ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

1. Linguagens Regulares

Referência:
SIPSER, M. Introdução à Teoria da Computação.
2ª edição, Ed. Thomson

Prof. Marcelo S. Lauretto
marcelolauretto@usp.br www.each.usp.br/lauretto

Adaptação e complementação dos slides elaborados e gentilmente cedidos pela Profa. Ariane Machado Lima (EACH/USP)

Cap 1 – Linguagens regulares

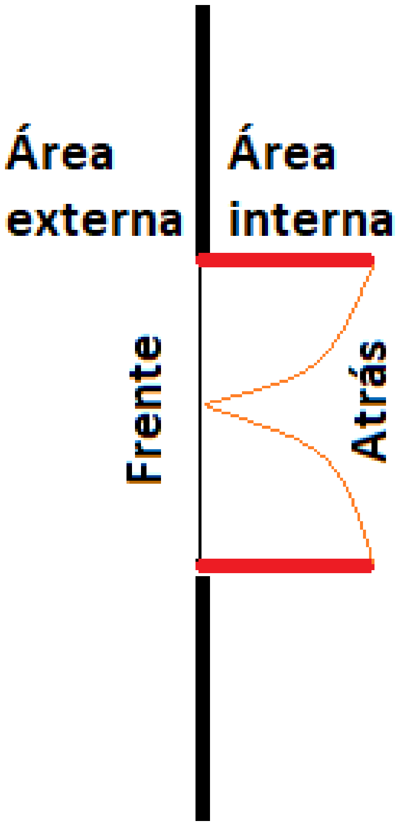
- Autômatos finitos
- Não determinismo
- Expressões regulares
- Gramáticas regulares
- Linguagens não-regulares

Autômatos finitos

- Necessidade de um modelo para entender (estudar) um computador
- Vários modelos computacionais com diferentes características (e complexidades)
- O modelo mais simples:
  - Máquina de estados finitos ou
  - Autômato de estados finitos ou
  - Autômato finito
  - Finite State Automaton (FSA)

Autômatos finitos

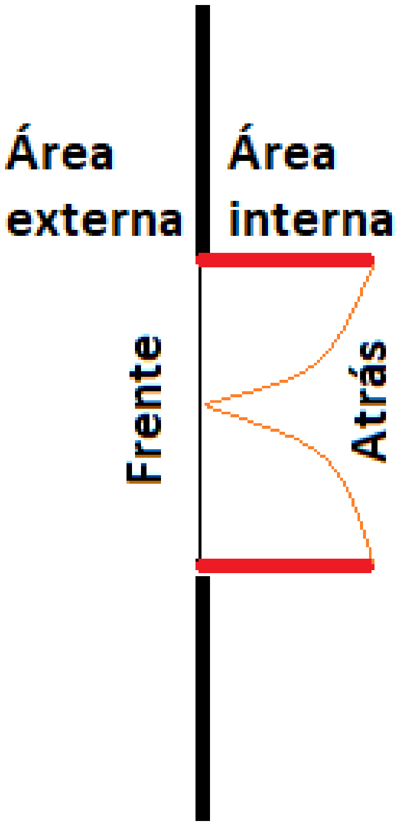
- O exemplo de um controlador de portas:
  - Estados:
    - Fechado / Aberto (Closed, Open)
  - Sinais de entrada nos sensores:
    - Frente, Atrás, Ambos, Nenhum (Front, Rear, Both, None)
  - Requisitos:
    - A porta deve abrir-se quando o sensor frontal detectar uma pessoa querendo entrar;
    - Quando a pessoa terminar de entrar, a porta deve fechar-se;
    - Por questão de segurança, não deve haver mudança de estado (Closed → Open ou Open → Closed) se houver uma pessoa atrás da porta.



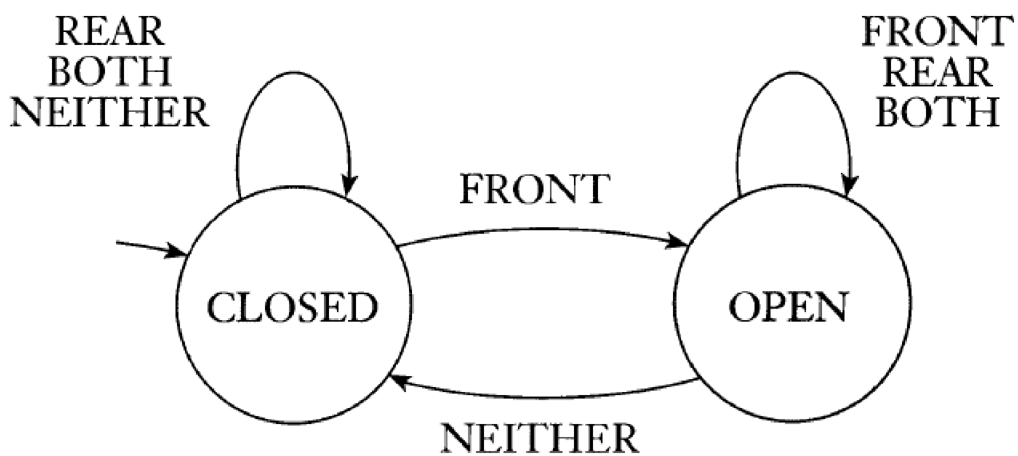
Autômatos finitos

- O exemplo de um controlador de portas:
  - Tabela de transição de estados:

	input signal			
state	NEITHER	FRONT	REAR	BOTH
CLOSED	CLOSED	OPEN	CLOSED	CLOSED
OPEN	CLOSED	OPEN	OPEN	OPEN



  - Diagrama de estados:



Autômatos finitos

- O exemplo de um controlador de portas:
  - Se considerarmos que o controlador inicia no estado Fechado, para cada série de sinais de entrada, a sequência de estados do controlador será única:
  - Ex: para a série

    F, R, N, F, B, N, R, N,

    o controlador passaria pela seguinte série de estados:

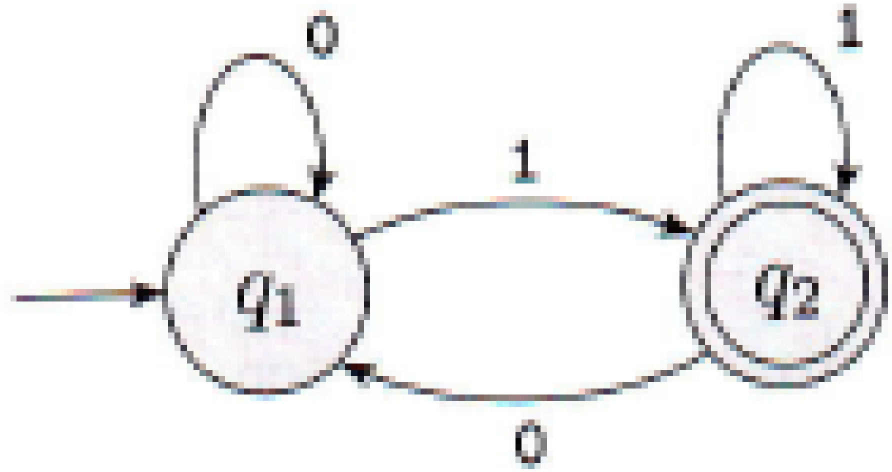
    C (inicial), O, O, C, O, O, C, C, C.

Autômatos finitos

- Autômatos finitos são mecanismos RECONHECEDORES
  - Ex: como seria o autômato para reconhecer strings binárias que começam e terminam com zero, podem ter 0s ou 1s no meio, com tamanho pelo menos 1?
    - 0, 00, 010, 000000, 0101110, ...

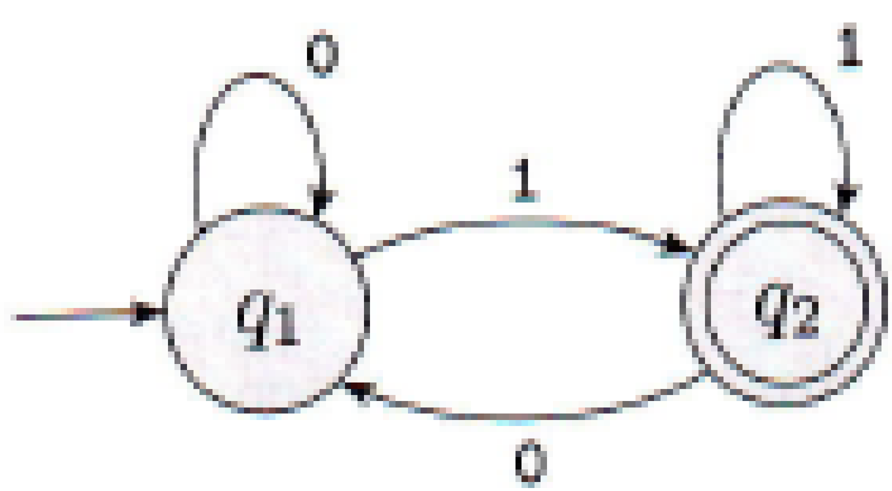
Autômatos finitos

- Diagrama de estados
  - Ex: o que esse autômato A3 reconhece?



Autômatos finitos

- Diagrama de estados
  - Ex: o que esse autômato A3 reconhece?

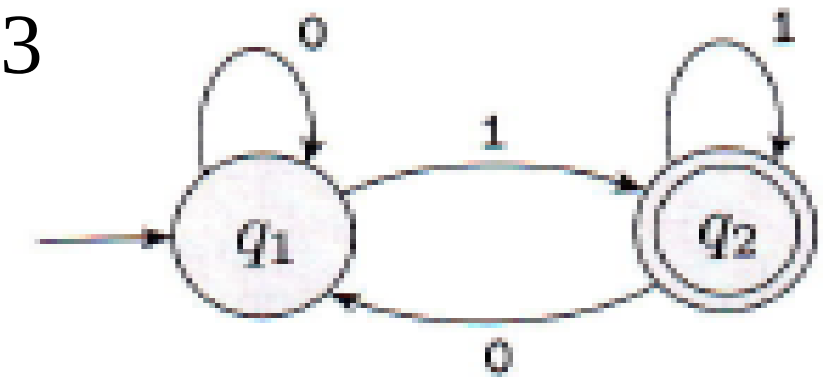


  - Resp: Sequências binárias que terminam em 1

Autômatos finitos

- A linguagem reconhecida por um autômato é o conjunto das cadeias (de símbolos de entrada) aceitas pelo autômato
- Ex:

A3



- L(A3) = {w | w é uma string binária e termina em 1}

Autômatos finitos

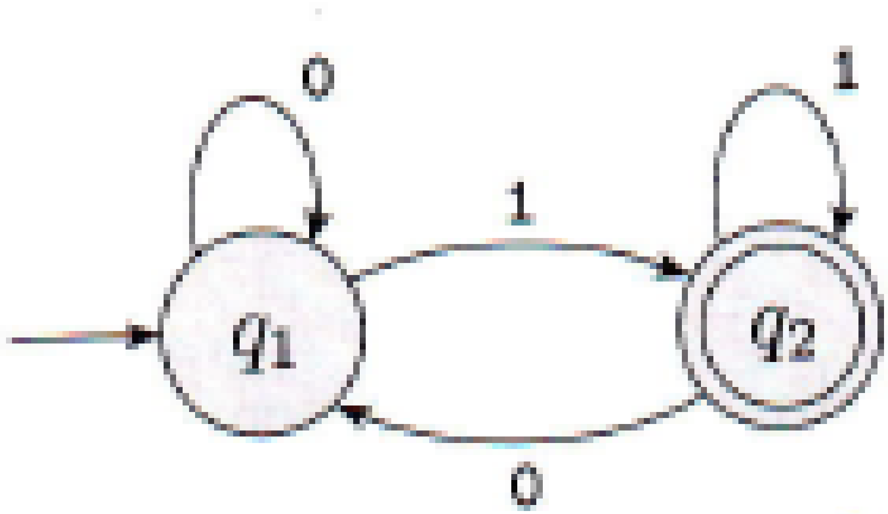
- Definição formal:

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]

Autômatos finitos

- Qual a definição formal do autômato A3?

