Propriedades de Linguagens Regulares

Lema do Bombeamento
Operações Fechadas sobre LR's
Aplicações

(H&U, 1969),(H&U, 1979),
(H;M;U, 2001) e (Menezes, 2002)

Lema do Bombeamento para LR

- Como decidir que uma linguagem é ou não regular?
  - Não bastaria não conseguir exibir um AF ou uma ER
- Toda linguagem regular satisfaz o Lema do bombeamento (LB).

- Se alguém apresenta a você uma falsa LR, use o LB para mostrar a contradição, pois ela não vai satizfazer o LB.

- O Lema do Bombeamento (Pumping Lemma) nos diz que qualquer cadeia suficientemente longa w de uma LR pode ser decomposta em 3 partes:

  w = xyz, de maneira que podemos construir outras cadeias da linguagem pela repetição da parte central y.

- Todas as cadeias da forma xy^kz são também da linguagem. Ou seja, podemos acionar a *bomba* quantas vezes quisermos, para criar quantas sentenças novas da linguagem desejarmos: xz, xyz, xyyz, xyyyz, …

Mostrando que uma Linguagem não é Regular

- Para mostrar que uma linguagem não é regular, mostramos que
  - não há como decompor uma cadeia (qualquer, arbitrariamente longa) da linguagem de forma que seja possível *bombear* e continuar na linguagem.

Lema do Bombeamento

Se L é uma linguagem regular, então:

existe uma constante natural n (que depende de L) tal que, para qualquer cadeia w de L com $|w| \geq n$, pode ser decomposta em três cadeias x, y, z ($w = xyz$) de forma que:

- **$|xy| \leq n$**
- **$y \neq \lambda$ (isto é, $|y| \geq 1$) e**
- **para qualquer $k \geq 0$, $xy^kz \in L$.**

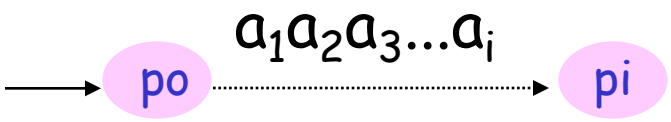
- **Demonstração (simplificada): Baseia-se no fato de que para as cadeias *longas* w é necessário usar pelo menos um loop de estados num AFD que aceite a linguagem.**

- **Assim, os símbolos de x são usados para chegarmos a um estado q do loop;**
- **os símbolos de y são usados para dar a volta no loop, de volta ao estado q;**
- **os símbolos de z são usados para ir de q até um estado final.**
- **Portanto, podemos dar quantas voltas no loop quisermos, e repetir y um número qualquer k de vezes: xy^kz.**

As cadeias curtas (comprimento < n) não são consideradas porque podem ser aceitas sem passar por qualquer loop.

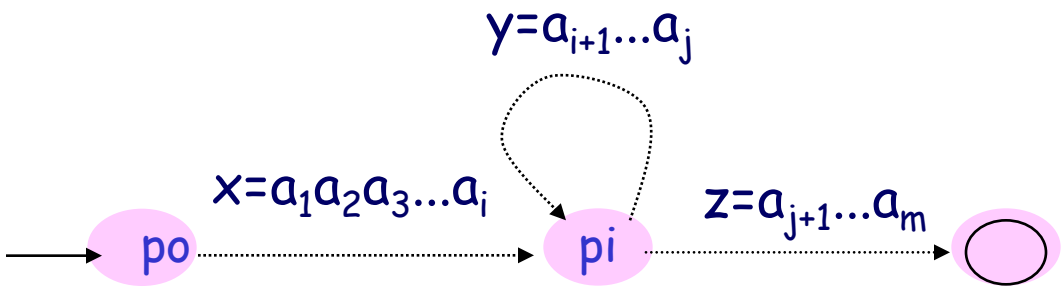
Prova formal do Lema do Bombeamento

Suponha que L seja regular. Então existe um AFD A, com n estados, que a reconhece. Considere qualquer string w de comprimento n ou maior: $w=a_1a_2a_3...a_m$, onde $m \geq n$, e cada a_i é um símbolo de entrada. Para i=0,1,...n, defina o estado p_i como $\delta(q_0,a_1a_2a_3...a_i)$. Isto é, p_i é o estado em que A se encontra depois de ler os primeiro i símbolos de w. Note que $p_0=q_0$.



