MAC 414
Autômatos, Computabilidade e Complexidade
aula 5 — 28/09/2020

O problema da pertinência

O problema da pertinência

Dados: uma linguagem L e uma palavra x, decidir

$$x \in L\,?$$

O problema da pertinência

Dados: uma linguagem L e uma palavra x, decidir

$$x \in L\,?$$

Se L é dada por um AD, fácil.

O problema da pertinência

Dados: uma linguagem L e uma palavra x, decidir

$$x \in L\,?$$

Se L é dada por um AD, fácil.

Se L é dada por um ER, é preciso, em princípio, encontrar uma fatoração adequada de x:

$$L : ((aa + baa)^* ba (bb + aaa)^* + bbb)^*$$

$$x : baaaaaabababbbbaaabbbaababbb$$

O problema da pertinência

Dados: uma linguagem L e uma palavra x, decidir

$$x \in L\,?$$

Se L é dada por um AD, fácil.

Se L é dada por um ER, é preciso, em princípio, encontrar uma fatoração adequada de x:

$$L : ((aa + baa)^* ba (bb + aaa)^* + bbb)^*$$
$$x : baaaaaababbbbbaaabbbaababbb$$

E como garantir que não tem, se for o caso?

Pertinência em AND

Pertinência em AND

Versão 1: sem arestas λ

Pertinência em AND

Versão 1: sem arestas λ

Idéia: lembrar onde poderia estar depois de ler cada caractere.

Data: $\mathscr{A}$ (sem λ), x

declare Q *para conter um conjunto de estados*

$Q \leftarrow S$

while $\sigma \leftarrow$ *nova letra* **do**

    Invariante: Q = *estados alcançados com o rótulo lido*

end

Data: $\mathscr{A}$ (sem λ), x

declare Q *para conter um conjunto de estados*

$Q \leftarrow S$

while $\sigma \leftarrow$ *nova letra* **do**

Invariante: Q = *estados alcançados com o rótulo lido*

Atualize Q

end

return $Q \cap F \neq \emptyset$

Data: $\mathscr{A}$ (sem λ), x

declare Q para conter um conjunto de estados

$Q \leftarrow S$

while $\sigma \leftarrow$ *nova letra* **do**

Invariante: Q = estados alcançados com o rótulo lido

Atualize Q

$Q' \leftarrow \emptyset$

for $q \in Q$ **do**

for $(q, \sigma, p) \in \Delta$ **do**

$Q' \leftarrow Q' \cup \{p\}$

end

end

$Q \leftarrow Q'$

end

return $Q \cap F \neq \emptyset$

Data: $\mathscr{A}$ (sem λ), x

declare Q para conter um conjunto de estados

$Q \leftarrow S$

while $\sigma \leftarrow$ *nova letra* **do**

  Invariante: Q = estados alcançados com o rótulo lido

  Atualize Q

  $Q' \leftarrow \emptyset$

  for $q \in Q$ **do**

    for $(q, \sigma, p) \in \Delta$ **do**

      $Q' \leftarrow Q' \cup \{p\}$

    end

  end

  $Q \leftarrow Q'$

end

return $Q \cap F \neq \emptyset$

Tempo: $\mathcal{O}(|\Delta|)$ por letra.

Como lidar com λ?

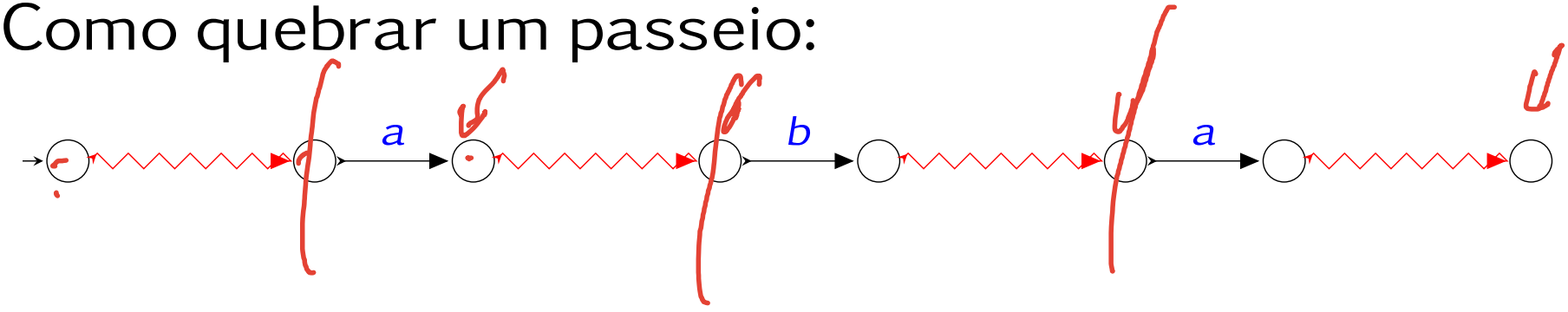
Como lidar com λ?

O invariante já falha no começo.

Como lidar com λ?

O invariante já falha no começo.

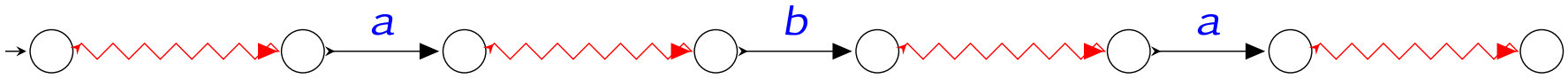
Como quebrar um passeio:



Como lidar com λ?

O invariante já falha no começo.

Como quebrar um passeio:



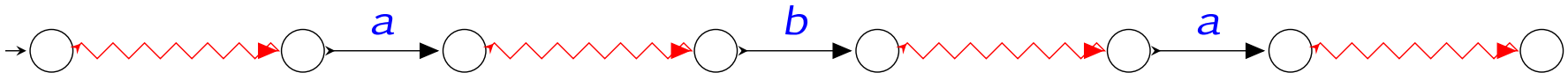
Precisamos da função

$$\text{Fecha}(\mathscr{A}, Q)$$

Como lidar com λ?

O invariante já falha no começo.

Como quebrar um passeio:



Precisamos da função

$$\text{Fecha}(\mathscr{A}, Q)$$

a partir de Q

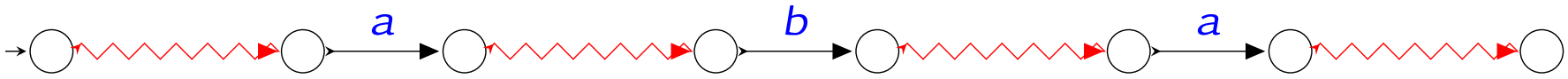
que devolve

todos estados atingíveis por passeios de rótulo λ

Como lidar com λ?

O invariante já falha no começo.

Como quebrar um passeio:



Precisamos da função

$$\text{Fecha}(\mathscr{A}, Q)$$

que devolve

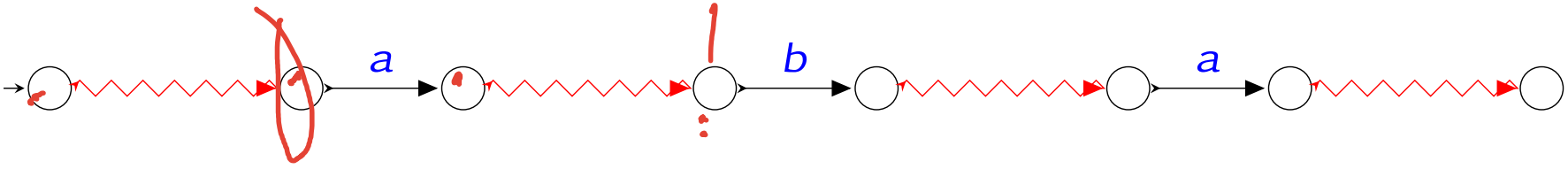
todos estados atingíveis por passeios de rótulo λ

Implementação:

Como lidar com λ?

O invariante já falha no começo.

