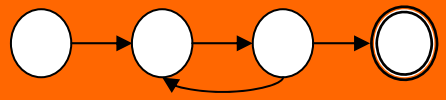
Automata e Linguagens Formais

CTC 34



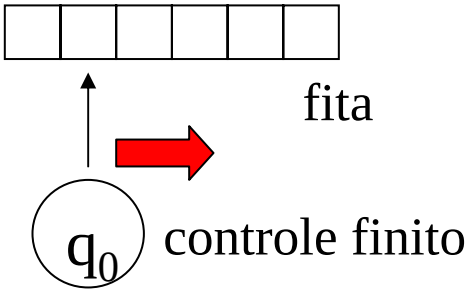
3

Prof. Carlos H. C. Ribeiro
carlos@ita.br
Ramal: 5895 Sala: 106

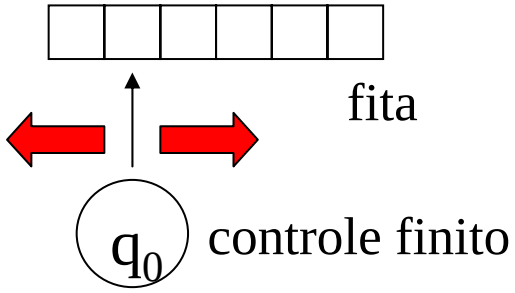
AFs Bidirecionais
AFs com Saída
- **Máquinas de Moore**
- **Máquinas de Mealy**

Linguagens Regulares: Propriedades
O *Pumping Lemma* (Lema do bombeamento)

Automata Finitos Bidirecionais (2AFD)



AFD



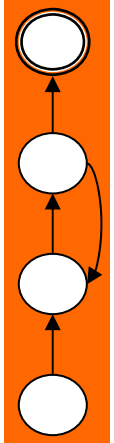
2AFD

Será que isto <u>aumenta</u> o poder do automato?

Um 2AFD é uma **quíntupla** $M = (Q, \Sigma, \delta, q_0, F)$

- Q = conjunto finito de estados, Σ = alfabeto
- δ = função de transição Q x $\Sigma \rightarrow$ Q x {L, R}
  - $\delta(q,a)=(p,L)$: lê a em q, vai p/ estado p e move p/ esquerda,
  - $\delta(q,a)=(p,R)$: lê a em q, vai p/ estado p e move p/ direita
- q_0 = estado inicial, F $\subseteq$Q = conjunto de estados finais

Para AFs, podemos definir operação do autômato para uma dada cadeia. Isto não serve para um 2AFD: preciso indicar a **direção** de movimento da fita **para cada novo símbolo** da cadeia que é lido...



2AFD: Operação sobre Cadeias

Def.: Uma **configuração instantânea** de um 2AFD *M* é uma cadeia $wqx \in \Sigma^*Q\,\Sigma^*$ tal que:

- *wx* é a cadeia de entrada;
- *q* é o estado atual do autômato;
- a cabeça está lendo o 1o. símbolo da subcadeia *x*.