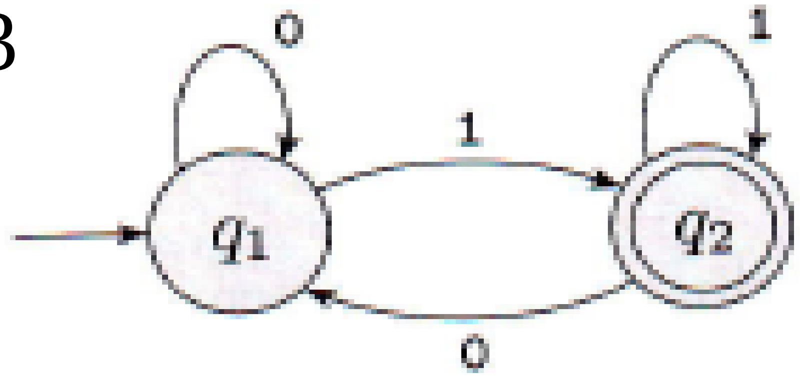


Um ***autômato finito*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito conhecido como os ***estados***,
2. Σ é um conjunto finito chamado o ***alfabeto***,
3. $\delta\colon Q \times \Sigma \longrightarrow Q$ 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]

Autômatos finitos

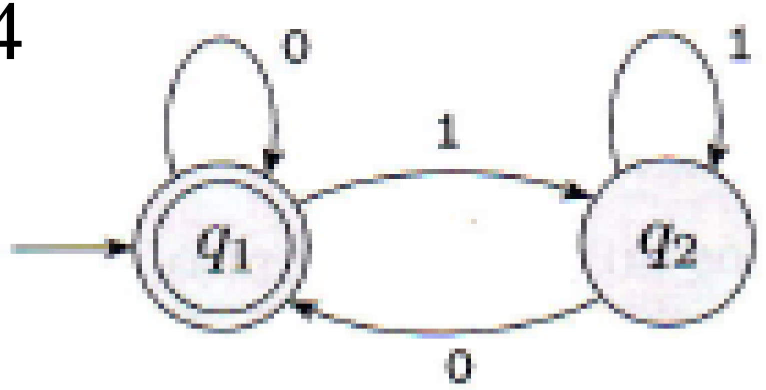
A3



L(A3) = {w | w é binária e termina com 1}

- Que linguagem o autômato abaixo reconhece? (Apenas mudou o estado final)

A4

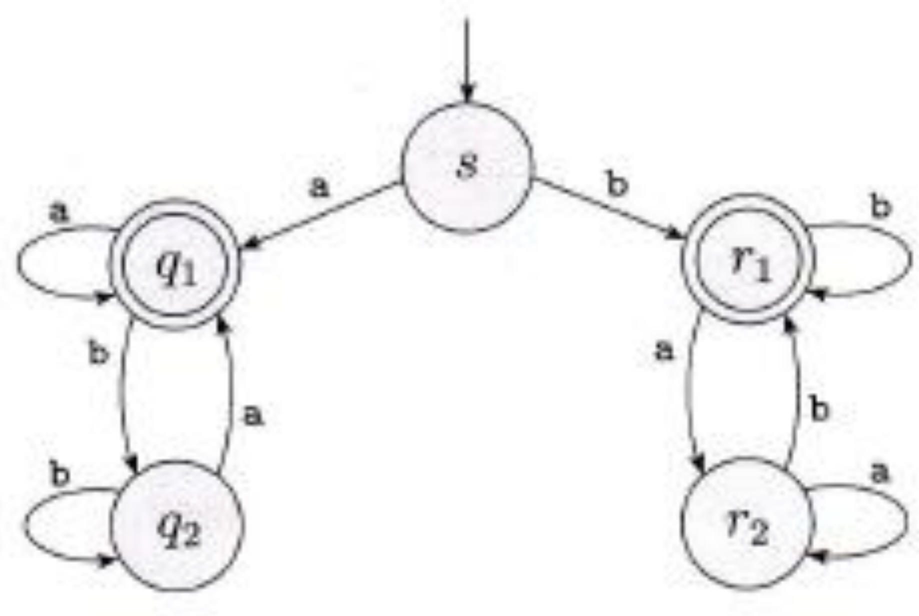


L(A4) = {ε} ∪ {w | w é binária e termina com 0}

$= \Sigma^* - L(A3)$

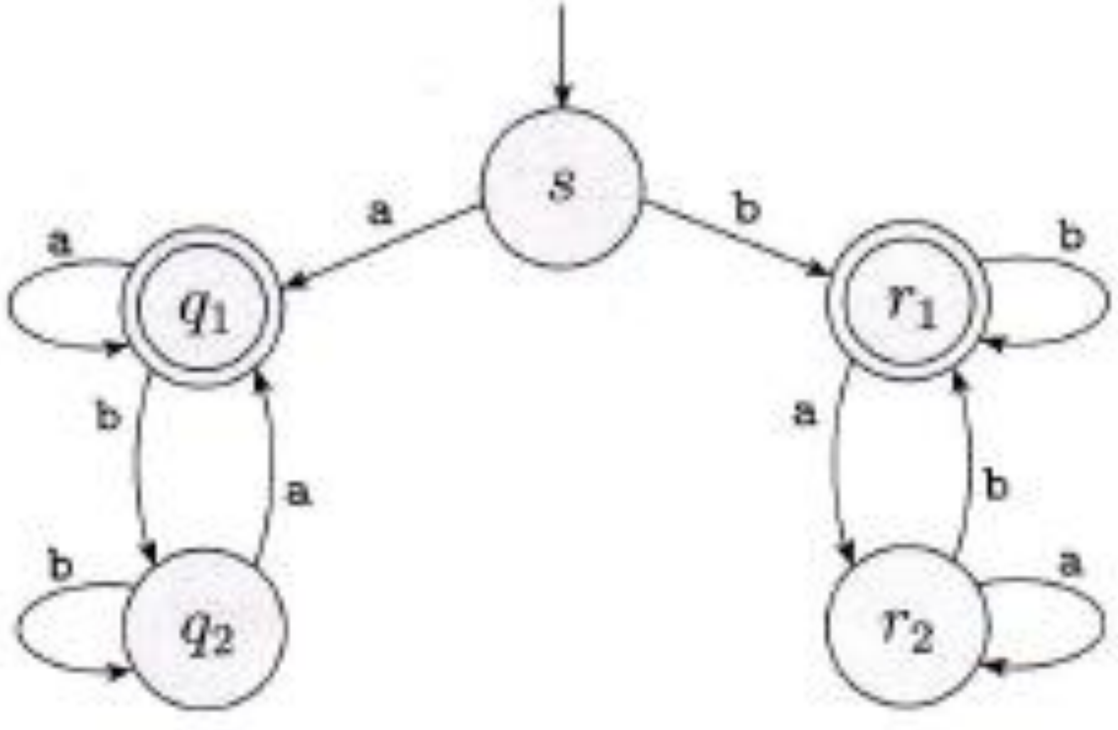
Autômatos finitos

- Que linguagem esse autômato reconhece?



Autômatos finitos

- Que linguagem esse autômato reconhece?



- Resp: Cadeias que comecem e terminem com o mesmo símbolo

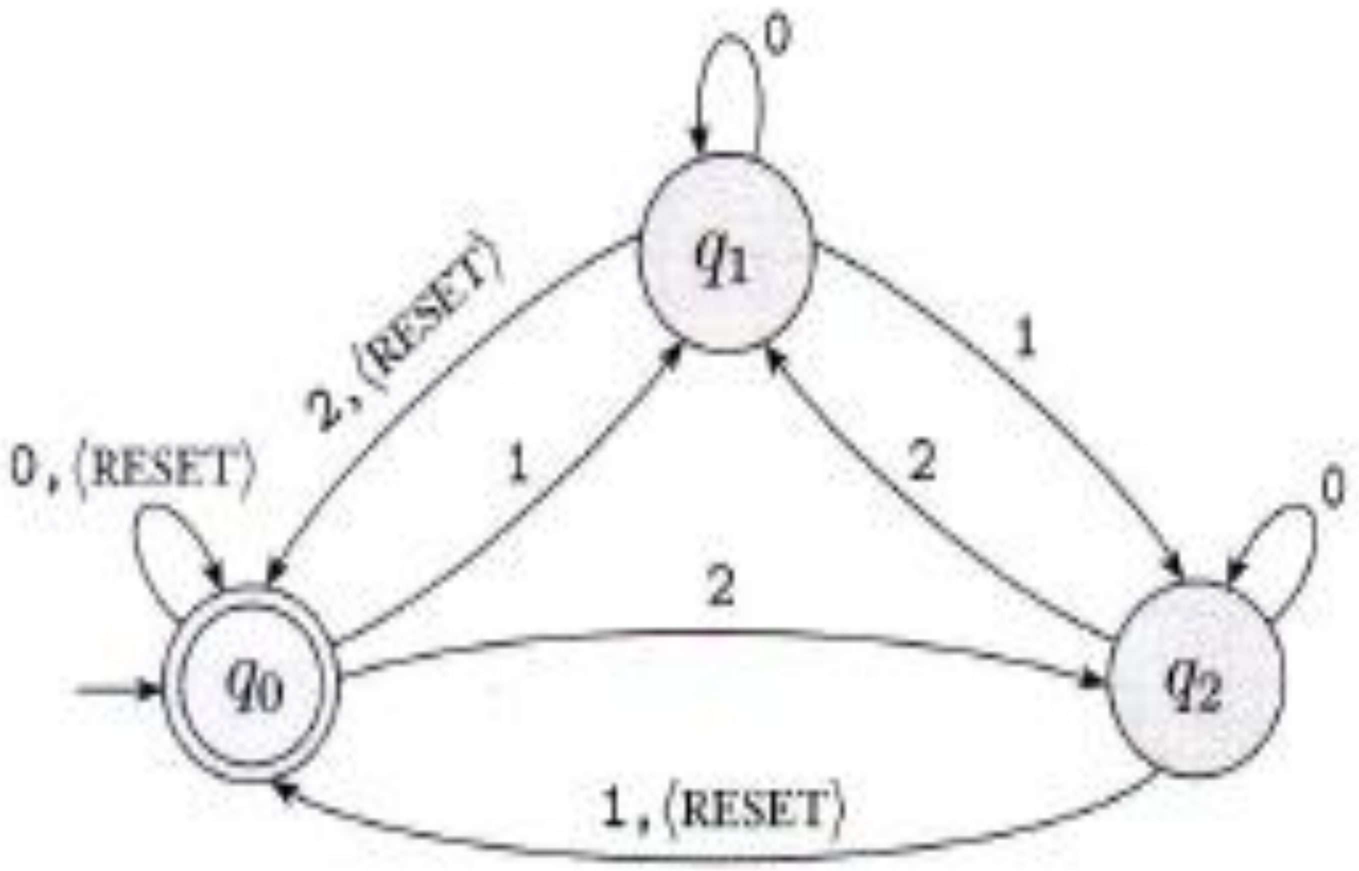
Projetando autômatos

- Pense que você é um autômato
- A cadeia de entrada pode ser arbitrariamente grande
- Sua memória é finita (o número de estados é finito)
- A transição se dá dados o estado atual e o próximo símbolo de entrada
- Você recebe um símbolo por vez, e não sabe quando a cadeia vai acabar (você precisa ter sempre uma “resposta corrente”)

Exercício

- Projete um autômato (diagrama de estados) que, dado Σ = {0,1,2,<RESET>}, aceita a cadeia de entrada se a soma dos números módulo 3 for igual a 0 (ou seja, se a soma for um múltiplo de 3).
  <RESET> zera o contador
  - $A_3 = \left\{ w_1 w_2 \dots w_n | w_j \in \{0,1,2\},\ \left(\sum_{j=1}^{n} w_j \right) \bmod 3 = 0 \right\}$

Exercício - solução



Autômatos finitos

- Pode ser mais conveniente projetar o autômato usando a definição formal ao invés do diagrama de estados
  - Ex: generalização do autômato anterior para aceitar somas múltiplas de i:
  - $A_i = \left\{w_1 w_2 \dots w_n | w_j \in \{0,1,\dots,i-1\},\ \left(\sum_{j=1}^{n} w_j\right) \bmod i = 0\right\}$

Autômatos finitos

- Pode ser mais conveniente projetar o autômato usando a definição formal ao invés do diagrama de estados
  - Ex: generalização do autômato anterior para aceitar somas múltiplas de i:
  - $A_i = \{w_1 w_2 \ldots w_n | w_j \in \{0,1,\ldots,i-1\},\ \left(\sum_{j=1}^{n} w_j\right) \bmod i = 0\}$

We describe the machine B_i formally as follows: $B_i = (Q_i, \Sigma, \delta_i, q_0, \{q_0\})$, where Q_i is the set of i states $\{q_0, q_1, q_2, \ldots, q_{i-1}\}$, and we design the transition function δ_i so that for each j, if B_i is in q_j, the running sum is j, modulo i. For each q_j let

$$\begin{aligned}
&\delta_i(q_j, 0) = q_j, \\
&\delta_i(q_j, 1) = q_k, \text{ where } k = j + 1 \text{ modulo } i, \\
&\delta_i(q_j, 2) = q_k, \text{ where } k = j + 2 \text{ modulo } i, \text{ and} \\
&\delta_i(q_j, \langle\text{RESET}\rangle) = q_0.
\end{aligned}$$

Definição formal de computação

Seja $M = (Q, \Sigma, \delta, q_0, F)$ um autômato finito e suponha que $w = w_1 w_2 \cdots w_n$ seja uma cadeia onde cada w_i é um membro do alfabeto Σ. Então M ***aceita*** w se existe uma seqüência de estados $r_0, r_1, \ldots, r_n$ em Q com três condições:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, para $i = 0, \ldots, n-1$, e
3. $r_n \in F$.

Linguagem Regular

- Uma linguagem é chamada linguagem regular se algum autômato finito a reconhece

- Vamos ver suas propriedades
  - Saber se uma linguagem é regular ou não para sabermos se podemos ou não implementar um autômato finito que a reconheça

Operações regulares

Sejam A e B linguagens. Definimos as operações regulares ***união***, ***concatenação*** e ***estrela*** da seguinte forma.

- **União**: $A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$.
- **Concatenação**: $A \circ B = \{xy \mid x \in A \text{ e } y \in B\}$.
- **Estrela**: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ e cada } x_i \in A\}$.

