ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 4
Não Determinismo e Expressões Regulares

Profa. Ariane Machado Lima
ariane.machado@usp.br

Aulas anteriores

- Equivalência entre AFDs e AFNs
- Fechamento de linguagens regulares sob a operação de união
  - Prova usando AFDs
  - Prova usando AFNs

Hoje

- Fechamento de linguagens regulares sob as operações de concatenação e estrela
  - Prova usando AFNs

- Expressões regulares

Fechamento sob concatenação

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 $A_1 \circ A_2$.

Concatenação: $A \circ B = \{xy | \ x \in A \text{ e } y \in B\}$

Fechamento sob concatenação

- O que acham?

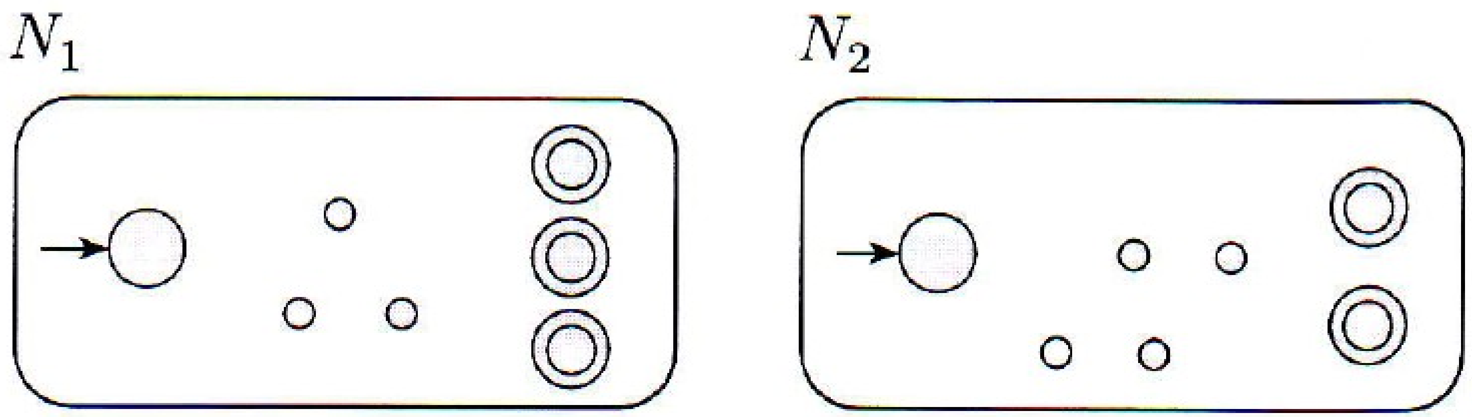
TEOREMA 1.26 ....

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

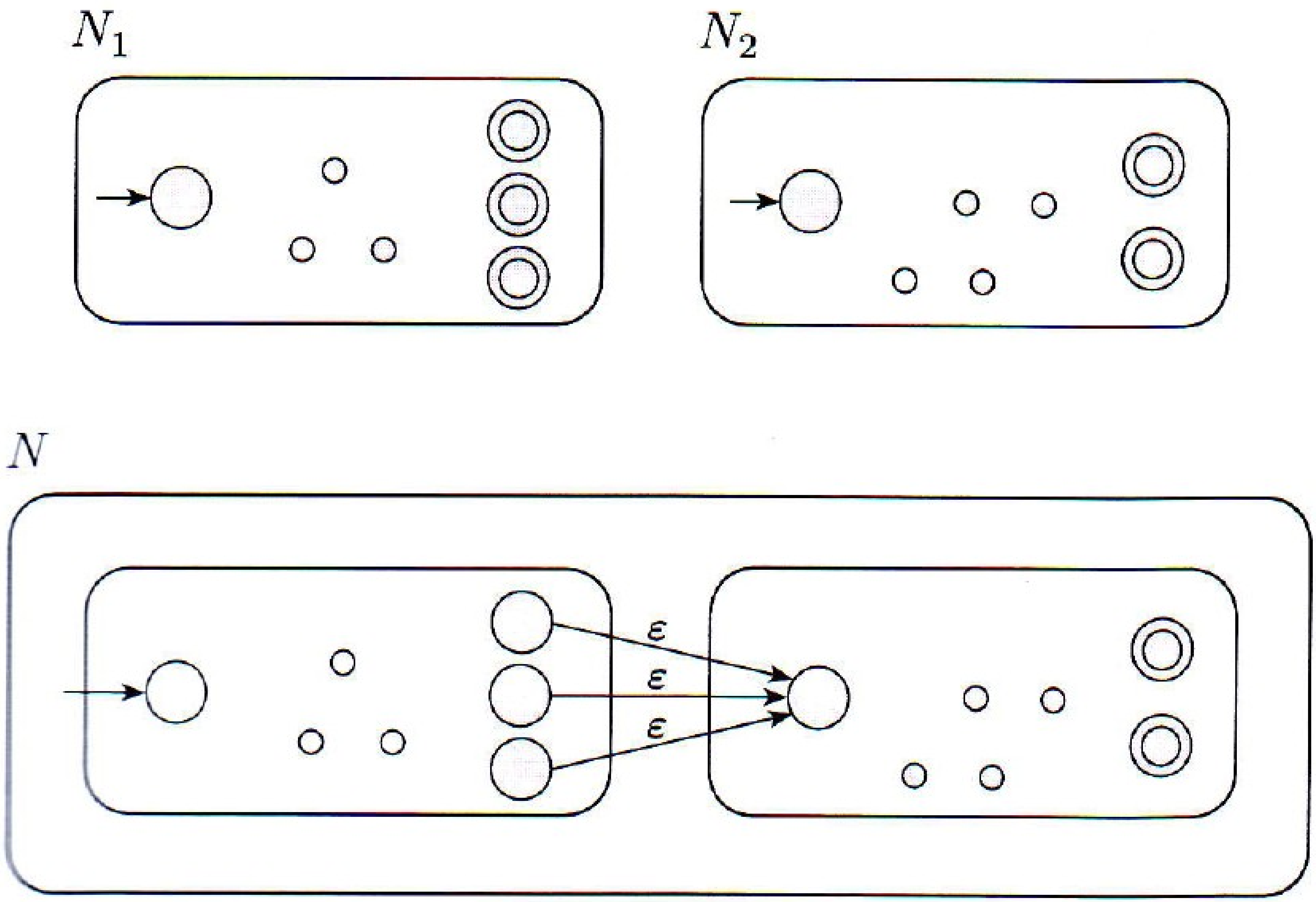
Em outras palavras, se A_1 e A_2 são linguagens regulares, então o mesmo acontece com $A_1 \circ A_2$.

- Prova?

Fechamento sob concatenação: prova



Fechamento sob concatenação: prova



Fechamento sob concatenação: prova

Suponha que $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ reconheça A_1, e que
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ reconheça A_2.

Construa $N = (Q, \Sigma, \delta, q_1, F_2)$ para reconhecer $A_1 \circ A_2$.

