

ACH2043
INTRODUÇÃO À TEORIA DA
COMPUTAÇÃO

Minimização de Autômatos Finitos

Prof. Marcelo S. Lauretto

marcelolauretto@usp.br

www.each.usp.br/lauretto

Comentários iniciais

- Um dos resultados teóricos mais importantes para a classe de linguagens regulares:
- “Toda linguagem regular possui um autômato finito determinístico, mínimo e único que a reconhece”

Comentários iniciais

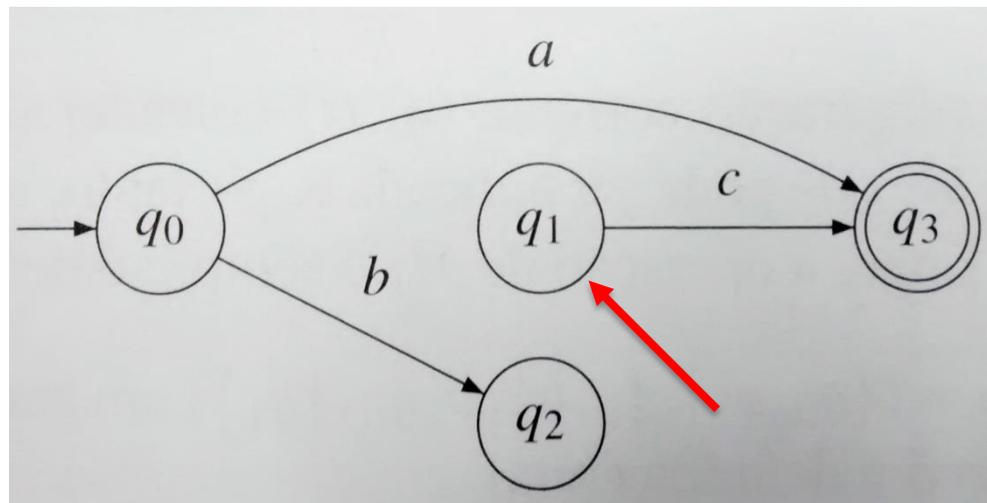
- Importância desse resultado:
 - Foi demonstrado que esse resultado é válido apenas para a classe das linguagens regulares
 - Possibilita a construção de reconhecedores sintáticos extremamente compactos e eficientes
 - É possível automatizar a minimização de autômatos finitos
 - Autômato finito mínimo é único para cada linguagem regular, possibilitando a elaboração de novos métodos no estudo de linguagens formais
 - P.ex. pode-se verificar a equivalência de duas linguagens regulares através da redução dos correspondentes autômatos finitos às suas versões equivalentes mínimas

Método de minimização de estados: etapas

- Partimos do pressuposto de que o autômato a ser minimizado é determinístico
 - Não possui transições com cadeia vazia
- Processo de minimização do número de estados ocorre em duas etapas:
 1. Eliminação de estados inacessíveis e estados inúteis
 2. Agrupamento e fusão de estados equivalentes

Estados inacessíveis

- Um estado q_i é dito inacessível se não existe no autômato qualquer caminho, formado por transições válidas, que leve do estado inicial até q_i
- Estados inacessíveis não contribuem para o poder de reconhecimento do autômato, já que nenhuma cadeia $w \in \Sigma^*$ pode levar o autômato do estado inicial até esses estados.



Estados inacessíveis

- Algoritmo para eliminação de estados inacessíveis:

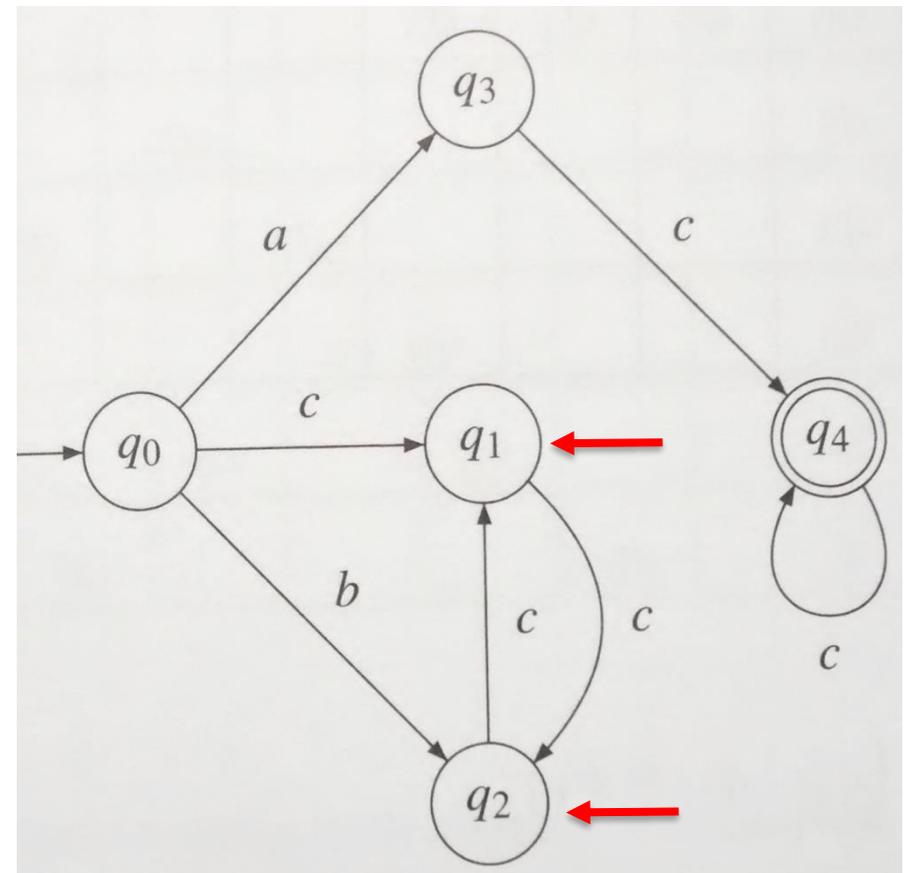
Duas marcas para cada estado: acessível e finalizado; o método inicia com todos os estados em branco