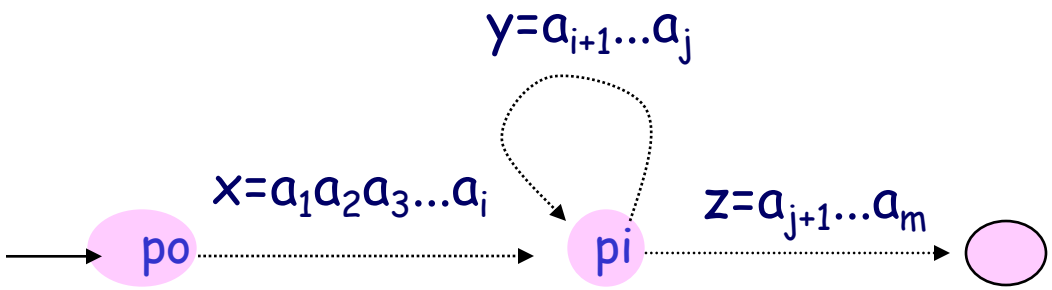
Como só existem n estados, não é possível que os n+1 diferentes p_i, para i=0, 1...n sejam distintos. Desse modo, podemos encontrar dois inteiros diferentes, i e j, com $0 \leq i < j \leq n$, tais que $p_i = p_j$. Agora, podemos dividir w = xyz como a seguir:

1. $x = a_1a_2a_3...a_i$
2. $y = a_{i+1}a_{i+2}...a_j$
3. $z = a_{j+1}a_{j+2}...a_m$



Observe que x pode ser vazio (i=0) e z pode ser vazio, se j=n=m. Mas y não pode ser vazio, pois i é estritamente menor que j. Também é verdade que $|xy| \leq n$. Falta verificar a última condição:

Vejamos o que acontece com A para entradas xy^kz para qualquer $k \geq 0$:



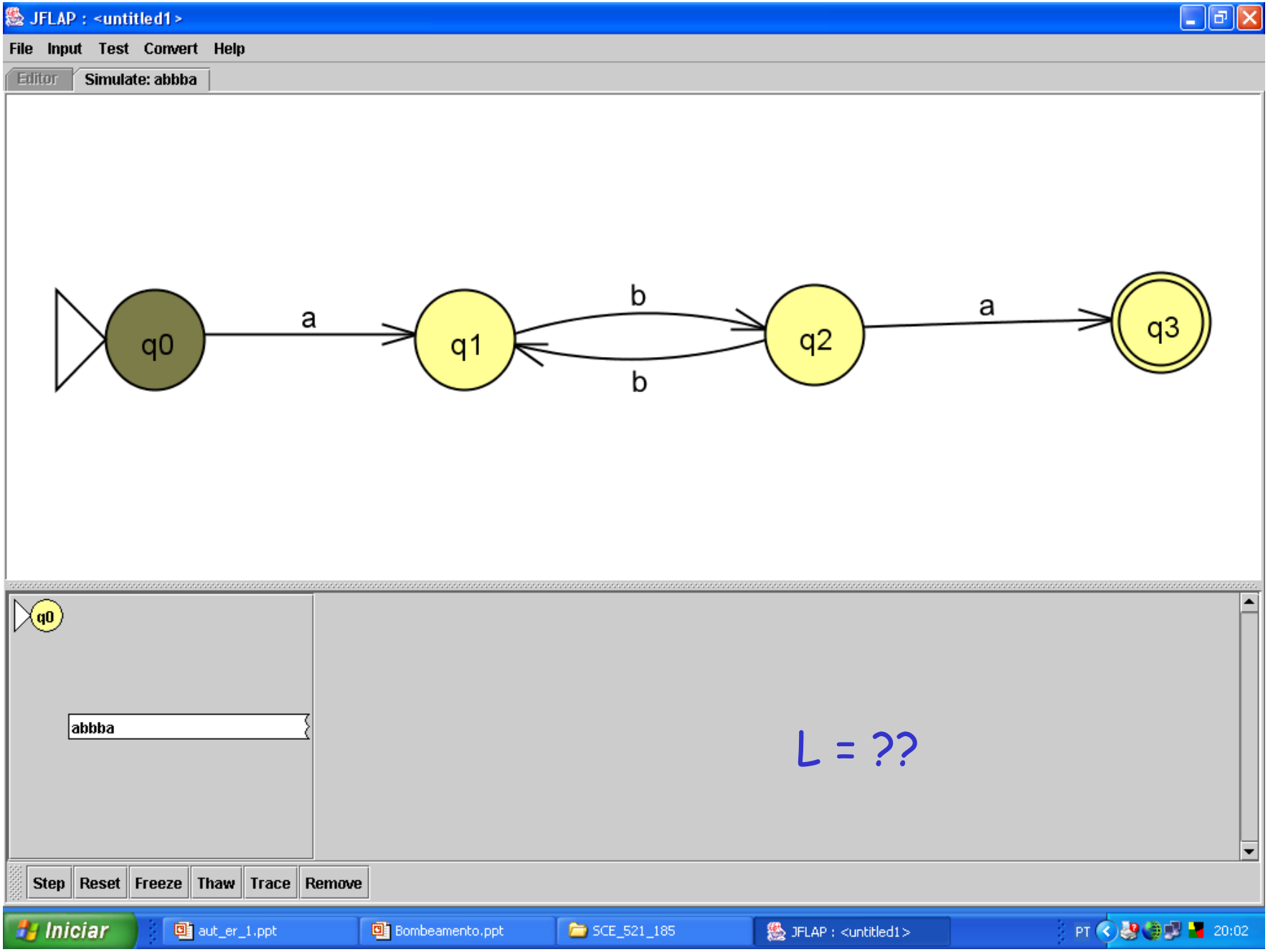
- Se k=0 (w=xz): *A* vai de po para pi na entrada x. Tendo em vista que $p_i=p_j$, *A* deve ir de p_i para o estado final, para a entrada z. Desse modo, *A* aceita xz.

