então {q1, q4}
    - {q1, q4} é marcado → marcar {q1, q2}.
      - {q1, q2} encabeça lista: marcar {q0, q4}.
      - {q0, q4} não encabeça lista: interrompe o processo.
  - Para b: δ(q2, b) = q5 e δ(q1, b) = q0, então {q0, q5}
    - {q0, q5} é não-marcado...
      - Incluir {q1, q2} na lista de {q0, q5}.



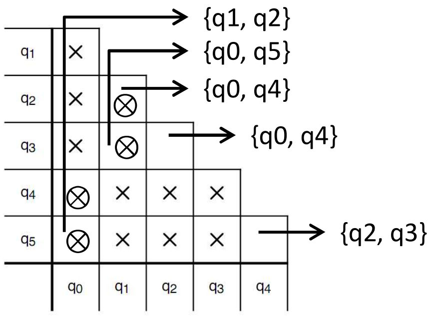
- Analisando {q1, q3}:
  - Para a: δ(q1, a) = q1 e δ(q3, a) = q5, então {q1, q5}
    - {q1, q5} é marcados → marcar {q1, q3}
      - {q1, q3} encabeça lista: marcar {q0, q5}
      - {q0, q5} encabeça lista: marcar {q1, q2}
      - {q1, q2} encabeça lista: marcar {q0, q4}.
      - {q0, q4} não encabeça lista: interrompe o processo.
  - Para b: δ(q3, b) = q4 e δ(q1, b) = q0, então {q0, q4}
    - {q0, q4} é marcado: marcar {q1, q3}
      - {q1, q3} encabeça uma lista: (ver acima)



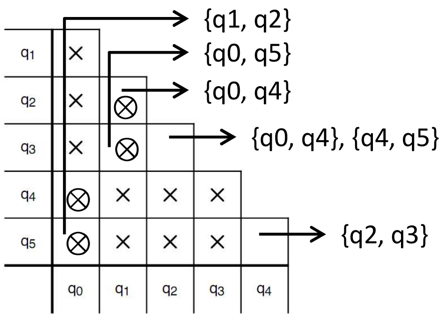
Algoritmo de Minimização

[EX] Algoritmo de minimização

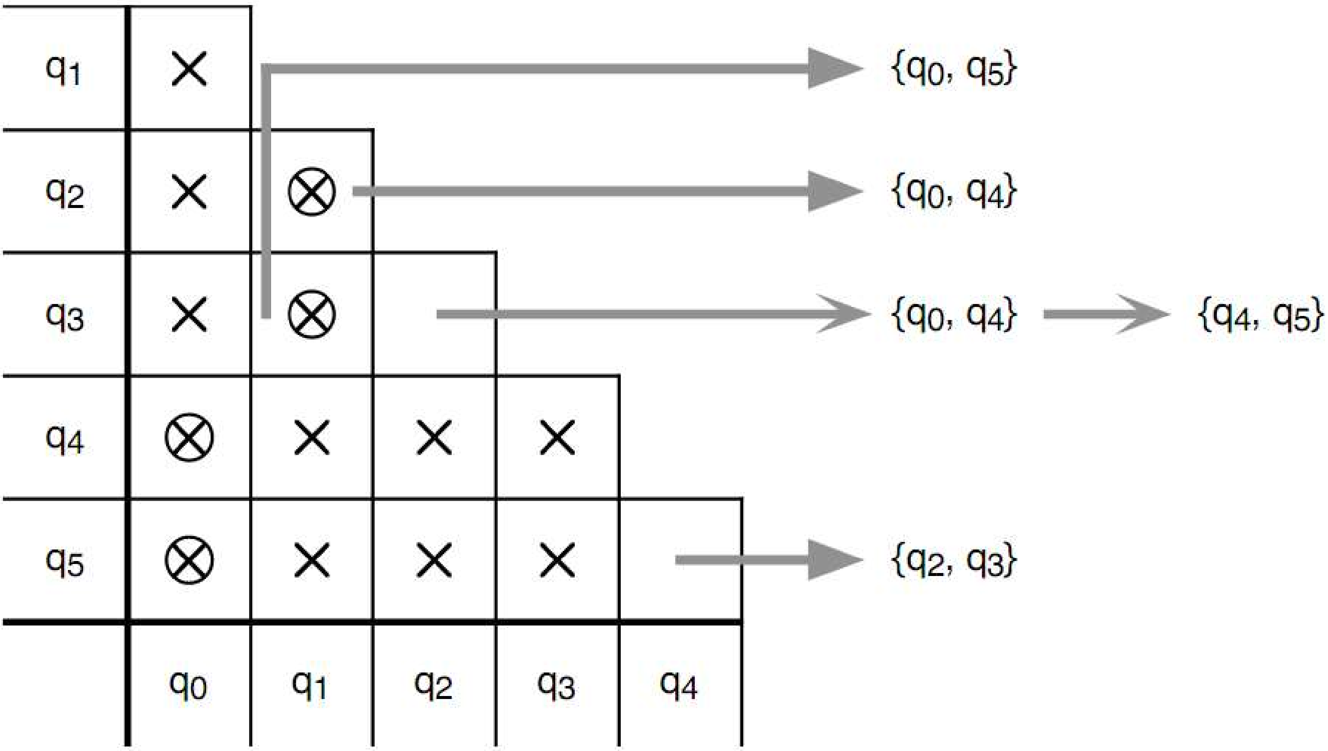
- Analisando {q2, q3}:
  - Para a: δ(q2, a) = q4 e δ(q3, a) = q5, então {q4, q5}
  - Para b: δ(q2, b) = q5 e δ(q3, b) = q4, então {q4, q5}
    - {q4, q5} é não-marcado
      - incluir {q2, q3} na lista de {q4, q5}