1. $Q = Q_1 \cup Q_2$.
   Os estados de N são todos os estados de N_1 e N_2.
2. O estado q_1 é o mesmo que o estado inicial de N_1.
3. Os estados de aceitação F_2 são os mesmos que os estados de aceitação de N_2.
4. Defina δ de modo que para qualquer $q \in Q$ e qualquer $a \in \Sigma_\varepsilon$,

Fechamento sob concatenação: prova

Suponha que $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ reconheça A_1, e que
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ reconheça A_2.

Construa $N = (Q, \Sigma, \delta, q_1, F_2)$ para reconhecer $A_1 \circ A_2$.

1. $Q = Q_1 \cup Q_2$.
   Os estados de N são todos os estados de N_1 e N_2.
2. O estado q_1 é o mesmo que o estado inicial de N_1.
3. Os estados de aceitação F_2 são os mesmos que os estados de aceitação de N_2.
4. Defina δ de modo que para qualquer $q \in Q$ e qualquer $a \in \Sigma_\varepsilon$,

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

Fechamento sob op. estrela

TEOREMA 1.49

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

Estrela: $A^* = \{x_1 x_2 \ldots x_k |\ k \geq 0 \text{ e cada } x_i \in A\}$

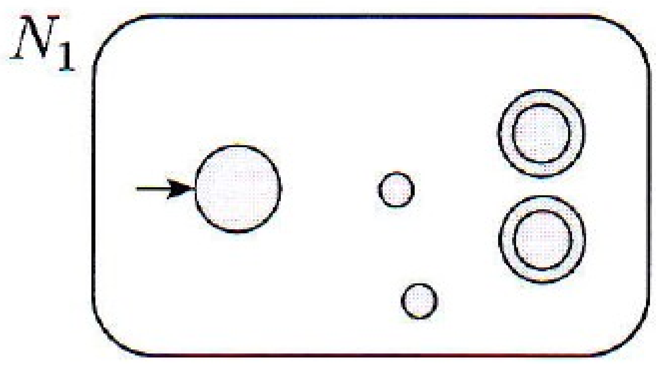
Fechamento sob op. estrela

TEOREMA 1.49

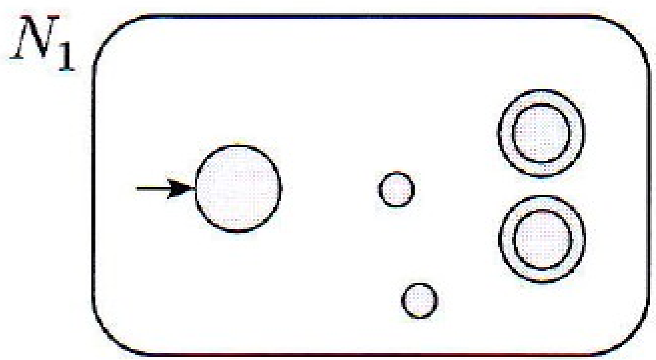
A classe de linguagens regulares é fechada sob a operação estrela.

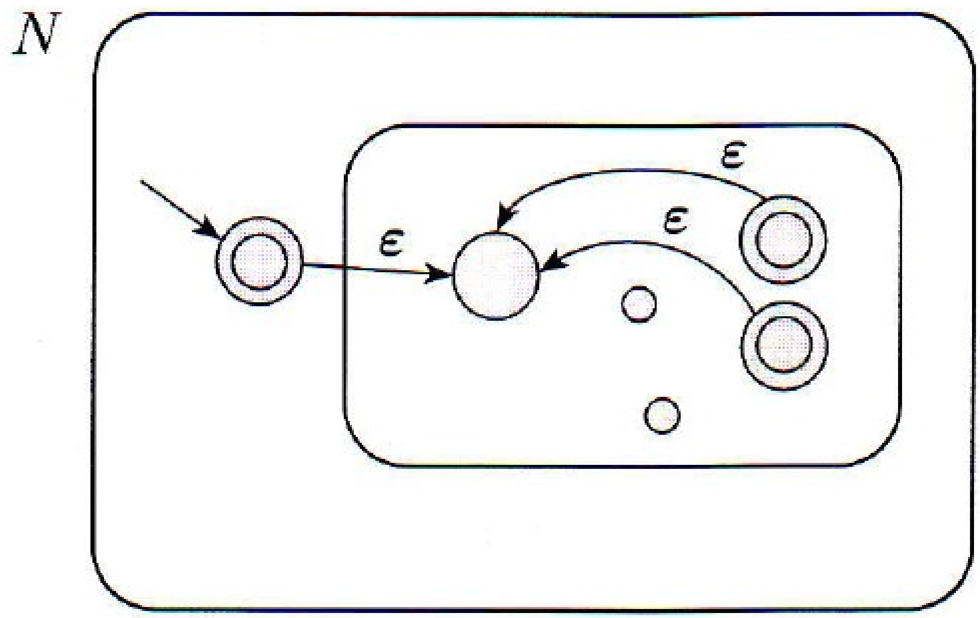
- Prova?

Fechamento sob op. estrela - prova



Fechamento sob op. estrela - prova





Fechamento sob op. estrela - prova

PROVA Suponha que $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ reconheça A_1. Construa $N = (Q, \Sigma, \delta, q_0, F)$ para reconhecer A_1^*.

1. $Q = \{q_0\} \cup Q_1$.
   Os estados de N são os estados de N_1 mais um novo estado inicial.
2. O estado q_0 é o novo estado inicial.
3. $F = \{q_0\} \cup F_1$.
   Os estados de aceitação são os antigos estados de aceitação mais o novo estado inicial.
4. Defina δ de modo que para qualquer $q \in Q$ e qualquer $a \in \Sigma_\varepsilon$,

Fechamento sob op. estrela - prova

PROVA Suponha que $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ reconheça A_1. Construa $N = (Q, \Sigma, \delta, q_0, F)$ para reconhecer A_1^*.

1. $Q = \{q_0\} \cup Q_1$.
   Os estados de N são os estados de N_1 mais um novo estado inicial.
2. O estado q_0 é o novo estado inicial.
3. $F = \{q_0\} \cup F_1$.
   Os estados de aceitação são os antigos estados de aceitação mais o novo estado inicial.
4. Defina δ de modo que para qualquer $q \in Q$ e qualquer $a \in \Sigma_\varepsilon$,

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

1.3 - Expressões regulares

Expressões regulares

- Uma linguagem é um conjunto de cadeias
- Um conjunto de cadeias pode ser descrito por uma expressão
- Ex: como você escreve no campo “Pesquisar” do computador que você quer localizar todos os arquivos que começam com “ACH2043” e terminam com “.pdf”?

Expressões regulares

- Uma linguagem é um conjunto de cadeias
- Um conjunto de cadeias pode ser descrito por uma expressão
- Ex: como você escreve no campo "Pesquisar" do computador que você quer localizar todos os arquivos que começam com "ACH2043" e terminam com ".pdf"?

ACH2043*.pdf

Expressões regulares

- Uma linguagem é um conjunto de cadeias
- Um conjunto de cadeias pode ser descrito por uma expressão
- Ex: como você escreve no campo “Pesquisar” do computador que você quer localizar todos os arquivos que começam com “ACH2043” e terminam com “.pdf”?