Como quebrar um passeio:



Precisamos da função

$$\text{Fecha}(\mathscr{A}, Q)$$

a partir de Q

que devolve

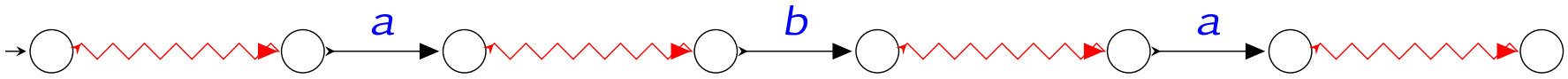
todos estados atingíveis por passeios de rótulo λ

Implementação: uma busca no subgrafo de $G_{\mathscr{A}}$ formado pelas arestas λ.

Como lidar com λ?

O invariante já falha no começo.

Como quebrar um passeio:



Precisamos da função

$$\text{Fecha}(\mathscr{A}, Q)$$

que devolve

todos estados atingíveis por passeios de rótulo λ

Implementação: uma busca no subgrafo de $G_{\mathscr{A}}$ formado pelas arestas λ.

Tempo linear no número de arestas λ.

Data: $\mathscr{A}$, x

declare Q *para conter um conjunto de estados*

$Q \leftarrow \text{Fecha}(\mathscr{A}, S)$

while $\sigma \leftarrow$ *nova letra* **do**

    Invariante: Q = *estados alcançados com o rótulo lido*

end

Data: $\mathscr{A}$, x

declare Q *para conter um conjunto de estados*

$Q \leftarrow \text{Fecha}(\mathscr{A}, S)$

while $\sigma \leftarrow$ *nova letra* **do**

Invariante: Q = *estados alcançados com o rótulo lido*

Atualize Q

end

return $Q \cap F \neq \emptyset$

Data: $\mathscr{A}$, x

declare Q *para conter um conjunto de estados*

$Q \leftarrow \text{Fecha}(\mathscr{A}, S)$

while $\sigma \leftarrow$ *nova letra* **do**

  Invariante: Q = *estados alcançados com o rótulo lido*

  Atualize Q

  $Q' \leftarrow \emptyset$

  for $q \in Q$ **do**

    for $(q, \sigma, p) \in \Delta$ **do**

      $Q' \leftarrow Q' \cup \{p\}$

    end

  end

  $Q \leftarrow Fecha(Q')$

end

return $Q \cap F \neq \emptyset$

Data: $\mathscr{A}$, x

declare Q *para conter um conjunto de estados*

$Q \leftarrow \text{Fecha}(\mathscr{A}, S)$

while $\sigma \leftarrow$ *nova letra* **do**

  Invariante: Q = *estados alcançados com o rótulo lido*

  Atualize Q

  $Q' \leftarrow \emptyset$

  for $q \in Q$ **do**

    for $(q, \sigma, p) \in \Delta$ **do**

      $Q' \leftarrow Q' \cup \{p\}$

    end

  end

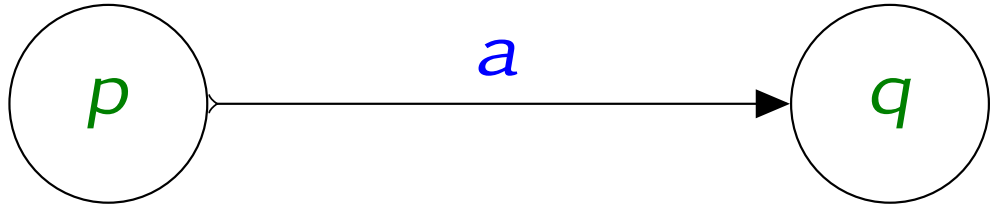
  $Q \leftarrow \mathit{Fecha}(Q')$

end

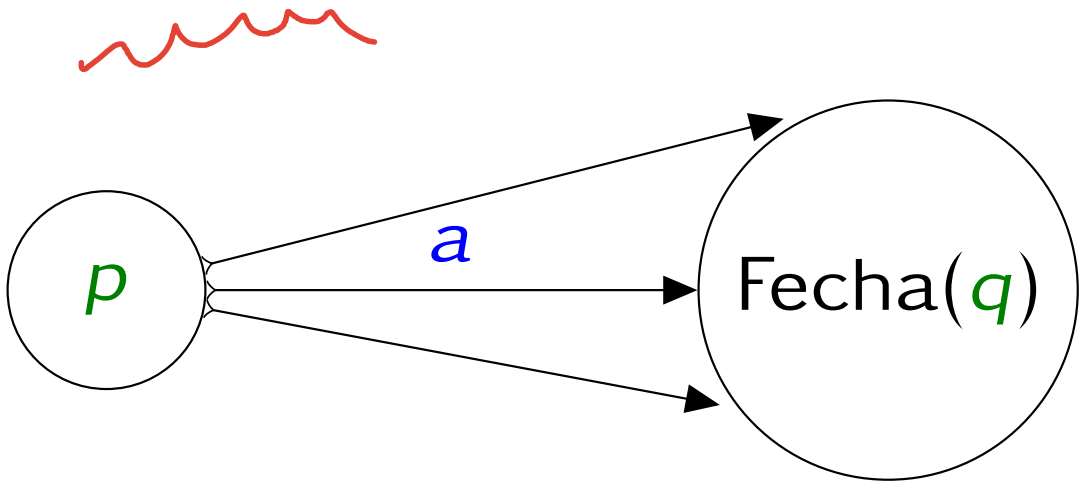
return $Q \cap F \neq \emptyset$

Tempo: $\mathcal{O}(|\Delta|)$ por letra.

Eliminação de λ



Eliminação de λ



Exemplo

Relembrando a construção econômica (existe a construção de Glushkov que é ainda mais econômica):

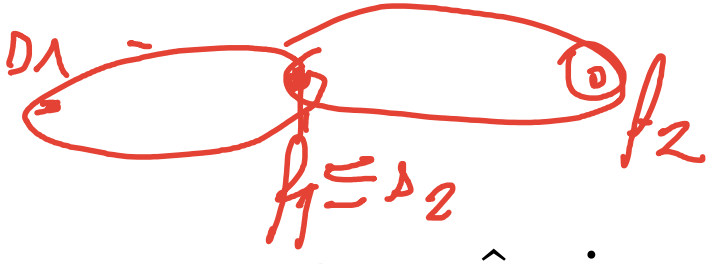
Exemplo

Relembrando a construção econômica (existe a construção de Glushkov que é ainda mais econômica):