- Analisando {q4, q5}:
  - Para a: δ(q4, a) = q3 e δ(q5, a) = q2, então {q2, q3}
  - Para b: δ(q4, b) = q2 e δ(q5, b) = q3, então {q2, q3}
    - {q2, q3} é não-marcado
      - Incluir {q4, q5} na lista de {q2, q3}



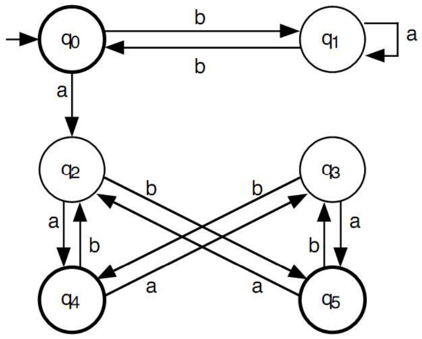
[EX] Algoritmo de minimização

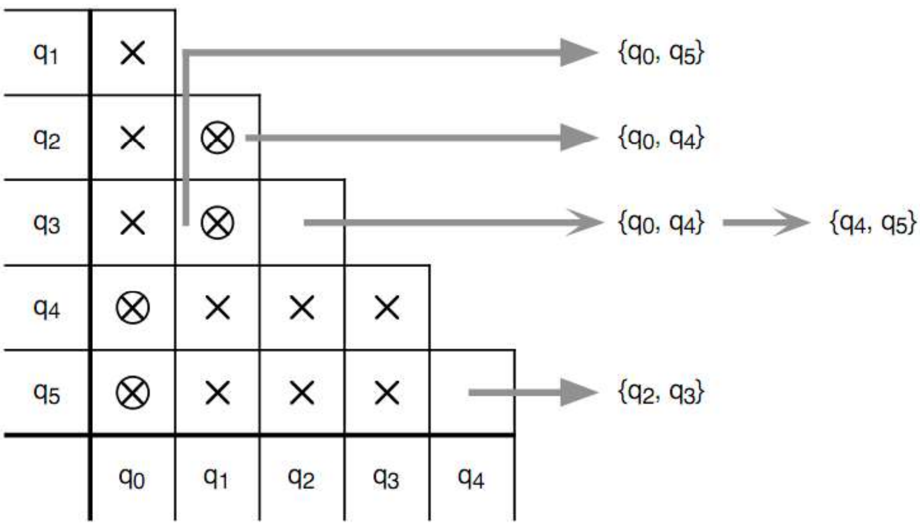


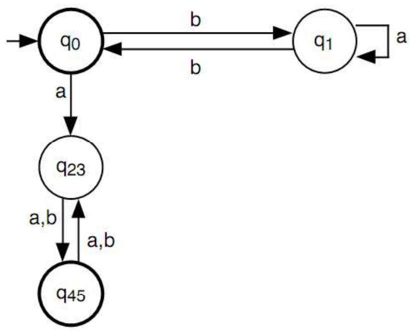
[AULA 09] Propriedades das LRs – Minimização de AFD SIN 131 – Introdução à Teoria da Computação (PER-3)
Algoritmo de Minimização

[EX] Algoritmo de minimização

- Passo 4. { q_2, q_3 } e { q_4, q_5 } são não-marcados
  - q_{23}: unificação dos estados q_2 e q_3.
  - q_{45}: unificação dos estados finais q_4 e q_5.