  ACH2043*.pdf

- Isso é uma expressão que descreve um conjunto

Expressões regulares

- Exemplo: o que essa expressão descreve?

$(\{0\} \cup \{1\}) \circ 0^*$

Expressões regulares

- Exemplo: o que essa expressão descreve?

({0} U {1}) ° 0*

Cadeias que comecem com 0 ou 1e terminem em zero ou mais 0's

Expressões regulares

- Exemplo: o que essa expressão descreve?

({0} U {1}) ° 0*

Cadeias que comecem com 0 ou 1e terminem em zero ou mais 0's

- Simplificação da notação:

(0 U 1)0*

Expressões regulares

- Regras de precedência (da maior para a menor):

  Estrela

  Concatenação

  União

- Outros exemplos:
  - (0 U 1)*
  - $0\Sigma^*$
  - $0\Sigma^* \cup \Sigma^*1$

Expressões regulares

DEFINIÇÃO 1.52

Digamos que R é uma ***expressão regular*** se R for

1. a para algum a no alfabeto Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, onde R_1 e R_2 são expressões regulares,
5. $(R_1 \circ R_2)$, onde R_1 e R_2 são expressões regulares, ou
6. (R_1^*), onde R_1 é uma expressão regular.

Nos itens 1 e 2, as expressões regulares a e ε representam as linguagens $\{a\}$ e $\{\varepsilon\}$, respectivamente. No item 3, a expressão regular $\emptyset$ representa a linguagem vazia. Nos itens 4, 5 e 6, as expressões representam as linguagens obtidas tomando-se a união ou concatenação das linguagens R_1 e R_2, ou a estrela da linguagem R_1, respectivamente.

Expressões regulares (ER)

- Abreviações:

  $R^+ = RR^*$

  $R^k = R.....R$ (concatenação de k R's)

- Se R é uma ER, dizemos que L(R) é a linguagem descrita por R

Exemplos de ERs

- $0^*10^* =$
- $\Sigma^*1\Sigma^* =$
- $(\Sigma\Sigma\Sigma)^* =$
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 =$

- $1^*\varnothing =$
- $\varnothing^* =$

Exemplos de ERs

- $0^*10^* = \{w \mid w$ contém um ÚNICO 1$\}$
- $\Sigma^*1\Sigma^* =$
- $(\Sigma\Sigma\Sigma)^* =$
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 =$

- $1^*\varnothing =$
- $\varnothing^* =$

Exemplos de ERs

- $0^*10^* = \{w \mid w$ contém um ÚNICO 1$\}$
- $\Sigma^*1\Sigma^* = \{w \mid w$ contém PELO MENOS UM 1$\}$
- $(\Sigma\Sigma\Sigma)^* =$
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 =$

- $1^*\emptyset =$
- $\emptyset^* =$

Exemplos de ERs

- $0^*10^* = \{w \mid w$ contém um ÚNICO 1$\}$
- $\Sigma^*1\Sigma^* = \{w \mid w$ contém PELO MENOS UM 1$\}$
- $(\Sigma\Sigma\Sigma)^* = \{w \mid$ o comprimento de w é múltiplo de 3$\}$
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 =$
- $1^*\emptyset =$
- $\emptyset^* =$

Exemplos de ERs

- 0*10* = {w | w contém um ÚNICO 1}
- Σ*1Σ* = {w | w contém PELO MENOS UM 1}
- (ΣΣΣ)* = {w | o comprimento de w é múltiplo de 3}
- 0Σ*0 U 1Σ*1 U 0 U 1 = {w | w começa e termina com o mesmo símbolo}
- 1*Ø =
- Ø* =

Exemplos de ERs

- 0^*10^* = {w | w contém um ÚNICO 1}
- $\Sigma^*1\Sigma^*$ = {w | w contém PELO MENOS UM 1}
- $(\Sigma\Sigma\Sigma)^*$ = {w | o comprimento de w é múltiplo de 3}
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$ = {w | w começa e termina com o mesmo símbolo}
- $1^*\varnothing = \varnothing$ (concatenação com $\varnothing$ produz $\varnothing$)
- $\varnothing^* =$

Exemplos de ERs

- $0^*10^* = \{w \mid w$ contém um ÚNICO 1$\}$
- $\Sigma^*1\Sigma^* = \{w \mid w$ contém PELO MENOS UM 1$\}$
- $(\Sigma\Sigma\Sigma)^* = \{w \mid$ o comprimento de w é múltiplo de 3$\}$
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w$ começa e termina com o mesmo símbolo$\}$
- $1^*\emptyset = \emptyset$ (concatenação com $\emptyset$ produz $\emptyset$)
- $\emptyset^* = \{ \varepsilon \}$ (operador * concatena qualquer número de cadeias para obter uma cadeia no resultado)