Operações regulares

Suponha que o alfabeto Σ seja o alfabeto padrão de 26 letras $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$. Se $A = \{\texttt{legal}, \texttt{ruim}\}$ e $B = \{\texttt{garoto}, \texttt{garota}\}$, então

Operações regulares

Suponha que o alfabeto Σ seja o alfabeto padrão de 26 letras $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$. Se $A = \{\mathtt{legal}, \mathtt{ruim}\}$ e $B = \{\mathtt{garoto}, \mathtt{garota}\}$, então

$$A \cup B =$$

Operações regulares

Suponha que o alfabeto Σ seja o alfabeto padrão de 26 letras $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$. Se $A = \{\mathtt{legal}, \mathtt{ruim}\}$ e $B = \{\mathtt{garoto}, \mathtt{garota}\}$, então

$$A \cup B = \{\mathtt{legal}, \mathtt{ruim}, \mathtt{garoto}, \mathtt{garota}\}$$

Operações regulares

Suponha que o alfabeto Σ seja o alfabeto padrão de 26 letras $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$. Se $A = \{\mathtt{legal}, \mathtt{ruim}\}$ e $B = \{\mathtt{garoto}, \mathtt{garota}\}$, então

$$A \cup B = \{\mathtt{legal}, \mathtt{ruim}, \mathtt{garoto}, \mathtt{garota}\}$$

$$A \circ B =$$

Operações regulares

Suponha que o alfabeto Σ seja o alfabeto padrão de 26 letras $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$. Se $A = \{\texttt{legal}, \texttt{ruim}\}$ e $B = \{\texttt{garoto}, \texttt{garota}\}$, então

$$A \cup B = \{\texttt{legal}, \texttt{ruim}, \texttt{garoto}, \texttt{garota}\}$$

$$A \circ B = \{\texttt{legalgaroto}, \texttt{legalgarota}, \texttt{ruimgaroto}, \texttt{ruimgarota}\}$$

Operações regulares

Suponha que o alfabeto Σ seja o alfabeto padrão de 26 letras $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$. Se $A = \{\mathtt{legal}, \mathtt{ruim}\}$ e $B = \{\mathtt{garoto}, \mathtt{garota}\}$, então

$$A \cup B = \{\mathtt{legal}, \mathtt{ruim}, \mathtt{garoto}, \mathtt{garota}\}$$

$$A \circ B = \{\mathtt{legalgaroto}, \mathtt{legalgarota}, \mathtt{ruimgaroto}, \mathtt{ruimgarota}\}$$

$$A^* =$$

Operações regulares

Suponha que o alfabeto Σ seja o alfabeto padrão de 26 letras $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$. Se $A = \{\texttt{legal}, \texttt{ruim}\}$ e $B = \{\texttt{garoto}, \texttt{garota}\}$, então

$$A \cup B = \{\texttt{legal}, \texttt{ruim}, \texttt{garoto}, \texttt{garota}\}$$

$$A \circ B = \{\texttt{legalgaroto}, \texttt{legalgarota}, \texttt{ruimgaroto}, \texttt{ruimgarota}\}$$

$$\begin{aligned} A^* = \{&\varepsilon, \texttt{legal}, \texttt{ruim}, \texttt{legallegal}, \texttt{legalruim}, \texttt{ruimlegal}, \texttt{ruimruim}, \\ &\texttt{legallegallegal}, \texttt{legallegalruim}, \texttt{legalruimlegal}, \\ &\texttt{legalruimruim}, \ldots\}. \end{aligned}$$

Fechamento sob união

TEOREMA 1.25

A classe de linguagens regulares é fechada sob a operação de união.

Em outras palavras, se A_1 e A_2 são linguagens regulares, o mesmo acontece com $A_1 \cup A_2$.

Fechamento sob união

Prova:

- sugestões?

Fechamento sob união

- Prova:
  - sugestões?
  - construímos um autômato M que simule ao mesmo tempo M1 e M2

Fechamento sob união - Prova

Suponha que M_1 reconheça A_1, onde $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, e que M_2 reconheça A_2, onde $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construa M para reconhecer $A_1 \cup A_2$, onde $M = (Q, \Sigma, \delta, q_0, F)$.

Fechamento sob união - Prova

Suponha que M_1 reconheça A_1, onde $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, e que
M_2 reconheça A_2, onde $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construa M para reconhecer $A_1 \cup A_2$, onde $M = (Q, \Sigma, \delta, q_0, F)$.

1. $Q = \{(r_1, r_2) | \; r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$.
Esse conjunto é o ***produto cartesiano*** dos conjuntos Q_1 e Q_2 e é escrito $Q_1 \times Q_2$. Trata-se do conjunto de todos os pares de estados, sendo o primeiro de Q_1 e o segundo de Q_2.

Fechamento sob união - Prova

Suponha que M_1 reconheça A_1, onde $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, e que
M_2 reconheça A_2, onde $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construa M para reconhecer $A_1 \cup A_2$, onde $M = (Q, \Sigma, \delta, q_0, F)$.

1. $Q = \{(r_1, r_2) | \; r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$.
Esse conjunto é o ***produto cartesiano*** dos conjuntos Q_1 e Q_2 e é escrito $Q_1 \times Q_2$. Trata-se do conjunto de todos os pares de estados, sendo o primeiro de Q_1 e o segundo de Q_2.

2. Σ, o alfabeto, é o mesmo em M_1 e M_2. Neste teorema e em todos os teoremas similares subseqüentes, assumimos por simplicidade que ambas M_1 e M_2 têm o mesmo alfabeto de entrada Σ. O teorema permanece verdadeiro se elas tiverem alfabetos diferentes, Σ_1 e Σ_2. Aí então modificaríamos a prova para tornar $\Sigma = \Sigma_1 \cup \Sigma_2$.

Fechamento sob união - Prova

3. δ, a função de transição, é definida da seguinte maneira. Para cada $(r_1, r_2) \in Q$ e cada $a \in \Sigma$, faça

$$\delta\big((r_1, r_2), a\big) = \big(\delta_1(r_1, a), \delta_2(r_2, a)\big).$$

Logo, δ obtém um estado de M (que na realidade é um par de estados de M_1 e M_2), juntamente com um símbolo de entrada, e retorna o próximo estado de M.

Fechamento sob união - Prova

3. δ, a função de transição, é definida da seguinte maneira. Para cada $(r_1, r_2) \in Q$ e cada $a \in \Sigma$, faça

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)).$$

Logo, δ obtém um estado de M (que na realidade é um par de estados de M_1 e M_2), juntamente com um símbolo de entrada, e retorna o próximo estado de M.

4. q_0 é o par (q_1, q_2).

Fechamento sob união - Prova

3. δ, a função de transição, é definida da seguinte maneira. Para cada $(r_1, r_2) \in Q$ e cada $a \in \Sigma$, faça

$$\delta\big((r_1, r_2), a\big) = \big(\delta_1(r_1, a), \delta_2(r_2, a)\big).$$

Logo, δ obtém um estado de M (que na realidade é um par de estados de M_1 e M_2), juntamente com um símbolo de entrada, e retorna o próximo estado de M.

4. q_0 é o par (q_1, q_2).

5. F é o conjunto de pares nos quais um dos membros é um estado de aceitação de M_1 ou M_2. Podemos escrevê-lo como

$$F = \{(r_1, r_2) | \, r_1 \in F_1 \text{ ou } r_2 \in F_2\}.$$

Essa expressão é a mesma que $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

Fechamento sob união - Prova

3. δ, a função de transição, é definida da seguinte maneira. Para cada $(r_1, r_2) \in Q$ e cada $a \in \Sigma$, faça

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)).$$

Logo, δ obtém um estado de M (que na realidade é um par de estados de M_1 e M_2), juntamente com um símbolo de entrada, e retorna o próximo estado de M.

E se fosse "e"?

4. q_0 é o par (q_1, q_2).

5. F é o conjunto de pares nos quais um dos membros é um estado de aceitação de M_1 ou M_2. Podemos escrevê-lo como

$$F = \{(r_1, r_2) | \, r_1 \in F_1 \text{ ou } r_2 \in F_2\}.$$

Essa expressão é a mesma que $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

Fechamento sob união - Prova

3. δ, a função de transição, é definida da seguinte maneira. Para cada $(r_1, r_2) \in Q$ e cada $a \in \Sigma$, faça

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)).$$

Logo, δ obtém um estado de M (que na realidade é um par de estados de M_1 e M_2), juntamente com um símbolo de entrada, e retorna o próximo estado de M.

E se fosse "e"?
Intersecção!

4. q_0 é o par (q_1, q_2).

5. F é o conjunto de pares nos quais um dos membros é um estado de aceitação de M_1 ou M_2. Podemos escrevê-lo como

$$F = \{(r_1, r_2) | \ r_1 \in F_1 \text{ ou } r_2 \in F_2\}.$$

Essa expressão é a mesma que $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

Fechamento sob concatenação

- O que acham?

Fechamento sob concatenação

- O que acham?

TEOREMA 1.26

A classe de linguagens regulares é fechada sob a operação de concatenação.

Em outras palavras, se A_1 e A_2 são linguagens regulares, então o mesmo acontece com $A_1 \circ A_2$.

Fechamento sob concatenação

- O que acham?

TEOREMA 1.26

A classe de linguagens regulares é fechada sob a operação de concatenação.

Em outras palavras, se A_1 e A_2 são linguagens regulares, então o mesmo acontece com $A_1 \circ A_2$.

- Prova?

Fechamento sob concatenação

- O que acham?

TEOREMA 1.26 ....

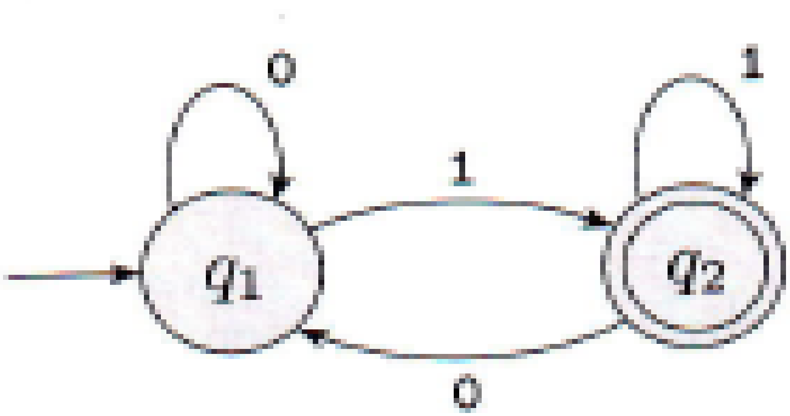
A classe de linguagens regulares é fechada sob a operação de concatenação.

Em outras palavras, se A_1 e A_2 são linguagens regulares, então o mesmo acontece com $A_1 \circ A_2$.

- Prova? Precisamos do conceito de não-determinismo

Autômatos Finitos Determinísticos (AFD)

- Dado um estado atual e um símbolo de entrada sabemos exatamente para onde ir (está determinado)



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***,

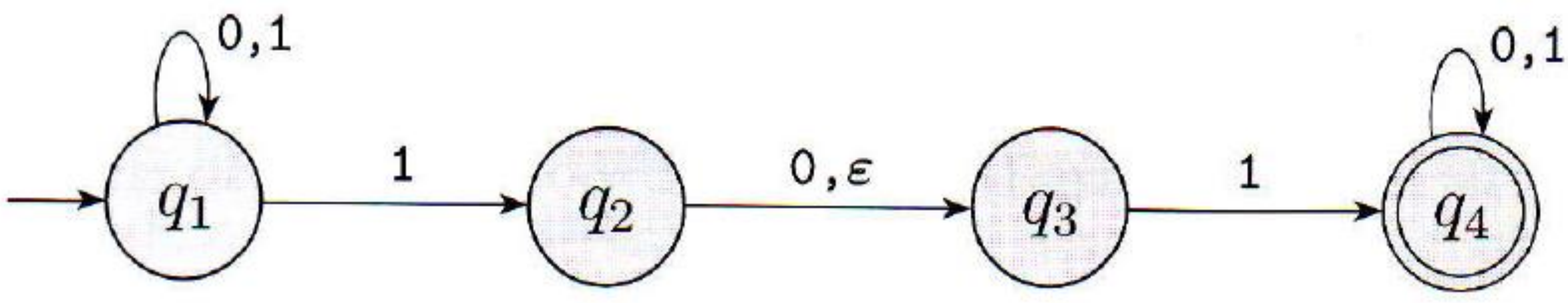
3. $\delta \colon Q \times \Sigma \longrightarrow Q$ é a ***função de transição***,[1]

4. $q_0 \in Q$ é o ***estado inicial***, e

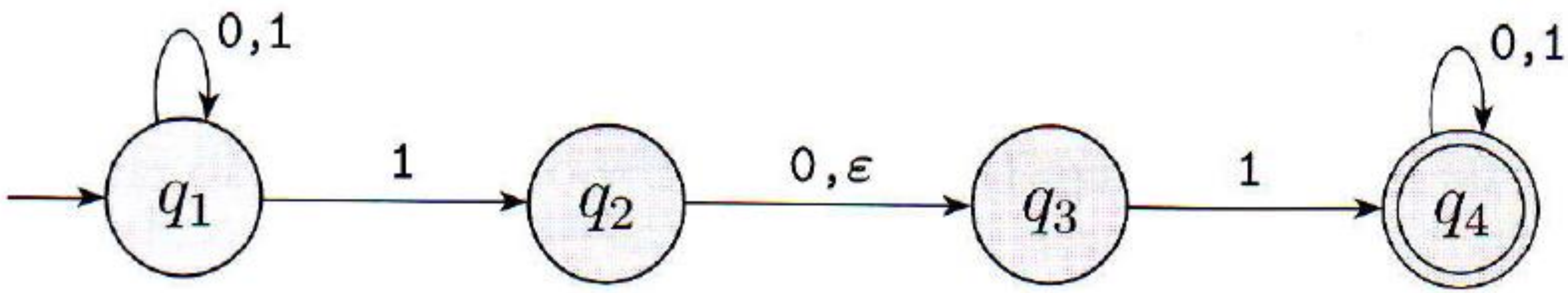
5. $F \subseteq Q$ é o ***conjunto de estados de aceitação***.[2]

Autômatos Finitos Não Determinísticos (AFN)

- Um estado pode ter 0 ou mais transições (setas saindo) para cada símbolo de Σ
- Um estado pode ter setas rotuladas por ε



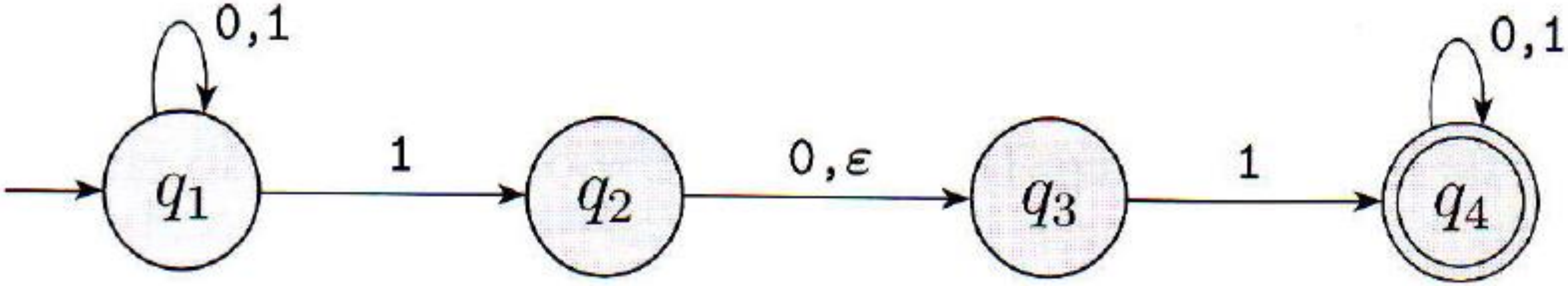
Autômatos Finitos
Não Determinísticos (AFN)



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. $\delta\colon Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ 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.

