

MAC 414

Autômatos, Computabilidade e  
Complexidade

aula 6 — 30/09/2020

# ER a partir de autômatos

# ER a partir de autômatos

É o passo que falta para o Teorema de Kleene.

# ER a partir de autômatos

É o passo que falta para o Teorema de Kleene.

Talvez o menos importante na prática.

# ER a partir de autômatos

É o passo que falta para o Teorema de Kleene.

Talvez o menos importante na prática.

Vamos ver três algoritmos, porque se aprende coisas diferentes com eles.

# Super-Autômatos

# Super-Autômatos

Gente mais séria (Sipser) chama de AND generalizados.

# Super-Autômatos

Gente mais séria (Sipser) chama de AND generalizados.

Generaliza AND 2-t. A diferença é que o rótulo de uma aresta pode ser uma ER arbitrária.



# Super-Autômatos

Gente mais séria (Sipser) chama de AND generalizados.

Generaliza AND 2-t. A diferença é que o rótulo de uma aresta pode ser uma ER arbitrária.

Fora isso, tudo igual: o rótulo de um passeio é o produto dos rótulos nas arestas. A linguagem reconhecida é a união das linguagens dadas pelos rótulos de passeios vencedores.

# Super-Autômatos

# Super-Autômatos

Formalmente:

$$\mathcal{A} = (K, \Sigma, \rho, s, f)$$

onde a novidade é que

$$\rho : \underline{K \times K} \rightarrow \mathcal{E}$$

associa a cada “possível aresta” um rótulo.

# Super-Autômatos

Formalmente:

$$\mathcal{A} = (K, \Sigma, \rho, s, f)$$

onde a novidade é que

$$\rho : K \times K \rightarrow \mathcal{E}$$

associa a cada “possível aresta” um rótulo.

Vamos exigir que para  $\rho(p, s) = \emptyset = \rho(f, p)$ .

# Super-Autômatos

Formalmente:

$$\mathcal{A} = (K, \Sigma, \rho, s, f)$$

onde a novidade é que

$$\rho : K \times K \rightarrow \mathcal{E}$$

associa a cada “possível aresta” um rótulo.

Vamos exigir que para  $\rho(p, s) = \emptyset = \rho(f, p)$ . Ou seja, sem aresta entrando em  $s$  ou saindo de  $f$ .

# Super-Autômatos

Formalmente:

$$\mathcal{A} = (K, \Sigma, \rho, s, f)$$

onde a novidade é que

$$\rho : K \times K \rightarrow \mathcal{E}$$

associa a cada “possível aresta” um rótulo.

Vamos exigir que para  $\rho(p, s) = \emptyset = \rho(f, p)$ . Ou seja, sem aresta entrando em  $s$  ou saindo de  $f$ .

Se  $P$  é passeio em  $G$ , a linguagem  $L(P)$  é a dada pelo produto dos rótulos das arestas em  $P$ .

# Super-Autômatos

Formalmente:

$$\mathcal{A} = (K, \Sigma, \rho, s, f)$$

onde a novidade é que

$$\rho : K \times K \rightarrow \mathcal{E}$$

associa a cada “possível aresta” um rótulo.

Vamos exigir que para  $\rho(p, s) = \emptyset = \rho(f, p)$ . Ou seja, sem aresta entrando em  $s$  ou saindo de  $f$ .

Se  $P$  é passeio em  $G$ , a linguagem  $L(P)$  é a dada pelo produto dos rótulos das arestas em  $P$ .

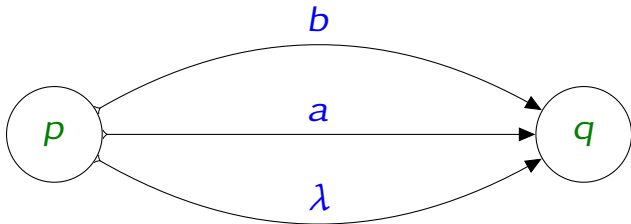
$$L(\mathcal{A}) = \bigcup \{L(P) \mid P : s \rightsquigarrow f\}$$