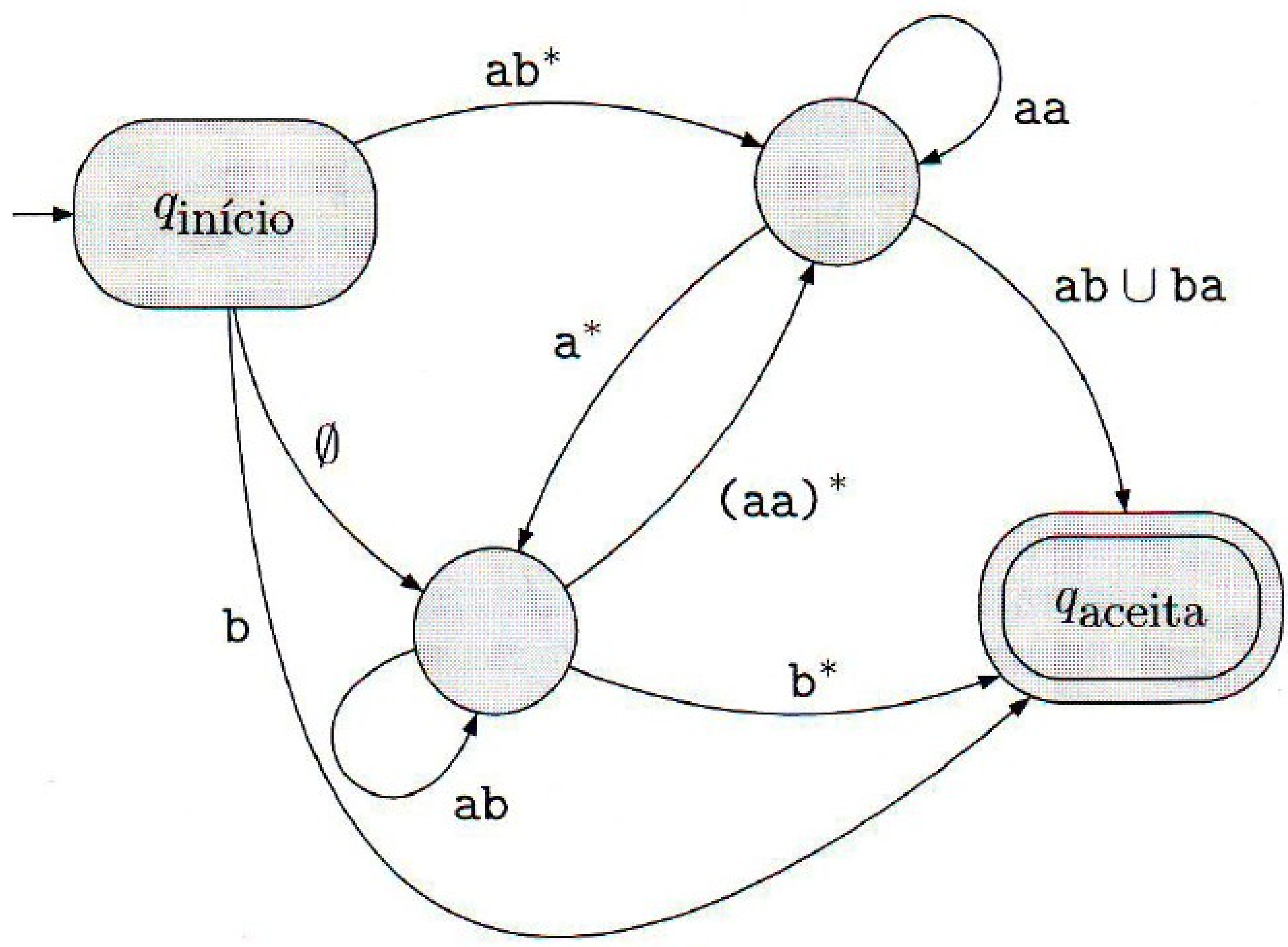
ERs: igualdades válidas

- $R \cup \emptyset = R$
- $R \circ \varepsilon = R$

- Trocar as operações acima pode invalidar as igualdades. Quando?

Autômatos Finitos Não-Determinísticos Generalizados (AFNGs)

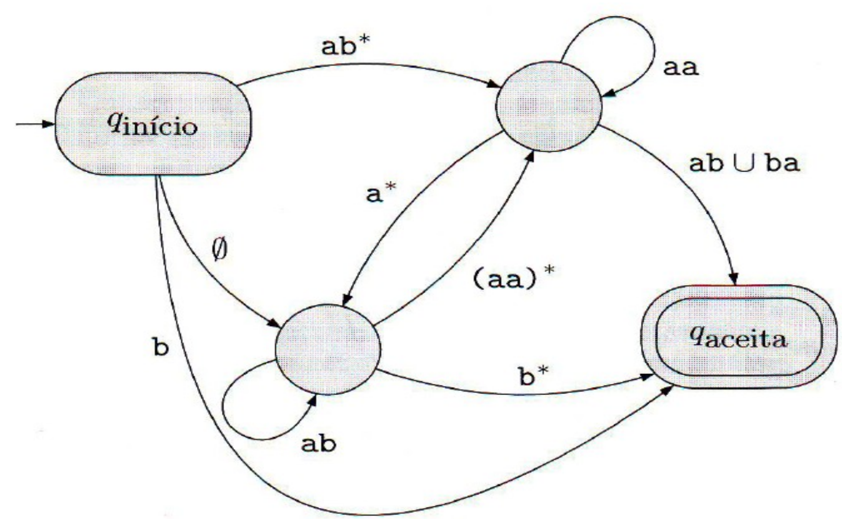
Rótulos das arestas podem ser expressões regulares



Autômatos Finitos Não-Determinísticos Generalizados (AFNGs)

Por conveniência, requeremos que os AFNGs tenham sempre um formato especial que atenda às seguintes condições:

- O estado inicial tem setas de transição saindo para todos os outros estados, mas nenhuma seta chegando de qualquer outro estado.
- Existe apenas um estado de aceitação, e ele tem setas chegando de todos os outros estados, mas nenhuma seta saindo para qualquer outro estado. Além disso, o estado de aceitação não é o mesmo que o estado inicial.
- Com exceção dos estados inicial e de aceitação, uma seta sai de cada estado para todos os outros e também de cada estado para ele mesmo.



Autômatos Finitos Não-Determinísticos Generalizados (AFNGs)

DEFINIÇÃO 1.64

Um ***autômato finito não-determinístico generalizado*** é uma 5-upla,
$(Q, \Sigma, \delta, q_{\text{início}}, q_{\text{aceita}})$, onde

1. Q é o conjunto finito de estados,
2. Σ é o alfabeto de entrada,
3. $\delta\colon (Q - \{q_{\text{aceita}}\}) \times (Q - \{q_{\text{início}}\}) \longrightarrow \mathcal{R}$ é a função de transição,
4. $q_{\text{início}}$ é o estado inicial, e
5. q_{aceita} é o estado de aceitação.

Autômatos Finitos Não-Determinísticos Generalizados (AFNGs)

Um AFNG aceita uma cadeia w em Σ^* se $w = w_1 w_2 \cdots w_k$, onde cada w_i está em Σ^*, e existe uma seqüência de estados $q_0, q_1, \ldots, q_k$ tal que

Autômatos Finitos Não-Determinísticos Generalizados (AFNGs)

Um AFNG aceita uma cadeia w em Σ^* se $w = w_1 w_2 \cdots w_k$, onde cada w_i está em Σ^*, e existe uma seqüência de estados $q_0, q_1, \ldots, q_k$ tal que

1. $q_0 = q_{\text{início}}$ é o estado inicial,
2. $q_k = q_{\text{aceita}}$ é o estado de aceitação, e
3. para cada i, temos $w_i \in L(R_i)$, onde $R_i = \delta(q_{i-1}, q_i)$; em outras palavras, R_i é a expressão sobre a seta de q_{i-1} a q_i.

Exp. regulares e autômatos finitos

- Qual a relação entre linguagens descritas por expressões regulares e linguagens reconhecidas por autômatos finitos?

Exp. regulares e autômatos finitos

- Qual a relação entre linguagens descritas por expressões regulares e linguagens reconhecidas por autômatos finitos?
  - Geram a mesma linguagem
  - Expressões regulares são equivalentes a autômatos fintos

Equivalência de ERs e AFs

TEOREMA 1.54

Uma linguagem é regular se e somente se alguma expressão regular a descreve.

Equivalência de ERs e AFs – Parte 1

LEMA 1.55

Se uma linguagem é descrita por uma expressão regular, então ela é regular.

Prova: vamos construir um AFN que reconheça L(R), e portanto L(R) será regular

Equivalência de ERs e AFs – Parte 1

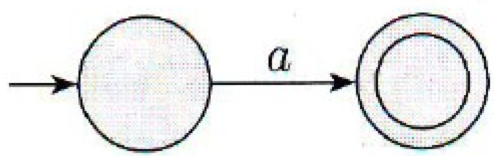
Digamos que R é uma ***expressão regular*** se R for

1. a para algum a no alfabeto Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, onde R_1 e R_2 são expressões regulares,
5. $(R_1 \circ R_2)$, onde R_1 e R_2 são expressões regulares, ou
6. (R_1^*), onde R_1 é uma expressão regular.

Equivalência de ERs e AFs – Parte 1

Digamos que R é uma ***expressão regular*** se R for

1. a para algum a no alfabeto Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, onde R_1 e R_2 são expressões regulares,
5. $(R_1 \circ R_2)$, onde R_1 e R_2 são expressões regulares, ou
6. (R_1^*), onde R_1 é uma expressão regular.