Autômatos Finitos Não Determinísticos (AFN)



1. $Q = \{q_1, q_2, q_3, q_4\}$,

2. $\Sigma = \{0,1\}$,

3. δ é dado como

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

,

4. q_1 é o estado inicial, e

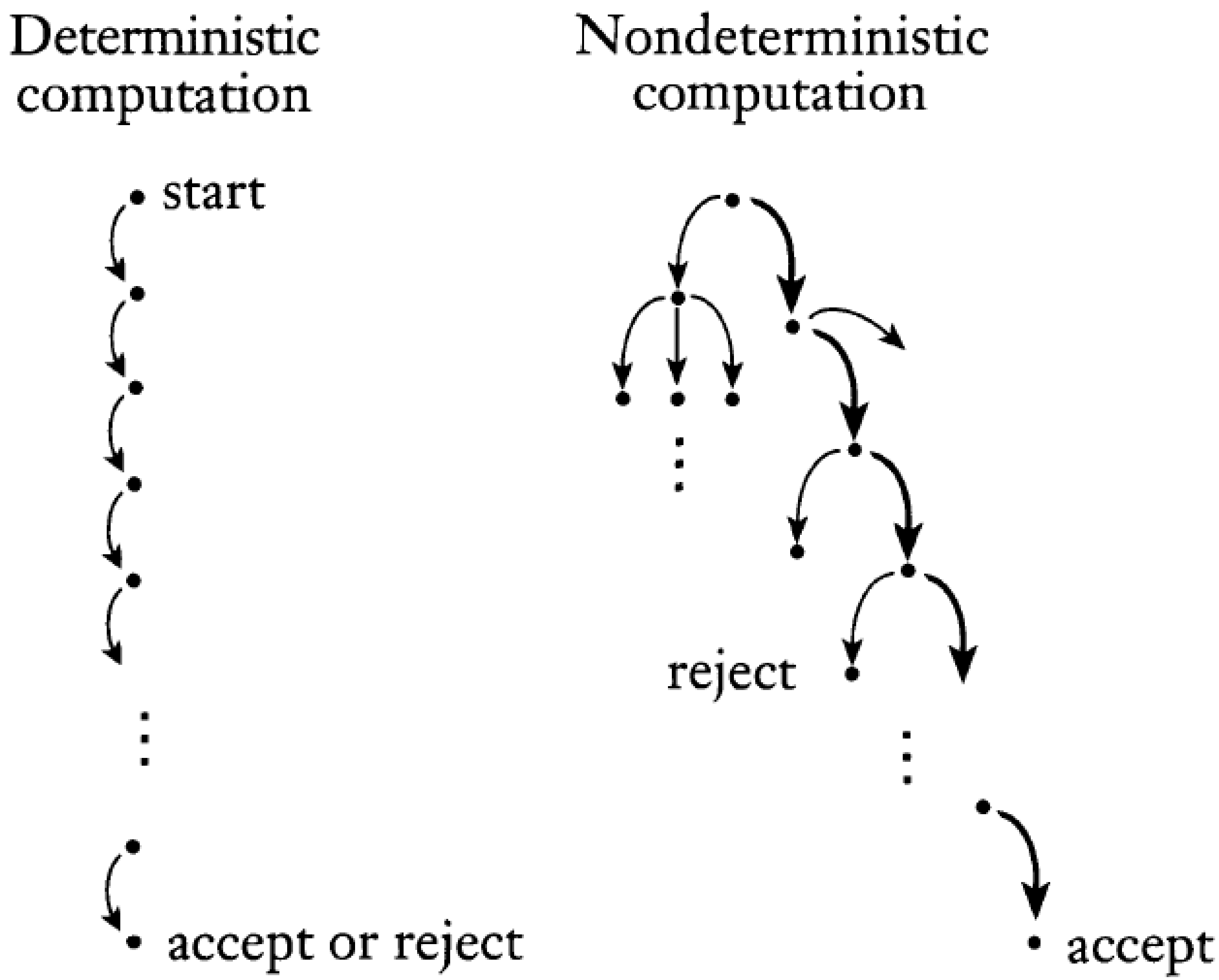
5. $F = \{q_4\}$.

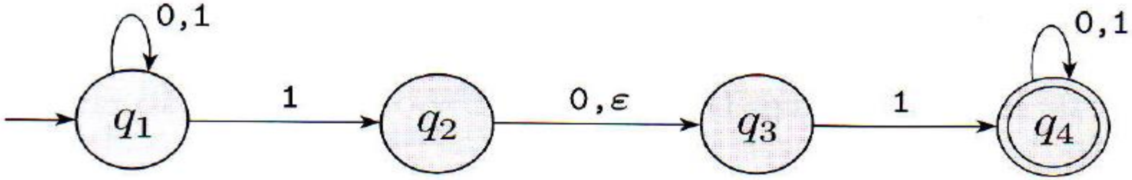
Funcionamento de um AFN

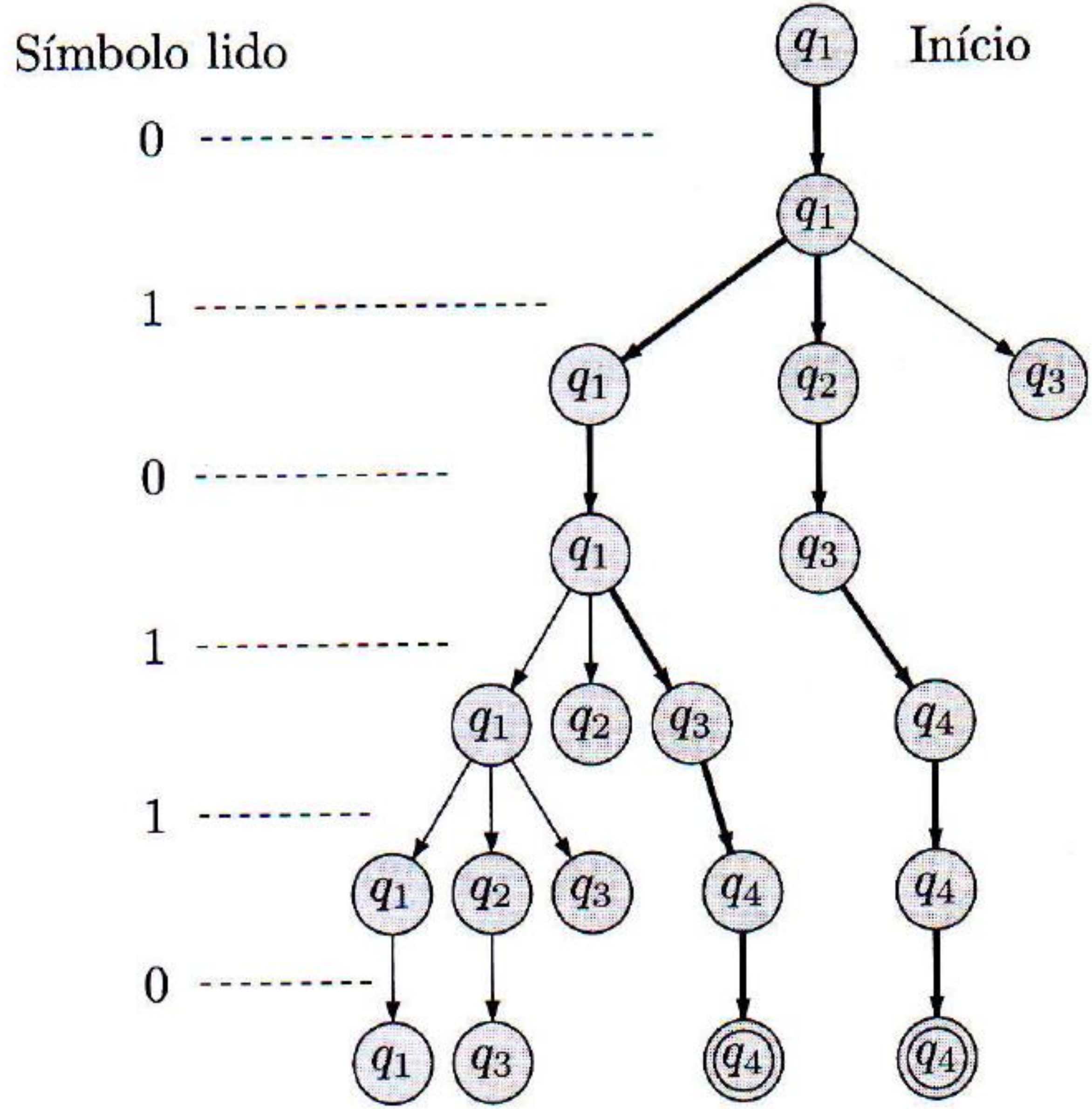
- Sempre que o autômato se depara com um não-determinismo (símbolo repetido ou ε) faz uma cópia de si e cada cópia segue com uma alternativa, em paralelo.
- Se uma cópia aceitar a cadeia, então o AFN aceita a cadeia

Funcionamento de um AFN

- Diferença entre computações determinísticas e não-determinísticas:




q_1
q_2
q_3
q_4
0,1
1
0, ε
1
0,1


Símbolo lido
Início
0
1
0
1
1
0
q_1
q_2
q_3
q_4

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.

Equivalência entre AFDs e AFNs

Prova: um estado para cada subconjunto

Primeiro vamos desconsiderar setas ε

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

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a).$$ [4]

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.

- Agora considerando setas ε:

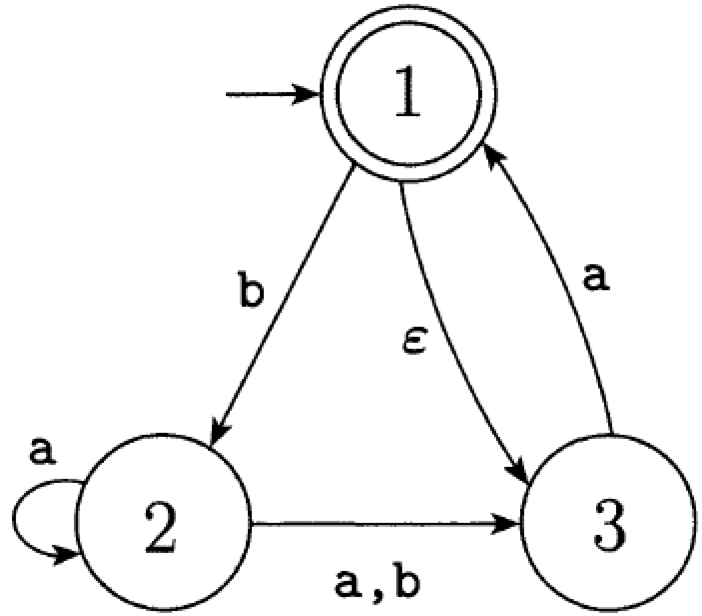
$$E(R) = \{q|\ q \text{ pode ser atingido a partir de } R \text{ viajando-se ao longo de 0 ou mais setas } \varepsilon\}.$$

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

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

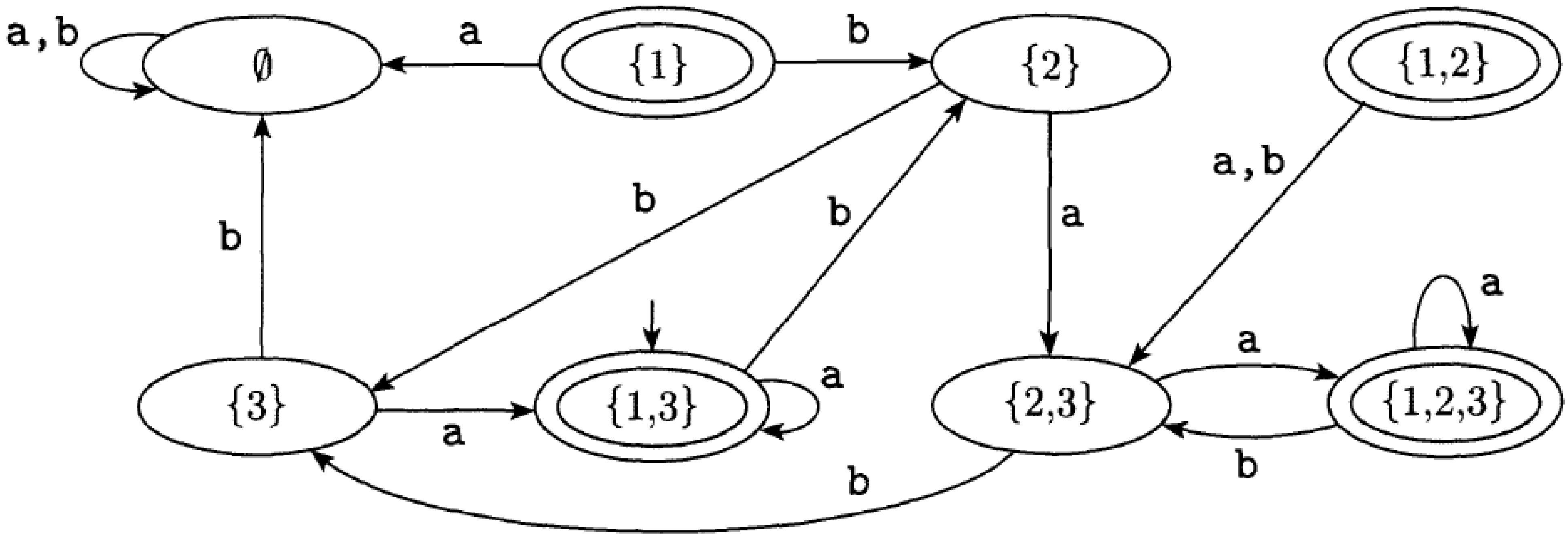
Equivalência entre AFDs e AFNs

- Exemplo:
  Converter o AFN abaixo em um AFD equivalente



Equivalência entre AFDs e AFNs

- Exemplo (cont):
  AFD resultante:



  – Note que o estado {1,2} é inacessível.