Produto

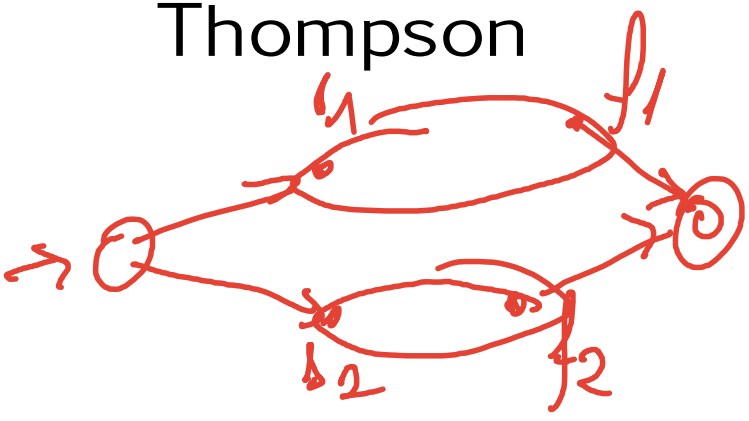
Thompson

econômico

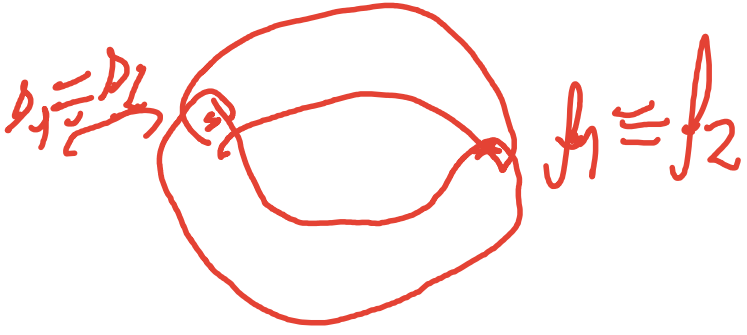


União

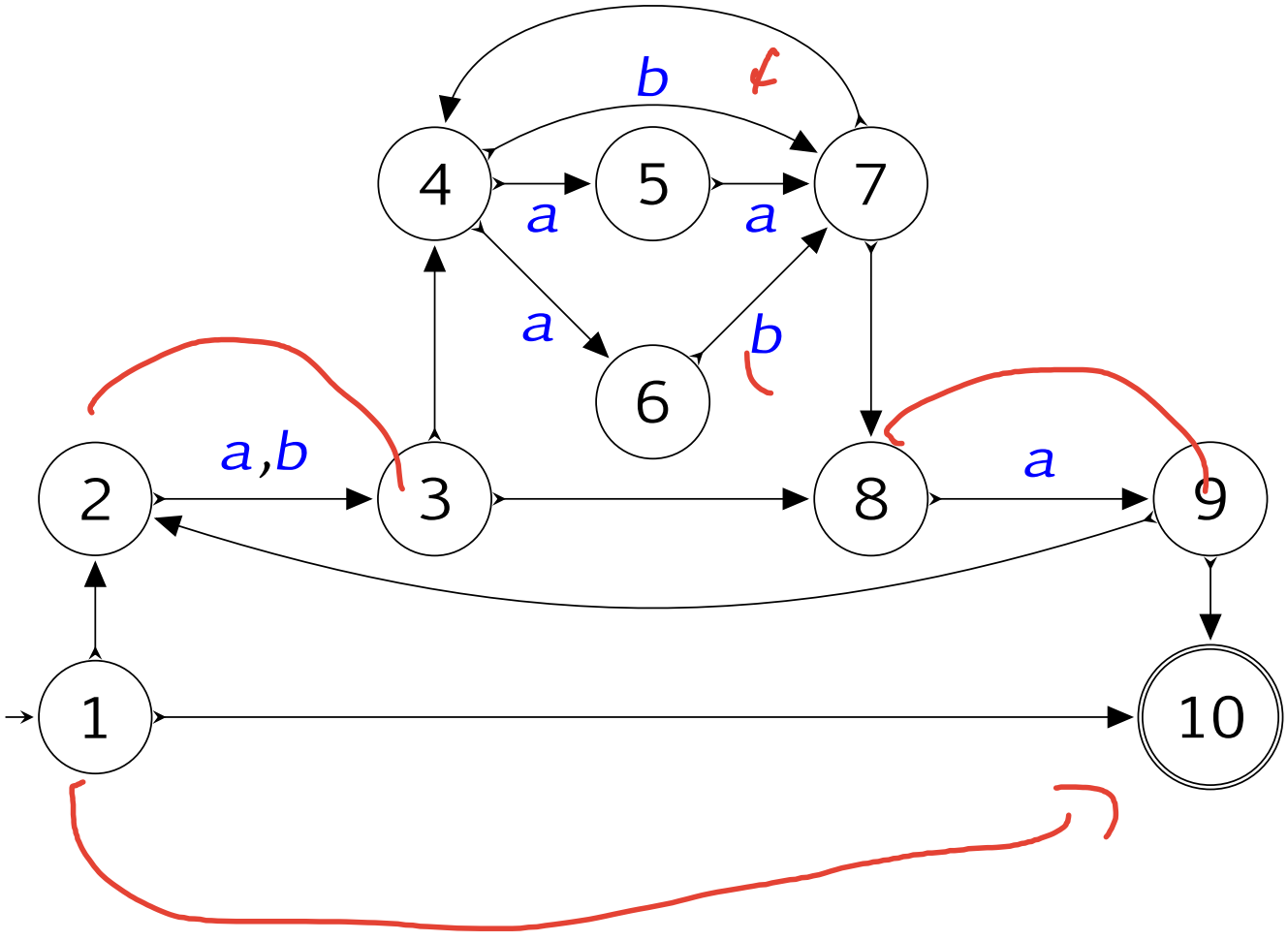
Thompson



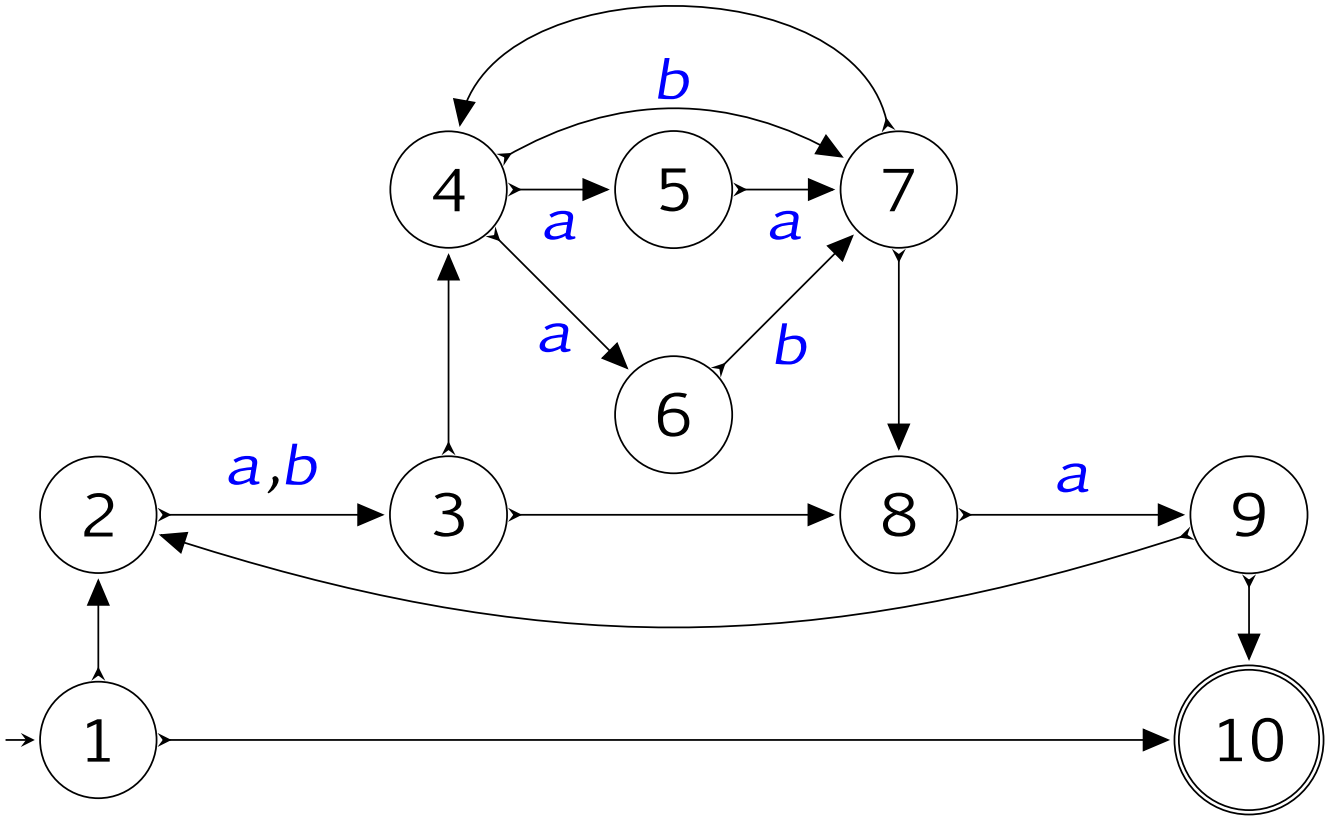
econômico



$$((a+b)(b+aa+ab)^* a)^*$$



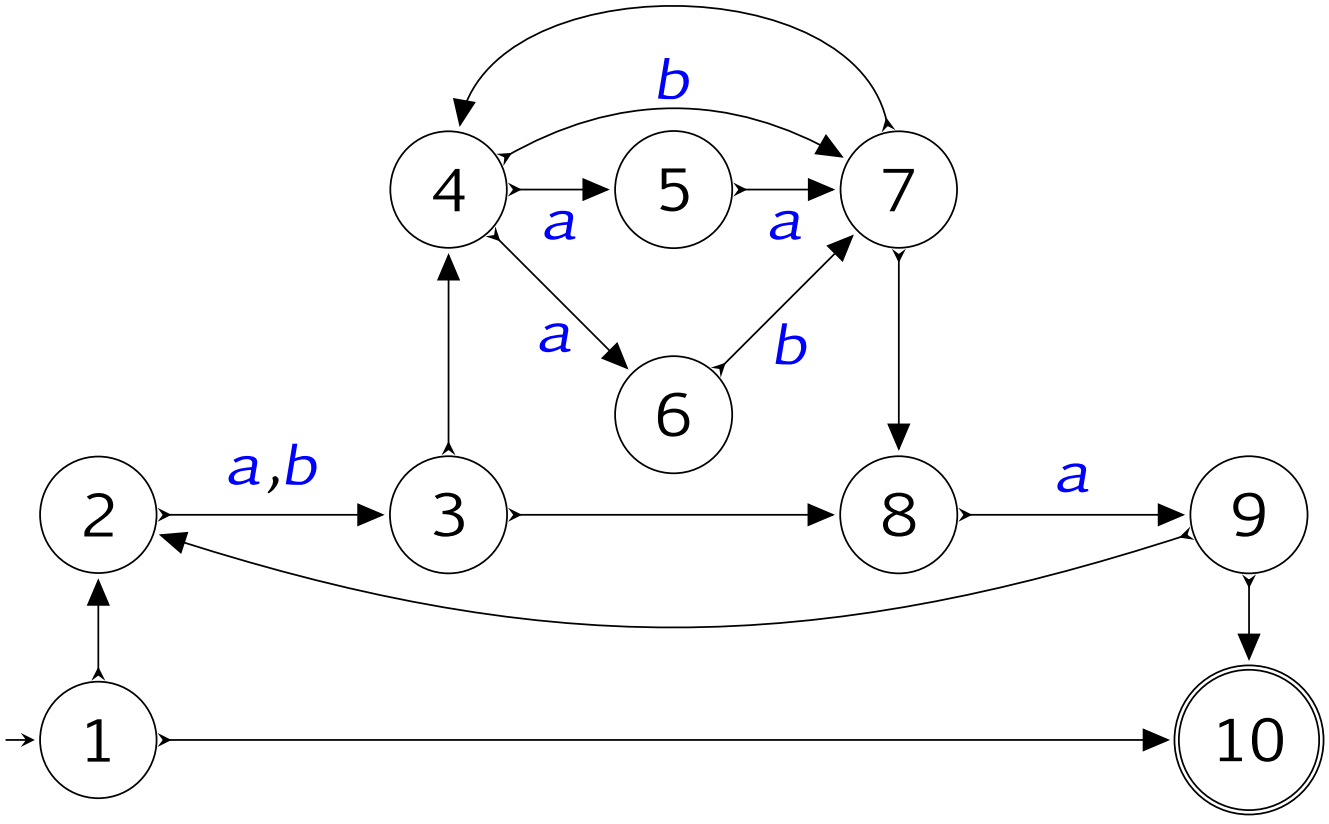
$((a+b)(b+aa+ab)^* a)^*$



$ababa$

Q

$((a+b)(b+aa+ab)^* a)^*$

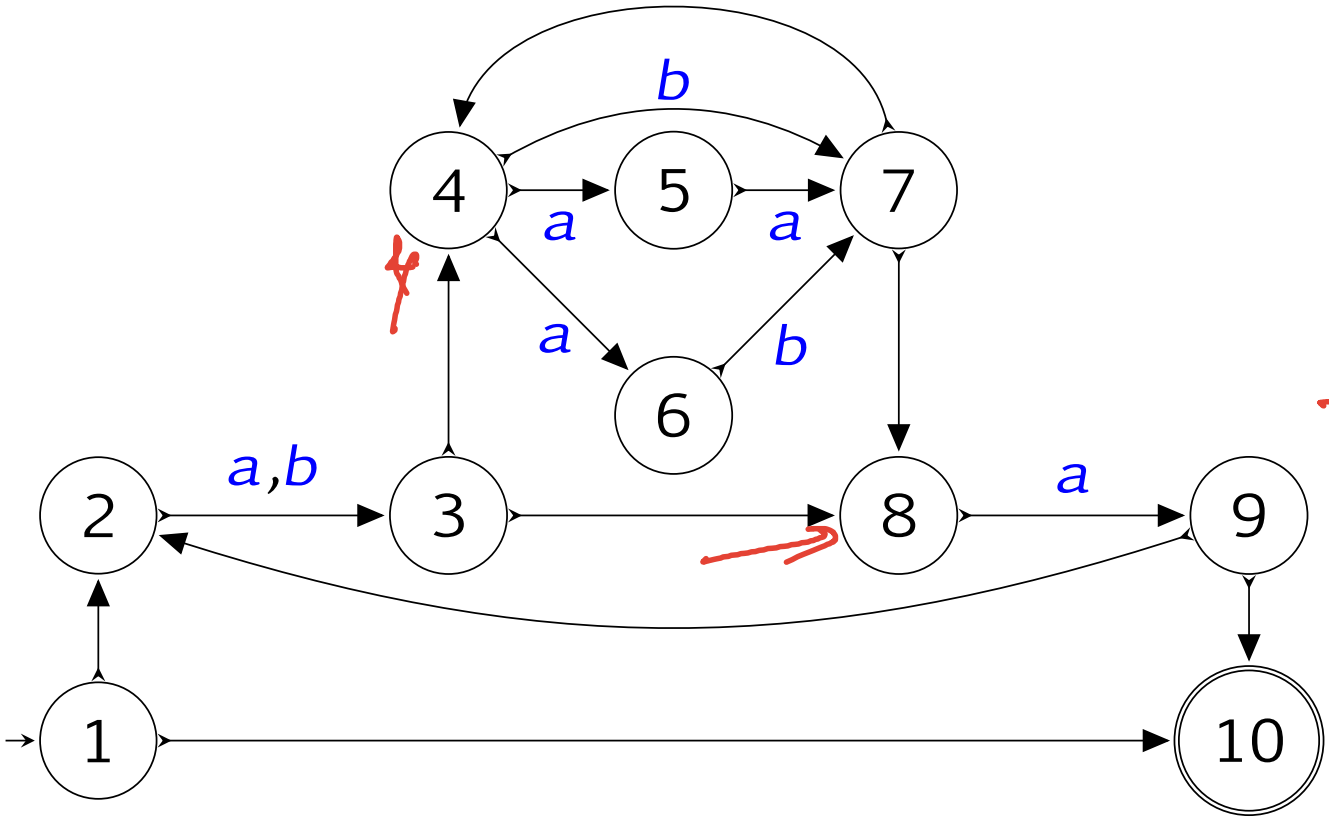


$ababa$

Q

1,2,10

$((a+b)(b+aa+ab)^* a)^*$



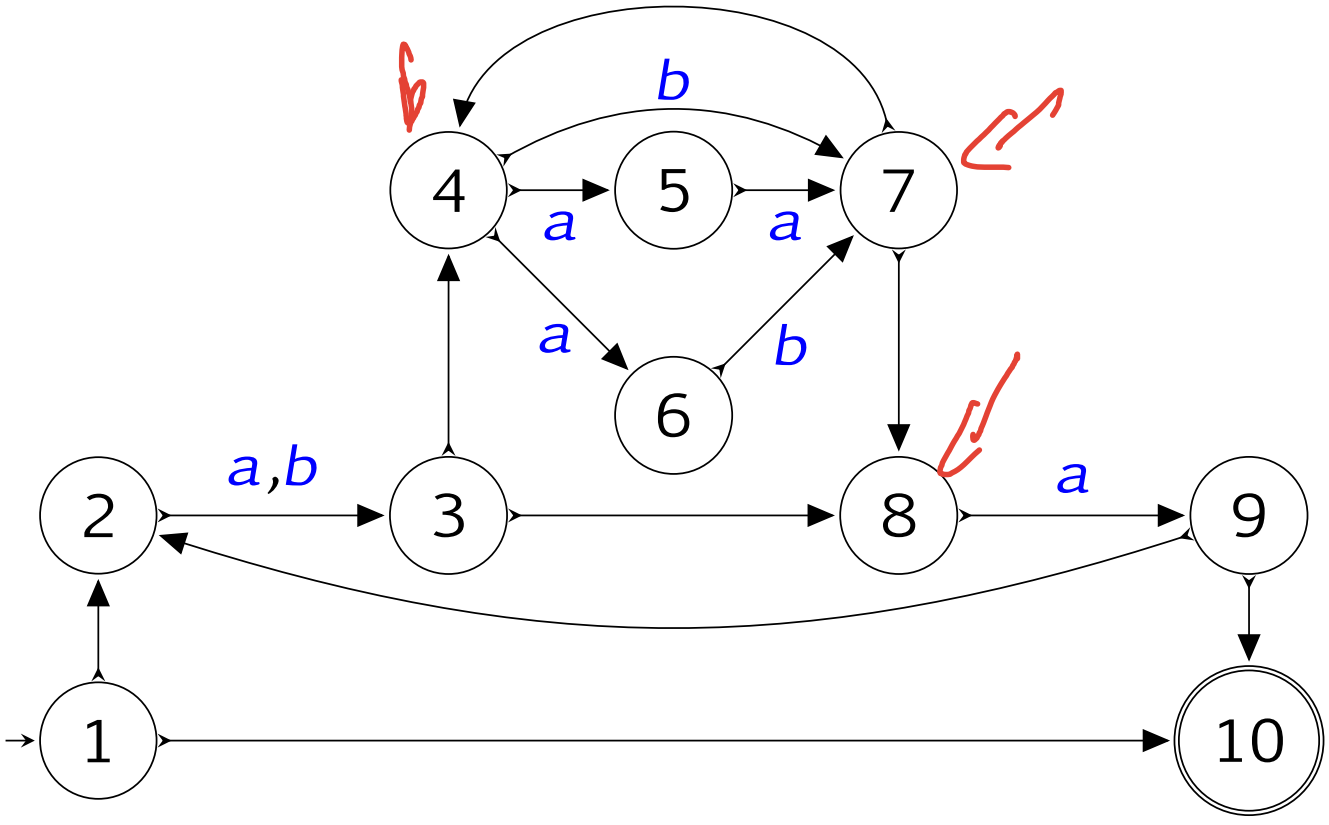
ababa

Q

1,2,10

3,4,8

$((a+b)(b+aa+ab)^* a)^*$



ababa

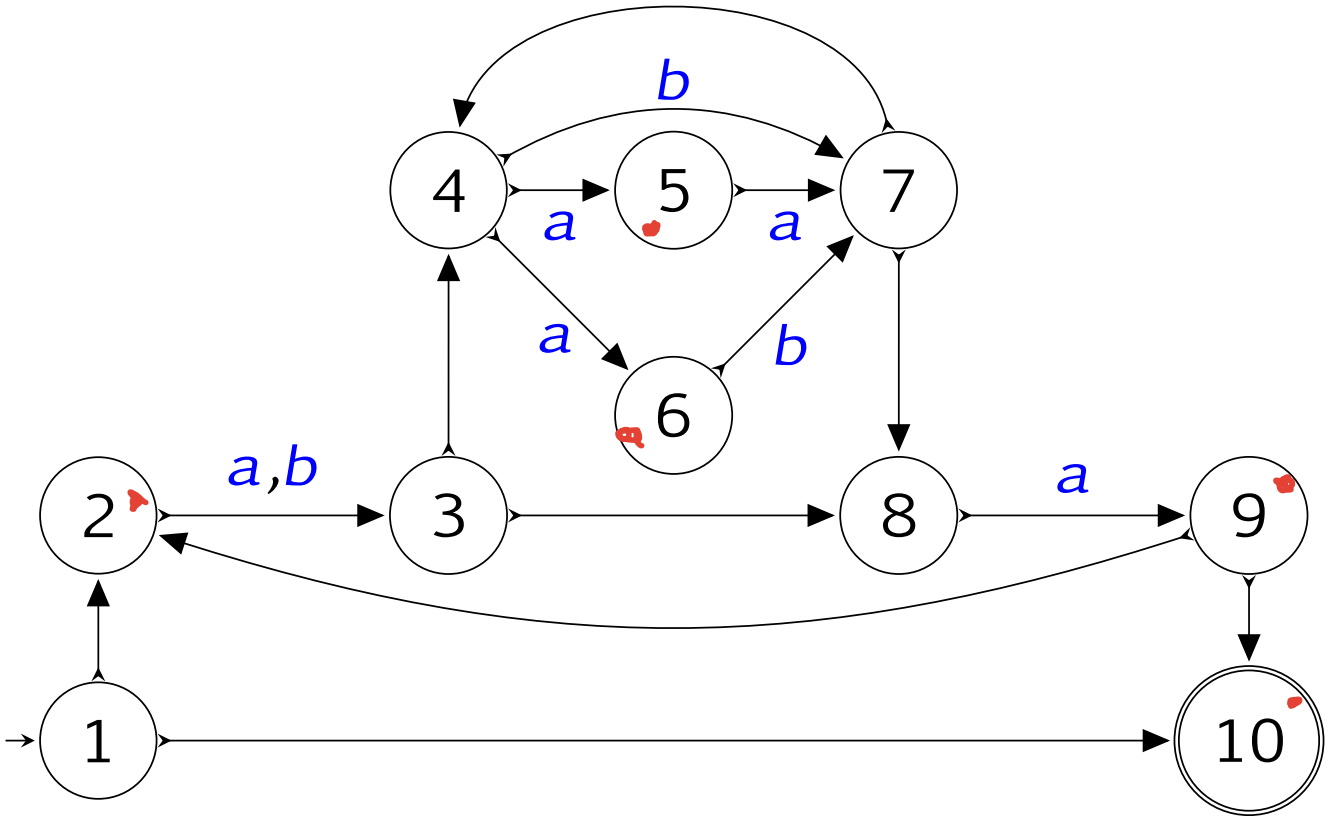
Q

1,2,10

3,4,8

7,4,8

$((a+b)(b+aa+ab)^*a)^*$



ababa

Q

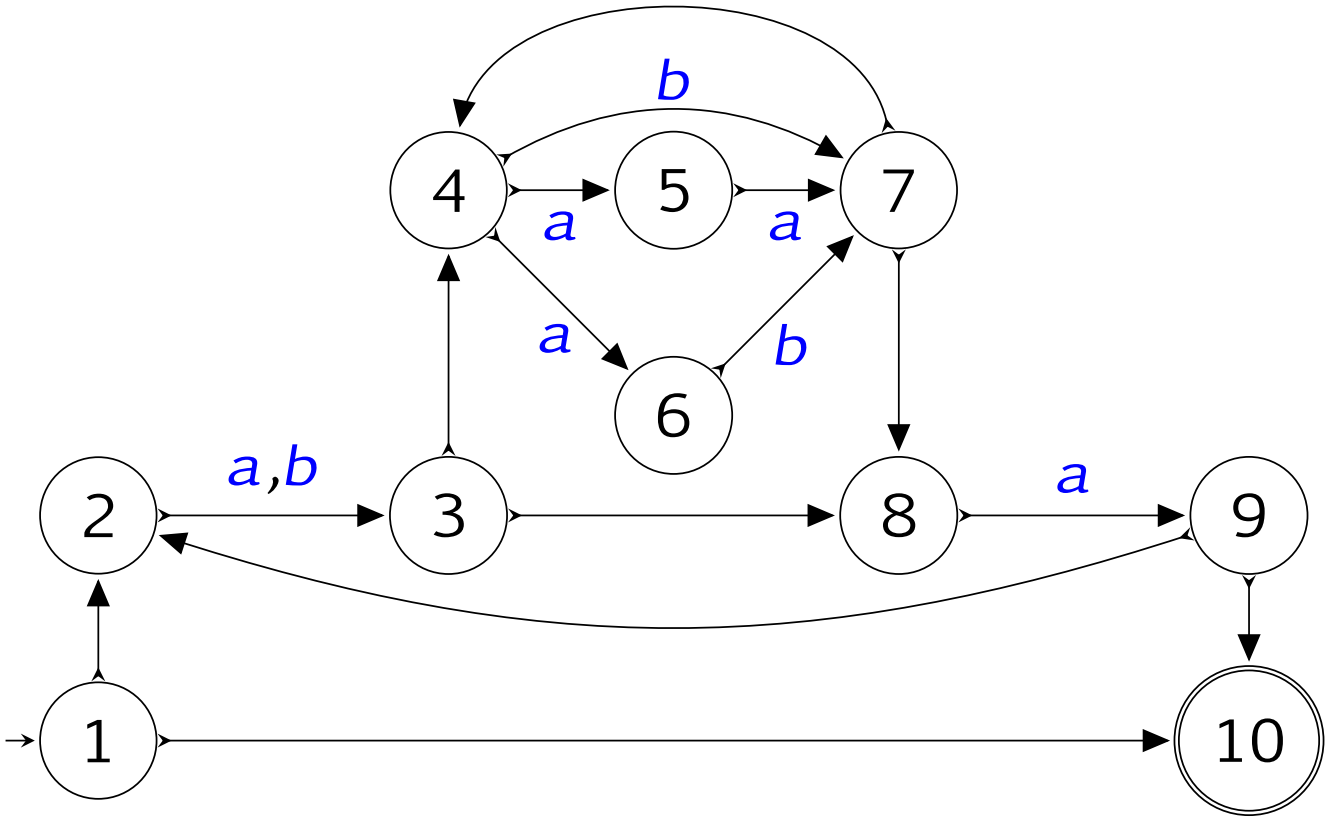
1,2,10

3,4,8

7,4,8

5,6,9,2,10

$((a+b)(b+aa+ab)^* a)^*$



ababa

Q

1,2,10

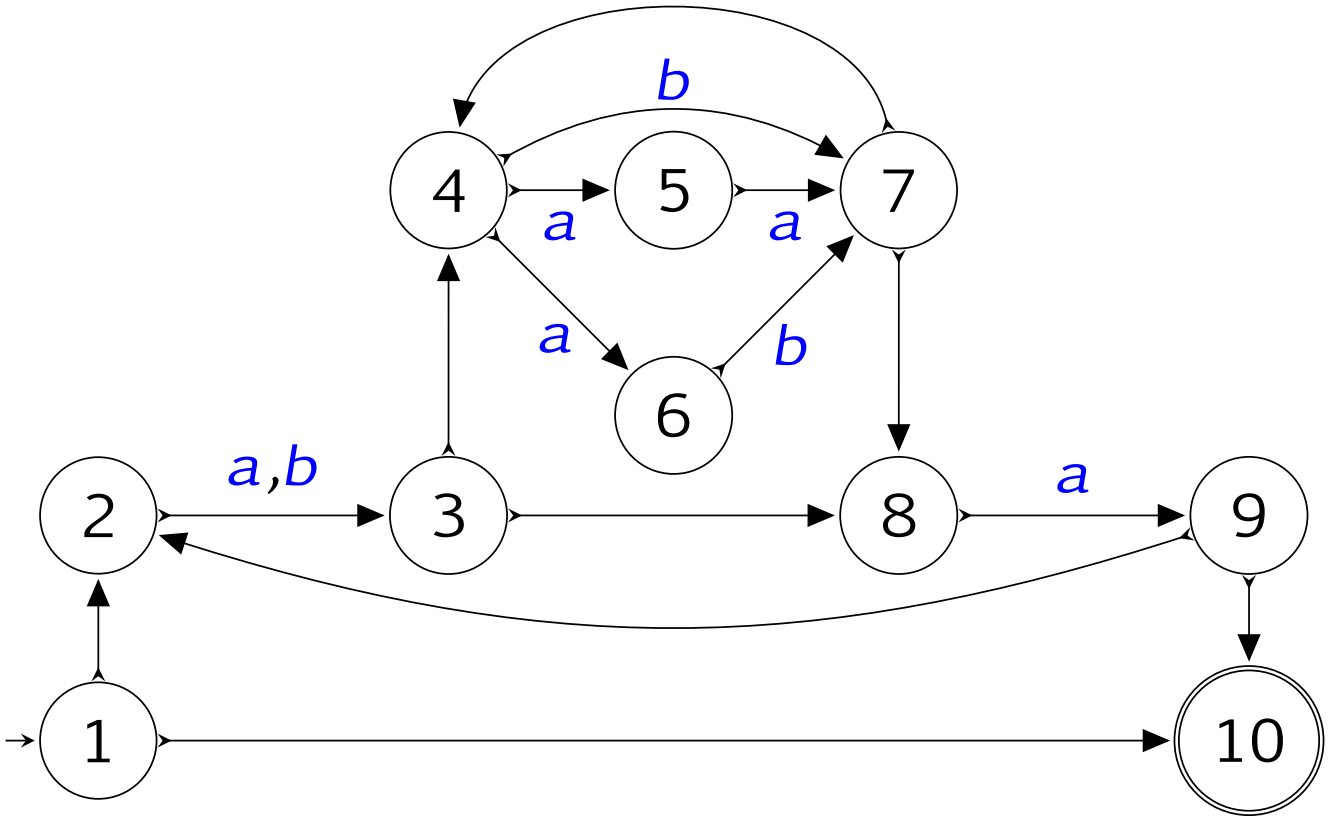
3,4,8

7,4,8

5,6,9,2,10

7,3,4,6,8

$((a+b)(b+aa+ab)^*a)^*$



ababa

Q

1,2,10

3,4,8

7,4,8

5,6,9,2,10

7,3,4,6,8

5,6,9,2,10

A construção dos subconjuntos

Versão sem λ

A construção dos subconjuntos