Equivalência de ERs e AFs – Parte 1

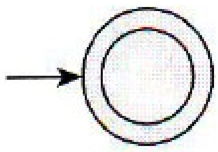
Digamos que R é uma ***expressão regular*** se R for

1. a para algum a no alfabeto Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, onde R_1 e R_2 são expressões regulares,
5. $(R_1 \circ R_2)$, onde R_1 e R_2 são expressões regulares, ou
6. (R_1^*), onde R_1 é uma expressão regular.

Equivalência de ERs e AFs – Parte 1

Digamos que R é uma ***expressão regular*** se R for

1. a para algum a no alfabeto Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, onde R_1 e R_2 são expressões regulares,
5. $(R_1 \circ R_2)$, onde R_1 e R_2 são expressões regulares, ou
6. (R_1^*), onde R_1 é uma expressão regular.



Equivalência de ERs e AFs – Parte 1

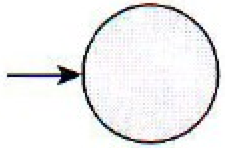
Digamos que R é uma ***expressão regular*** se R for

1. a para algum a no alfabeto Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, onde R_1 e R_2 são expressões regulares,
5. $(R_1 \circ R_2)$, onde R_1 e R_2 são expressões regulares, ou
6. (R_1^*), onde R_1 é uma expressão regular.

Equivalência de ERs e AFs – Parte 1

Digamos que R é uma ***expressão regular*** se R for

1. a para algum a no alfabeto Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, onde R_1 e R_2 são expressões regulares,
5. $(R_1 \circ R_2)$, onde R_1 e R_2 são expressões regulares, ou
6. (R_1^*), onde R_1 é uma expressão regular.



Equivalência de ERs e AFs – Parte 1

Digamos que R é uma ***expressão regular*** se R for

1. a para algum a no alfabeto Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, onde R_1 e R_2 são expressões regulares,
5. $(R_1 \circ R_2)$, onde R_1 e R_2 são expressões regulares, ou
6. (R_1^*), onde R_1 é uma expressão regular.

Equivalência de ERs e AFs – Parte 1

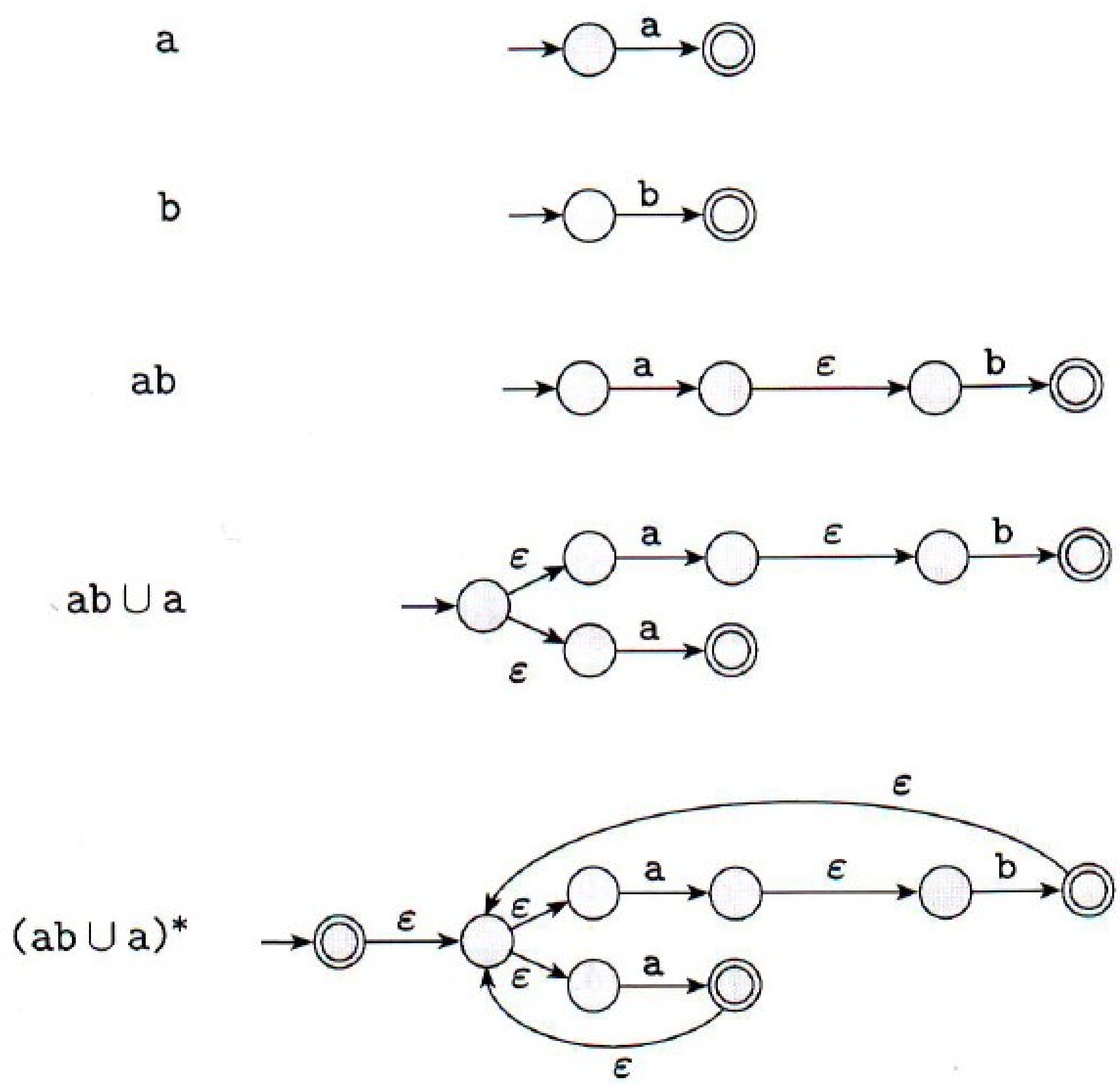
Digamos que R é uma ***expressão regular*** se R for

1. a para algum a no alfabeto Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, onde R_1 e R_2 são expressões regulares,
5. $(R_1 \circ R_2)$, onde R_1 e R_2 são expressões regulares, ou
6. (R_1^*), onde R_1 é uma expressão regular.