1. Marque o estado inicial como acessível (mas não como finalizado)
2. Enquanto houver um estado q_i marcado como acessível mas não finalizado:
 - Para cada estado q_j ainda não marcado, tal que haja uma transição $q_i \xrightarrow{a} q_j$, marque q_j como acessível
 - Quando todos os estados vizinhos de q_i tiverem sido inspecionados, marque q_i como finalizado.
3. Elimine todos os estados não marcados

Obs: Busca pode ser em largura ou em profundidade

Estados inúteis

- Um estado q_i é dito inútil se não existe no autômato qualquer caminho, formado por transições válidas, que leve de q_i até algum estado de aceitação
- Nenhuma cadeia $w \in \Sigma^*$ conduz o autômato de um estado inútil até um dos estados finais.



Estados inúteis

- Algoritmo para eliminação de estados inúteis:
Duas marcas para cada estado: útil e finalizado; o método inicia com todos os estados em branco
 1. Marque todos os estados de aceitação como estados úteis (mas não como finalizados)
 2. Enquanto houver um estado q_i marcado como útil mas não finalizado:
 - Para cada estado q_j ainda não marcado, tal que haja uma transição $q_j \xrightarrow{a} q_i$, marque q_j como útil
 - Quando todos os estados satisfazendo a condição (a) tiverem sido inspecionados, marque q_i como finalizado.
 3. Elimine todos os estados não marcados

Dica: usualmente, é mais eficiente criar um grafo transposto (invertendo as orientações das transições)

Equivalência entre estados

- Notação:
 - Dado um estado p qualquer do AFD e uma cadeia x , denota-se $(p, x) \vdash^* (q, \varepsilon)$ quando, partindo do estado p , o AFD parar sobre o estado q após consumir toda a cadeia x
- **Definição:** Considere um AFD $M = (Q, \Sigma, \delta, q_0, F)$ e dois estados $q_1, q_2 \in Q$. Diz-se que a cadeia $x \in \Sigma^*$ distingue q_1 de q_2 se $(q_1, x) \vdash^* (q_3, \varepsilon)$, $(q_2, x) \vdash^* (q_4, \varepsilon)$ e, de forma exclusiva, ou $q_3 \in F$ ou $q_4 \in F$
- Em outras palavras, uma cadeia x distingue q_1 de q_2 quando, para ser integralmente consumida a partir de cada um desses dois estados, ela conduzir o autômato a um estado final em apenas um desses casos

Equivalência entre estados

- **Definição:** Dois estados q_1, q_2 são ditos k -indistinguíveis (denotado por $q_1 \equiv^k q_2$) se e apenas se não houver cadeia x , $|x| \leq k$, que permita distinguir q_1 de q_2
- De acordo com a definição acima, para quaisquer pares de estados $q_i, q_j \in Q$, valem:
 - $q_i \equiv^0 q_j$ se e somente se ambos forem de aceitação ou nenhum for de aceitação:
$$q_i \equiv^0 q_j \Leftrightarrow (q_i, q_j \in F \vee q_i, q_j \in Q - F)$$
 - $q_i \equiv^k q_j$ se e somente se:
 - $q_i \equiv^{k-1} q_j$ e
 - $\forall a \in \Sigma, \delta(q_i, a) \equiv^{k-1} \delta(q_j, a)$

Equivalência entre estados

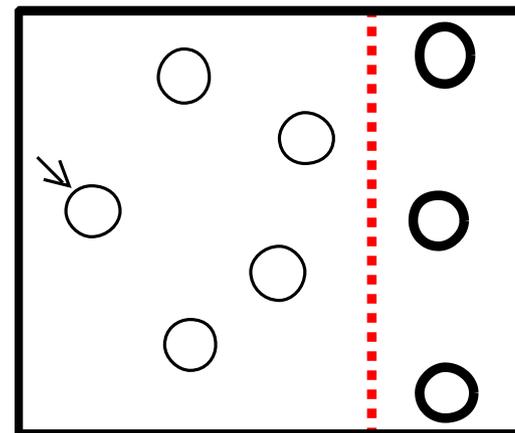
- **Definição:** Dois estados q_1, q_2 são ditos indistinguíveis ou equivalentes (denotado por $q_1 \equiv q_2$) se e apenas se eles forem k-indistinguíveis para todo $k \geq 0$
- A definição acima poderia sugerir que, para poder-se saber se dois estados são equivalentes, seria necessário verificar todas as cadeias de tamanho arbitrário \rightarrow Impossível!
- O teorema a seguir garante a existência de um método que, em um número finito de passos, identifica os estados equivalentes.