- Se k >0 ($w=xy^kz$): *A* vai de po a pi sobre a entrada x, circula de p_i para p_i k vezes para a entrada y^k, e depois vai para o estado de aceitação para a entrada z. Dessa forma, para qualquer $k\geq 0$, xy^kz também é aceito por *A*; ou seja, xy^kz está em L.□ □ □

Provando que uma linguagem L não é regular

- Seja L o conjunto das cadeias w sobre {0,1} tal que $w=0^n1^n$.
- Supondo que L seja regular, pelo Lema do Bombeamento, se $w \in L$, então $w=xyz$ e $y \neq \varepsilon$, e $|xy| \leq n$, e $xz \in L$ e $xy^kz \in L$.
- Se $|xy| \leq n$ e como xy é um prefixo de w, então x e y consistem apenas em símbolos 0.
- Pelo Lema, então $xz \in L$. Porém, xz deve ter n 1´s, já que nenhum apareceu em xy. Por outro lado, xz tem menos de n 0´s, pois perdemos os 0´s de y. Como $y \neq \varepsilon$, não pode haver mais que n-1 0´s entre x e z.
- Assim, supondo L regular, concluímos um absurdo: o de que $w=xz=0^{n-1}1^n \in L$.
- Logo, L não é regular.

Exemplo provando que L é Regular



- Seja L = {$ab^n a$| n >=1 e ímpar} ou {$ab^{2n+1}a$ | n>=0}

- Pelo Lema do Bombeamento, temos que:

- seja n = 4
- Para w = abbba |w| = 5 >= 4

  Sejam:

  - x = a
  - y = bb |xy| <= 4; |y| >= 1
  - z = ba

Para todo k >= 0 $a(bb)^k ba \in L$

Exemplo provando que L **não** é regular

$L = \{a^i b^i \mid i \geq 0\}$ (Já vimos que L é uma LLC.)

Seja i = n + 1.

Considere a cadeia z = $a^i b^i$. Qualquer decomposição z = uvw deve ter em v o mesmo número de a´s e de b´s, para que a propriedade de que o número de a´s é igual ao de b´s se mantenha nas cadeias u v^k w.

- Se isso não acontecer, quando acrescentarmos mais um v (aumentando k de 1), obteremos uma cadeia fora da linguagem. Portanto, v deve ser da forma $a^j b^j$, com j >0, já que v não pode ser vazia.
- Mas nesse caso, u v^2 w conterá a cadeia $a^j b^j a^j b^j$, com pelo menos um a e depois um b, o que não pode acontecer na linguagem.