# De AND para Super-AND



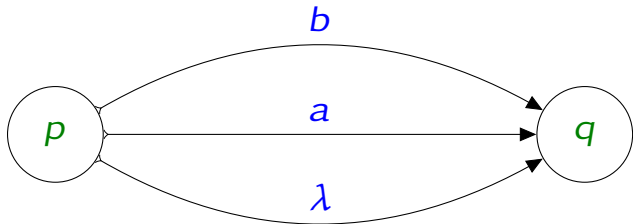
# De AND para Super-AND

Definição:

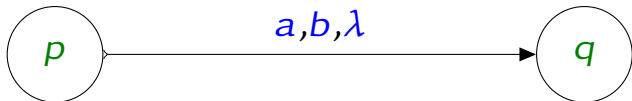


# De AND para Super-AND

Definição:

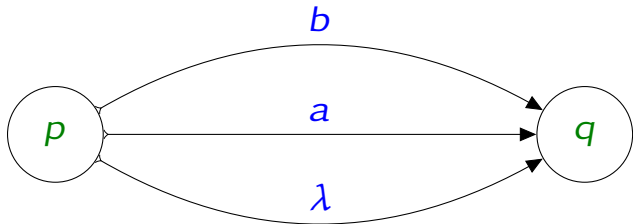


Desenho:

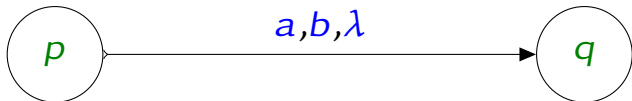


# De AND para Super-AND

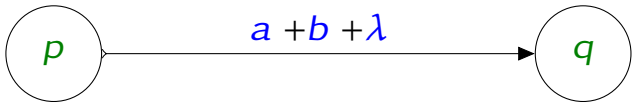
Definição:



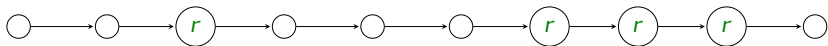
Desenho:



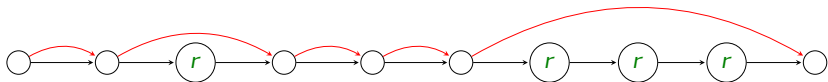
Agora:



# Uso de um estado $r$



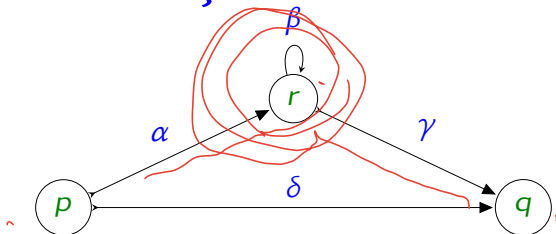
# Uso de um estado $r$



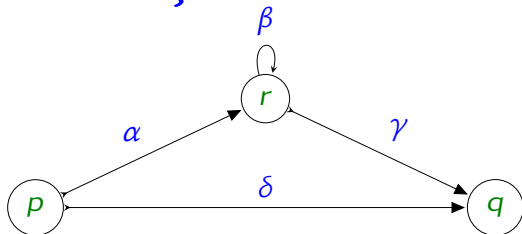
Todo caminho pode ser quebrado em caminhos cujo único vértice interior é  $r$ .

# Eliminação do estado $r$

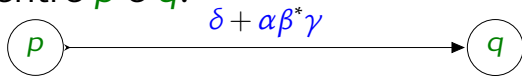
# Eliminação do estado $r$



# Eliminação do estado $r$

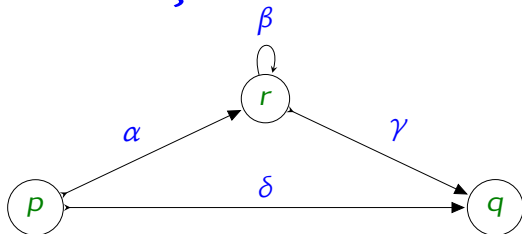


Eliminando  $r$  entre  $p$  e  $q$ :

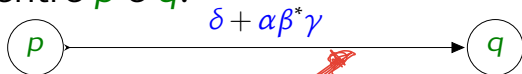




# Eliminação do estado $r$

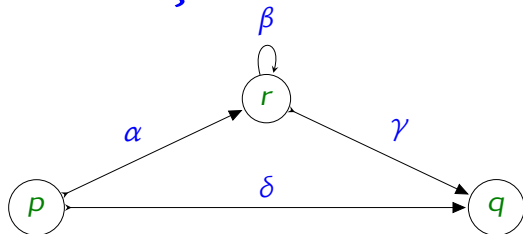


Eliminando  $r$  entre  $p$  e  $q$ :

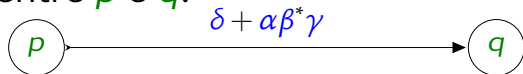


Pode ser  $p = q$ .