Equivalência entre estados

- **Teorema:** Seja $M = (Q, \Sigma, \delta, q_0, F)$ um AFD com n estados, e considere dois estados quaisquer q_1, q_2 de M . Então, $q_1 \equiv q_2$ se e somente se $q_1 \equiv^{n-2} q_2$.
 - Esse teorema afirma que, para se garantir a equivalência de dois estados em um AFD com n estados, é suficiente garantir sua $(n - 2)$ -indistinguibilidade, ou seja, não há necessidade de se considerar cadeias de comprimento maior que $n - 2$
- Ideia da demonstração:
 - $q_1 \equiv q_2 \Rightarrow q_1 \equiv^{n-2} q_2$: imediato, pois $q_1 \equiv^k q_2 \Rightarrow q_1 \equiv^{k-1} q_2$
 - $q_1 \equiv^{n-2} q_2 \Rightarrow q_1 \equiv q_2$:
 - Trivial se o autômato tiver estados que são somente de aceitação ou de rejeição: nesse caso, quaisquer pares de estados q_1, q_2 são indistinguíveis, pois ambos levam qualquer cadeia w simultaneamente para um estado de aceitação ou de rejeição (assumindo que a função de transição $\delta: Q \times \Sigma \rightarrow Q$ seja total)

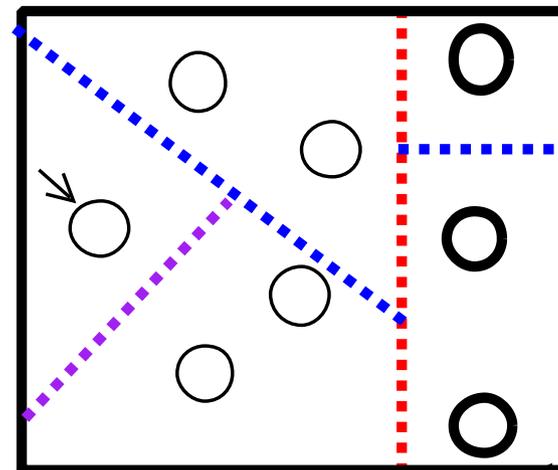
Equivalência entre estados

- Ideia da demonstração (cont):
 - $q_1 \equiv^{n-2} q_2 \Rightarrow q_1 \equiv q_2$ (cont):
 - Consideremos agora o caso mais geral em há tanto estados finais como não-finais em M .
 - De acordo com o critério \equiv^0 , o conjunto Q pode ser particionado inicialmente em dois grandes grupos:
 - O primeiro formado pelos estados finais (F)
 - O segundo pelos estados não finais ($Q - F$)
 - Trata-se, portanto, do primeiro de uma série de sucessivos refinamentos do o objetivo de determinar as classes de equivalências de estados de M .



Equivalência entre estados

- Ideia da demonstração (cont):
 - $q_1 \equiv^{n-2} q_2 \Rightarrow q_1 \equiv q_2$ (cont):
 - Executa-se, em seguida, para cada um dos dois subconjuntos obtidos através de \equiv^0 , seu refinamento (particionamento) através de relações \equiv^i , $i = 1, 2, 3, \text{etc.}$
 - Como M possui n estados (sendo alguns finais e outros não), o maior subconjunto de Q criado através de \equiv^0 possui no máximo $n - 1$ estados; portanto, haverá no máximo $n - 2$ refinamentos sucessivos de \equiv^0 gerando conjuntos de classes de equivalência, distintas umas das outras.



Equivalência entre estados

- Ideia da demonstração (cont):
 - $q_1 \equiv^{n-2} q_2 \Rightarrow q_1 \equiv q_2$ (cont):
 - Para completar a demonstração, basta provar que cada um dos $n - 2$ particionamentos distintos sucessivos (no máximo) refere-se ao uso correspondente de cadeias de comprimento $1, 2, \dots, n - 2$, para efetuar o teste de distinguibilidade do par de estados.
 - Consequentemente, não há possibilidade de ocorrer um novo particionamento distinto dos anteriores para cadeias de comprimento k se os particionamentos obtidos para cadeias de comprimento $k - 1$ e $k - 2$ se mostrarem idênticos.
 - Para provar essa afirmação, considere-se o conjunto de todas as classes de equivalência de M que satisfazem simultaneamente às relações \equiv^k e \equiv^{k+1} . Nesse caso, essas mesmas classes de equivalência satisfazem a \equiv^{k+2} , \equiv^{k+3} e assim sucessivamente.