Versão sem λ

Dado um AND $\mathscr{A} = (K, \Sigma, \Delta, S, F)$ sem transições λ, o autômato dos subconjuntos de $\mathscr{A}$ é

$$\hat{\mathscr{A}} = (2^K, \Sigma, \delta, S, \mathscr{F})$$

onde

$$\delta(Q, \sigma) = \{p \in K \mid \exists q \in Q \text{ tq } (q, \sigma, p) \in \Delta\}.$$

A construção dos subconjuntos

Versão sem λ

Dado um AND $\mathscr{A} = (K, \Sigma, \Delta, S, F)$ sem transições λ, o autômato dos subconjuntos de $\mathscr{A}$ é

$$\hat{\mathscr{A}} = (2^K, \Sigma, \delta, S, \mathscr{F})$$

onde

$$\delta(Q, \sigma) = \{p \in K \mid \exists q \in Q \text{ tq } (q, \sigma, p) \in \Delta\}.$$
$$\mathscr{F} = \{Q \subseteq K \mid Q \cap F \neq \emptyset\}$$

Teorema

$L(\hat{\mathscr{A}}) = L(\mathscr{A})$.

Teorema

$L(\hat{\mathscr{A}}) = L(\mathscr{A})$.

Teorema

$L(\hat{\mathscr{A}}) = L(\mathscr{A})$.

Dem: Basta provar que para todo $x \in \Sigma^*$,

$$Sx = \{p \in K \mid \text{existe passeio } S \xrightarrow{x} p \text{ em } G_{\mathscr{A}}\}.$$

Teorema

$L(\hat{\mathscr{A}}) = L(\mathscr{A})$.