# Eliminação do estado $r$



Eliminando  $r$  entre  $p$  e  $q$ :



Pode ser  $p = q$ .

$$\alpha\beta^*\gamma = \begin{cases} \emptyset & \text{se } \alpha = \emptyset \text{ ou } \gamma = \emptyset \\ \alpha\gamma & \text{se } \beta = \emptyset \end{cases}$$

# Algoritmo de eliminação

Data:  $\mathcal{A}$  *suport*

while  $K \neq \{s, f\}$  do

  escolha  $r \in K \setminus \{s, f\}$

  for  $(p, q) \in K \times K$  do

    | elimine  $r$  entre  $p$  e  $q$

  end

$K \leftarrow K \setminus \{r\}$

*$K \setminus \{r\} \times K \setminus \{r\}$*

end

return  $\rho(s, f)$

# Algoritmo de eliminação

Data:  $\mathcal{A}$

while  $K \neq \{s, f\}$  do

    /\* Invariante: cada aresta  $p \rightarrow s$  codifica  
    todos os passeios  $p \rightsquigarrow s$  cujos vértices  
    internos são estados eliminados \*/

    escolha  $r \in K \setminus \{s, f\}$

    for  $(p, q) \in K \times K$  do

        | elimine  $r$  entre  $p$  e  $q$

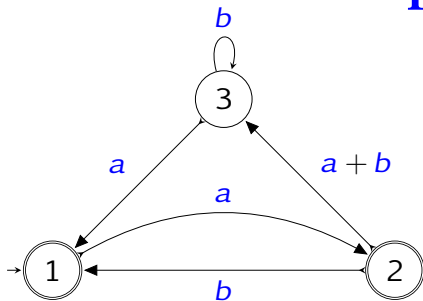
    end

$K \leftarrow K \setminus \{r\}$

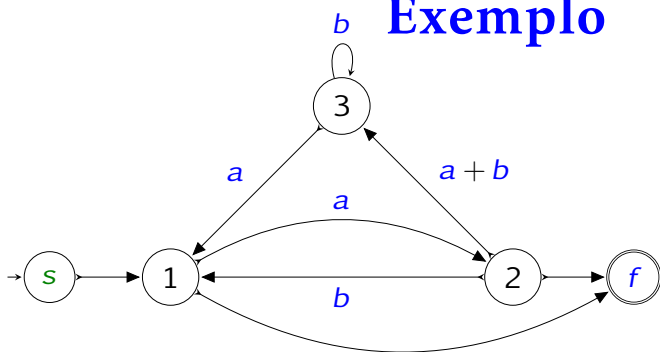
end

return  $\rho(s, f)$

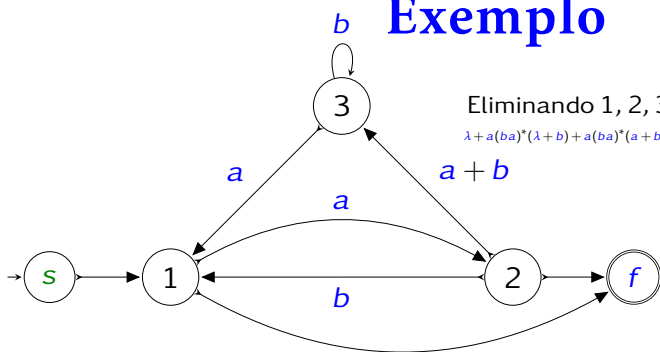
# Exemplo



# Exemplo



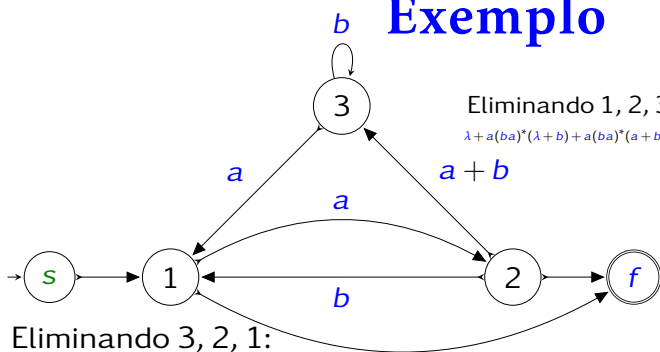
# Exemplo



Eliminando 1, 2, 3:

$$\lambda + a(ba)^*(\lambda + b) + a(ba)^*(a + b)(b + aa(ba)^*(a + b))^*(a + aa(ba)^*(\lambda + b))$$

# Exemplo

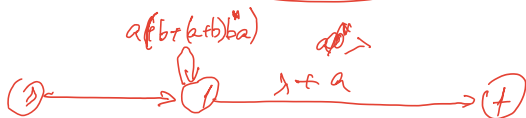


Eliminando 1, 2, 3:

$$\lambda + a(ba)^*(\lambda + b) + a(ba)^*(a + b)(b + aa(ba)^*(a + b))^*(a + aa(ba)^*(\lambda + b))$$

Eliminando 3, 2, 1:

Floyd-Warshall  
~~caminhos~~  
 distâncias





# Método da semântica da chegada

# Método da semântica da chegada

Daqui para a frente AD, para simplificar, mas pode ser adaptado para AND.

# Método da semântica da chegada

Daqui para a frente AD, para simplificar, mas pode ser adaptado para AND.

Para provar que um AD *funciona*, é preciso ter algum invariante para o processamento de uma palavra.

# Método da semântica da chegada

Daqui para a frente AD, para simplificar, mas pode ser adaptado para AND.

Para provar que um AD *funciona*, é preciso ter algum invariante para o processamento de uma palavra.

Alguma afirmativa da forma *o estado atual é  $q$  sse o que foi lido tem a estrutura...*

# Método da semântica da chegada

Daqui para a frente AD, para simplificar, mas pode ser adaptado para AND.

Para provar que um AD *funciona*, é preciso ter algum invariante para o processamento de uma palavra.

Alguma afirmativa da forma *o estado atual é  $q$  sse o que foi lido tem a estrutura...*

Um jeito genérico de descrever essa estrutura é declarar as linguagens

$$L_q = \{x \mid sx = q\}.$$

# Método da semântica da chegada

$$L_q = \{x \mid sx = q\}.$$

# Método da semântica da chegada

$$L_q = \{x \mid sx = q\}.$$



Para cada estado, dá para chegar através das arestas que apontam para ele:

# Método da semântica da chegada

$$L_q = \{x \mid sx = q\}.$$

Para cada estado, dá para chegar através das arestas que apontam para ele:

$$L_q = \bigcup_{p \in K} \{L_p \sigma \mid p\sigma = q\}$$



# Método da semântica da chegada

$$L_q = \{x \mid sx = q\}.$$

Para cada estado, dá para chegar através das arestas que apontam para ele:

$$L_q = \bigcup_{p \in K} \{L_p \sigma \mid p\sigma = q\} = \bigcup_{p \in K} L_p \{\sigma \mid p\sigma = q\}.$$

# Método da semântica da chegada

$$L_q = \{x \mid sx = q\}.$$

Para cada estado, dá para chegar através das arestas que apontam para ele:

$$L_q = \bigcup_{p \in K} \{L_p \sigma \mid p\sigma = q\} = \bigcup_{p \in K} L_p \{\sigma \mid p\sigma = q\}.$$

$s$  é especial:

$$L_s = \bigcup_{p \in K} \{L_p \sigma \mid p\sigma = s\} \cup \{\lambda\}.$$

# Método da semântica da chegada

$$L_q = \{x \mid sx = q\}.$$

Para cada estado, dá para chegar através das arestas que apontam para ele:

$$L_q = \bigcup_{p \in K} \{L_p \sigma \mid p\sigma = q\} = \bigcup_{p \in K} L_p \underbrace{\{\sigma \mid p\sigma = q\}}.$$

$s$  é especial:

$$L_s = \bigcup_{p \in K} \{L_p \sigma \mid p\sigma = s\} \cup \{\lambda\}.$$

No fim:

$$L(\mathcal{A}) = \bigcup_{q \in F} L_q$$



Usando notação de soma:

Usando notação de soma:

$$a_{pq} = \sum \{\sigma \mid p\sigma = q\} \quad p, q \in K$$

$$b_q = \begin{cases} \emptyset & q \in K, q \neq s \\ \{\lambda\} & q = s \end{cases}$$

Usando notação de soma:

$$a_{pq} = \sum \{\sigma \mid p\sigma = q\} \quad p, q \in K$$

$$b_q = \begin{cases} \emptyset & q \in K, q \neq s \\ \{\lambda\} & q = s \end{cases}$$

Considere o sistema linear:

$$X_q = \sum_p X_p a_{pq} + b_q \quad q \in K$$

Usando notação de soma:

$$a_{pq} = \sum \{\sigma \mid p\sigma = q\} \quad p, q \in K$$

$$b_q = \begin{cases} \emptyset & q \in K, q \neq s \\ \{\lambda\} & q = s \end{cases}$$

Considere o sistema linear:

$$X_q = \sum_p X_p a_{pq} + b_q \quad q \in K$$

$$\text{Ou: } X = XA + b$$



Usando notação de soma:

$$a_{pq} = \sum \{\sigma \mid p\sigma = q\} \quad p, q \in K$$

$$b_q = \begin{cases} \emptyset & q \in K, q \neq s \\ \{\lambda\} & q = s \end{cases}$$

Considere o sistema linear:

$$X_q = \sum_p X_p a_{pq} + b_q \quad q \in K$$

Ou:  $X = XA + b$

Pelo visto antes, o vetor  $(L_q)_{q \in K}$  é solução.

Usando notação de soma:

$$a_{pq} = \sum \{\sigma \mid p\sigma = q\} \quad p, q \in K$$

$$b_q = \begin{cases} \emptyset & q \in K, q \neq s \\ \{\lambda\} & q = s \end{cases}$$

Considere o sistema linear:

$$X_q = \sum_p X_p a_{pq} + b_q \quad q \in K$$

Ou:  $X = XA + b$

Pelo visto antes, o vetor  $(L_q)_{q \in K}$  é solução.

Falta:

1. Mostrar que a solução é única.

Usando notação de soma:

$$a_{pq} = \sum \{\sigma \mid p\sigma = q\} \quad p, q \in K$$

$$b_q = \begin{cases} \emptyset & q \in K, q \neq s \\ \{\lambda\} & q = s \end{cases}$$

Considere o sistema linear:

$$X_q = \sum_p X_p a_{pq} + b_q \quad q \in K$$

Ou:  $X = XA + b$

Pelo visto antes, o vetor  $(L_q)_{q \in K}$  é solução.

Falta:

1. Mostrar que a solução é única.
2. Dar um algoritmo para resolver.

# Outro sistema

# Outro sistema

Dado AD  $\mathcal{A} = (K, \Sigma, \delta, s, F)$ , considere a família de autômatos  $\mathcal{A}_q = (K, \Sigma, \delta, q, F)$ ,  $q \in K$ , e seja  $L_q = L(\mathcal{A}_q)$ .

# Outro sistema

Dado AD  $\mathcal{A} = (K, \Sigma, \delta, s, F)$ , considere a família de autômatos  $\mathcal{A}_q = (K, \Sigma, \delta, q, F)$ ,  $q \in K$ , e seja  $L_q = L(\mathcal{A}_q)$ .

Argumento parecido com o anterior leva a definir:

$$a_{pq} = \sum \{ \sigma \mid p\sigma = q \} \quad p, q \in K \text{ (mesma matriz)}$$

$$b_q = \begin{cases} \{ \lambda \} & q \in F \\ \emptyset & q \notin F. \end{cases}$$

# Outro sistema

Dado AD  $\mathcal{A} = (K, \Sigma, \delta, s, F)$ , considere a família de autômatos  $\mathcal{A}_q = (K, \Sigma, \delta, q, F)$ ,  $q \in K$ , e seja  $L_q = L(\mathcal{A}_q)$ .