Equivalência entre estados

- Ideia da demonstração (cont):

- $q_1 \equiv^{n-2} q_2 \Rightarrow q_1 \equiv q_2$ (cont):

- Considere-se, por exemplo, uma situação hipotética em que:

- a relação \equiv^k particiona um certo conjunto Q em três subconjuntos Q_0, Q_1 e Q_2 ;
- a relação \equiv^{k+1} preserva o particionamento da relação \equiv^k inalterado;
- a relação \equiv^{k+2} produz uma partição diferente, digamos Q_0, R, S e Q_2 , com $R \cup S = Q_1, R \cap S = \emptyset$:

$$\equiv^k : Q_0, Q_1, Q_2$$

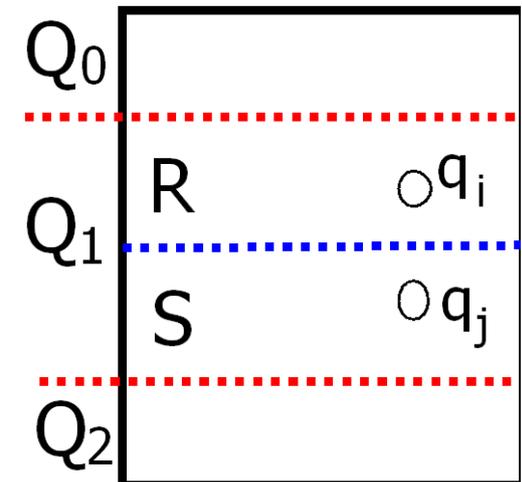
$$\equiv^{k+1} : Q_0, Q_1, Q_2$$

$$\equiv^{k+2} : Q_0, R, S, Q_2$$

- Admitindo-se, por hipótese, que Q_1 seja particionado em duas novas classes de equivalência R, S , isso significa que existem $q_1, q_2 \in Q_1$ tais que $q_1 \not\equiv^{k+2} q_2$. Mas para que isso fosse verdade, seria necessário, de acordo com a definição, que:

1) $q_1 \not\equiv^{k+1} q_2$, ou

2) $\delta(q_1, a) \not\equiv^{k+1} \delta(q_2, a)$ para algum $a \in \Sigma$.



Equivalência entre estados

- Ideia da demonstração (cont):
 - $q_1 \equiv^{n-2} q_2 \Rightarrow q_1 \equiv q_2$ (cont):
 - A condição (1) é falsa, pois de acordo com a hipótese original, $q_1, q_2 \in Q_1$ e portanto $q_1 \equiv^{k+1} q_2$.
 - A condição (2) também é falsa, pois se $q_1 \equiv^{k+1} q_2$ então $\delta(q_1, a) \equiv^k \delta(q_2, a)$; como, por hipótese, as partições produzidas pelas relações \equiv^k e \equiv^{k+1} são idênticas, então $\delta(q_1, a) \equiv^{k+1} \delta(q_2, a)$.
 - Fica, portanto, demonstrado que:
 - Na hipótese de serem obtidos dois conjuntos idênticos de classes de equivalência para k e $k + 1$, não haverá mais necessidade de se analisar a equivalência de tais classes para valores maiores do que k .
 - Para um autômato finito com n estados, haverá no máximo $n - 1$ conjuntos distintos de classes de equivalência (\equiv^0 e os demais $n - 2$), cada qual associado a cadeias de comprimento 0 até $n - 2$, não havendo, portanto, necessidade de se examinar a equivalência de tais classes para cadeias de comprimento superior a $n - 2$. \square

Etapas para Minimização de Estados

- Remoção dos estados inacessíveis e inúteis
- Identificação dos pares de estados equivalentes entre si;
- Agrupamentos dos estados em classes de equivalência, cada uma identificada pelo seu representante
- Criação de um novo AFD (mínimo)
 - Novos estados correspondem aos representantes das classes de equivalências do AFD original;
 - Novas transições são obtidas do AFD original, substituindo a referência aos estados originais pelos respectivos representantes

Identificação dos pares de estados equivalentes - algoritmo

- Entrada: um AFD $M = (Q, \Sigma, \delta, q_0, F)$
- Saída: Uma partição Q_0, Q_1, \dots, Q_K do conjunto Q de estados, de tal forma que seus elementos correspondem às mais amplas classes de equivalências de estados existentes em Q .
 1. Divide-se o conjunto original de estados de M nos dois subconjuntos que compõem sua partição inicial:
 - Subconjunto dos estados finais;
 - Subconjunto dos estados não finais.
 - Justificativa: Em um AFD, um estado final, qualquer que seja ele, é sempre distinguível de um estado não final (já que a cadeia vazia os distingue).
 2. Essa partição inicial corresponde, portanto, ao resultado da aplicação da relação \equiv^0 ao conjunto Q .