Provas de fechamento sob operações de união, concatenação e estrela

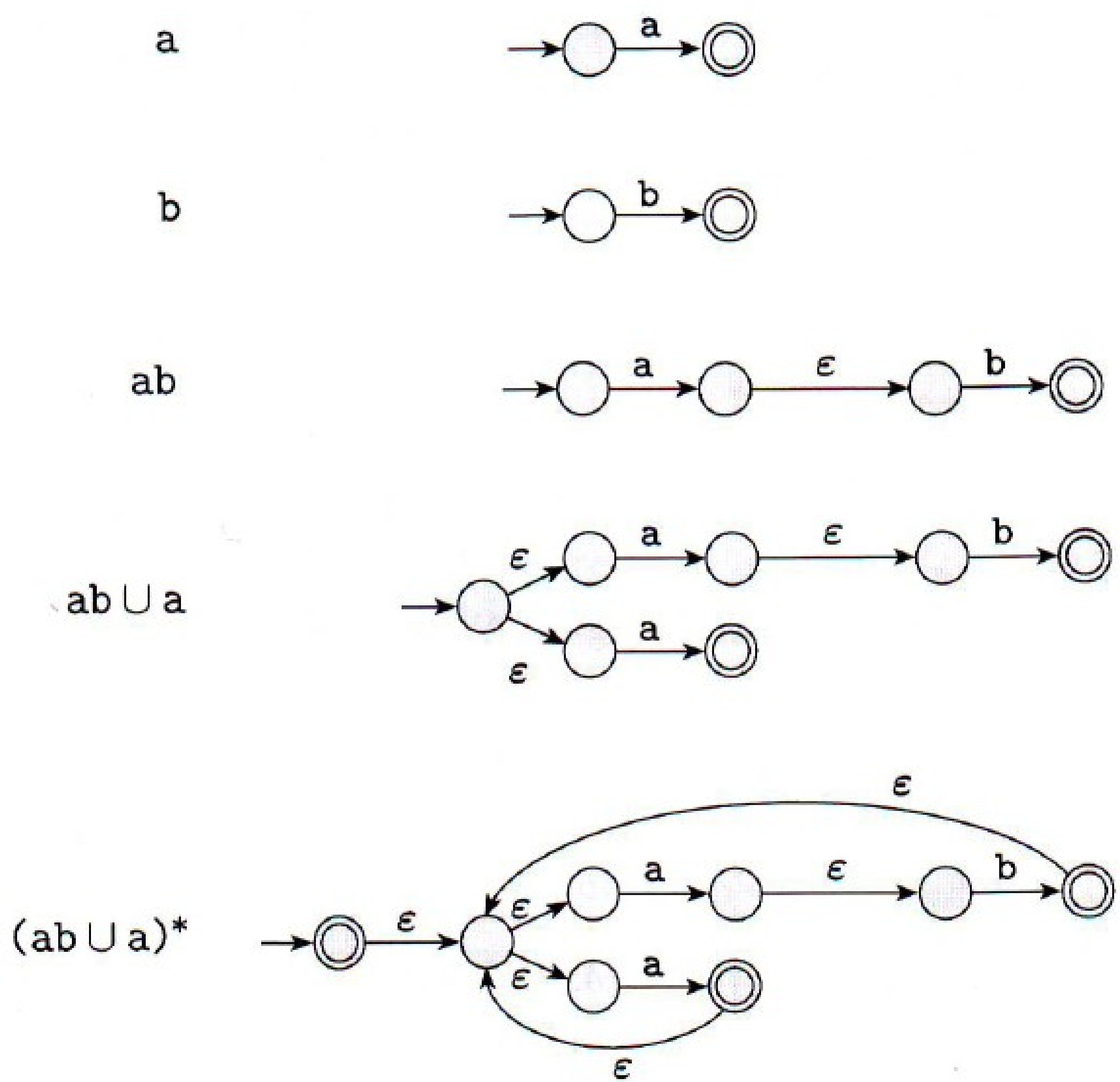
Equivalência de ERs e AFs – Parte 1

- Observação: essa prova fornece um mecanismo para construção de AFNs a partir de ERs.
- Ex: (ab U a)*
-
-



Equivalência de ERs e AFs – Parte 1

- Observação: essa prova fornece um mecanismo para construção de AFNs a partir de ERs.
- Ex: (ab U a)*
- O AFN resultante não necessariamente possui o número mínimo de estados
- Como seria o AFN com apenas 2 estados para essa expressão?



Equivalência de ERs e AFs – Parte 2

LEMA 1.60

Se uma linguagem é regular, então ela é descrita por uma expressão regular.

Ideia da Prova: se L é regular então um existe um AFD que a descreve. Se dado um AFD eu sempre conseguir escrever uma ER equivalente, então está feito.

Equivalência de ERs e AFs – Parte 2

- Vamos:

  1) mostrar como converter AFDs em AFNGs

  2) mostrar como converter AFNGs em ERs

Conversão de AFD em AFNG

- Novo estado inicial apontando para o antigo com uma seta ε
- Novo estado final com setas ε chegando dos estados finais antigos (que deixam de ser finais)
- Setas com múltiplos rótulos (ou múltiplas setas entre 2 estados na mesma direção) viram uma seta com a união dos rótulos
- Setas com rótulo Ø onde não havia setas (e deveria ter no AFNG)

Equivalência de ERs e AFs – Parte 2

Conversão de AFNG em ER

- O AFNG tem k estados, k >= 2 (estados inicial e de aceitação são distintos)
- Se k = 2, há só uma aresta contendo a expressão regular que descreve a linguagem reconhecida pelo AFNG
- Se k > 2, construímos um AFNG equivalente com k-1 estados
- Continuo o processo até que k =2

Equivalência de ERs e AFs – Parte 2

Conversão de AFNG em ER

- O AFNG tem k estados, k >= 2 (estados inicial e de aceitação são distintos)
- Se k = 2, há só uma aresta contendo a expressão regular que descreve a linguagem reconhecida pelo AFNG
- Se k > 2, construímos um AFNG equivalente com k-1 estados
- Continuo o processo até que k =2

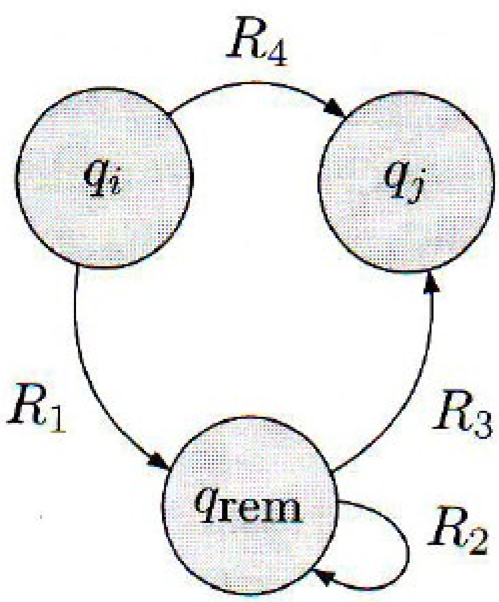
Equivalência de ERs e AFs – Parte 2

Conversão de AFNG em ER

- Construindo um AFNG com k-1 estados:
  - Escolha q_{rem} (estado não inicial e não final)
  - Remova q_{rem}
  - Ajuste a expressão de CADA transição $q_i \rightarrow q_j$ de forma a ter compensar a remoção de q_{rem}

Conversão de AFNG em ER

- Ex:

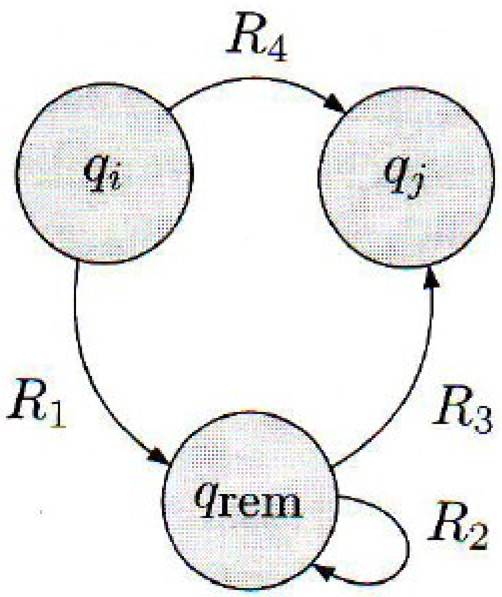


antes

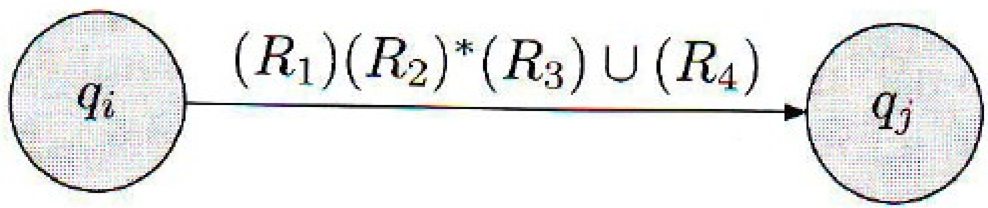
Equivalência de ERs e AFs – Parte 2

Conversão de AFNG em ER

- Ex:



antes



depois

Conversão de AFNG em ER

Também para $q_i \leftarrow q_j$ e quando $i = j$

Equivalência de ERs e AFs – Parte 2

Conversão de AFNG em ER

CONVERT(G):

1. Seja k o número de estados de G.
2. Se $k = 2$, então G deve consistir de um estado inicial, um estado de aceitação, e uma única seta conectando os dois rotulada com uma expressão regular R.
   Retorne a expressão R.
3. Se $k > 2$, selecionamos qualquer $q_{\text{rem}} \in Q$ diferente de $q_{\text{início}}$ e de q_{aceita} e seja G' o AFNG $(Q', \Sigma, \delta', q_{\text{início}}, q_{\text{aceita}})$, onde

$$Q' = Q - \{q_{\text{rem}}\},$$

e para qualquer $q_i \in Q' - \{q_{\text{aceita}}\}$ e qualquer $q_j \in Q' - \{q_{\text{início}}\}$ seja

$$\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4),$$

para $R_1 = \delta(q_i, q_{\text{rem}})$, $R_2 = \delta(q_{\text{rem}}, q_{\text{rem}})$, $R_3 = \delta(q_{\text{rem}}, q_j)$ e $R_4 = \delta(q_i, q_j)$.

4. Compute CONVERT(G') e retorne esse valor.

AFIRMATIVA 1.65 ....

Para qualquer AFNG G, CONVERT(G) é equivalente a G.

- Prova: por indução sobre k (número de estados de G)
  - **Base**: se k = 2

Base: Prove que a afirmação é verdadeira para $k = 2$ estados. Se G tem apenas dois estados, ele só pode ter uma única seta, que vai do estado inicial para o estado de aceitação. A expressão regular que é o rótulo sobre essa seta descreve todas as cadeias que propiciam a G chegar ao estado de aceitação. Logo, essa expressão é equivalente a G.

AFIRMATIVA 1.65

Para qualquer AFNG G, CONVERT(G) é equivalente a G.

- Prova: por indução sobre k (número de estados de G
  - **Indução**: vale para k-1 estados. Vamos provar que vale para k estados

    Ou seja, vamos mostrar que cada chamada da recursão produz um autômato equivalente a G

AFIRMATIVA 1.65

Para qualquer AFNG G, CONVERT(G) é equivalente a G.

- Prova: por indução sobre k (número de estados de G
  - **Indução**: vale para k-1 estados. Vamos provar que vale para k estados
    - Primeiro: provamos que, se G reconhece uma cadeia w, G' = CONVERT(G) também reconhece (<=)
    - Segundo: provamos que, se G' reconhece uma cadeia w, G também reconhece (=>)

AFIRMATIVA 1.65

Para qualquer AFNG G, CONVERT(G) é equivalente a G.

- Prova: por indução sobre k (número de estados de G
  - **Indução**: vale para k-1 estados. Vamos provar que vale para k estados
    - Primeiro: provamos que, se G reconhece uma cadeia w, G' = CONVERT(G) também reconhece (<=)
    - Segundo: provamos que, se G' reconhece uma cadeia w, G também reconhece (=>)

AFIRMATIVA 1.65 ....

Para qualquer AFNG G, CONVERT(G) é equivalente a G.

- Prova: por indução sobre k (número de estados de G
  - **Indução**: vale para k-1 estados. Vamos provar que vale para k estados (vou remover um estado)
    - Primeiro: provamos que, se G reconhece uma cadeia w, G' = CONVERT(G) também reconhece (<=)

G reconhece uma cadeia w => existe pelo menos um caminho por G

$q_{início}, q_1, q_2, \ldots, q_{aceita}$

AFIRMATIVA 1.65

Para qualquer AFNG G, CONVERT(G) é equivalente a G.

- Prova: por indução sobre k (número de estados de G
  - **Indução**: vale para k-1 estados. Vamos provar que vale para k estados (vou remover um estado)
    - Primeiro: provamos que, se G reconhece uma cadeia w, G' = CONVERT(G) também reconhece (<=)

G reconhece uma cadeia w => existe pelo menos um caminho por G

$q_{início}, q_1, q_2, \ldots, q_{aceita}$

Se nenhum desses estados é q_{rem}, G' aceita w (pois as expressões antigas (de G) rotulando cada transição desse caminho estão contidas nas novas expressões de G', como parte da união)

AFIRMATIVA 1.65

Para qualquer AFNG G, CONVERT(G) é equivalente a G.

- Prova: por indução sobre k (número de estados de G
  - **Indução**: vale para k-1 estados. Vamos provar que vale para k estados (vou remover um estado)
    - Primeiro: provamos que, se G reconhece uma cadeia w, G' = CONVERT(G) também reconhece (<=)

G reconhece uma cadeia w => existe pelo menos um caminho por G

$q_{início}, q_1, q_2, \ldots, q_{aceita}$

Se nenhum desses estados é q_{rem}, G' aceita w (pois as expressões antigas (de G) rotulando cada transição desse caminho estão contidas nas novas expressões de G', como parte da união)

Se o caminho contém q_{rem}, cada retirada de q_{rem}'s consecutivos não altera o fato de G' aceitar w, pois q_i/q_j anterior/posterior a essa série q_{rem}'s possuem em G' uma transição q_i->q_j com uma expressão que descreve cadeias que levam de q_i a q_j via q_{rem}

Logo, G' também reconhece w

AFIRMATIVA 1.65

Para qualquer AFNG G, CONVERT(G) é equivalente a G.

- Prova: por indução sobre k (número de estados de G
  - **Indução**: vale para k-1 estados. Vamos provar que vale para k estados
    - Primeiro: provamos que, se G reconhece uma cadeia w, G' = CONVERT(G) também reconhece (<=)
    - Segundo: provamos que, se G' reconhece uma cadeia w, G também reconhece (=>)

Cada transição de G' q_i->q_j descreve cadeias que, em G, vão de q_i a q_j diretamemente ou via q_{rem}

Logo, G reconhece w.