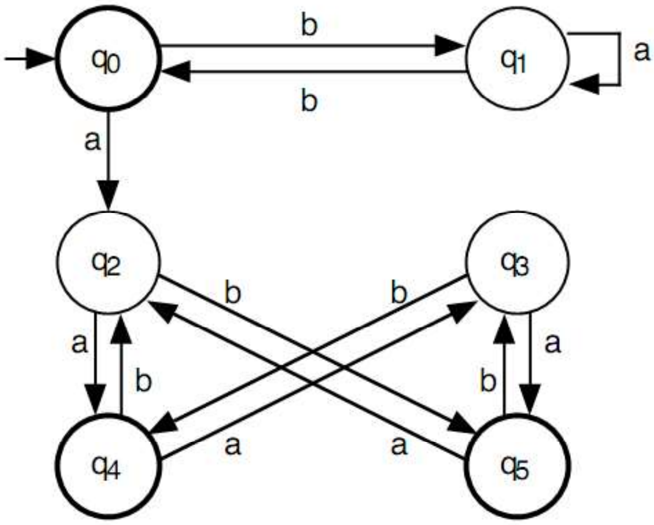




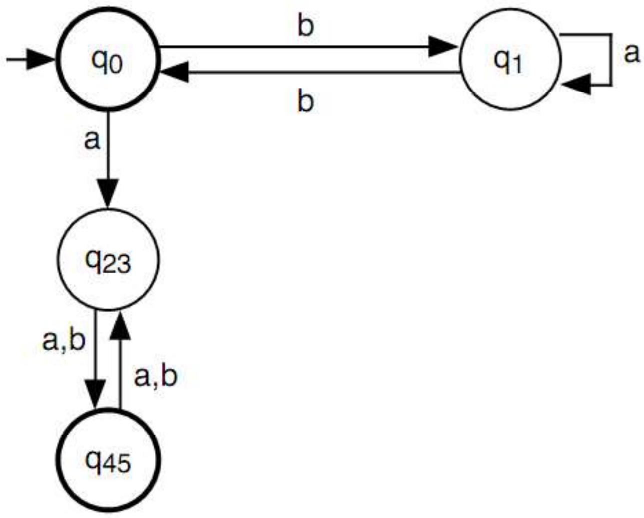


[EX] Algoritmo de minimização

- AFD



- AFD Mínimo



[EX] Algoritmo de minimização

- Teorema: Autômato Finito Mínimo
  - O autômato construído usando o algoritmo de minimização
    - AFD com menor número de estados que aceita a linguagem
- Teorema: Unicidade do Autômato Finito Mínimo
  - AFD mínimo de uma linguagem é único
    - A menos de isomorfismo
    - Usual ser referido como o (e não como um) autômato mínimo.
- Isomorfismo de AFD:
    - Diferencia-se, eventualmente, na identificação (nome) dos estados

PROPRIEDADES DAS LRs

Operações fechadas sobre as LR

- Operações sobre LR podem ser usadas para:
  - Construir novas linguagens a partir de linguagens conhecidas:
    - Álgebra de LR.
  - Provar propriedades e construir algoritmos.

- Classe de Linguagens Regulares é **fechada** para:
  - União;
  - Concatenação;
  - Complemento;
  - Intersecção.

União e Concatenação

- Decorrem trivialmente da definição de expressão regular (ER).

Complemento

- Suponha L uma LR sobre Σ*. Então existe um AFD M:

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

  - tal que ACEITA(M) = L.