Dem: Basta provar que para todo $x \in \Sigma^*$,

$$Sx = \{p \in K \mid \text{existe passeio } S \xrightarrow{x} p \text{ em } G_{\mathscr{A}}\}.$$

Indução padrão.

Teorema

$L(\hat{\mathscr{A}}) = L(\mathscr{A})$.

Dem: Basta provar que para todo $x \in \Sigma^*$,

$$Sx = \{p \in K \mid \text{existe passeio } S \xrightarrow{x} p \text{ em } G_{\mathscr{A}}\}.$$

Indução padrão.
Daí segue:

$$\begin{aligned} x \in L(\hat{\mathscr{A}}) &\Leftrightarrow Sx \in \mathscr{F} \\ &\Leftrightarrow Sx \cap F \neq \emptyset \\ &\Leftrightarrow \text{existe } p \in Sx \text{ tq } p \in F \\ &\Leftrightarrow \text{existe passeio } S \xrightarrow{x} F \text{ em } G_{\mathscr{A}} \\ &\Leftrightarrow x \in L(\mathscr{A}). \end{aligned}$$

□

A construção dos subconjuntos

A construção dos subconjuntos

Dado um AND $\mathscr{A} = (K, \Sigma, \Delta, S, F)$, o autômato dos subconjuntos de $\mathscr{A}$ é

$$\hat{\mathscr{A}} = (2^K, \Sigma, \delta, \text{Fecha}(\mathscr{A}, S), \mathscr{F})$$

onde