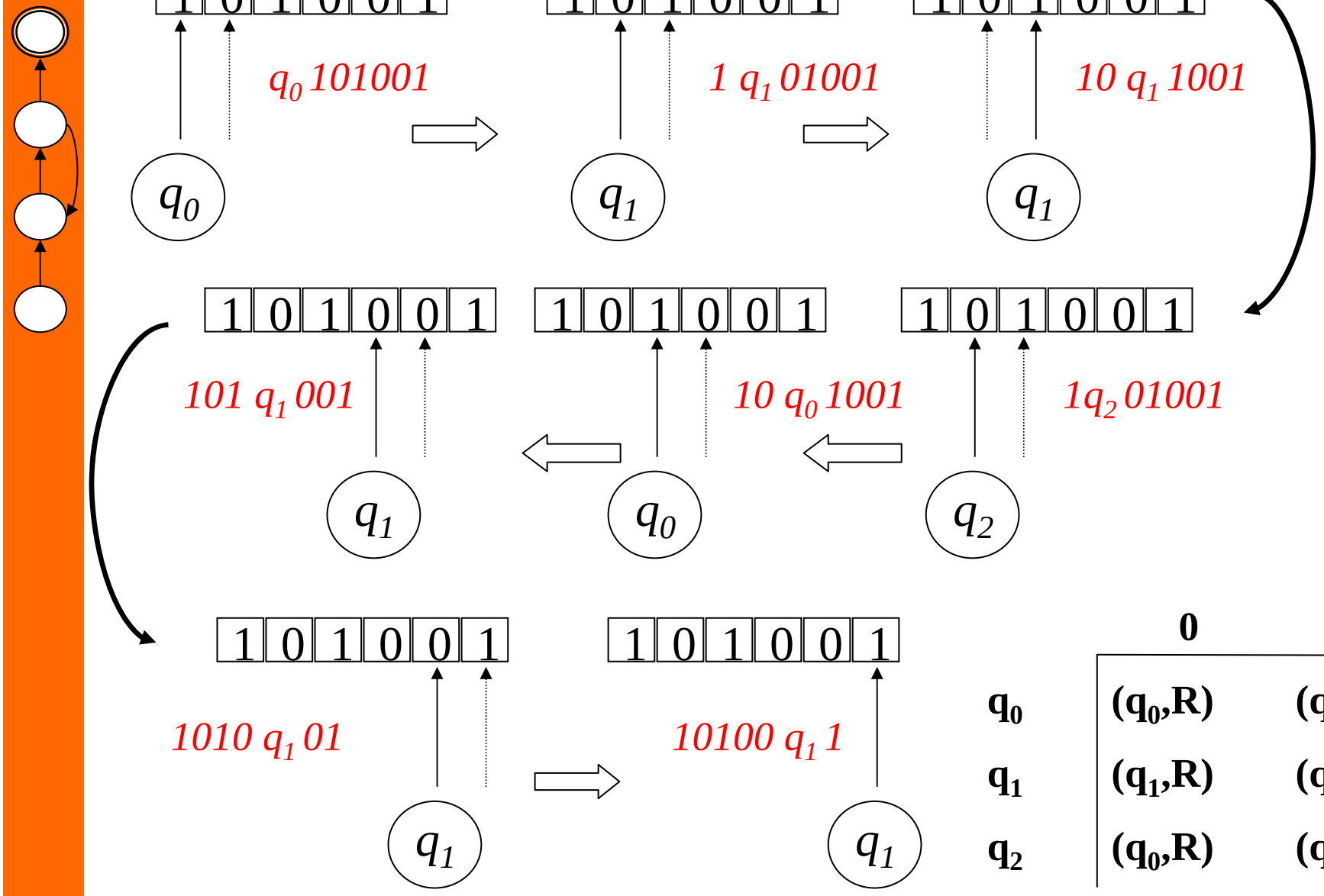
Def.: A **transição** $\vdash$ em um 2AFD é:

$a_1a_2 \ldots a_{i-1}\, q\, a_i \ldots a_n \vdash a_1a_2 \ldots a_{i-1}a_i\, p\, a_{i+1} \ldots a_n$ se $\delta(q,a_i) = (p, R)$

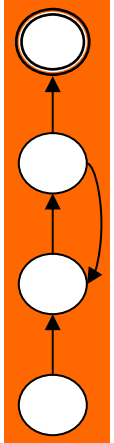
$a_1a_2 \ldots a_{i-1}\, q\, a_i \ldots a_n \vdash a_1a_2 \ldots a_{i-2}\, p\, a_{i-1}a_i \ldots a_n$ se $\delta(q,a_i) = (p,L)$ e $i>1$

Def.: Uma **linguagem *L(M)* aceita por um 2AFD** é formada por cadeias ***w*** tais que

$$q_o\boldsymbol{w} \overset{*}{\vdash} \boldsymbol{w}p \text{ para algum } p \in F.$$



	0	1
q_0	(q_0,R)	(q_1,R)
q_1	(q_1,R)	(q_2,L)
q_2	(q_0,R)	(q_2,L)



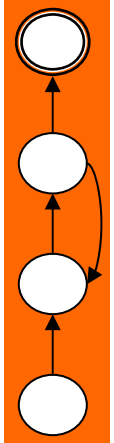
2AFD Só Aceitam Linguagens Regulares

Teorema: Se *L* é aceita por um 2AFD, então L é uma linguagem regular.

Prova: Hopcroft/Ullman, págs. 40-41

Ou seja:

Um 2AFD é equivalente a um AFD



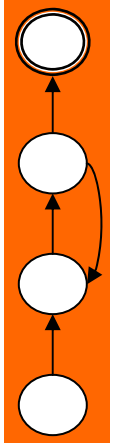
Automata Finitos com Saída

Até agora, só vimos AFs que apenas indicam se aceitam ou não uma cadeia de entrada.

Estudaremos agora dois tipos de automata que produzem saídas.

Máquinas de Moore: saídas associadas ao estado.