- Ou seja, nenhuma decomposição é possível, contrariando o Lema, e podemos concluir que L não é regular.

- OU
- Considere a cadeia $z = 0^n1^n$; z pode ser dividida em 3 pedaços = uvw onde para i >= 0 a cadeia uv^iw pertence a L. Vamos considerar 3 casos para mostrar que este resultado é impossível:

- 1) a cadeia v consiste somente de 0´s. Neste caso, a cadeia uvvw tem mais 0´s do que 1 violando o LB.
- 2) a cadeia v consiste somente de 1´s. Este caso também dá uma contradição.
- 3) a cadeia v consiste de 0´s e 1´s. Neste caso a cadeia uvvw tem o mesmo número de 0´s e 1´s mas não estão na ordem desejada pois v = 0101. E assim ela não é membro de L o que é uma contradição.

- A contradição é inevitável se nós aceitamos a suposição de que B é regular. Assim, B não é regular.

Exercícios

Mostre que as linguagens abaixo não são regulares:

1) $L = \{ xx^r \mid x \in \{0,1\}^* \}$
2) $L = \{0^n 1 0^n \mid n >= 1\}$
3) $L = \{0^n 1^m 2^n \mid$ n e m são inteiros quaisquer$\}$
4) $L = \{0^n 1^m \mid n <= m\}$
5) $L = \{0^n 1^{2n} \mid n >= 1\}$

Operações que preservam a propriedade de ser uma LR

- Existem muitas operações que, quando aplicadas a LR resultam em uma LR (dizemos que são fechadas). Entre elas temos: união, complemento e intersecção (operações booleanas), concatenação, fechamento, diferença e reverso ($a_1a_2...a_n \rightarrow a_n...a_2a_1$).

- Veremos a prova para as 6 primeiras propriedades acima.

- Elas são úteis também como ferramentas de construção de autômatos.

- Por exemplo, a propriedade da união ajuda a construir o autômato para
  - L(M) = { x $\in$ {0,1}* | o nro de 1's em x é múltiplo de 3 OU de 4}
  - e também o AF para o Analisador Léxico de um compilador

Propriedade de Fechamento das Linguagens da Hierarquia de Chomsky

Fechamento sobre	LR	LLC	LSC	LEF
união	S			
concatenação	S			
fechamento	S			
complemento	S			
intersecção	S			
diferença	S			
reverso	S			

Lema 3.1 (H&U, 69) – União

- A classe das LR é fechada sobre a união. Se L1 é LR e L2 é LR então $L1 \cup L2$ também é.
- (Prova via ER: direta)
- PROVA: Sejam L1 e L2 reconhecidas por AF $M1 = (Q1, \Sigma1, \delta1, qo, F1)$ e $M2 = (Q2, \Sigma2, \delta2, ro, F2)$. Suponha $Q1 \cap Q2 = \emptyset$ e $so \notin Q1$; $so \notin Q2$.

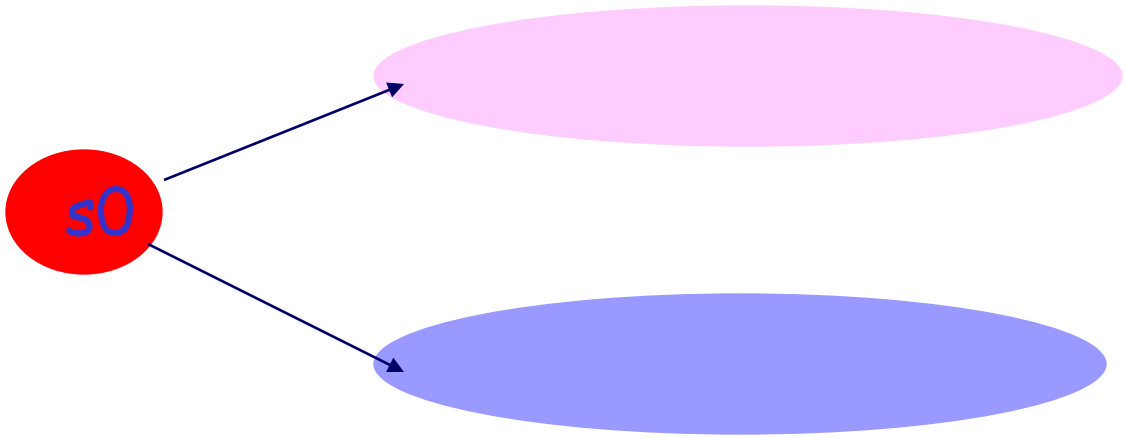
$M3 = (Q1 \cup Q2 \cup \{so\}, \Sigma1 \cup \Sigma2, \delta3, so, F)$ é um AFND onde:

- Se $\lambda \in L1$ ou $L2$ então $F = F1 \cup F2 \cup \{so\}$ cc. $F = F1 \cup F2$

- $\delta 3$
  1) $\delta 3(so,a) = \{\delta 1(qo,a), \delta 2(ro,a)\}$ para todo $a \in \Sigma 1 \cup \Sigma 2$
  2) $\delta 3(q,a) = \delta 1(q,a)$ para todo $q \in Q1$ e $a \in \Sigma 1$
  3) $\delta 3(q,a) = \delta 2(q,a)$ para todo $q \in Q2$ e $a \in \Sigma 2$

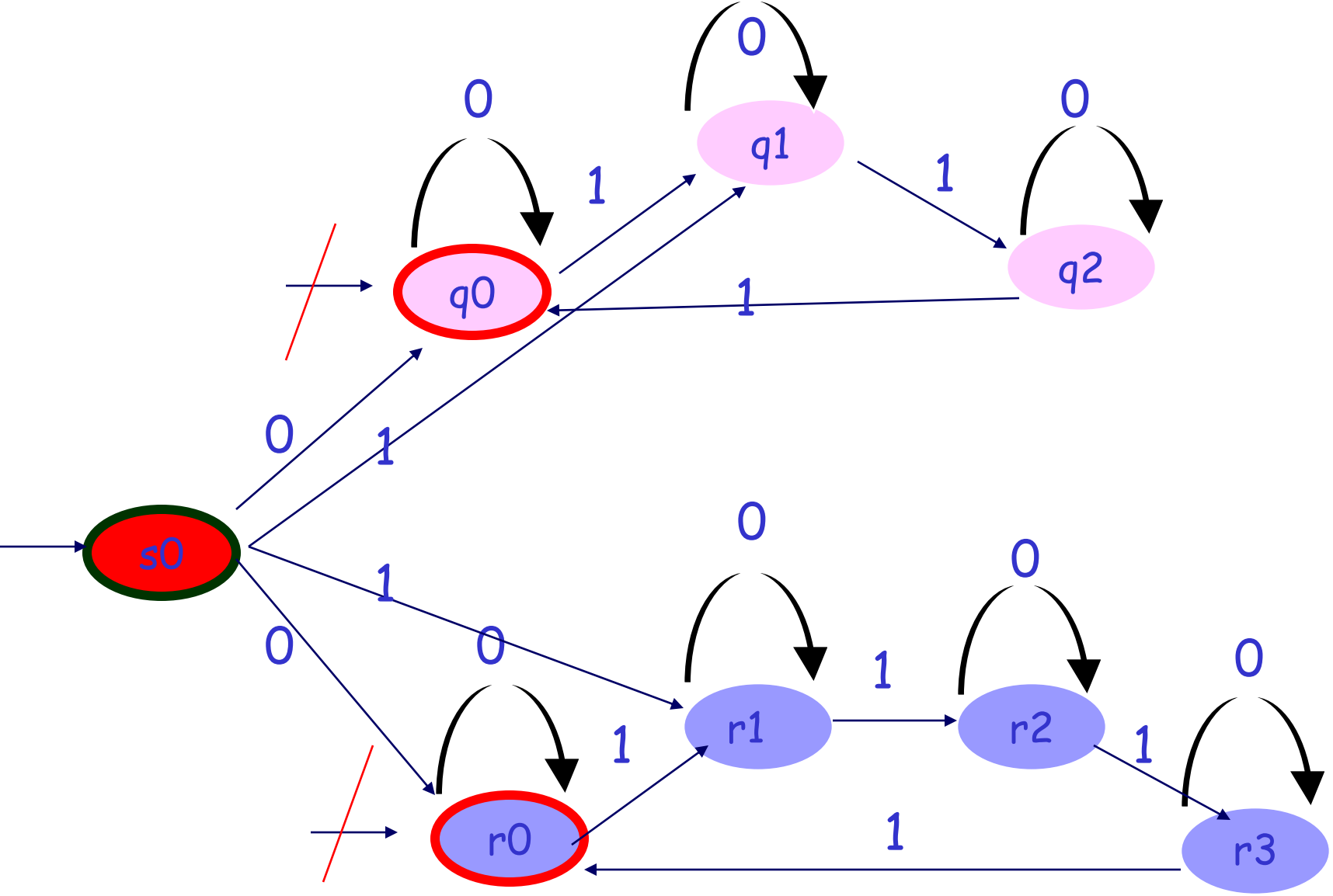
  Continuam os mesmos

$$L(M3) = L(M1) \cup L(M2)$$



Exemplo

- Use o Lema 3.1 para fazer um AFND que reconheça $L(M) = \{ x \in \{0,1\}^* \mid$ o nro de 1's em x é múltiplo de 3 OU de 4$\}$