Equivalência entre AFDs e AFNs

- **Corolário 1.40:**
  Uma linguagem é regular se e somente se algum autômato finito não-determinístico a reconhece.

AFDs e AFNs

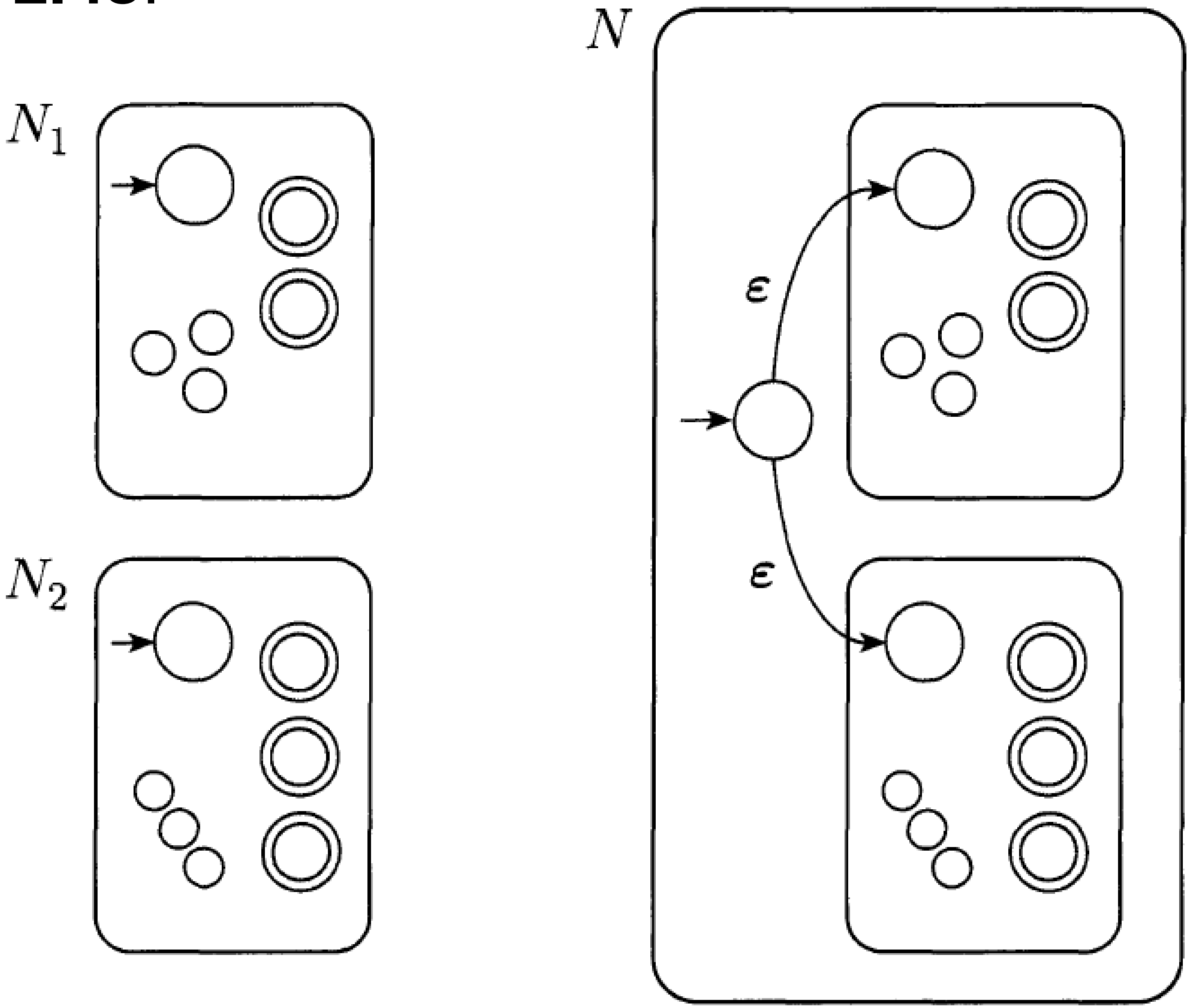
- Por que o teorema de equivalência é importante?
  - Pode-se optar por um outro dependendo do objetivo
    - AFDs são mais eficientes
    - AFNs podem:
      - ser mais fáceis de serem projetados
      - facilitar demonstração de teoremas
      - ser úteis em versões probabilísticas

Fechamento sob operações regulares

- **Teorema 1.45**:
  A classe de linguagens regulares é fechada sob a operação união.
  - Ideia da prova: Suponha que temos duas linguagens regulares A_1 e A_2. Queremos provar que $A_1 \cup A_2$ é regular. A ideia é tomar os dois AFNs, N_1 para A_1 e N_2 para A_2, e combiná-los em um novo AFN N que reconhece $A_1 \cup A_2$.

Fechamento sob operações regulares

- **Teorema 1.45**:



Fechamento sob operações regulares

- **Teorema 1.45**:
  – Dem:

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.
   The states of N are all the states of N_1 and N_2, with the addition of a new start state q_0.
2. The state q_0 is the start state of N.
3. The accept states $F = F_1 \cup F_2$.
   The accept states of N are all the accept states of N_1 and N_2. That way N accepts if either N_1 accepts or N_2 accepts.

Fechamento sob operações regulares

- **Teorema 1.45**:
  - Dem (cont):

4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

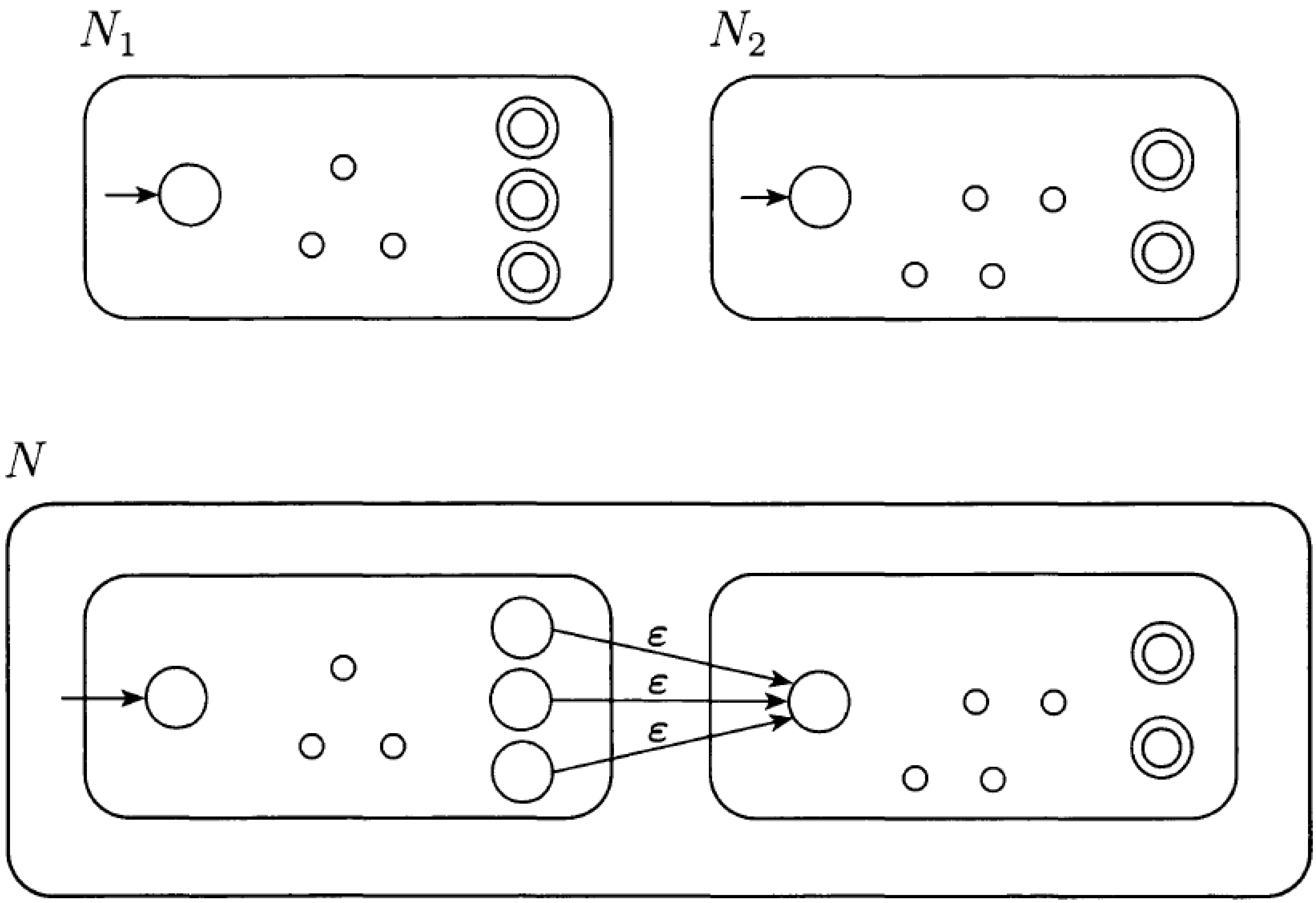
$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

Fechamento sob operações regulares

- **Teorema 1.47**:
  A classe de linguagens regulares é fechada sob a operação concatenação.
  - Ideia da prova: Suponha que temos duas linguagens regulares A_1 e A_2. Queremos provar que $A_1 \circ A_2$ é regular. A ideia é tomar os dois AFNs, N_1 para A_1 e N_2 para A_2, e combiná-los em um novo AFN N que reconhece $A_1 \circ A_2$.

Fechamento sob operações regulares

- **Teorema 1.47**:



Fechamento sob operações regulares

- **Teorema 1.47**:
  - Dem:

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$.

1. $Q = Q_1 \cup Q_2$.
   The states of N are all the states of N_1 and N_2.
2. The state q_1 is the same as the start state of N_1.
3. The accept states F_2 are the same as the accept states of N_2.

Fechamento sob operações regulares

- **Teorema 1.47**:
  - Dem (cont):

4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

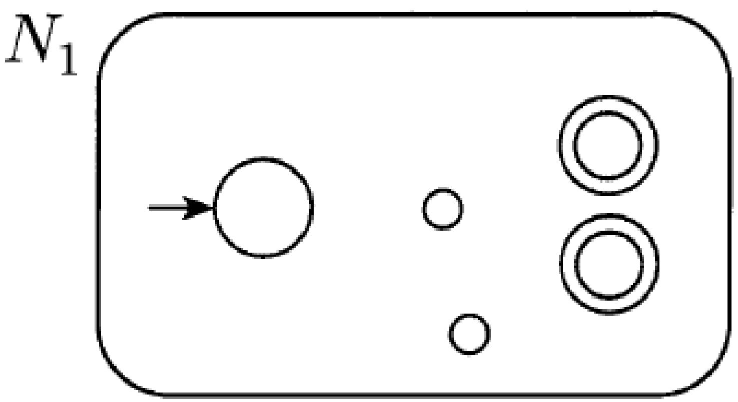
$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2. \end{cases}$$

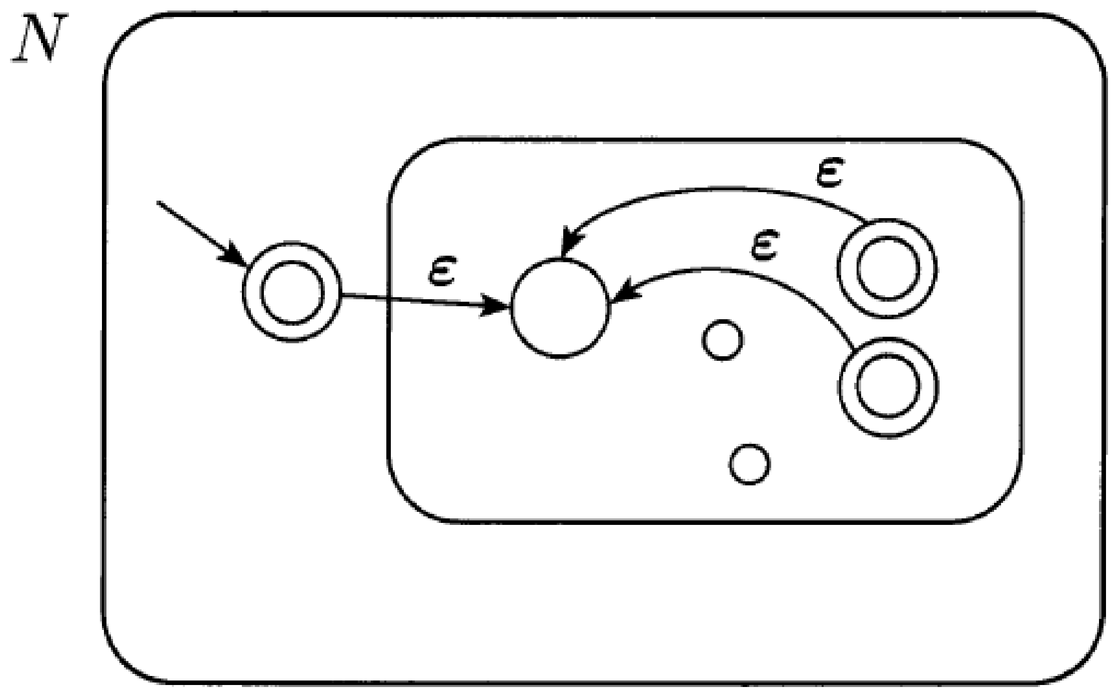
Fechamento sob operações regulares

- **Teorema 1.49**:
  A classe de linguagens regulares é fechada sob a operação estrela.
  - Ideia da prova: Suponha que temos uma linguagem regular A_1. Queremos provar que ${A_1}^*$ é regular. A ideia é tomar o AFN N_1 para A_1, e construir, a partir dele, um novo AFN N que reconhece ${A_1}^*$.