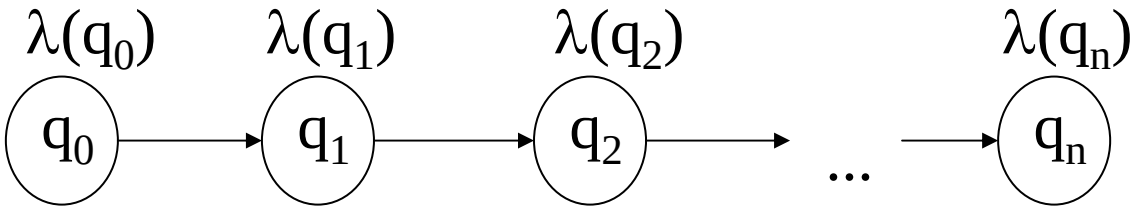
Máquinas de Mealy: saídas associadas à transição.



Máquinas de Moore

Def.: Uma máquina de Moore *M* é uma sextupla $(Q,\Sigma,\Delta,\delta,\lambda,q_o)$, onde Δ é o alfabeto da saída e $\lambda:Q\rightarrow\Delta$ define a saída associada a cada estado.

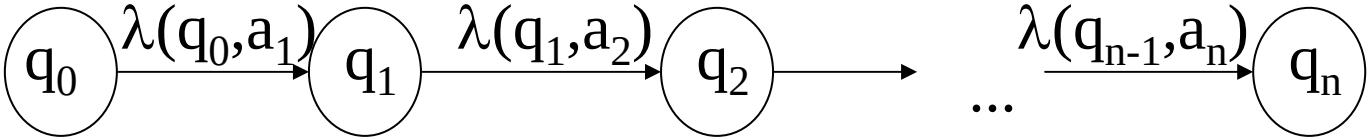
Entrada $a_1a_2...a_n$ **Estados** $q_0q_1...q_n$ **Saída** $\lambda(q_0)\lambda(q_1)...\lambda(q_n)$

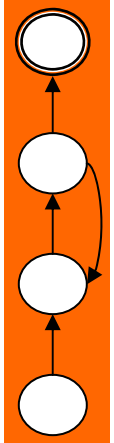


Máquinas de Mealy

Def.: Uma máquina de Mealy *M* é uma sextupla $(Q,\Sigma,\Delta,\delta,\lambda,q_o)$, onde Δ é o alfabeto da saída e $\lambda:Q \times \Sigma\rightarrow\Delta$ define a saída associada a cada transição.

Entrada $a_1a_2...a_n$ **Estados** $q_0q_1...q_n$ **Saída** $\lambda(q_0,,a_1)\lambda(q_1,a_2)...\lambda(q_{n-1},a_n)$





Equivalência: Moore → Mealy

Teorema 1 : Se $M_1 = (Q,\Sigma,\Delta,\delta,\lambda,q_o)$, é uma máquina de Moore, então existe uma máquina de Mealy M_2 equivalente, desde que ignoremos a primeira saída da máquina de Moore.

Prova: Seja $M_2 = (Q,\Sigma,\Delta,\delta,\lambda',q_o)$ uma máquina de Mealy tal que $\lambda'(q,a) = \lambda(\delta(q,a))$. Então:

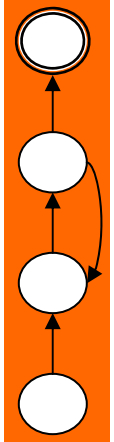
a) M_2 realiza as mesmas transições de M_1, sob o mesmo conj. de estados e alfabeto.

b) M_2 produz as mesmas saídas de M_1, a menos da primeira:

M_1 produz $\lambda(q_0) = \lambda(\delta(q_0,\varepsilon))$. M_2 produz $\lambda'(q_0, \varepsilon) = \varepsilon$ (ignoro)

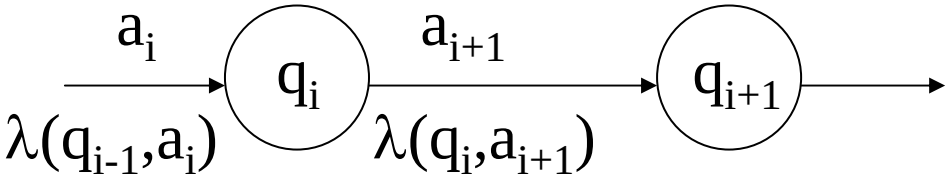
M_1 produz $\lambda(q_1) = \lambda(\delta(q_0,a_1))$. M_2 produz $\lambda'(q_0, a_1)$ ok!

M_1 produz $\lambda(q_2) = \lambda(\delta(q_1,a_2))$. M_2 produz $\lambda'(q_1, a_2)$ ok!