(Continua)

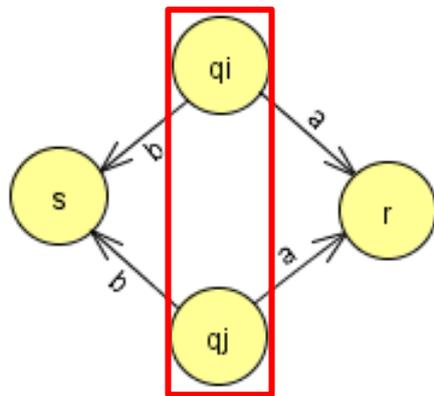
Identificação dos pares de estados equivalentes - algoritmo

(Continuação)

3. Para cada um dos subconjuntos obtidos em (1), refiná-los em novas partições, segundo o critério:
 - Dois estados q_i, q_j de um mesmo subconjunto Q_i , obtido de uma partição prévia do conjunto Q de estados, são equivalentes se e somente se:
 - q_i e q_j têm transições definidas sobre o mesmo conjunto de símbolos $S \subseteq \Sigma$, e
 - Para cada um desses símbolos $a \in S$:
 - $\delta(q_i, a) = \delta(q_j, a)$ ou
 - $\delta(q_i, a) \neq \delta(q_j, a)$ mas $\delta(q_i, a)$ e $\delta(q_j, a)$ são equivalentes.
 - Caso contrário, q_i e q_j não são equivalentes e devem, portanto, ensejar uma partição de Q_i .

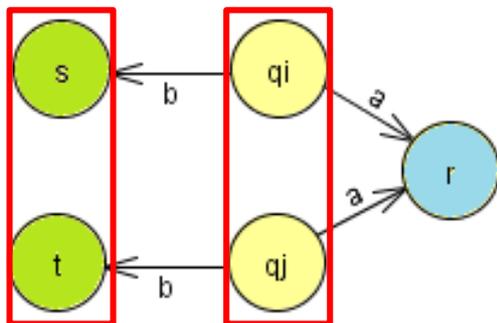
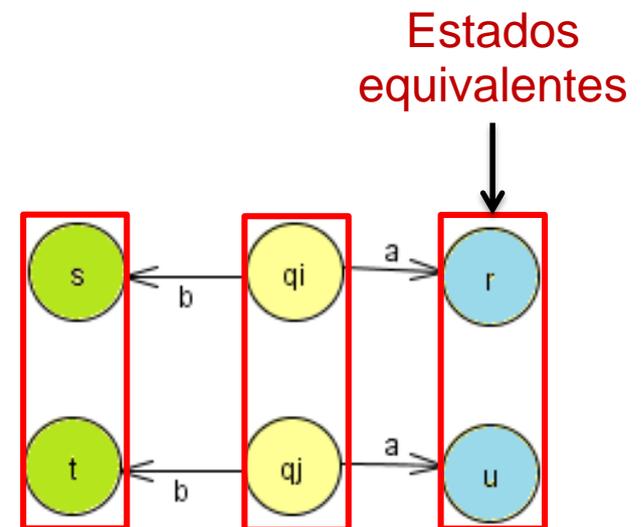
Identificação dos pares de estados equivalentes - algoritmo

- Representações dos casos que satisfazem à condição 2b do algoritmo:



Transições com as mesmas entradas para estados idênticos

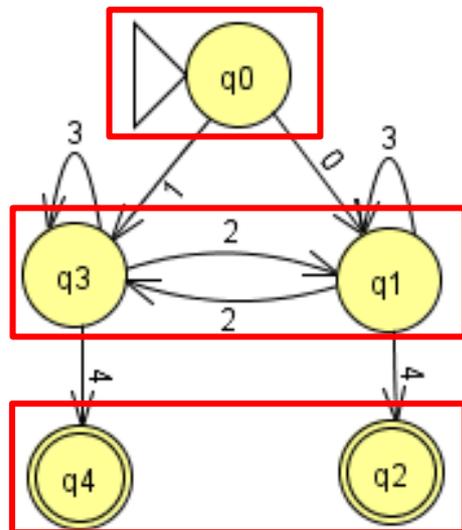
Transições com as mesmas entradas para estados equivalentes



Transições com as mesmas entradas para estados idênticos e equivalentes

Identificação dos pares de estados equivalentes - implementação

- Para a identificação de estados equivalentes, a estrutura de dados a ser preenchida é uma matriz binária $E[i, j]$, indicando se os estados distintos q_i e q_j são ou não equivalentes
- Exemplo: considere o AFD abaixo e respectiva matriz $E[i, j]$