M3 = ?

Lema 3.2 (H&U, 69) - Complemento

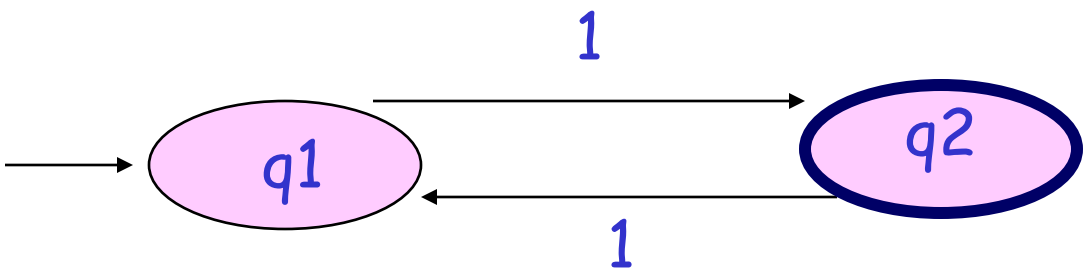
- A classe dos conjuntos aceitos por um AF é fechada sobre o complemento (Se L é uma LR então $\overline{L}$ também é).

- Prova: M1 = (Q, $\Sigma 1$,$\delta 1$, qo,F) é um AF que aceita um conjunto L1. Seja $\Sigma 2$ um alfabeto contendo $\Sigma 1$ (pode ser o mesmo) e d um estado $\notin$ Q. M2 aceita $\Sigma 2^*$ - L1.

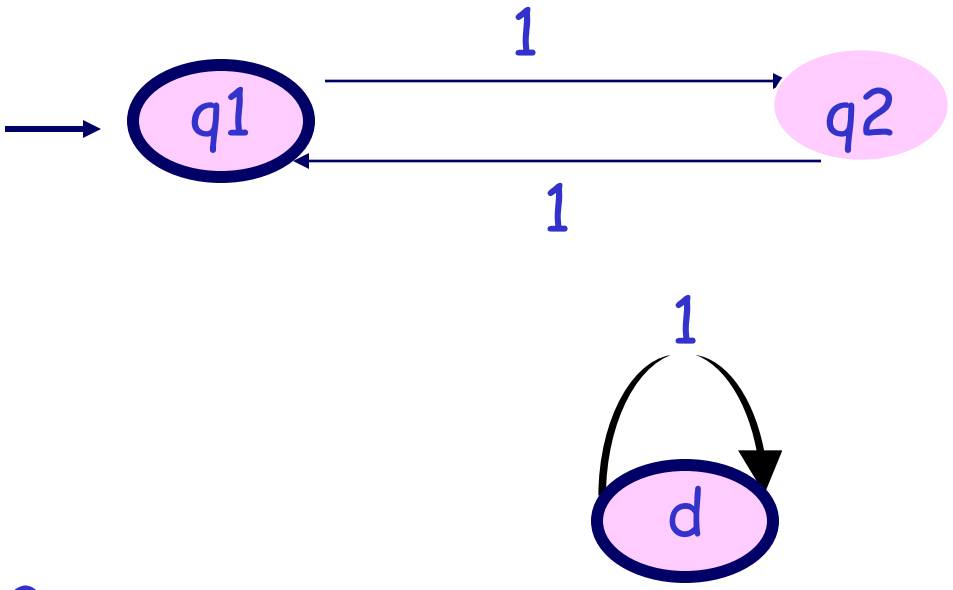
- M2 = $(Q \cup \{d\}, \Sigma 2, \delta 2, qo, (Q - F) \cup \{d\})$