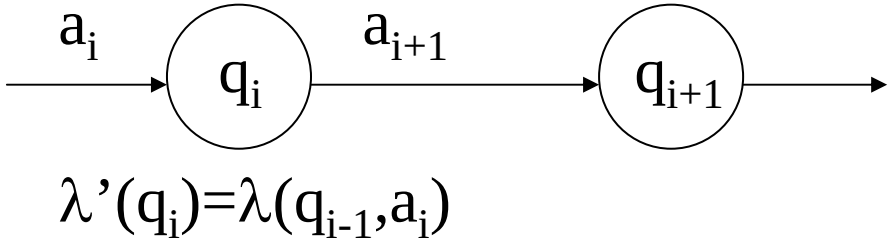


Equivalência: Mealy → Moore

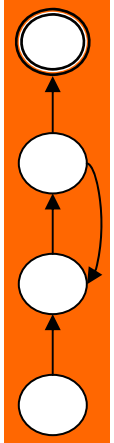
Mealy:



Moore:



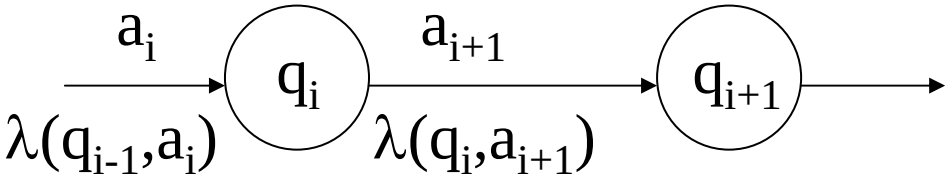
Não funciona! Por def., λ' deveria mapear estados em saídas ($Q \rightarrow \Delta$), não dependendo de estados e entradas anteriores... E o que fazer se existirem 2 transições distintas p/ um estado?



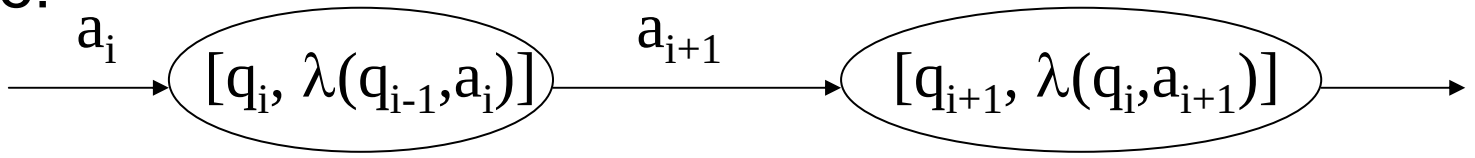
Equivalência: Mealy → Moore

Solução: representar a saída produzida pela última transição Mealy como parte do estado Moore.

Mealy:

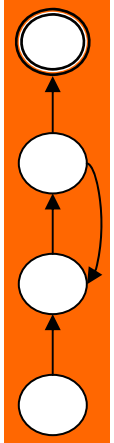


Moore:



$\lambda'([q_i, \lambda(q_{i-1},a_i)])=\lambda(q_{i-1},a_i)$

λ' mapeia "estados" $Q \times \Delta \rightarrow \Delta$, independentemente de estados/entradas anteriores



Equivalência: Mealy → Moore

Teorema 2 : Se $M_1 = (Q,\Sigma,\Delta,\delta,\lambda,q_o)$, é uma máquina de Mealy, então existe uma máquina de Moore M_2 equivalente.

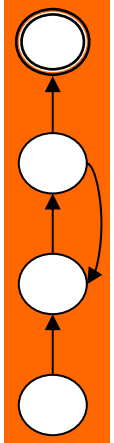
Prova: Seja $M_2 = (Qx\ \Delta,\Sigma,\Delta,\delta',\lambda',[q_o,b_0])$ uma máquina de Moore tal que:

i) b_0 é um elemento qualquer de Δ.

ii) $\delta'([q,b],a)=[\delta(q,a), \lambda(q,a)]$

iii) $\lambda'([q,b])=b$

M_2 produz as mesmas saídas de M_1, sobre um conjunto de estados univocamente mapeado a partir dos estados de M_1.



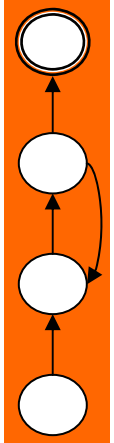
Linguagens Regulares: Propriedades de Fechamento

Relembrando a definição de linguagem regular:

- Ø, {ε} e {a}, para todo a∈Σ, são linguagens regulares.
- **Passo recursivo:** X e Y linguagens regulares ⇒ X ∪ Y, XY e X^* são linguagens regulares.
- **Fechamento:** X é linguagem regular apenas se obtido dos elementos básicos por um número finito de aplicações do passo recursivo.