- Construir um AFD M_C tal que ACEITA(M_C) = ~L

$$MC = (\Sigma, Q_C, \delta_C, q_0, F_C)$$

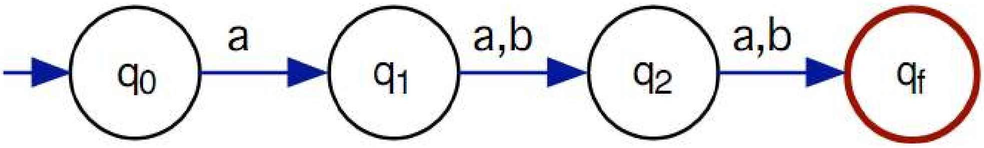
- Como?
  - Introduzir um novo estado (não final) ***d***:
    - Destino de todas as transações indefinidas;
    - $\boldsymbol{Q_C} = \boldsymbol{Q} \cup \{ \boldsymbol{d} \}$ (suponha $\boldsymbol{d} \notin \boldsymbol{Q}$).
  - Um ciclo em ***d*** para todo símbolo de ***Σ***
    - Garante a leitura de toda a entrada.
  - Transformar estados finais em não finais e vice-versa:
    - $\boldsymbol{F_C} = \boldsymbol{Q_C} - \boldsymbol{F}$

- Claramente, o autômato finito M_C é tal que

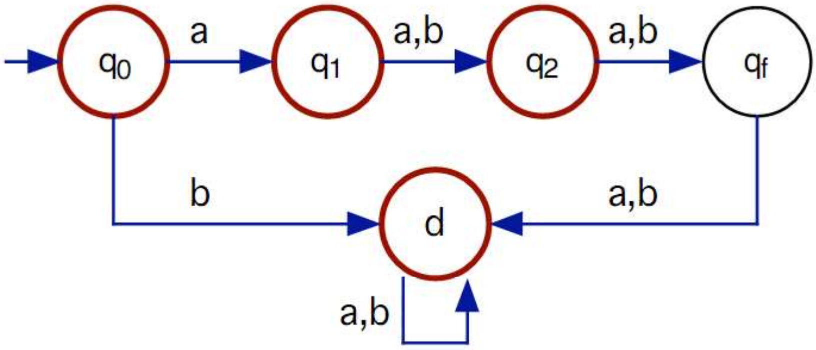
ACEITA(M_C) = ~L ou seja ACEITA(M_C) = REJEITA(M)

Complemento

- $M = (\{ a, b \}, \{ q_0, q_1, q_2, q_f \}, \delta, q_0, \{ q_f \})$



- $MC = (\{ a, b \}, \{ q_0, q_1, q_2, q_f, d \}, \delta_C, q_0, \{ q_0, q_1, q_2, d \})$



Intersecção

- Suponha L_1 e L_2 LR
- Propriedade de DeMorgan para conjuntos

$$L_1 \cap L_2 = \sim(\sim L_1 \cup \sim L_2)$$

- Como a Classe das LR é fechada para **complemento** e **união**:
  - Então também é fechada para a intersecção.

OUTRAS PROPRIEDADES

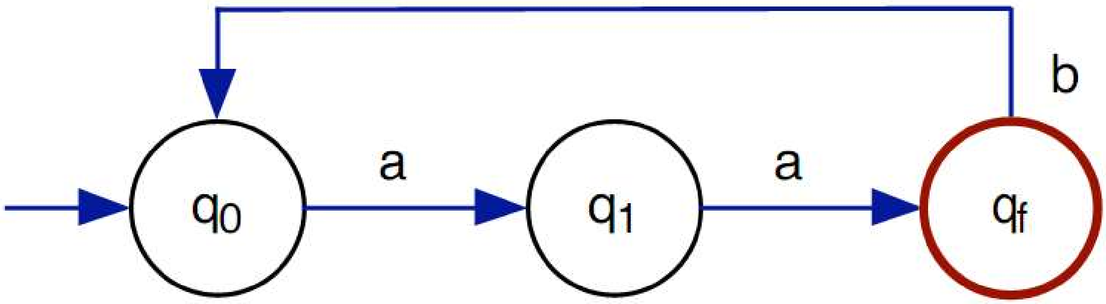
LR finita, infinita ou vazia

- Se L é uma LR aceita por um autômato finito M = $(\Sigma, Q, \delta, q_0, F)$ com n estados, então L é:
  - Vazia:
    - Sse M **não aceita** qualquer palavra w tal que
    - $|w| < n$
  - Infinita:
    - Sse M **aceita** pelo menos uma palavra w tal que
    - $n \leq | w | < 2n$
  - Finita:
    - Sse M **não aceita** qualquer palavra w tal que
    - $n \leq | w | < 2n$
      - Contraposição sobre L infinita.