Argumento parecido com o anterior leva a definir:

$$a_{pq} = \sum \{\sigma \mid p\sigma = q\} \quad p, q \in K \text{ (mesma matriz)}$$

$$b_q = \begin{cases} \{\lambda\} & q \in F \\ \emptyset & q \notin F. \end{cases}$$

E o sistema

$$X = AX + b$$

# Outro sistema

Dado AD  $\mathcal{A} = (K, \Sigma, \delta, s, F)$ , considere a família de autômatos  $\mathcal{A}_q = (K, \Sigma, \delta, q, F)$ ,  $q \in K$ , e seja  $L_q = L(\mathcal{A}_q)$ .

Argumento parecido com o anterior leva a definir:

$$a_{pq} = \sum \{\sigma \mid p\sigma = q\} \quad p, q \in K \text{ (mesma matriz)}$$

$$b_q = \begin{cases} \{\lambda\} & q \in F \\ \emptyset & q \notin F. \end{cases}$$

E o sistema

$$X = AX + b$$

Queremos encontrar

$$L(\mathcal{A}) = L_s.$$



# Como resolver

# Como resolver

Analogia com sistema de equações numéricas.

# Como resolver

Analogia com sistema de equações numéricas.

Uma equação:  $x = ax + b$  (e vou supor que  $0 < a < 1$ )

# Como resolver

Analogia com sistema de equações numéricas.

Uma equação:  $x = ax + b$  (e vou supor que  $0 < a < 1$ )

Então  $x = (1 - a)^{-1}b$

# Como resolver

Analogia com sistema de equações numéricas.

Uma equação:  $x = ax + b$  (e vou supor que  $0 < a < 1$ )

Então  $x = (1 - a)^{-1}b = (1 + a + a^2 + a^3 + \dots)b$

# Como resolver

Analogia com sistema de equações numéricas.

Uma equação:  $x = ax + b$  (e vou supor que  $0 < a < 1$ )

Então  $x = (1 - a)^{-1}b = (1 + a + a^2 + a^3 + \dots)b = a^*b$

# Como resolver

Analogia com sistema de equações numéricas.

Uma equação:  $x = ax + b$  (e vou supor que  $0 < a < 1$ )

Então  $x = (1 - a)^{-1}b = (1 + a + a^2 + a^3 + \dots)b = a^*b$

# Como resolver

Analogia com sistema de equações numéricas.

Uma equação:  $x = ax + b$  (e vou supor que  $0 < a < 1$ )

Então  $x = (1 - a)^{-1}b = (1 + a + a^2 + a^3 + \dots)b = a^*b$

Isto funciona com linguagens! (ex 1 do listão)



# Como resolver

Analogia com sistema de equações numéricas.

Uma equação:  $x = ax + b$  (e vou supor que  $0 < a < 1$ )

Então  $x = (1 - a)^{-1}b = (1 + a + a^2 + a^3 + \dots)b = a^*b$

Isto funciona com linguagens! (ex 1 do listão)

Agora é só usar o método de eliminação de variáveis, como num sistema usual.

# Como resolver

Analogia com sistema de equações numéricas.

Uma equação:  $x = \underline{ax + b}$  (e vou supor que  $0 < a < 1$ )

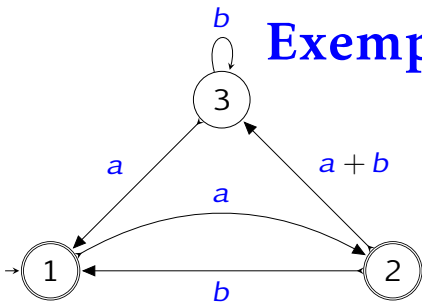
Então  $x = (1 - a)^{-1}b = (1 + a + a^2 + a^3 + \dots)b = \underline{a^*b}$

Isto funciona com linguagens! (ex 1 do listão)

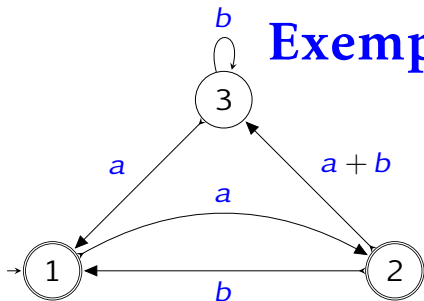
Agora é só usar o método de eliminação de variáveis, como num sistema usual.

A ref. 7 Aspectos Teóricos da Computação tem isso em detalhes.