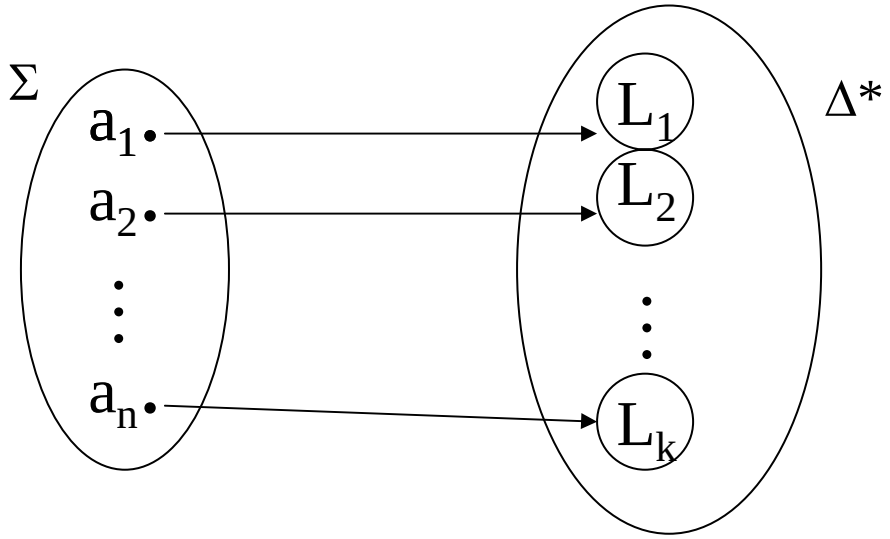
Uma linguagem é regular sss é reconhecida por um AFD.

- **Teorema:** ***A classe das linguagens regulares é fechada sob complementação.*** Se *L* é uma linguagem regular sobre Σ, então $\overline{L} = \Sigma^* - L$ é regular.
- **Teorema:** ***A classe das linguagens regulares é fechada sob intersecção.*** Se L_1 e L_2 são linguagens regulares, então $L_1 \cap L_2$ é linguagem regular.



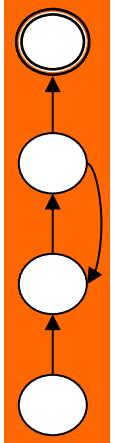
Substituições

Uma substituição *f* é um mapeamento de um alfabeto Σ em subconjuntos de Δ^*, para algum alfabeto Δ:



Estendendo a definição para Cadeias: $f(\varepsilon)= \varepsilon$, $f(xa)=f(x)f(a)$

E para linguagens: $f(L) = \bigcup_{x \in L} f(x)$



Substituições: Exemplos e Fechamento

Exemplo 1: $\Sigma=\{0, 1\}$ $\Delta=\{a, b\}$

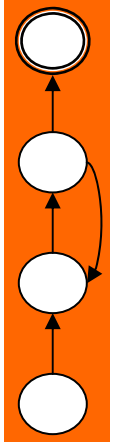
$f(0)=\mathbf{a}$, $f(1)=\mathbf{b}^*$ $\Rightarrow$ $f(010)=f(0)f(1)f(0)=\mathbf{ab}^*\mathbf{a}$

Exemplo 2: $\Sigma=\{0, 1\}$ $\Delta=\{a, b\}$

$f(0)=\mathbf{a}$, $f(1)=\mathbf{b}^*$, $L=\mathbf{0}^*\mathbf{(0+1)1}^*$ $\Rightarrow$ $f(L)=\mathbf{a}^*\mathbf{(a+b}^*\mathbf{)(b}^*\mathbf{)}^*=\mathbf{a}^*\mathbf{b}^*$

Teorema: ***A classe das linguagens regulares é fechada sob substituições.*** Se L é uma linguagem regular sobre Σ, então $f(L)$ é regular (sobre Δ).

Y. Bar-Hilell, M. Perles e E. Shamir [1961]



Homomorfismos e Homomorfismos Inversos

Def.: Um homomorfismo h é uma substituição que mapeia um alfabeto Σ em subconjuntos unitários de Δ^*, para algum alfabeto Δ.

Homomorfismos mapeiam símbolos em cadeias.