- $\delta 2(q,a) = \delta 1(q,a)$ para cada $q \in Q$ e $a \in \Sigma 1$ (continua o anterior)
- $\delta 2(q,a) = d$ para cada $q \in Q$ e $a \in (\Sigma 2 - \Sigma 1)$ do particular estado
- $\delta 2(d,a) = d$ para cada $a \in \Sigma 2$

- Exemplo: Use o Lema 3.2 para reconhecer o complemento de L(M1) = ($x \in \{1\}^* |$ x possui um nro ímpar de 1's}.
- Faça para $\Sigma 2 = \Sigma 1 = \{1\}$ e para $\Sigma 2 \supset \Sigma 1$, isto é, $\Sigma 2 = \{0...9\}$



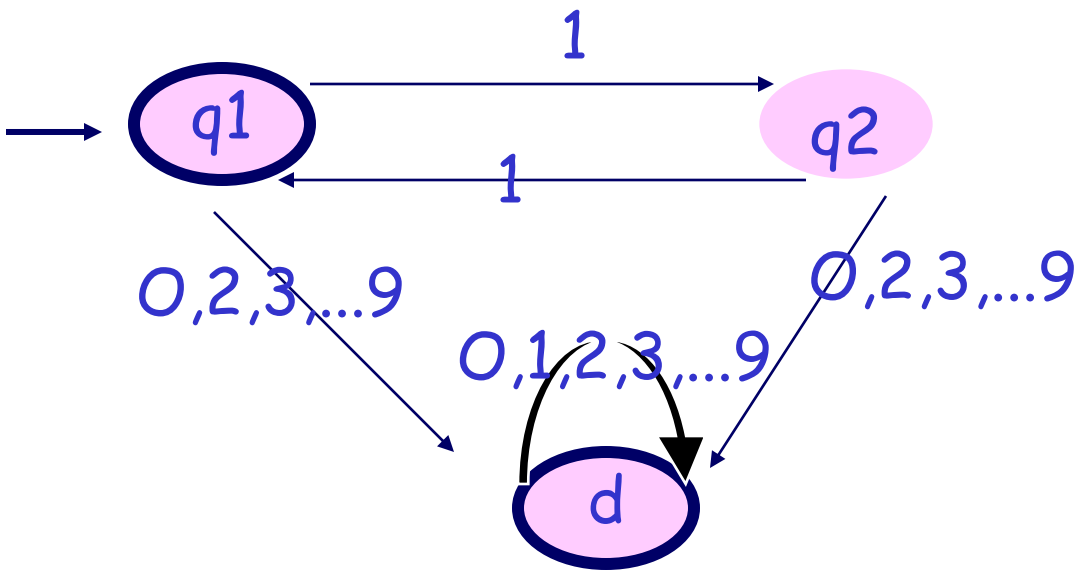
M1: reconhece ímpar



M2: Reconhece par

D é um estado inacessível que pode ser eliminado. Portanto M pode ser reduzido

$\overline{M}$ = ?



Só não reconhece cadeias de 1´s de tamanho ímpar

$\overline{M}$ = ?

Teo 3.6 (H&U, 69) - Intersecção

- A classe das LR é fechada sobre a intersecção. Se L1 é LR e L2 é LR então $L1 \cap L2$ também é.

- Prova: dos Lemas 3.1 e 3.2 e da Lei de Morgan, desde que a própria intersecção usa operações de união e complemento.

$$L1 \cap L2 = \overline{\overline{L1} \cup \overline{L2}}$$

- A classe dos conjuntos aceitos por um AF forma uma Álgebra de Boole. Isto é, é uma coleção de conjuntos fechados sobre a união, complemento e intersecção.