[EX] LR finita, infinita ou vazia

- A linguagem é infinita sse **aceita uma palavra** w tal que:
  - $n \leq |w| < 2n$:
  - aabaa é aceita
  - $3 \leq |aabaa| < 6$

- Logo, a linguagem é infinita.



Igualdade de LRs

- Teorema mostra que:
  - Existe um algoritmo para verificar se dois autômatos finitos são equivalentes:
    - Reconhecem a mesma linguagem.

- Importante consequência:
  - Existe um algoritmo que permite verificar se duas implementações são equivalentes.

Igualdade de LRs

- Se M_1 e M_2 são AF, então existe um algoritmo para determinar se:

$$\text{ACEITA}(M_1) = \text{ACEITA}(M_2)$$

- <u>PROVA:</u>
- Suponha M_1 e M_2 AFs
  - $\text{ACEITA}(M_1) = L_1$ e $\text{ACEITA}(M2) = L_2$
- Portanto, é possível construir um AF M3:
  - Tal que $\text{ACEITA}(M_3) = L_3$

$$L_3 = (L_1 \cap \sim L_2) \cup (\sim L_1 \cap L_2)$$

- Claramente, $L_1 = L_2$ sse L_3 é vazia.
  - Existe um algoritmo para verificar se uma LR é vazia.

BIBLIOGRAFIA

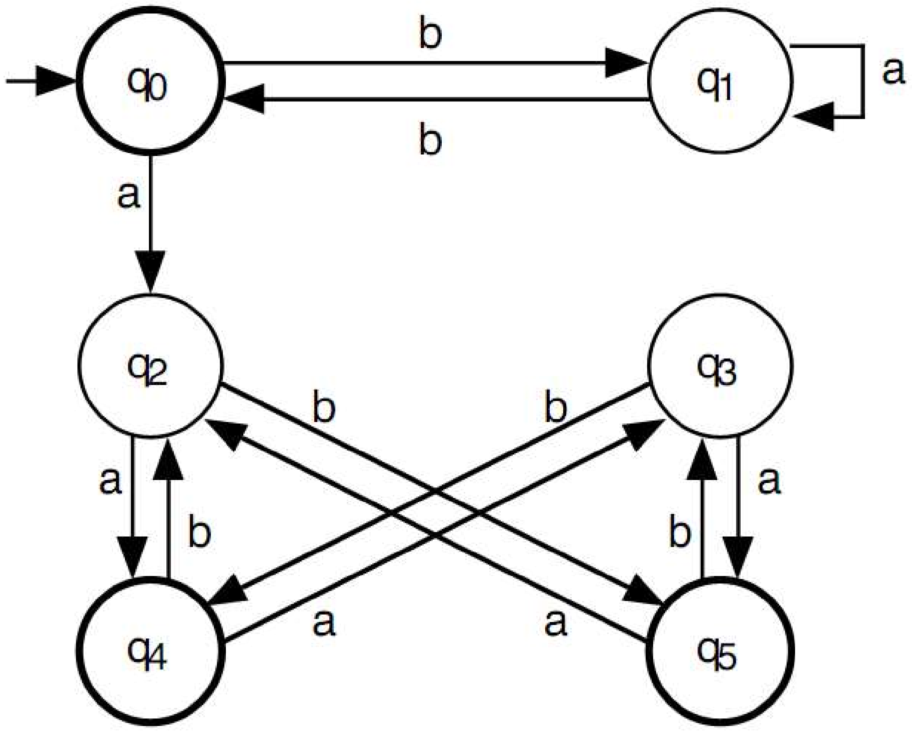
- MENEZES, P. B. **Linguagens formais e autômatos**, 6. ed., Bookman, 2011.
  - Capítulo 4.
  - + Slides disponibilizados pelo autor do livro.