Fechamento sob operações regulares

- **Teorema 1.49**:





Fechamento sob operações regulares

- **Teorema 1.49**:
  - Dem:

PROOF Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.
Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*.

1. $Q = \{q_0\} \cup Q_1$.
   The states of N are the states of N_1 plus a new start state.
2. The state q_0 is the new start state.
3. $F = \{q_0\} \cup F_1$.
   The accept states are the old accept states plus the new start state.

Fechamento sob operações regulares

- **Teorema 1.49**:
  - Dem (cont):

4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

Expressões regulares

- Expressões aritméticas (operadores $+, -, \times, \div$, etc): ex:

$$(5 + 3) \times 4$$

  - Operadores: números
  - Resultado: número
    - No exemplo acima, $(5 + 3) \times 4 = 32$
- Expressões regulares (com operadores $\cup, \circ, *$): ex:

$$(0 \cup 1)0^*$$

  - Operadores: linguagens
  - Resultado: linguagem
    - No exemplo acima, $(0 \cup 1)0^* = \{\text{strings que começam com 0 ou 1 seguido por qualquer número de zeros}\}$

Expressões regulares

- Interpretando a expressão regular $(0 \cup 1)0^*$:
  - Símbolos 0 e 1 representam as linguagens $\{0\}$ e $\{1\}$
  - $(0 \cup 1) = (\{0\} \cup \{1\}) = \{0,1\}$
  - $0^* = \{0\}^* = \{\varepsilon\} \cup \{\text{strings contendo qualquer número de } 0's\}$
  - $(0 \cup 1)0^* = (0 \cup 1) \circ 0^* = \{0,1\} \circ \{0\}^* = \{\text{strings que começam com } 0 \text{ ou } 1 \text{ seguido por qualquer número de zeros}\}$
  - Obs: operador de concatenação é usualmente implícito

- Aplicações típicas de expressões regulares em computação:
  - Busca de sequências que satisfazem certos padrões
  - Utilitários AWK e GREP em sistemas Linux; linguagem PERL e editores de texto

Expressões regulares

- Outro exemplo:

  $(0 \cup 1)^* = \{\text{todas as strings possíveis (inclusive vazias)de } 0'\text{s e } 1'\text{s}\}$

  – Se $\Sigma = \{0,1\}$, então podemos escrever Σ como uma abreviação de Símbolos 0 e 1 representam as linguagens $\{0\}$ e $\{1\}$. $(0 \cup 1)$

  – De forma mais geral, se Σ é um alfabeto:
    - A expressão regular Σ descreve a linguagem de todas as strings de tamanho 1 sobre esse alfabeto;
    - A expressão regular Σ^* descreve a linguagem de todas as strings sobre esse alfabeto.

  – Outros exemplos:
    - $\Sigma^* 1$ é a linguagem de todas as strings que terminam em 1
    - $(0\Sigma^*) \cup (\Sigma^* 1)$ consiste de todas as strings que começam com 0 ou terminam com 1

Expressões regulares

- Definição 1.52:

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. (R_1^*), where R_1 is a regular expression.

In items 1 and 2, the regular expressions a and ε represent the languages $\{a\}$ and $\{\varepsilon\}$, respectively. In item 3, the regular expression $\emptyset$ represents the empty language. In items 4, 5, and 6, the expressions represent the languages obtained by taking the union or concatenation of the languages R_1 and R_2, or the star of the language R_1, respectively.

Expressões regulares

- Definição acima é indutiva:
  - Define-se expressões regulares em termos de expressões regulares menores
- Observações:
  - Precedência de operadores:
    - Estrela (*), Concatenação ($\circ$), União ($\cup$)
  - R^+ é uma abreviação para RR^+
    - Ou seja, R^+ denota a concatenação de 1 ou mais palavras de R
    - Analogamente, R^k representa a concatenação de *k* palavras de R
  - Quando queremos distinguir entre a expressão regular R e a linguagem por ela representada, denotamos $L(R)$ a linguagem de R

Expressões regulares

- Exemplos:

1. $0^*10^* = \{w \mid w \text{ contains a single } 1\}$.

2. $\Sigma^*1\Sigma^* = \{w \mid w \text{ has at least one } 1\}$.

3. $\Sigma^*001\Sigma^* = \{w \mid w \text{ contains the string } 001 \text{ as a substring}\}$.

4. $1^*(01^+)^* = \{w \mid \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}$.

5. $(\Sigma\Sigma)^* = \{w \mid w \text{ is a string of even length}\}$.[5]

6. $(\Sigma\Sigma\Sigma)^* = \{w \mid \text{the length of } w \text{ is a multiple of three}\}$.

7. $01 \cup 10 = \{01, 10\}$.

8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$.

---

[5] The ***length*** of a string is the number of symbols that it contains.

Expressões regulares

- Exemplos:

9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$.
The expression $0 \cup \varepsilon$ describes the language $\{0, \varepsilon\}$, so the concatenation operation adds either 0 or ε before every string in 1^*.

10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$.

11. $1^*\emptyset = \emptyset$.
Concatenating the empty set to any set yields the empty set.

12. $\emptyset^* = \{\varepsilon\}$.
The star operation puts together any number of strings from the language to get a string in the result. If the language is empty, the star operation can put together 0 strings, giving only the empty string.

Expressões regulares

- Identidades importantes:

$R \cup \emptyset = R$.
Adding the empty language to any other language will not change it.

$R \circ \varepsilon = R$.
Joining the empty string to any string will not change it.

- Atenção:

$R \cup \varepsilon$ may not equal R.
For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \cup \varepsilon) = \{0, \varepsilon\}$.

$R \circ \emptyset$ may not equal R.
For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \circ \emptyset) = \emptyset$.

Expressões regulares

- Exemplo de expressões regulares em compilação:
  - Objetos elementares em uma linguagem de programação, denominados *tokens*, tais como nomes de variáveis e constantes, podem ser descritas com expressões regulares
  - Por exemplo, uma constante numérica que pode incluir uma parte fracional e/ou um sinal pode ser descrita como um membro da linguagem

$$(+ \cup - \cup \varepsilon)\left(D^+ \cup D^+ . D^* \cup D^* . D^+\right)$$

$$\text{where } D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

    - Exemplos: 72 ; 3.14159 ; $+7$; $-.01$

Equivalência entre expressões regulares e autômatos finitos

- Expressões regulares e autômatos finitos são equivalentes em seu poder descritivo
  - Expressões regulares podem ser convertidas em autômatos finitos e vice-versa

THEOREM 1.54 ....

A language is regular if and only if some regular expression describes it.

- Este teorema tem duas direções;
  - Pode-se enunciar e provar cada direção como um lema separado

Equivalência entre expressões regulares e autômatos finitos

LEMMA 1.55

If a language is described by a regular expression, then it is regular.

- Ideia da demonstração:
  - Digamos que temos uma expressão regular *R* descrevendo uma linguagem *A*.
  - Mostramos como converter *R* em um AFN (NFA, em inglês) que reconhece *A*.
  - Como *A* possui um AFN que a reconhece, então *A* é regular.

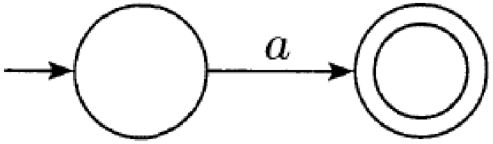
Equivalência entre expressões regulares e autômatos finitos

LEMMA 1.55 ....

If a language is described by a regular expression, then it is regular.

PROOF Let's convert R into an NFA N. We consider the six cases in the formal definition of regular expressions.

1. $R = a$ for some a in Σ. Then $L(R) = \{a\}$, and the following NFA recognizes $L(R)$.



Note that this machine fits the definition of an NFA but not that of a DFA because it has some states with no exiting arrow for each possible input symbol. Of course, we could have presented an equivalent DFA here but an NFA is all we need for now, and it is easier to describe.

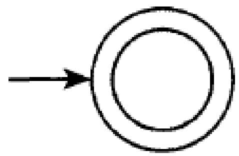
Formally, $N = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$, where we describe δ by saying that $\delta(q_1, a) = \{q_2\}$ and that $\delta(r, b) = \emptyset$ for $r \neq q_1$ or $b \neq a$.

Equivalência entre expressões regulares e autômatos finitos

LEMMA 1.55

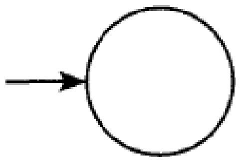
If a language is described by a regular expression, then it is regular.

2. $R = \varepsilon$. Then $L(R) = \{\varepsilon\}$, and the following NFA recognizes $L(R)$.



Formally, $N = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$, where $\delta(r, b) = \emptyset$ for any r and b.

3. $R = \emptyset$. Then $L(R) = \emptyset$, and the following NFA recognizes $L(R)$.



Formally, $N = (\{q\}, \Sigma, \delta, q, \emptyset)$, where $\delta(r, b) = \emptyset$ for any r and b.

Equivalência entre expressões regulares e autômatos finitos

LEMMA 1.55

If a language is described by a regular expression, then it is regular.

4. $R = R_1 \cup R_2$.

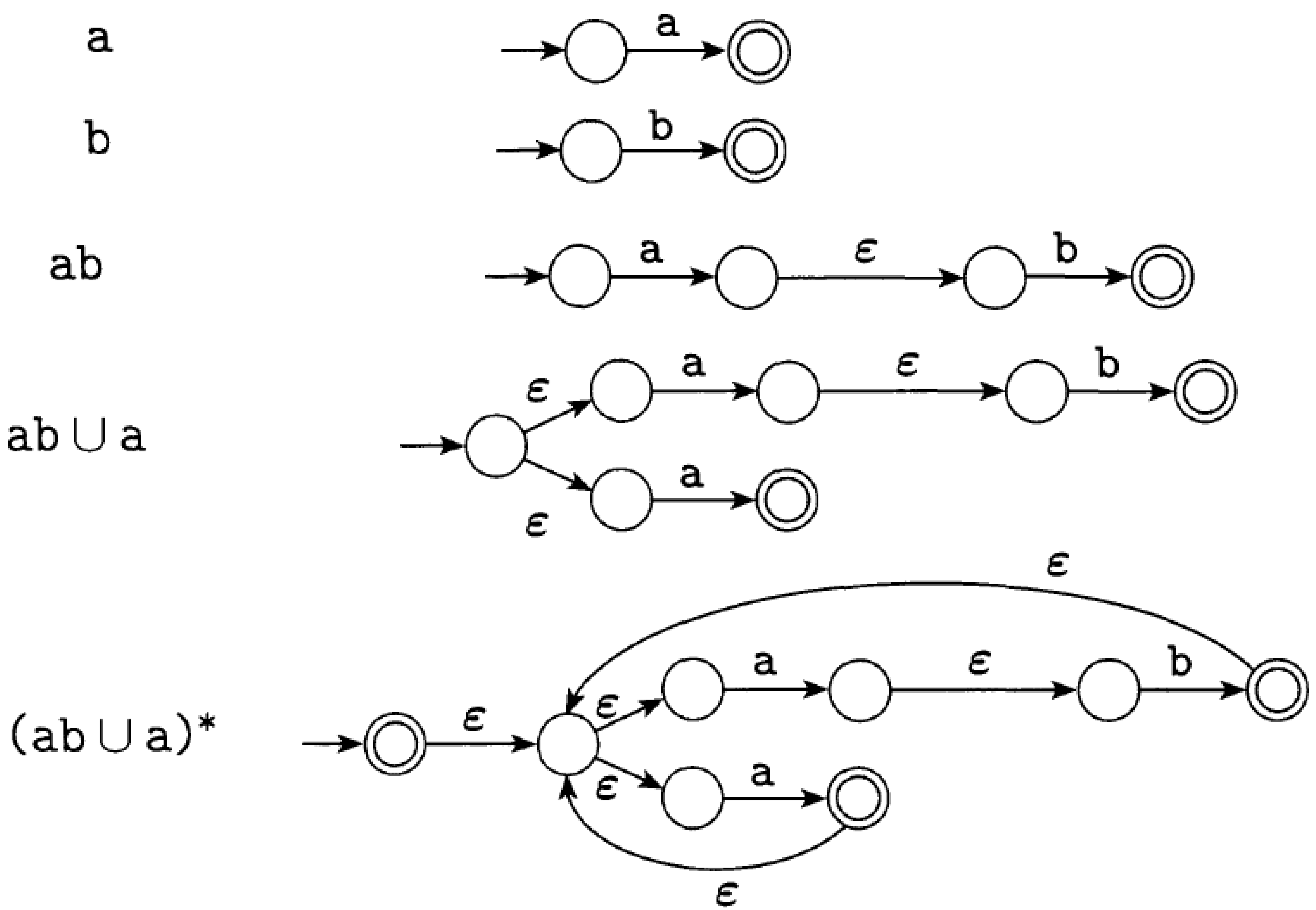
5. $R = R_1 \circ R_2$.