Def.: Um homomorfismo inverso h^{-1} da linguagem L é $h^{-1}(L)=\{x \mid h(x)\in L\}$

Def.: Um homomorfismo inverso h^{-1} da cadeia w é $h^{-1}(w)=\{x \mid h(x)=w\}$

Exemplo 1: $\Sigma=\{0, 1\}$, $\Delta=\{a, b\}$

$h(0) = aa$, $h(1) = aba \Rightarrow h(010) = h(0)h(1)h(0) = aaabaaa$

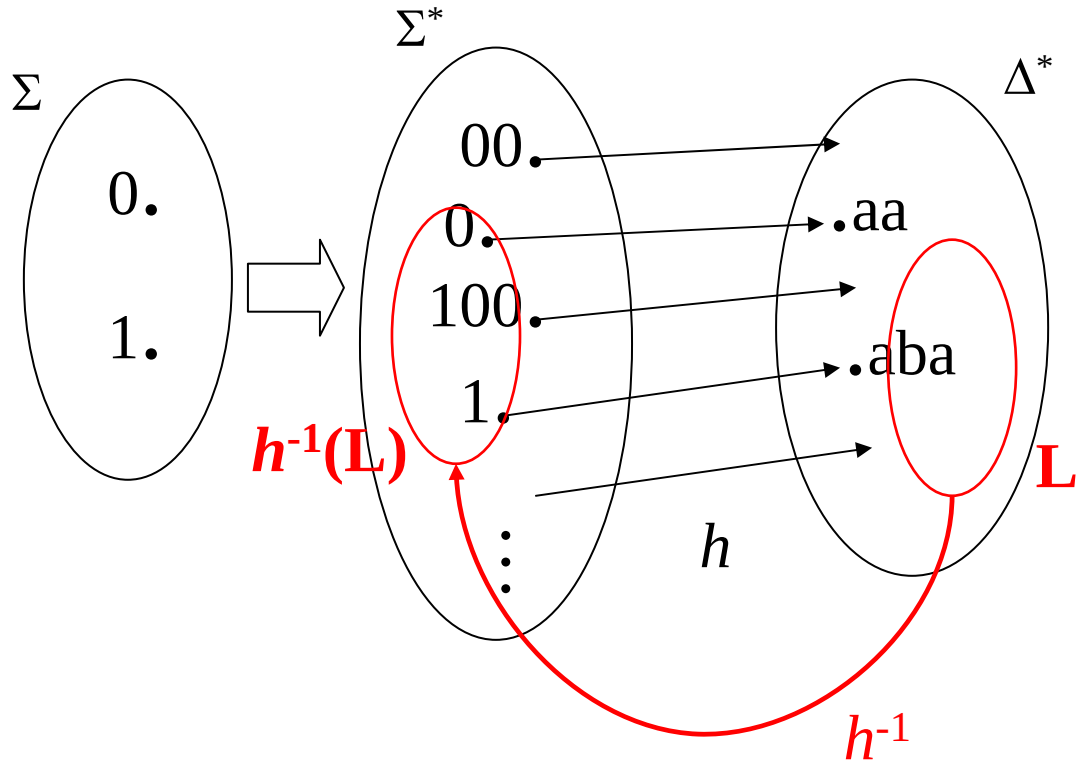
Exemplo 2: $\Sigma=\{0, 1\}$, $\Delta=\{a, b\}$

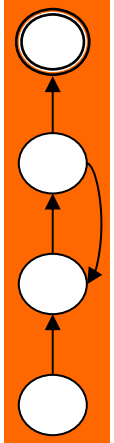
$h(0) = aa$, $h(1) = aba$, $L = \mathbf{(01)}^* \Rightarrow h(L) = \mathbf{(aaaba)}^*$

Homomorfismos Inversos: Exemplo

$\Sigma=\{0, 1\}$ $\Delta=\{a, b\}$

$h(0) = aa$, $h(1) = aba$, $L = \mathbf{(ab+ba)^*a}$ $h^{-1}(L) = ?$





Homomorfismos Inversos: Exemplo

$\Sigma=\{0, 1\}$ $\Delta=\{a, b\}$ $h(0) = aa$, $h(1) = aba$, L = **(ab+ba)*a** $h^{-1}(L) = ?$

i) Cadeia w em L começando com b: não pode ser $h(x)$ para nenhuma cadeia x de 0's e 1's! (pois $h(0)$ e $h(1)$ começam com a).

Logo, cadeias w em L que <u>podem</u> ser imagem de cadeias x em Σ^* devem começar com a.

ii) Caso 1: w=a Impossível (h sobre alfabeto Σ é Cadeia de pelo menos dois símbolos).

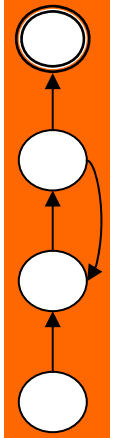
Caso 2: w=abw', para algum w' em **(ab+ba)*a**

Neste caso, devo ter $h^{-1}(w)$ começando com 1 (aba), de modo a garantir o b na segunda posição. Mas aí, tenho 2 possibilidades:

Caso 2.1: w'=a. Neste caso: w=aba, e portanto **$h^{-1}(w) = 1$**.