# Exemplo



# Exemplo



$$X = XA + B$$

$$X = BA^*$$

$$X_1 = aX_2 + \lambda$$

$$X_2 = bX_1 + (a+b)X_3 + \lambda$$

$$X_3 = aX_1 + bX_3$$

$$\left. \begin{aligned}
 X_3 &= b^* a X_1 \\
 X_1 &= aX_2 + \lambda \\
 X_2 &= (a+b)b^* a X_1 + \lambda \\
 &\quad (+ \phi X_2)
 \end{aligned} \right\}$$

$$\left. \begin{aligned}
 X_1 &= a(a+b)b^* a X_1 + \lambda \\
 &= a(a+b)b^* a X_1 + \lambda \\
 X_1 &= (a(a+b)b^* a)^* (\lambda + \lambda)
 \end{aligned} \right\}$$

$$X_1 = aX_2 + \lambda$$

$$X_2 = bX_1 + (a+b)X_3 + \lambda$$

$$X_3 = aX_1 + bX_3$$

---

$$X_3 = b^* a X_1$$

$$X_2 = bX_1 + (a+b)^* b^* X_1 + \lambda$$

$$= (b + (a+b)^* b^*) X_1 + \lambda$$

$$X_1 = a(b + (a+b)^* b^*) X_1 + \lambda + a$$

$$X_1 = \left( a(b + (a+b)^* b^*) \right)^* (\lambda + a)$$

# O Teorema de Kleene

Agora está completo: Expressões regulares, Autômatos determinísticos e não-determinísticos descrevem as mesmas classes de linguagens.

# O Teorema de Kleene

Agora está completo: Expressões regulares, Autômatos determinísticos e não-determinísticos descrevem as mesmas classes de linguagens.

A demonstração envolveu vários algoritmos.

# O Teorema de Kleene

Agora está completo: Expressões regulares, Autômatos determinísticos e não-determinísticos descrevem as mesmas classes de linguagens.

A demonstração envolveu vários algoritmos.

Uma análise da eficiência aparece em [www.ime.usp.br/~am/414-17/complexidade.html](http://www.ime.usp.br/~am/414-17/complexidade.html).



# Como provar que uma linguagem não é regular?

# Como provar que uma linguagem não é regular?

Depende de como ela é dada.

# Como provar que uma linguagem não é regular?

Depende de como ela é dada.

Vamos abstrair o argumento usado para a  $A_n B_n$ .

# Lema do Bombeamento

## Lema

Se  $L$  é uma linguagem regular, então existe um inteiro  $N$  tal que, se  $x \in L$  então para toda fatoração  $x = yzw \in L$  com  $|z| = N$ , existe uma fatoração  $z = tuv$ , com  $u \neq \lambda$  tal que para todo  $n \geq 0$ ,  $ytu^n vw \in L$  (ou seja,  $ytu^*vw \subseteq L$ ).

# Lema do Bombeamento

## Lema

Se  $L$  é uma linguagem regular, então existe um inteiro  $N$  tal que, se  $x \in L$  então para toda fatoração  $x = yzw \in L$  com  $|z| = N$ , existe uma fatoração  $z = tuv$ , com  $u \neq \lambda$  tal que para todo  $n \geq 0$ ,  $ytu^n vw \in L$  (ou seja,  $ytu^*vw \subseteq L$ ).

# Lema do Bombeamento

## Lema

Se  $L$  é uma linguagem regular, então existe um inteiro  $N$  tal que, se  $x \in L$  então para toda fatoração  $x = yzw \in L$  com  $|z| = N$ , existe uma fatoração  $z = tuv$ , com  $u \neq \lambda$  tal que para todo  $n \geq 0$ ,  $ytu^n vw \in L$  (ou seja,  $ytu^*vw \subseteq L$ ).

## Lema

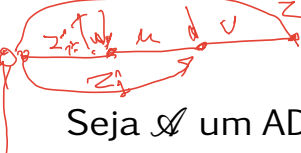
Se  $L$  é uma linguagem regular, então existe um inteiro  $N$  tal que, se  $x \in L$ , existe uma fatoração  $x = tuv$ , com  $|tu| \leq N$ , tal que  $u \neq \lambda$  e para todo  $n \geq 0$ ,  $ytu^n vw \in L$ .