6. $R = R_1^*$.

For the last three cases we use the constructions given in the proofs that the class of regular languages is closed under the regular operations. In other words, we construct the NFA for R from the NFAs for R_1 and R_2 (or just R_1 in case 6) and the appropriate closure construction.

Equivalência entre expressões regulares e autômatos finitos

- Ex: Converter $(ab \cup a)^*$ em um AFN:



Equivalência entre expressões regulares e autômatos finitos

LEMMA 1.60

If a language is regular, then it is described by a regular expression.

- Ideia da demonstração:
  - Queremos mostra que, se uma linguagem A é regular, então uma expressão regular a descreve.
  - Como A é regular, ele é reconhecida por um AFD (DFA, em inglês).
  - Descrevemos um procedimento para converter AFDs em expressões regulares equivalentes.
  - Procedimento em duas partes:
    - Convertemos o AFD em um ***autômato finito não-determinístico generalizado*** (AFNG em português; GNFA em inglês)
    - Convertemos o AFNG em uma expressão regular

Equivalência entre expressões regulares e autômatos finitos

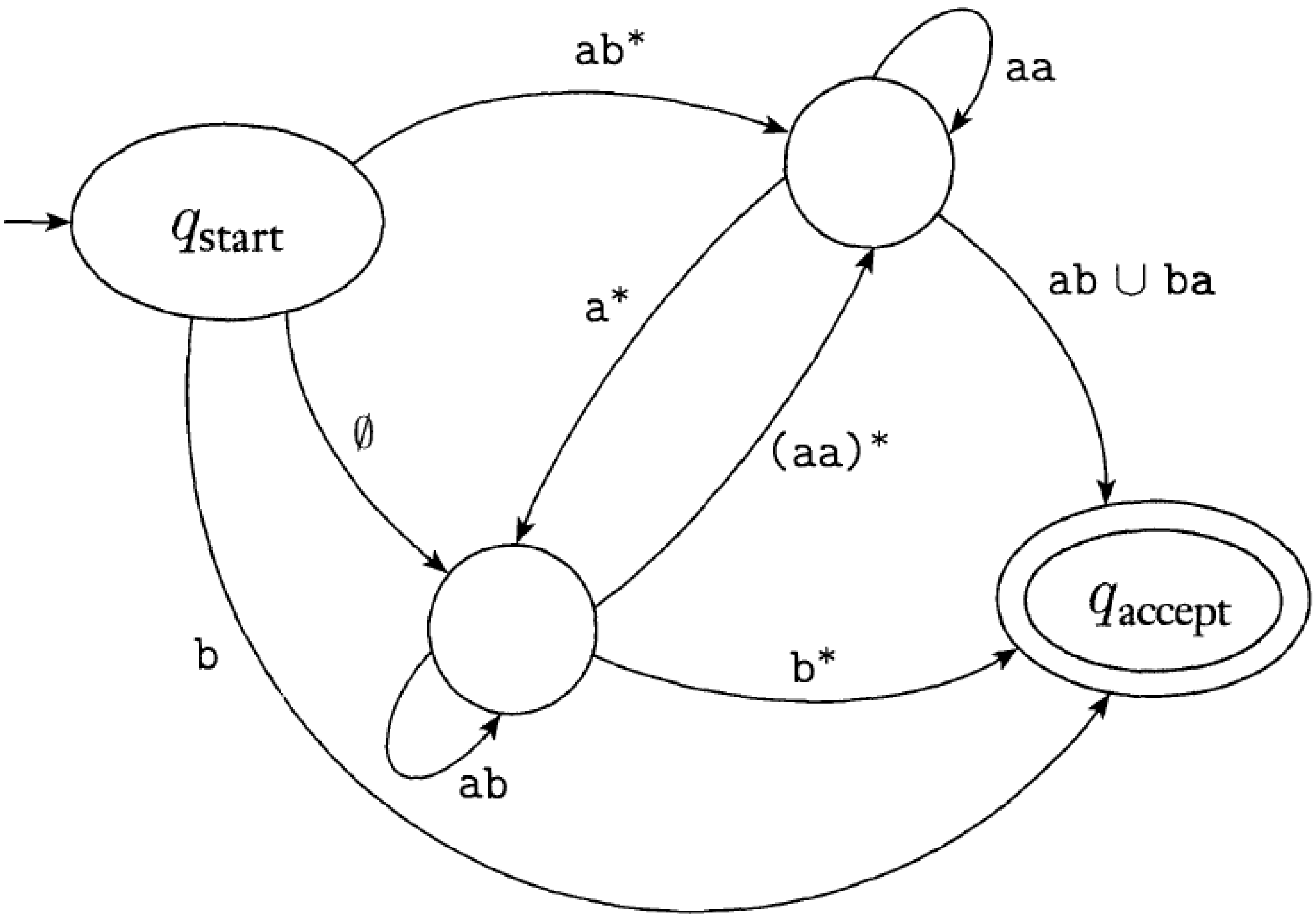
LEMMA 1.60

If a language is regular, then it is described by a regular expression.

- Autômato finito não-determinístico generalizado
  - AFN cujas transições são rotuladas por expressões regulares, ao invés de símbolos do alfabeto
    - AFNG lê blocos de símbolos da fita, não necessariamente um de cada vez como ocorre com os AFNs;
    - Se uma transição de um estado p_i para o estado p_j é rotulada por uma expressão regular *R*, então essa transição ocorre lendo-se um bloco de símbolos da entrada que constituam uma string descrita por *R.*
    - Um AFNG aceita uma entrada se sua computação puder resultar em um estado de aceitação ao final da leitura.

Equivalência entre expressões regulares e autômatos finitos

- Autômato finito não-determinístico generalizado



Equivalência entre expressões regulares e autômatos finitos

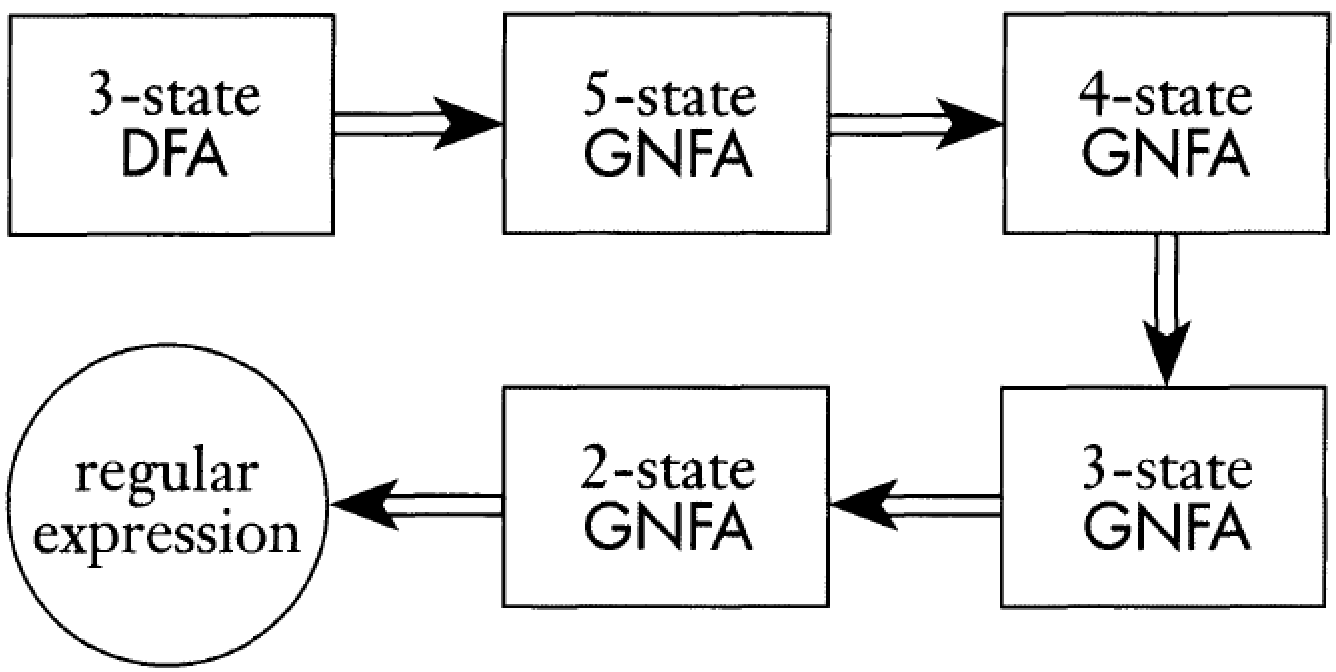
- Autômato finito não-determinístico generalizado
  - Condições:
    - Estado inicial possui setas de transições saindo para todos os demais estados, mas não possui setas de transições chegando de nenhum estado
    - Há apenas um único estado de aceitação, que possui setas chegando de todos os demais estados mas não setas saindo para outros estados
    - Exceto para os estados inicial e de aceitação, existe sempre uma seta saindo de todo estado para os demais estados, e também uma seta de cada estado para si mesmo.

Equivalência entre expressões regulares e autômatos finitos

- Conversão do AFNG para uma expressão regular
  - Suponha que o AFNG tenha k estados. Como ele possui um estado inicial e um de aceitação distintos, sabemos que $k \geq 2$;
  - Se $k > 2$, construímos um AFNG equivalente com $k - 1$ estados; este passo é repetido até ser reduzido a apenas 2 estados.
  - Quando $k = 2$, o AFNG possui uma única aresta do estado inicial para o estado de aceitação. O rótulo dessa aresta é a expressão regular equivalente.

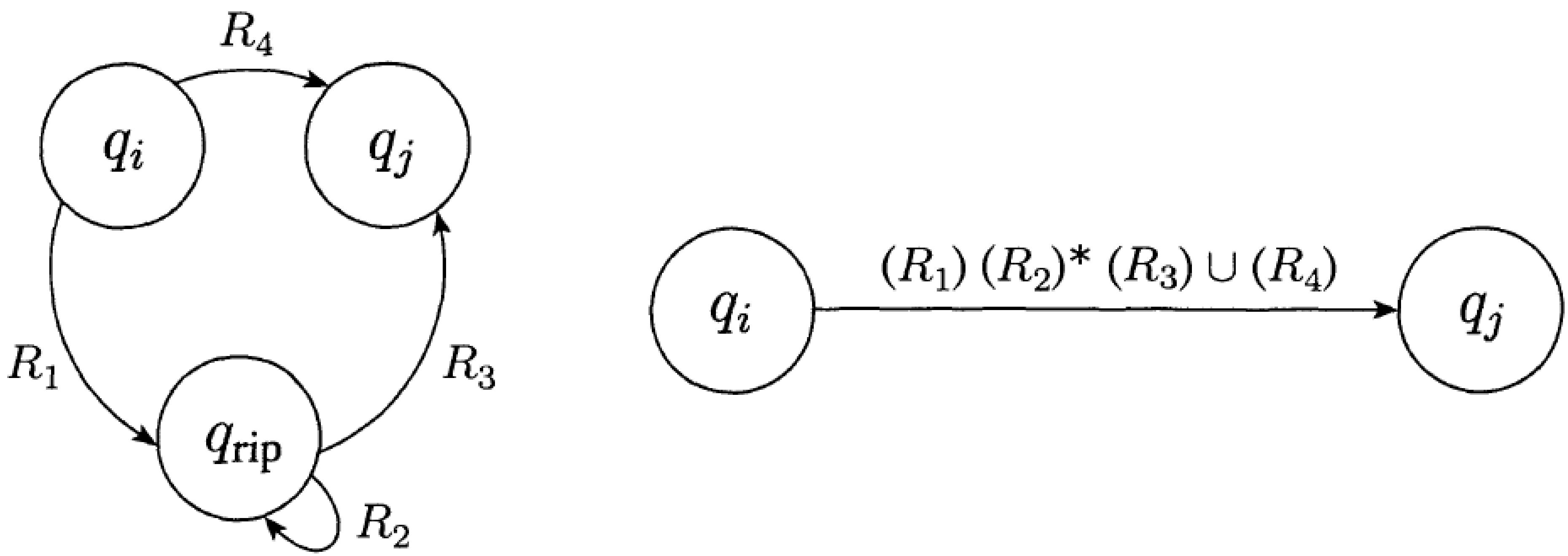
Equivalência entre expressões regulares e autômatos finitos

– Representação gráfica da conversão de um AFD inicial (com 3 estados) para a expressão regular equivalente:



Equivalência entre expressões regulares e autômatos finitos

– Eliminação de um estado intermediário do AFNG:

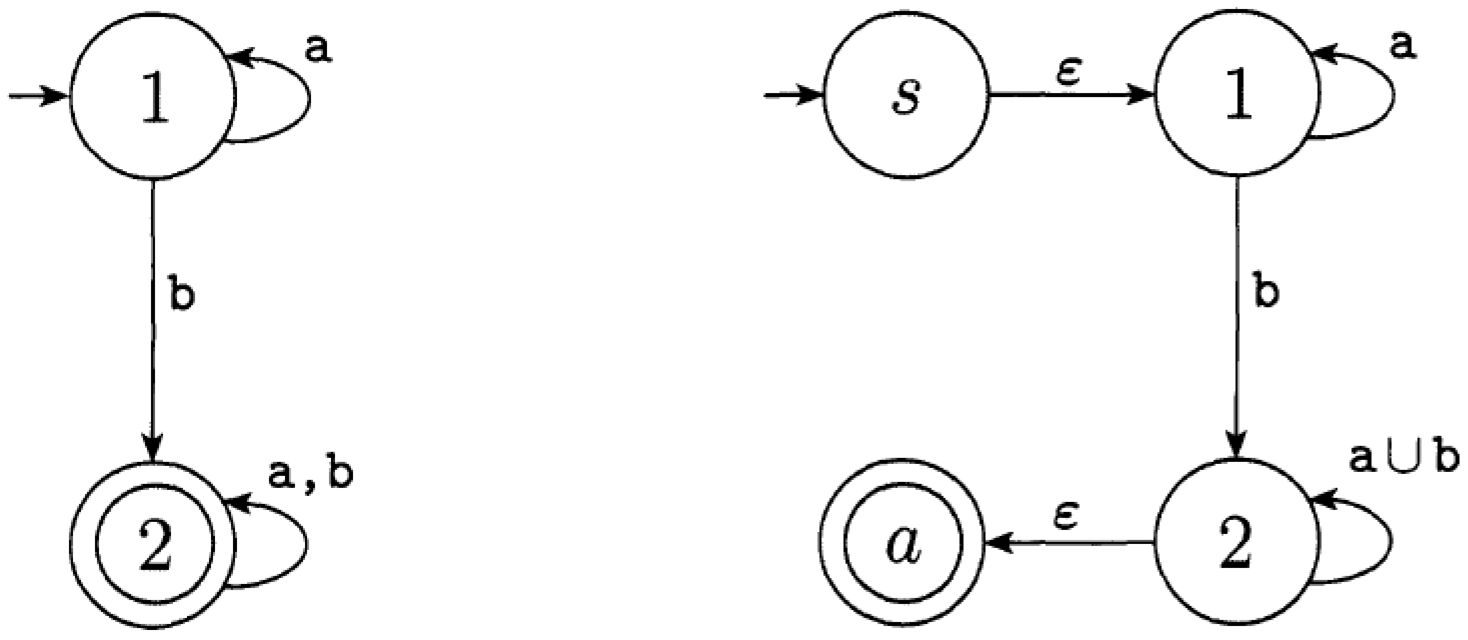


before after

We make this change for each arrow going from any state q_i to any state q_j, including the case where $q_i = q_j$. The new machine recognizes the original language.