$$\delta(Q, \sigma) = \text{Fecha}(\mathscr{A}, \{p \mid \exists q \in Q \text{ tq } (q, \sigma, p) \in \Delta)\}.$$

A construção dos subconjuntos

Dado um AND $\mathscr{A} = (K, \Sigma, \Delta, S, F)$, o autômato dos subconjuntos de $\mathscr{A}$ é

$$\hat{\mathscr{A}} = (2^K, \Sigma, \delta, \text{Fecha}(\mathscr{A}, S), \mathscr{F})$$

onde

$$\delta(Q, \sigma) = \text{Fecha}(\mathscr{A}, \{p \mid \exists q \in Q \text{ tq } (q, \sigma, p) \in \Delta)\}.$$
$$\mathscr{F} = \{q \subseteq K \mid \text{Fecha}(\mathscr{A}, Q) \cap F \neq \emptyset\}$$

A construção dos subconjuntos

Dado um AND $\mathscr{A} = (K, \Sigma, \Delta, S, F)$, o autômato dos subconjuntos de $\mathscr{A}$ é

$$\hat{\mathscr{A}} = (2^K, \Sigma, \delta, \text{Fecha}(\mathscr{A}, S), \mathscr{F})$$

onde

$$\delta(Q, \sigma) = \text{Fecha}(\mathscr{A}, \{p \mid \exists q \in Q \text{ tq } (q, \sigma, p) \in \Delta)\}.$$
$$\mathscr{F} = \{q \subseteq K \mid \text{Fecha}(\mathscr{A}, Q) \cap F \neq \emptyset\}$$

Teorema

$L(\hat{\mathscr{A}}) = L(\mathscr{A})$.