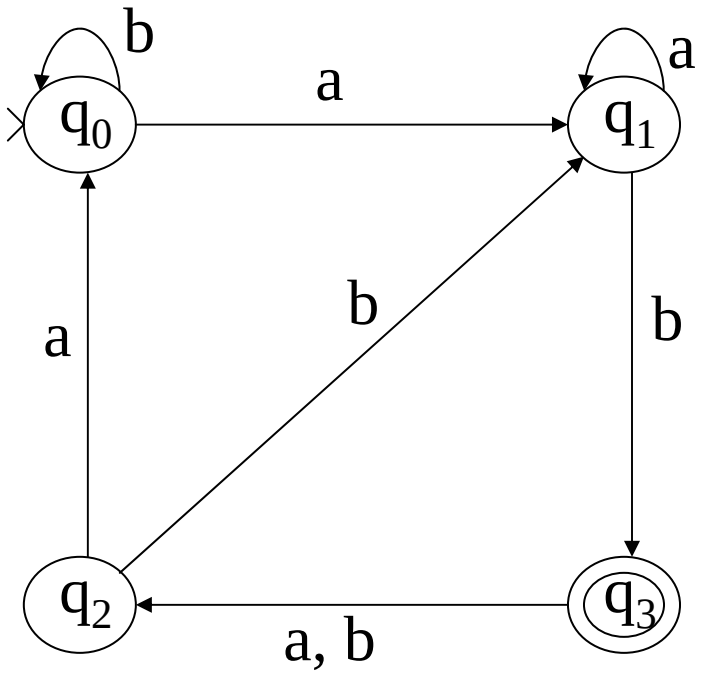
Caso 2.2: w'=abw'' $\Rightarrow$ w=ababw''. Impossível de ser gerado a partir de $h(0)$ e $h(1)$.

- **Teorema: *A classe das linguagens regulares é fechada sob homomorfismos.*** *Y. Bar-Hillel, M. Perles e E. Shamir [1961]*
- **Teorema: *A classe das linguagens regulares é fechada sob homomorfismos inversos.*** *S.Ginsburg e G. Rose [1963]*



O *Pumping Lemma* para Conjuntos Regulares

Análise Informal:



a**bab**baaab ✓

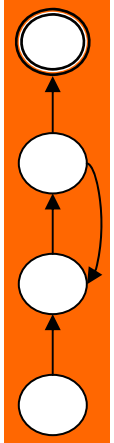
a**babbab**baaab ✓

...

a **(bab)^i** baaab ✓

u **v** **w**

Subcadeia "bombeada" (pumped)



O *Pumping Lemma* para Conjuntos Regulares

Lema: Seja *G* o grafo de estados de um AFD *M* com k estados. Então qualquer caminho de comprimento k em *G* contém um ciclo.

Prova: Caminho de comprimento k: k+1 estados.

Mas existem k nós em M: existirá um nó q_i que ocorre pelo menos duas vezes. A subtrajetória entre a primeira e a segunda ocorrências de q_i é o ciclo.

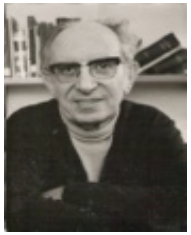
Corolário: Seja *G* o grafo de estados de um AFD *M* com k estados, e seja p um caminho de comprimento maior igual a k. Então p pode ser decomposto em subtrajetórias q, r e s (p=qrs), em que o comprimento de qr é menor ou igual a k e r é um ciclo.

O *Pumping Lemma* para Conjuntos Regulares

Lema (*Pumping Lemma*): Seja L uma linguagem regular aceita por um AFD M com k estados. Seja *z* uma Cadeia em L com comprimento maior ou igual a k. Então *z* pode ser escrito como *uvw*, onde:

$$length(uv) \leq k, length(v) > 0 \text{ e } uv^i w \in L, \forall i \geq 0$$

Y. Bar-Hillel, M. Perles e E. Shamir [1961]



Muito útil para provar que uma dada linguagem **não é** regular: Basta mostrar que não existe uma decomposição *uvw* de uma cadeia da linguagem que satisfaça as condições do lema.