AFIRMATIVA 1.65

Para qualquer AFNG G, CONVERT(G) é equivalente a G.

- Prova: por indução sobre k (número de estados de G
  - **Indução**: vale para k-1 estados. Vamos provar que vale para k estados
    - Primeiro: provamos que, se G reconhece uma cadeia w, G' = CONVERT(G) também reconhece (<=)
    - Segundo: provamos que, se G' reconhece uma cadeia w, G também reconhece (=>)

Cada transição de G' q_i->q_j descreve cadeias que, em G, vão de q_i a q_j diretamente ou via q_{rem}

Logo, G reconhece w.

LOGO...

AFIRMATIVA 1.65

Para qualquer AFNG G, CONVERT(G) é equivalente a G.

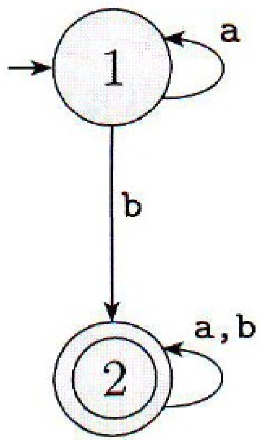
LEMA 1.60

Se uma linguagem é regular, então ela é descrita por uma expressão regular.

TEOREMA 1.54

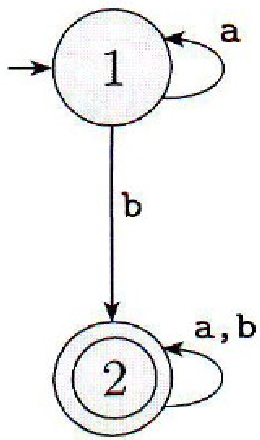
Uma linguagem é regular se e somente se alguma expressão regular a descreve.

Exemplo

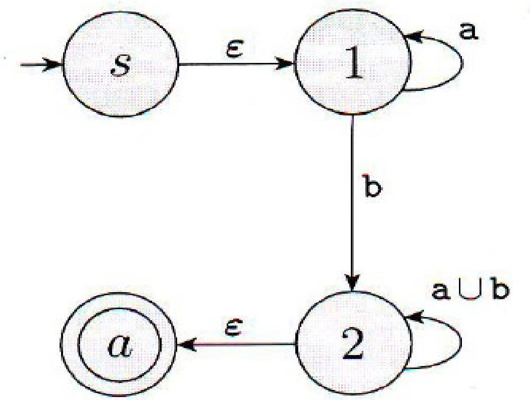


(a)

Exemplo

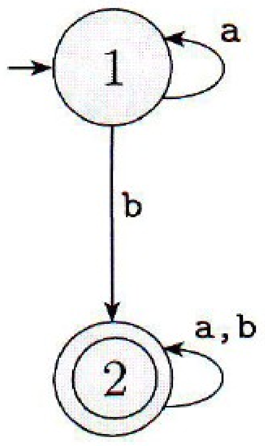


(a)

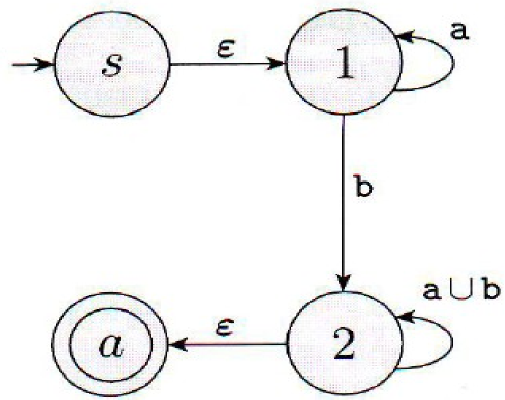


(b)

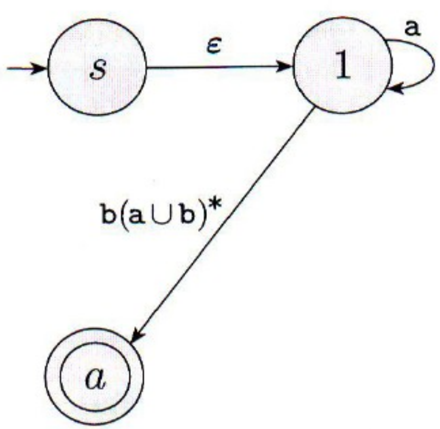
Exemplo



(a)

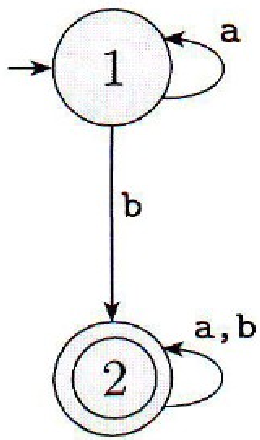


(b)

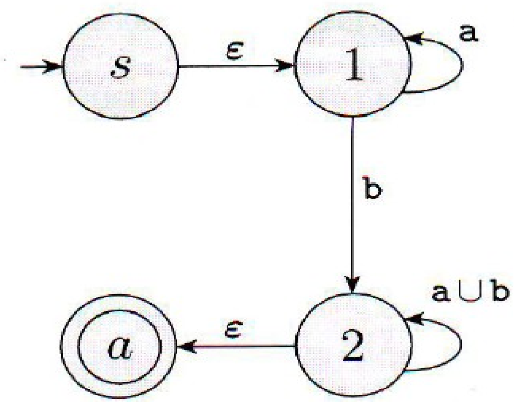


(c)

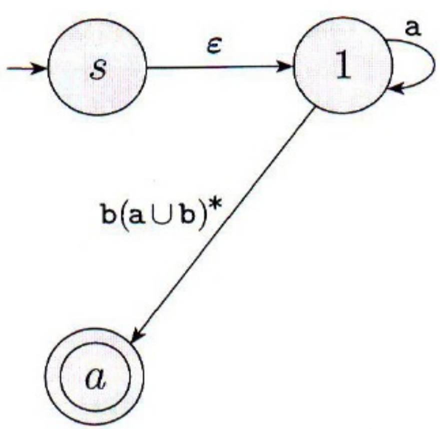
Exemplo



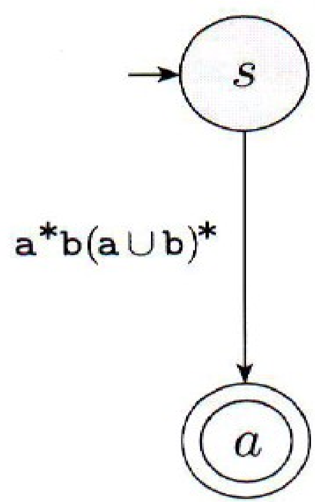
(a)



(b)



(c)



(d)

Aplicações de ERs

- Programas utilitários (ex: grep, awk)
- Linguagens de programação (ex: perl, python)
- Compiladores - definição de tokens para geração de analisadores léxicos
  - Ex: definição de uma constante numérica:

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

    Onde D = {0,1,2,3,4,5,6,7,8,9}