Corolário

N-Rec* $\subseteq$ *Rec

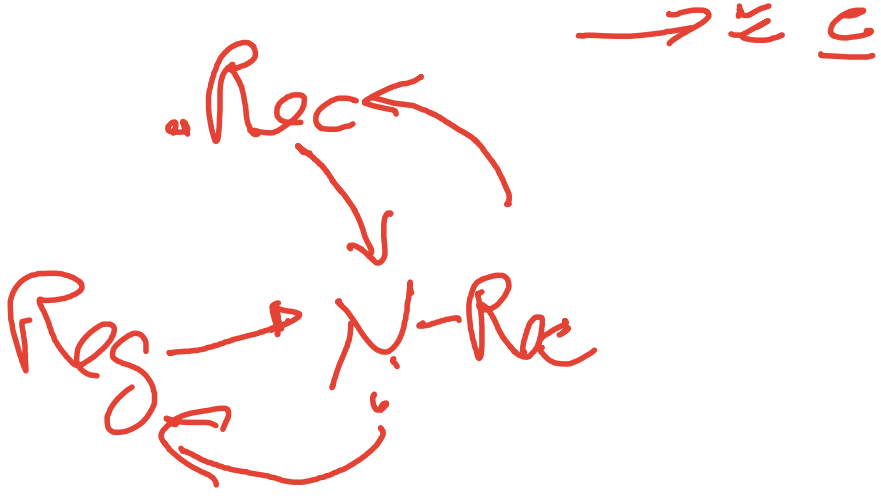
Corolário

N-Rec* $\subseteq$ *Rec

Corolário

N-Rec* $\subseteq$ *Rec

Mais um pedaço do Teorema de Kleene! Falta pouco.



Corolário

N-Rec* $\subseteq$ *Rec

Mais um pedaço do Teorema de Kleene! Falta pouco.

Antes de mostrar um exemplo...

Autômatos acessíveis

Autômatos acessíveis

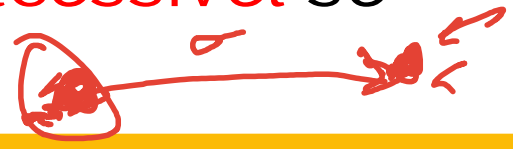
Um estado de um AND é acessível se existe passeio em $G_{\mathscr{A}}$ de S até ele.

Autômatos acessíveis

Um estado de um AND é acessível se existe passeio em $G_{\mathscr{A}}$ de S até ele. Um autômato é acessível se todos seus estados são acessíveis.

Autômatos acessíveis

Um estado de um AND é acessível se existe passeio em $G_{\mathscr{A}}$ de S até ele. Um autômato é acessível se todos seus estados são acessíveis.



Teorema

Seja $\mathscr{A}$ um AND, e $\dot{K}$ seu conjunto de estados acessíveis. Considere o autômato parte acessivel

$$\dot{\mathscr{A}} = (\dot{K}, \Sigma, \dot{\Delta}, S, F \cap \dot{K}),$$

onde $\dot{\Delta}$ consiste das arestas cujo início está em $\dot{K}$. Então, $\dot{\mathscr{A}}$ é acessível, $L(\dot{\mathscr{A}}) = L(\mathscr{A})$, e se $\mathscr{A}$ é determinístico, $\dot{\mathscr{A}}$ também é.

Autômatos acessíveis

Um estado de um AND é acessível se existe passeio em $G_{\mathscr{A}}$ de S até ele. Um autômato é acessível se todos seus estados são acessíveis.

Teorema

Seja $\mathscr{A}$ um AND, e $\dot{K}$ seu conjunto de estados acessíveis. Considere o autômato parte acessivel

$$\dot{\mathscr{A}} = (\dot{K}, \Sigma, \dot{\Delta}, S, F \cap \dot{K}),$$

onde $\dot{\Delta}$ consiste das arestas cujo início está em $\dot{K}$. Então, $\dot{\mathscr{A}}$ é acessível, $L(\dot{\mathscr{A}}) = L(\mathscr{A})$, e se $\mathscr{A}$ é determinístico, $\dot{\mathscr{A}}$ também é.

Autômatos acessíveis

Um estado de um AND é acessível se existe passeio em $G_{\mathscr{A}}$ de S até ele. Um autômato é acessível se todos seus estados são acessíveis.

Teorema

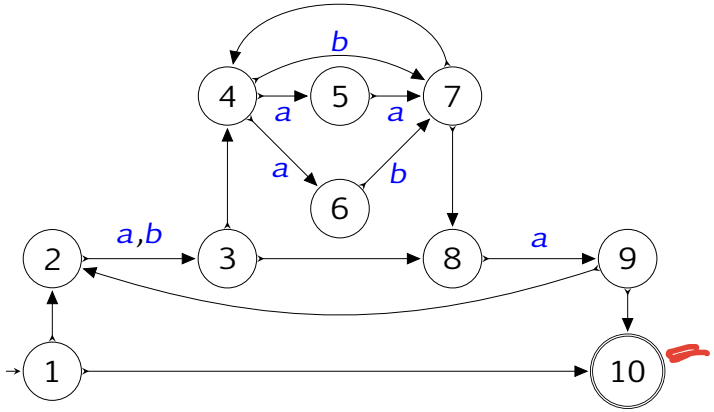
Seja $\mathscr{A}$ um AND, e $\dot{K}$ seu conjunto de estados acessíveis. Considere o autômato parte acessivel

$$\dot{\mathscr{A}} = (\dot{K}, \Sigma, \dot{\Delta}, S, F \cap \dot{K}),$$

onde $\dot{\Delta}$ consiste das arestas cujo início está em $\dot{K}$. Então, $\dot{\mathscr{A}}$ é acessível, $L(\dot{\mathscr{A}}) = L(\mathscr{A})$, e se $\mathscr{A}$ é determinístico, $\dot{\mathscr{A}}$ também é.