[FIM]

- FIM:
  - **[AULA 09]** Propriedades das LRs – Minimização de AFD
- Próxima aula:
  - **[AULA 10]** Máquina de Mealy e máquina de Moore

Apendice A – Minimização – Exemplo do livro

[EX] Algoritmo de minimização



[EX] Algoritmo de minimização

- Passo 1
  - Construção da tabela
- Passo 2
  - Marcação dos pares { estado final, estado não-final }

q_1	×				
q_2	×				
q_3	×				
q_4		×	×	×	
q_5		×	×	×	
	q_0	q_1	q_2	q_3	q_4

[EX] Algoritmo de minimização

- { q_0, q_4 }

  $\delta(q_0, a) = q_2$ $\delta(q_0, b) = q_1$
  $\delta(q_4, a) = q_3$ $\delta(q_4, b) = q_2$
  - { q_1, q_2 } e { q_2, q_3 } são não-marcados.
    - Incluir { q_0, q_4 } nas listas de { q_1, q_2 } e { q2, q3 }.
- { q_0, q_5 }

  $\delta(q_0, a) = q_2$ $\delta(q_0, b) = q_1$
  $\delta(q_5, a) = q_2$ $\delta(q_5, b) = q_3$
  - { q_1, q_3 } é não-marcado (e { q_2, q_2 } é trivialmente equivalente).
    - Incluir { q_0, q_5 } na lista de { q_1, q_3 }.
- { q_1, q_2 }

  $\delta(q_1, a) = q_1$ $\delta(q_1, b) = q_0$
  $\delta(q_2, a) = q_4$ $\delta(q_2, b) = q_5$
  - { q_1, q_4 } é marcado: marcar { q_1, q_2 }.
  - { q_1, q_2 } encabeça uma lista: marcar { q_0, q_4 }.

[EX] Algoritmo de minimização

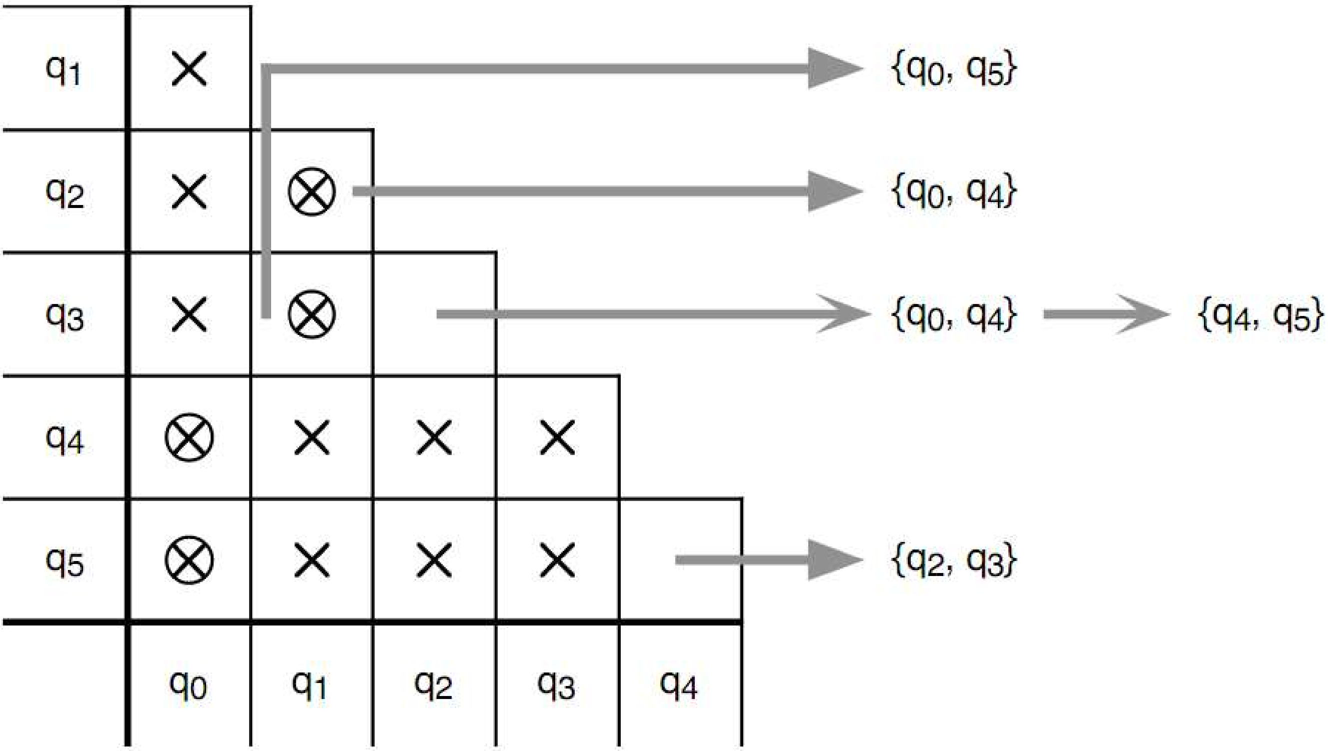
- { q_1, q_3 }

  $\delta(q_1, a) = q_1$ $\delta(q_1, b) = q_0$
  $\delta(q_3, a) = q_5$ $\delta(q_3, b) = q_4$
  - { q_1, q_5 } e { q_0, q_4 } são marcados: marcar { q_1, q_3 }
  - { q_1, q_3 } encabeça uma lista: marcar { q_0, q_5 }
- { q_2, q_3 }