	q0	q1	q2	q3	q4
q0		0	0	0	0
q1	0		0	1	0
q2	0	0		0	1
q3	0	1	0		0
q4	0	0	1	0	

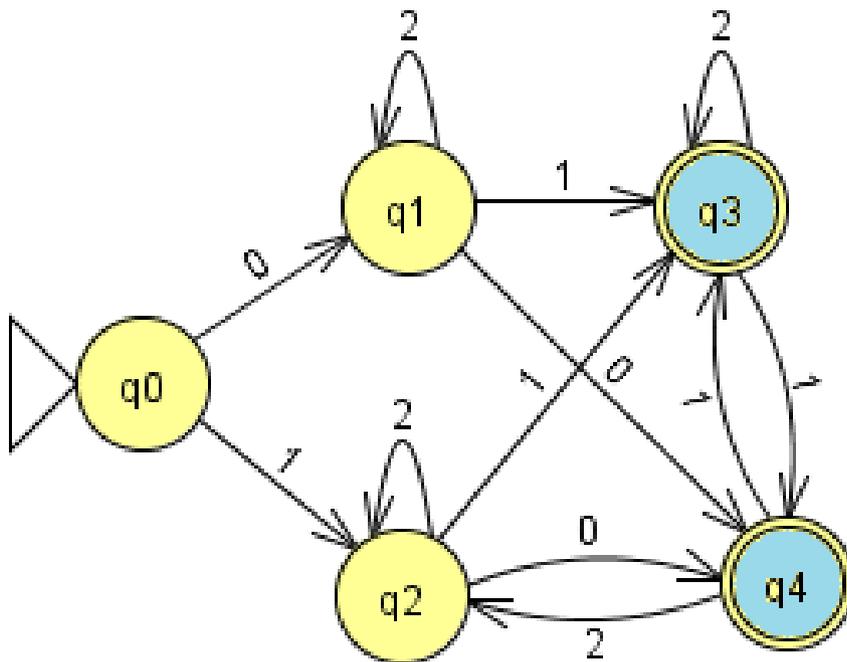
Estados equivalentes

Identificação dos pares de estados equivalentes - implementação

- $E[i, j]$ é atualizada iterativamente, como segue:
 1. Separação dos estados finais e não finais:
 - Para cada par de estados q_i, q_j , se ambos forem finais ou ambos forem não finais, defina $E[i, j] = E[j, i] = 1$; caso contrário, $E[i, j] = E[j, i] = 0$
 2. Marcação dos estados que não tenham transições sobre os mesmos símbolos:
 - Para cada par de estados q_i, q_j tais $E[i, j] = 1$, se houver algum símbolo $a \in \Sigma$ tal que $\delta(q_i, a)$ esteja definida mas $\delta(q_j, a)$ não esteja (ou vice-versa), então marque-os como não equivalentes:
 $E[i, j] = E[j, i] = 0$
 3. Repita o procedimento abaixo até que nenhuma nova posição de M seja modificada:
 - Para cada par de estados q_i, q_j tais $E[i, j] = 1$, se houver algum símbolo $a \in \Sigma$ tal que $\delta(q_i, a)$ e $\delta(q_j, a)$ não seja, equivalentes, então marque q_i, q_j como não equivalentes: $E[i, j] = E[j, i] = 0$

Identificação dos pares de estados equivalentes - implementação

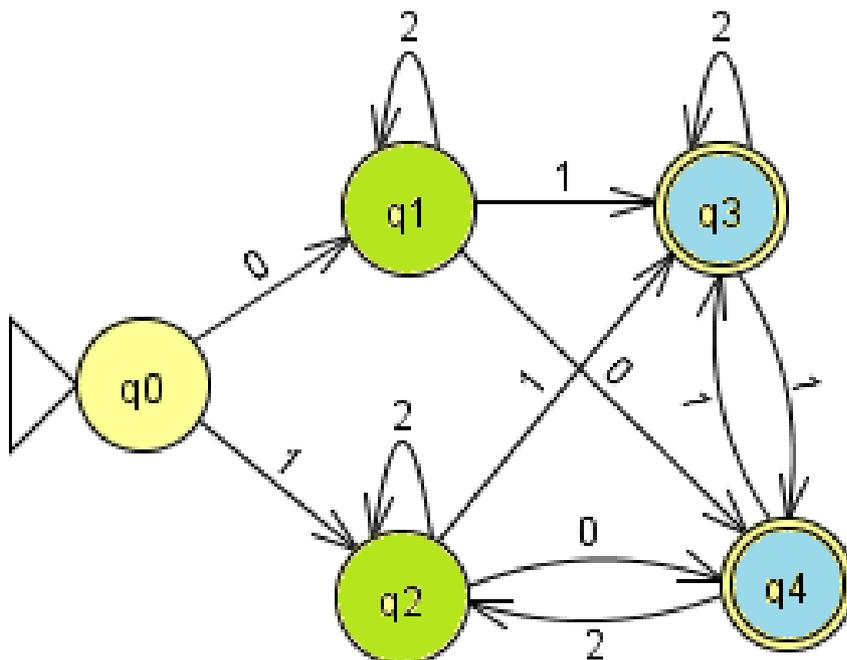
- Exemplo: considere o AFD abaixo e respectiva matriz $E[i, j]$
 - Separação dos estados finais e não finais



	q0	q1	q2	q3	q4
q0		1	1	0	0
q1	1		1	0	0
q2	1	1		0	0
q3	0	0	0		1
q4	0	0	0	1	

Identificação dos pares de estados equivalentes - implementação

- Exemplo: considere o AFD abaixo e respectiva matriz $E[i, j]$
- Marcação dos estados que não tenham transições sobre os mesmos símbolos