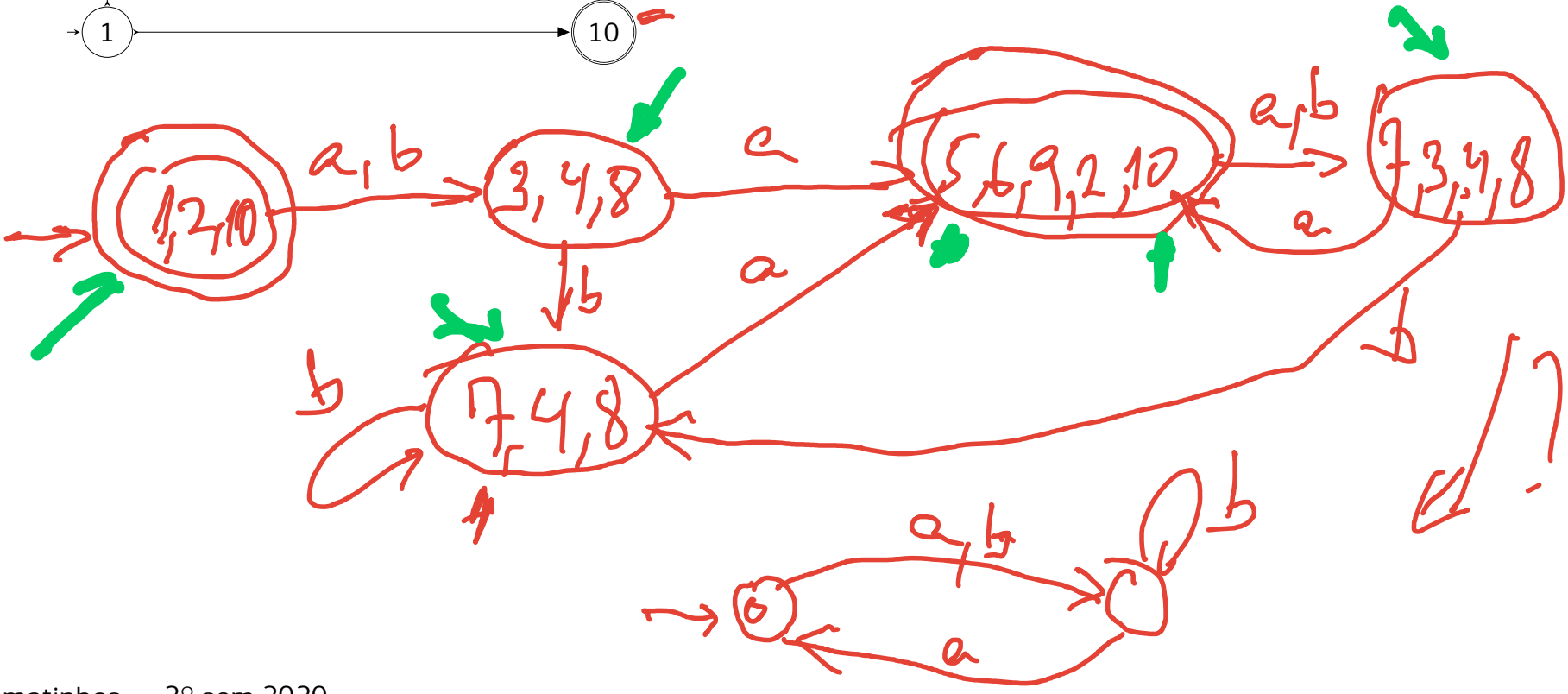
Dem: Os dois autômatos têm os mesmos passeios vencedores, com os mesmos rótulos. □

$((a+b)(b+aa+ab)^* a)^*$



ababab

1,2,10
3,4,8
7,4,8
5,6,9,2,10
7,3,6,8
5,6,9,2,10



Tem que ser exponencial?

Tem que ser exponencial?

Em geral, sim. Ainda vamos ver como provar isso.

Tem que ser exponencial?

Em geral, sim. Ainda vamos ver como provar isso.
E no mundo real?

Tem que ser exponencial?

Em geral, sim. Ainda vamos ver como provar isso.

E no mundo real?

Gnu grep usa várias técnicas, entre as quais, autômatos. No manual, seção **5 Performance:**

> The grep command operates partly via a set of automata that are designed for efficiency, and partly via a slower matcher that takes over when the fast matchers run into unusual features like back-references.
>
> [···]
>
> A back-reference such as '\1' can hurt performance significantly in some cases, since back-references cannot in general be implemented via a finite state automaton, and instead trigger a backtracking algorithm that can be quite inefficient. For example,[···] using an algorithm that is exponential in the worst case.

Boyer-Moore, Aho-Corasick

O que falta para o Teorema de Kleene?

Já vimos!

O que falta para o Teorema de Kleene?

Obter uma ER para a linguagem de um autômato.

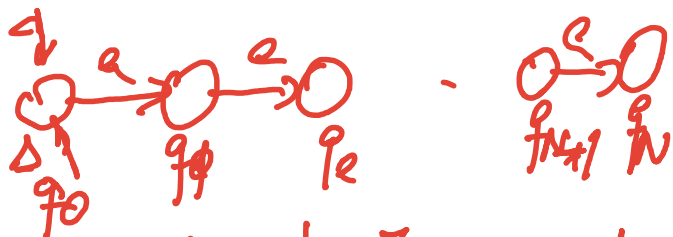
Ex de ling não regulas

$A_n B_n = \{ a^n b^n \mid n \geq 0 \}$

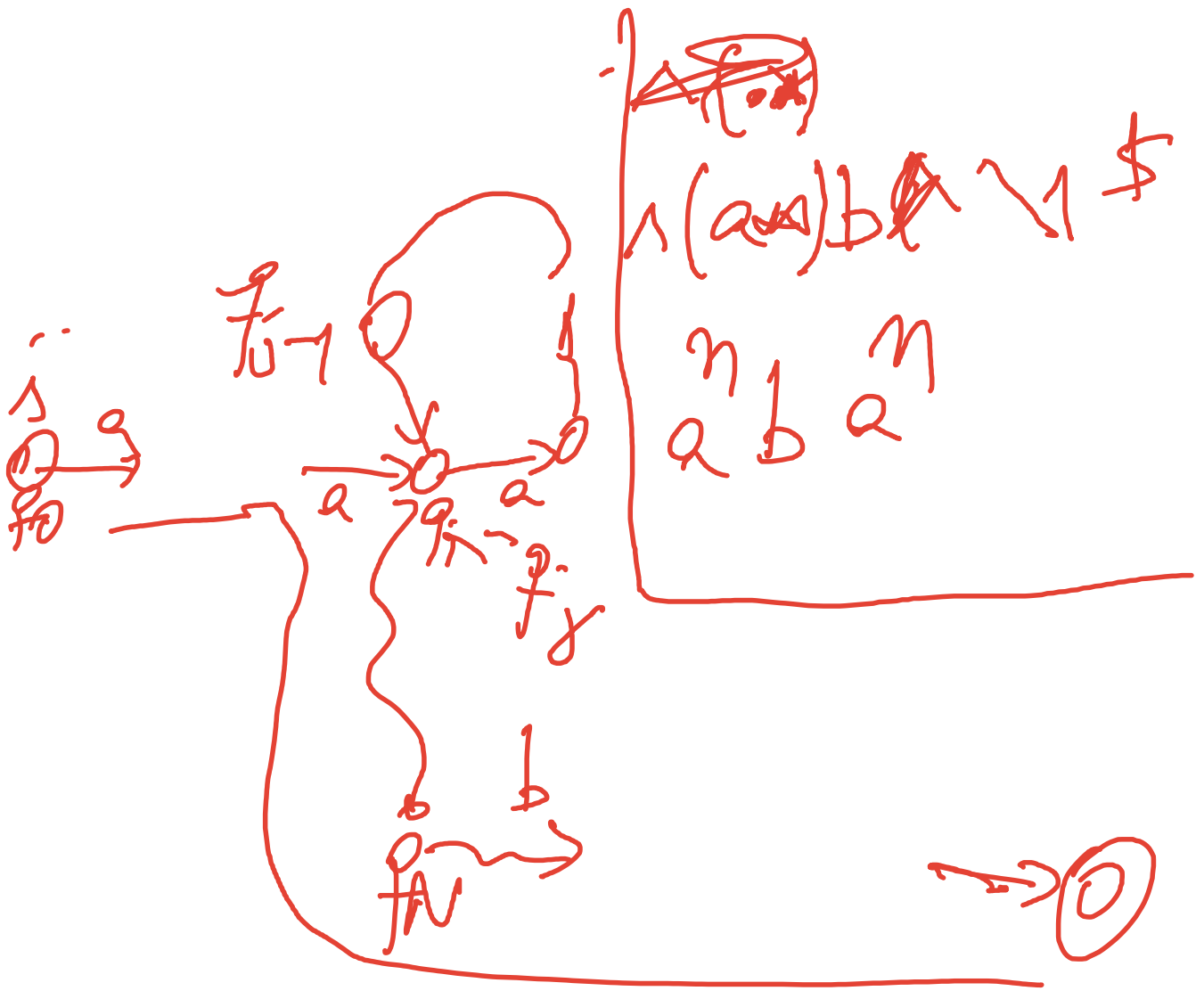
Dem: QM: não existe AD reconhecendo $A_n B_n$

Suponha que existe A tq $L(A) = A_n B_n$

$N = |K|$ $\quad x = a^N b^N \in A_n B_n$



$N+1$ estados! $\exists i < j$ tq $q_i = q_j$



$a^n b^n$ $n \geq 1$

$A_n B_n$? ×