Exemplo 3.51 Considere-se o autômato da Figura 3.68. A aplicação das propriedades enun-

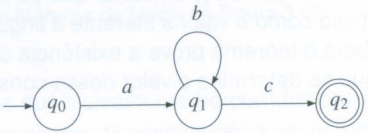


Figura 3.68 Aplicação do "Pumping Lemma" ao autômato finito que aceita ab^*c

ciadas através do "Pumping Lemma" a este autômato podem ser verificadas através do uso de cadeias de comprimento maior ou igual a 3, uma vez que ele possui três estados:

- Considere-se a cadeia $w = abc, |w| = 3$. Então, w pode ser reescrito como $xyz, |xy| \leqslant 3, 1 \leqslant |y| \leqslant 3$ e, finalmente, $xy^iz \in L, \forall i \geqslant 0$. Nesse caso, deve-se escolher $x = a, y = b, z = c$. Assim, $xz = ac, xyyz = abbc, xyyyz = abbbc$ etc. são todas cadeias que pertencem a L.
- Considere-se a cadeia $w = abbbc, |w| = 5$. Então, w pode ser reescrito como $xyz, |xy| \leqslant 3, 1 \leqslant |y| \leqslant 3$ e, finalmente, $xy^iz \in L, \forall i \geqslant 0$. Nesse caso podem-se fazer três escolhas distintas de subdivisão da cadeia w, todas em conformidade com os critérios do "Pumping Lemma":

  - $x = a, y = b, z = bbc$. As cadeias $(a)(b)^*(bbc)$ estão contidas em L.
  - $x = a, y = bb, z = bc$. As cadeias $(a)(bb)^*(bc)$ estão contidas em L.
  - $x = ab, y = b, z = c$. As cadeias $(ab)(b)^*(c)$ estão contidas em L.

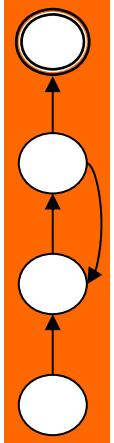
  Nem todas as subdivisões de uma cadeia w geram cadeias que produzem cadeias que pertencem à linguagem. Note-se, em particular, no exemplo acima, que seria possível relacionar, entre as subdivisões possíveis da cadeia de comprimento 5, as seguintes alternativas:

  i) $x = \varepsilon, y = a, z = bbbc$;

  ii) $x = \varepsilon, y = ab, z = bbc$;

  iii) $x = \varepsilon, y = abb, z = bc$.

Em todos esses casos, xy^iz gera cadeias que não pertencem a L. Qualquer que seja a cadeia escolhida, o "Pumping Lemma" garante apenas que, se ela possuir comprimento mínimo, então ao menos uma subdivisão xyz da mesma será possível de ser feita, de modo que todas as cadeias xy^iz também pertençam à linguagem.



Exemplo 1: L={$z \in$ {a, b}* | length(z) é quadrado perfeito}

Prova por contradição

Assuma, por absurdo, que L é regular.

Então L é aceita por algum AFD de k estados.

Pumping lemma: $z \in$ L e length(z) $\geq$ k $\Rightarrow$ $z = uvw$, length (uv) $\leq$ k, $v \neq \varepsilon$ e $uv^i w \in$ L.

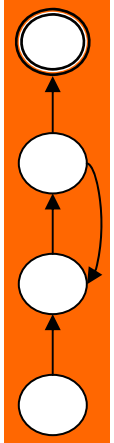
Vou mostrar que S ¤ T J] N Cadeia decomposta de acordo com o *Pumping Lemma* cujo comprimento X J O F V Z F I W F I T

Seja z tal que length(z)=k^2 (um quadrado perfeito)

Pumping Lemma: $z=uvw$, 0 < length(v) $\leq$ k, v é ciclo $\Rightarrow$ $uv^2w \in$ L

length(uv^2w) = length(uvw) + length (v) $\leq$ k^2 + k < k^2 + 2k + 1 = (k+1)^2

Ou seja: k^2 < length(uv^2w) < (k+1)^2 $\Rightarrow$ $uv^2w \in$ L não é quadrado perfeito (não pertence a L)!



Exemplo 2: $L=\{a^i b^i \mid i \geq 0\}$

Prova por contradição

Assuma, por absurdo, que L é regular.

Então L é aceita por algum AFD de k estados.

Pumping lemma: $z \in L$ e length(z) $\geq$ k $\Rightarrow$ $z = uvw$, length (uv) $\leq$ k, $v \neq \varepsilon$ e $uv^i w \in L$.

Vou mostrar que S ¤ T J] N cadeia decomposta de acordo com o *Pumping Lemma* que U J W Y J S ‹ F ¥ 1

Seja $z=a^k b^k$ (pertence a L, tem comprimento 2k)

Pumping Lemma: $z=uvw=a^i \boldsymbol{a^j} a^{k-i-j} b^k$, $i+j \leq$ k e j > 0.

Como *v* é o ciclo, posso realizar o bombeamento:

$$uv^2w = a^i \boldsymbol{a^j a^j} a^{k-i-j} b^k = a^k a^j b^k \notin L !$$