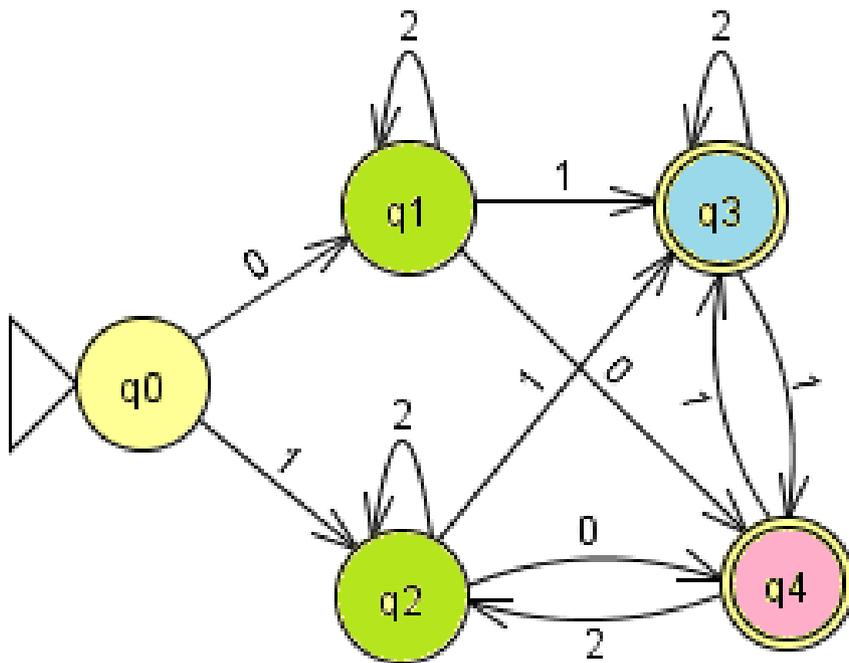


	q0	q1	q2	q3	q4
q0		0	0	0	0
q1	0		1	0	0
q2	0	1		0	0
q3	0	0	0		1
q4	0	0	0	1	

Estados q_1 e q_2 aceitam transições com o símbolo 2, enquanto q_0 não

Identificação dos pares de estados equivalentes - implementação

- Exemplo: considere o AFD abaixo e respectiva matriz $E[i, j]$
- Marcação dos estados que possuam transições não equivalentes



	q0	q1	q2	q3	q4
q0		0	0	0	0
q1	0		1	0	0
q2	0	1		0	0
q3	0	0	0		0
q4	0	0	0	0	

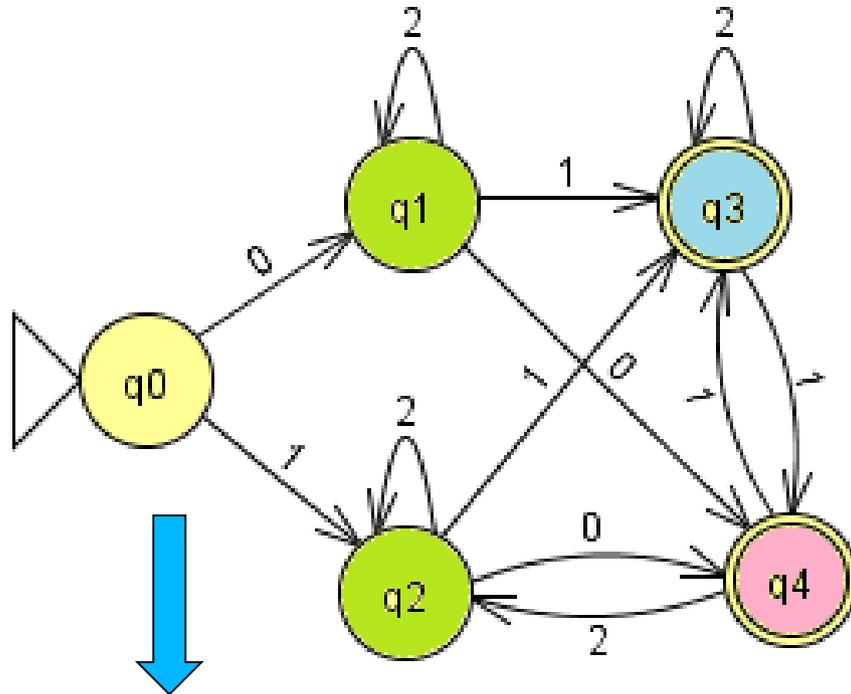
$\delta(q_3, 2)$ e $\delta(q_4, 2)$ não são equivalentes

Agrupamentos dos estados em classes de equivalência - implementação

- Identificação da classe de equivalência de cada estado do AFD é fornecida pelo vetor $\text{rep}[i]$:
 - $\text{rep}[i]$ = representante da classe de equivalência à qual o estado q_i pertence
- $\text{rep}[i]$ é atualizado como segue:
 - Inicialize $\text{rep}[i] = -1$, para todo q_i
 - Inicialize o contador $C = 0$
 - Enquanto houver algum estado q_i tal que $\text{rep}[i] = -1$, faça:
 - Incremente o contador: $C = C + 1$
 - $\text{rep}[i] = C - 1$
 - Para todos os estados q_j tais que $E[i, j] = 1$, atribua $\text{rep}[j] = \text{rep}[i]$

Identificação dos pares de estados equivalentes - implementação

- Exemplo: Identificação das classes de equivalência:



	q0	q1	q2	q3	q4
q0		0	0	0	0
q1	0		1	0	0
q2	0	1		0	0
q3	0	0	0		0
q4	0	0	0	0	



Classes de equivalência:

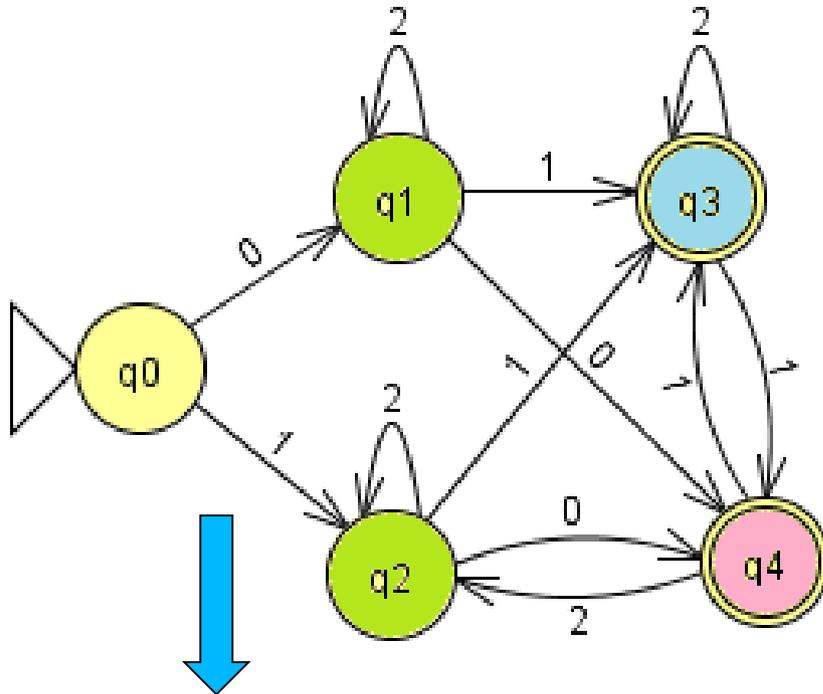
q	rep
q0	0
q1	1
q2	1
q3	2
q4	3

Criação do novo AFD mínimo - algoritmo

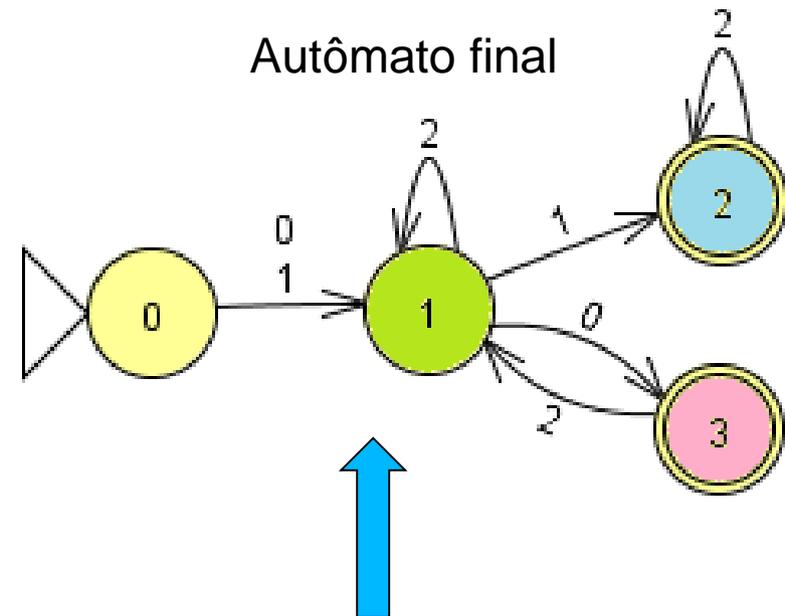
- Entrada: um AFD $M = (Q, \Sigma, \delta, q_0, F)$
- Saída: um AFD $M' = (Q', \Sigma, \delta', q_0', F')$ tal que $L(M') = L(M)$ e M' não contenha nenhum par de estados equivalentes
- Obter a matriz de estados equivalentes $E[i, j]$ e o vetor de representantes $rep[i]$, conforme descrito anteriormente.
- Construir M' tal que:
 - $Q' = \{0, 1, \dots, C - 1\}$: conjunto das classes de equivalência (cada qual representada por um índice)
 - δ' : Para cada transição no AFD original, $\delta(q_i, a)$, atribua $\delta'(rep[i], a) = rep[\delta(q, a)]$
 - $q_0' = rep[q_0]$, ou seja, o estado correspondente à classe de equivalência que contém $q_0 \in Q$
 - $F' = \{rep[q] \mid q \in F\}$, ou seja, todas as classes de equivalência contendo estados finais no AFD original

Criação do novo AFD mínimo - exemplo

- Exemplo: Criação do AFD final:



	q0	q1	q2	q3	q4
q0		0	0	0	0
q1	0		1	0	0
q2	0	1		0	0
q3	0	0	0		0
q4	0	0	0	0	



Classes de equivalência:

q	rep
q0	0
q1	1
q2	1
q3	2
q4	3

Bibliografia

- RAMOS, M. V. M.; NETO, J. J.; VEGA, I. S. Linguagens Formais – Teoria, Modelagem e Implementação. Ed. Bookman, 2009.