  $\delta(q_2, a) = q_4$ $\delta(q_2, b) = q_5$
  $\delta(q_3, a) = q_5$ $\delta(q_3, b) = q_4$
  - { q_4, q_5 } é não-marcado: incluir { q_2, q_3 } na lista de{ q_4, q_5 }
- { q_4, q_5 }

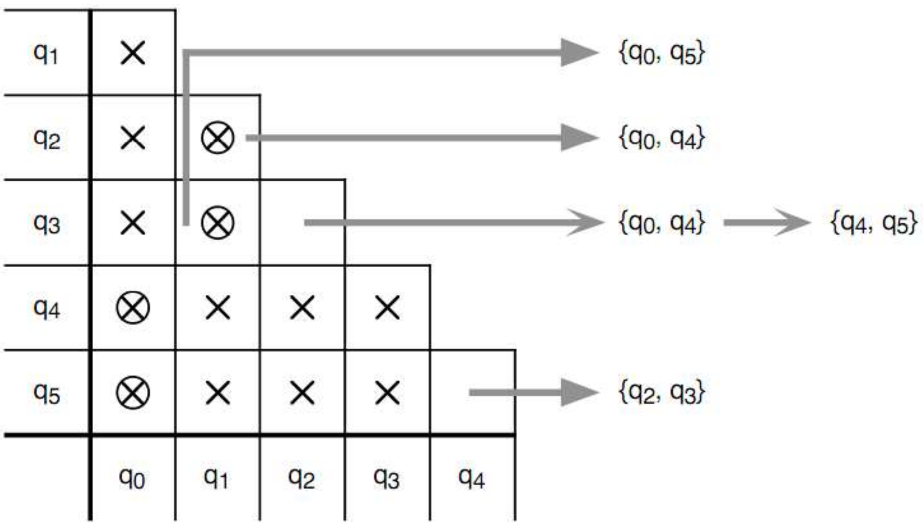
  $\delta(q_4, a) = q_3$ $\delta(q_4, b) = q_2$
  $\delta(q_5, a) = q_2$ $\delta(q_5, b) = q_3$
  - { q_2, q_3 } é não-marcado: incluir { q_4, q_5 } na lista de{ q_2, q_3 }

[EX] Algoritmo de minimização



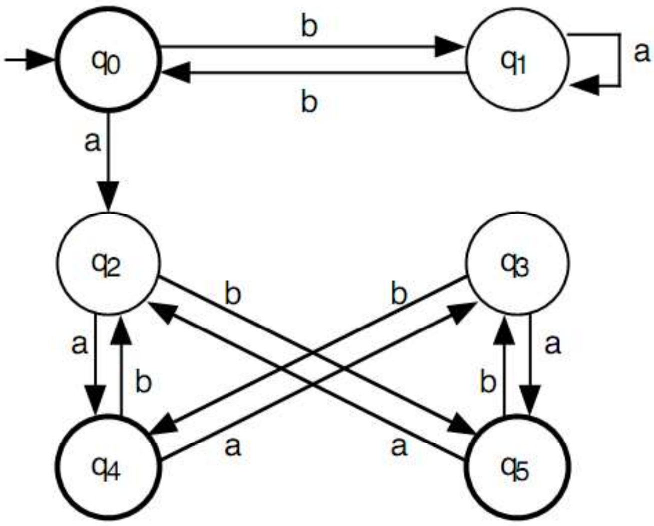
[EX] Algoritmo de minimização

- Passo 4. { q_2, q_3 } e { q_4, q_5 } são não-marcados
  - q_{23}: unificação dos estados q_2 e q_3.
  - q_{45}: unificação dos estados finais q_4 e q_5.



[EX] Algoritmo de minimização

- AFD



- AFD Mínimo