# Prova




Seja  $\mathcal{A}$  um AD reconhecendo  $L$  e seja  $N$  o número de estados. Considere uma palavra  $x \in L$ ; então,  $sx \in F$ . Dada uma fatoração  $x = yzw \in L$  com  $|z| = N$ , vamos denotar por  $z_i$  o prefixo de  $z$  de comprimento  $i$ . A sequência  $(syz_i)_{i=0,1,\dots,N}$  tem  $N + 1$  termos, e já que  $\mathcal{A}$  só tem  $N$  estados há uma repetição.

# Prova



Seja  $\mathcal{A}$  um AD reconhecendo  $L$  e seja  $N$  o número de estados. Considere uma palavra  $x \in L$ ; então,  $sx \in F$ . Dada uma fatoração  $x = yzw \in L$  com  $|z| = N$ , vamos denotar por  $z_i$  o prefixo de  $z$  de comprimento  $i$ . A sequência  $(syz_i)_{i=0,1,\dots,N}$  tem  $N + 1$  termos, e já que  $\mathcal{A}$  só tem  $N$  estados há uma repetição. Assim, existem  $0 \leq i < j \leq N$  tais que

$\Rightarrow syz_i = syz_j$ . Então  $z$  se fatora na forma  $z = tuv$ , onde  $t = z_i$  e  $tu = z_j$  (e  $u \neq \lambda$ , já que  $|u| = j - i > 0$ ), assim,  $syt = sytu$ . Isso implica que para todo  $n \geq 0$ ,  $sytu^n = sytu$ . Segue que  $sytu^n vw = sytuvw = sx \in F$ .



□



# Para que serve?

# Para que serve?

Provar que uma linguagem é regular:

# Para que serve?

Provar que uma linguagem é regular: **NUNCA!**

# Para que serve?

Provar que uma linguagem é regular: **NUNCA!**

Provar que uma linguagem não é regular.

①  $A^nB^n$

# Para que serve?

Provar que uma linguagem é regular: **NUNCA!**

Provar que uma linguagem não é regular.

①  $A_n B_n$

②  $\{x \in (a + b)^* \mid |x|_a > 1000|x|_b\}$

# Para que serve?

Provar que uma linguagem é regular: **NUNCA!**

Provar que uma linguagem não é regular.

- 1  $A_n B_n$
- 2  $\{x \in (a + b)^* \mid |x|_a > 1000|x|_b\}$
- 3  $\{x \in a^* \mid |x| \text{ é quadrado perfeito}\}$

# Para que serve?

Provar que uma linguagem é regular: **NUNCA!**

Provar que uma linguagem não é regular.

- 1  $A_n B_n$
- 2  $\{x \in (a + b)^* \mid |x|_a > 1000|x|_b\}$
- 3  $\{x \in a^* \mid |x| \text{ é quadrado perfeito}\}$
- 4  $\{x : |x|_a = |x|_b\} + (a + b)^*(aaa + bbb)(a + b)^*$

$$L = \{x \mid |x|_a > 1000 \mid x \mid_b\}$$

Suponha que  $L \in \text{REG}$ ; seja  $N$  dado pelo C.B.

$$x = \overset{N}{b} \overset{2000N}{a} \in L$$

$$\tau u = \overset{k}{b} \quad k \leq N$$

$$u = \overset{\mu}{b} \quad 0 < \mu$$

$$x = \overset{k-\mu}{b} \cdot \overset{\mu}{b} \cdot \overset{N-k}{a} \overset{2000N}{a}$$

$\overset{k-\mu}{b} \overset{2000N}{u} \overset{N-k}{b} \overset{2000N}{a}$  deveria estar em  $L$ . Mas não!

Contradição



$$L = \{a^n \mid n \text{ é quadrado perfeito}\} = \{a^{n^2} \mid n \in \mathbb{N}\}$$

Supondo  $L$  regular, seja  $n$  dado pelo ZB

Considere  $x = a^{n^2} \in L$

$$\begin{array}{c} k \\ \hline a \\ \hline n \end{array}$$

Então existe  $0 < k \leq n$  tq  $(a^k)^n = a^{kn} \in L$  p todo  $n$

Considere  $n=2$ .  $a^{2+k} \in L$

$$\mathbb{N}^2 < \mathbb{N}^{2+k} < (\mathbb{N}^+)^2 \quad *$$