Equivalência entre expressões regulares e autômatos finitos

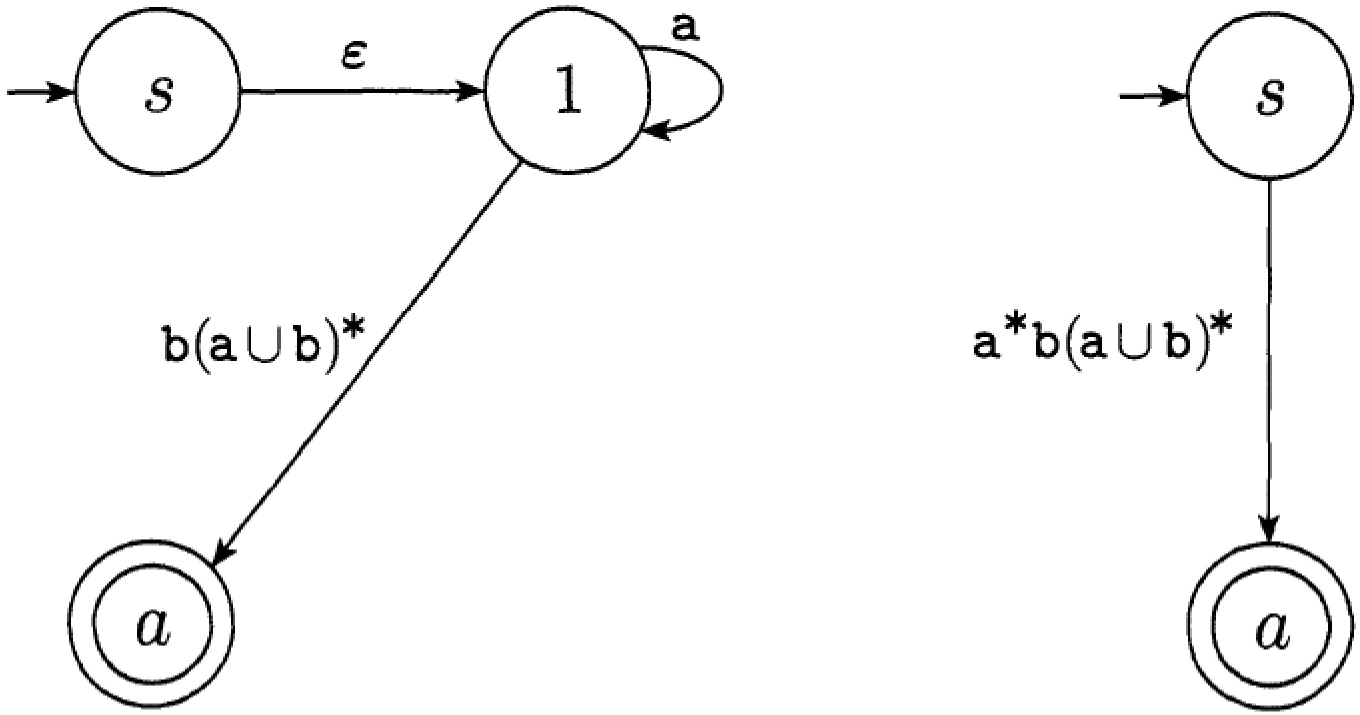
- Exemplo:



(a) (b)

Equivalência entre expressões regulares e autômatos finitos

- Exemplo:



(c) (d)

Linguagens não-regulares

- A linguagem B = $\{0^n 1^n \mid n >= 0\}$ é regular?
  - Notação: a^n = símbolo *a* repetido *n* vezes

- Como provar que uma linguagem não pode ser reconhecida por um autômato finito?

Linguagens não-regulares

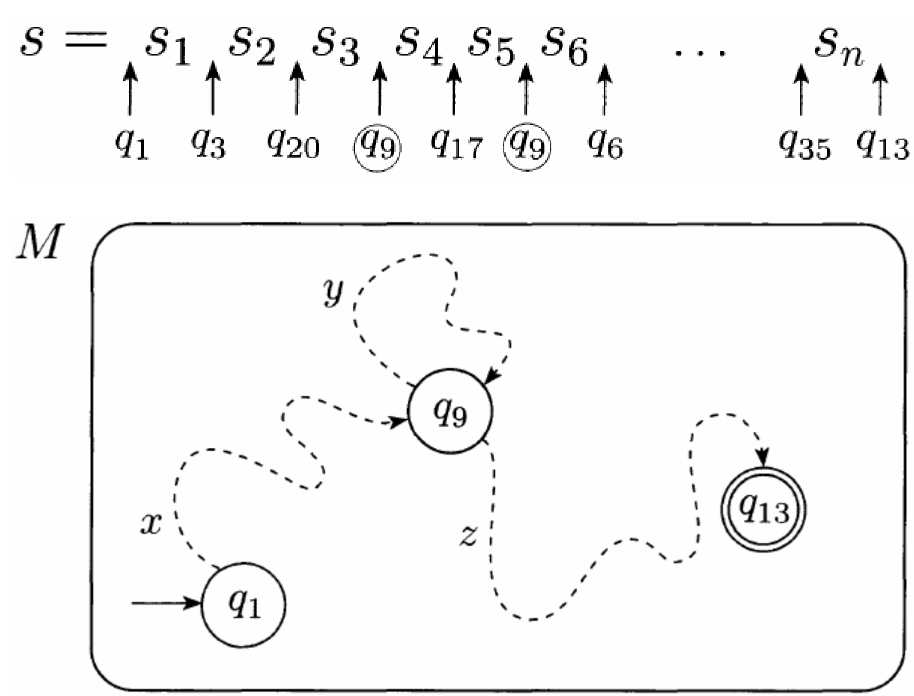
TEOREMA 1.70

Lema do bombeamento Se A é uma linguagem regular, então existe um número p (o comprimento de bombeamento) tal que, se s é qualquer cadeia de A de comprimento no mínimo p, então s pode ser dividida em três partes, $s = xyz$, satisfazendo as seguintes condições:

1. para cada $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, e
3. $|xy| \leq p$.

Linguagens não-regulares

- Ideia da prova:
  - Usamos p = número de estados do AFD *M* que reconhece *A*
  - Considere uma cadeia $s \in A$ de tamanho $n \geq p$;
  - Como $n \geq p$, então a computação de *M* sobre s repetirá algum estado; tomemos o 1º estado repetido (q_9 no exemplo abaixo)
  - Então, s pode ser dividida em 3 subcadeias, $s = xyz$, onde:
    - subcadeia x leva M do estado inicial até q_9 ($|x| \geq 0$)
    - subcadeia y leva M do estado q_9 até si mesmo ($|y| > 0$)
    - subcadeia z leva M do estado q_9 até um estado final ($|z| \geq 0$)
  - Note que qualquer cadeia $xy^iz, i \geq 0$, leva M do estado inicial até o estado final, e portanto pertence a A
  - Finalmente, $|xy| \leq p$, pois senão outro estado que não o q_9 teria sido repetido antes (absurdo, pois assumimos que q_9 foi o primeiro a repetir).



Linguagens não-regulares

PROVA Seja $M = (Q, \Sigma, \delta, q_1, F)$ um AFD que reconhece A e p o número de estados de M.

Seja $s = s_1 s_2 \cdots s_n$ uma cadeia em A de comprimento n, onde $n \geq p$. Seja $r_1, \ldots, r_{n+1}$ a seqüência de estados nos quais M passa enquanto processa s, de forma que $r_{i+1} = \delta(r_i, s_i)$ para $1 \leq i \leq n$. Essa seqüência tem comprimento $n+1$, que é pelo menos $p+1$. Entre os primeiros $p+1$ elementos da seqüência, dois devem ser o mesmo estado, pelo princípio da casa de pombos. Chamamos o primeiro desses de r_j e o segundo de r_l. Como r_l ocorre entre as primeiras $p+1$ posições da seqüência começando em r_1, temos que $l \leq p+1$. Agora, seja $x = s_1 \cdots s_{j-1}$, $y = s_j \cdots s_{l-1}$ e $z = s_l \cdots s_n$.

Como x leva M de r_1 para r_j, y leva M de r_j para r_j e z leva M de r_j para r_{n+1}, que é um estado de aceitação, M deve aceitar xy^iz para $i \geq 0$. Sabemos que $j \neq l$, e portanto $|y| > 0$; e $l \leq p+1$, e logo $|xy| \leq p$. Dessa forma, satisfizemos todas as condições do lema do bombeamento.

Linguagens não-regulares

EXEMPLO 1.73

Seja B a linguagem $\{0^n1^n | n \geq 0\}$. Usamos o lema do bombeamento para provar que B não é regular. A prova é por contradição.

Suponha, ao contrário, que B seja regular. Seja p o comprimento de bombeamento dado pelo lema do bombeamento. Escolha s como a cadeia 0^p1^p. Como s é um membro de B e tem comprimento maior que p, o lema do bombeamento garante que s pode ser dividida em três partes, $s = xyz$, onde para qualquer $i \geq 0$ a cadeia xy^iz está em B. Consideramos três casos para mostrar que esse resultado é impossível.

1. A cadeia y contém apenas 0s. Neste caso, a cadeia $xyyz$ tem mais 0s que 1s e, portanto, não é um membro de B, violando a condição 1 do lema do bombeamento. Esse caso é uma contradição.
2. A cadeia y contém somente 1s. Esse caso também dá uma contradição.
3. A cadeia y contém ambos, 0s e 1s. Nesse caso, a cadeia $xyyz$ pode ter o mesmo número de 0s e 1s, mas eles estarão fora de ordem, com alguns 1s antes de 0s. Logo, ela não é um membro de B, o que é uma contradição.

Linguagens não-regulares

EXEMPLO 1.74 ....

Seja $C = \{w \mid w$ tem número igual de 0s e 1s$\}$. Usamos o lema do bombeamento para provar que C não é regular. A prova é por contradição.

$0^p 1^p$

Se fizermos x e z serem a cadeia vazia e y ser a cadeia $0^p 1^p$, então $xy^i z$ sempre terá um número igual de 0s e 1s e, portanto, está em C.

Logo, *parece* que s pode ser bombeada.

Linguagens não-regulares

EXEMPLO 1.74

Seja $C = \{w | w \text{ tem número igual de 0s e 1s}\}$. Usamos o lema do bombeamento para provar que C não é regular. A prova é por contradição.

$0^p 1^p$

Se fizermos x e z serem a cadeia vazia e y ser a cadeia $0^p 1^p$, então $xy^i z$ sempre terá um número igual de 0s e 1s e, portanto, está em C.

Logo, *parece* que s pode ser bombeada.

Aqui a condição 3 no lema do bombeamento é útil.

Se $|xy| \leq p$, então y deve conter somente 0s; logo, $xyyz \notin C$.

Por conseguinte, s não pode ser bombeada.

Linguagens não-regulares

EXEMPLO 1.74

Seja $C = \{w|\, w$ tem número igual de 0s e 1s$\}$. Usamos o lema do bombeamento para provar que C não é regular. A prova é por contradição.

0^p1^p

Se fizermos x e z serem a cadeia vazia e y ser a cadeia 0^p1^p, então xy^iz sempre terá um número igual de 0s e 1s e, portanto, está em C.

Logo, *parece* que s pode ser bombeada.

Aqui a condição 3 no lema do bombeamento é útil.

Se $|xy| \leq p$, então y deve conter somente 0s; logo, $xyyz \notin C$.

Por conseguinte, s não pode ser bombeada.

Cuidado: $s = (01)^p$ $x = \varepsilon$, $y = 01$ e $z = (01)^{p-1}$

Linguagens não-regulares

EXEMPLO 1.77

$$E = \{0^i 1^j | i > j\}$$

$$s = 0^{p+1} 1^p.$$

Condição 3 => y contém somente zeros

xyyz ainda está em E

Linguagens não-regulares

EXEMPLO 1.77

$$E = \{0^i 1^j | i > j\}$$

$$s = 0^{p+1} 1^p.$$

Condição 3 => y contém somente zeros

xyyz ainda está em E

Mas $xy^0z = xz$ não está

Não esqueça que você pode tentar bombear para baixo!