Teo 3.8 (H&U, 69) - Concatenação

- A classe dos conjuntos aceitos por um AF é fechada sobre a concatenação. Se L1 é LR e L2 é LR, então L1.L2 é LR.
- (Prova via ER: direta)
- Prova: Seja M1 = (Q1, Σ1,δ1, q1,F1) aceitando L1 e M2 = (Q2, Σ2,δ2, q2,F2) aceitando L2.
- Assumimos Q1 e Q2 disjuntos e $\Sigma = \Sigma 1 = \Sigma 2$.
- M3 = (Q1 $\cup$ Q2, Σ,δ3, q1,F) é um AFND onde:

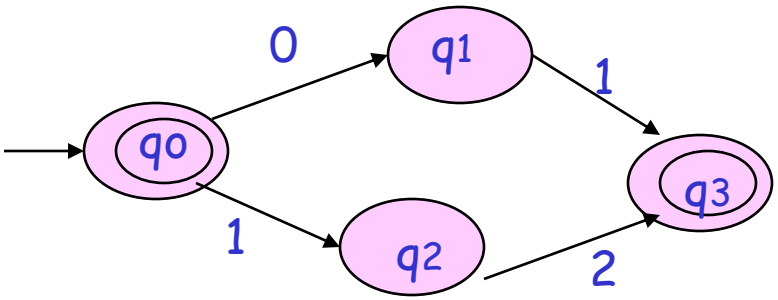
- $\delta 3(q,a) = \{\delta 1(q,a)\}$ para cada $q \in (Q1 - F1)$ e $a \in \Sigma$.

M3 age como M1 para o começo da cadeia possivelmente vazia

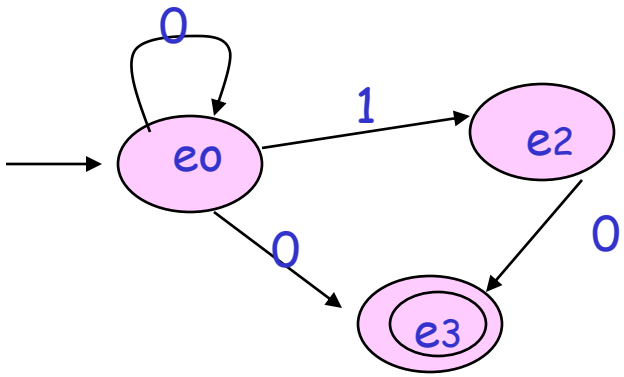
- $\delta 3(q,a) = \{\delta 1(q,a), \delta 2(q2,a)\}$ para cada $q \in F1$ e $a \in \Sigma$. M3 continua em M1 ou vai para M2
- $\delta 3(q,a) = \{\delta 2(q,a)\}$ para cada $q \in Q2$. M3 age como M2 depois que a cadeia de entrada pertence a L2.

- Se $\lambda \notin L2$ então F = F2 (vai aceitar em M2)
- Se $\lambda \in L2$ então $F = F1 \cup F2$ (aceita também cadeias de L1).

Exemplo

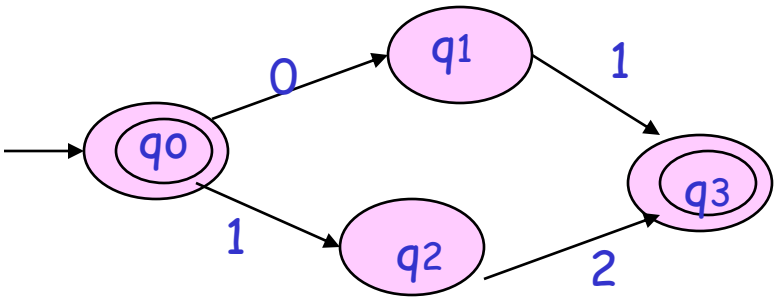


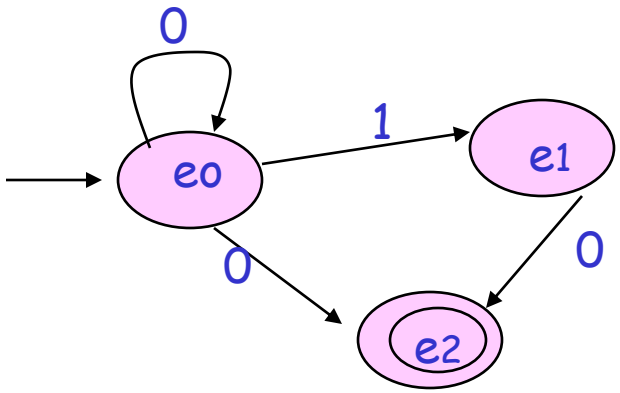
L1={01,12,λ}



L2= 0*(10+0)

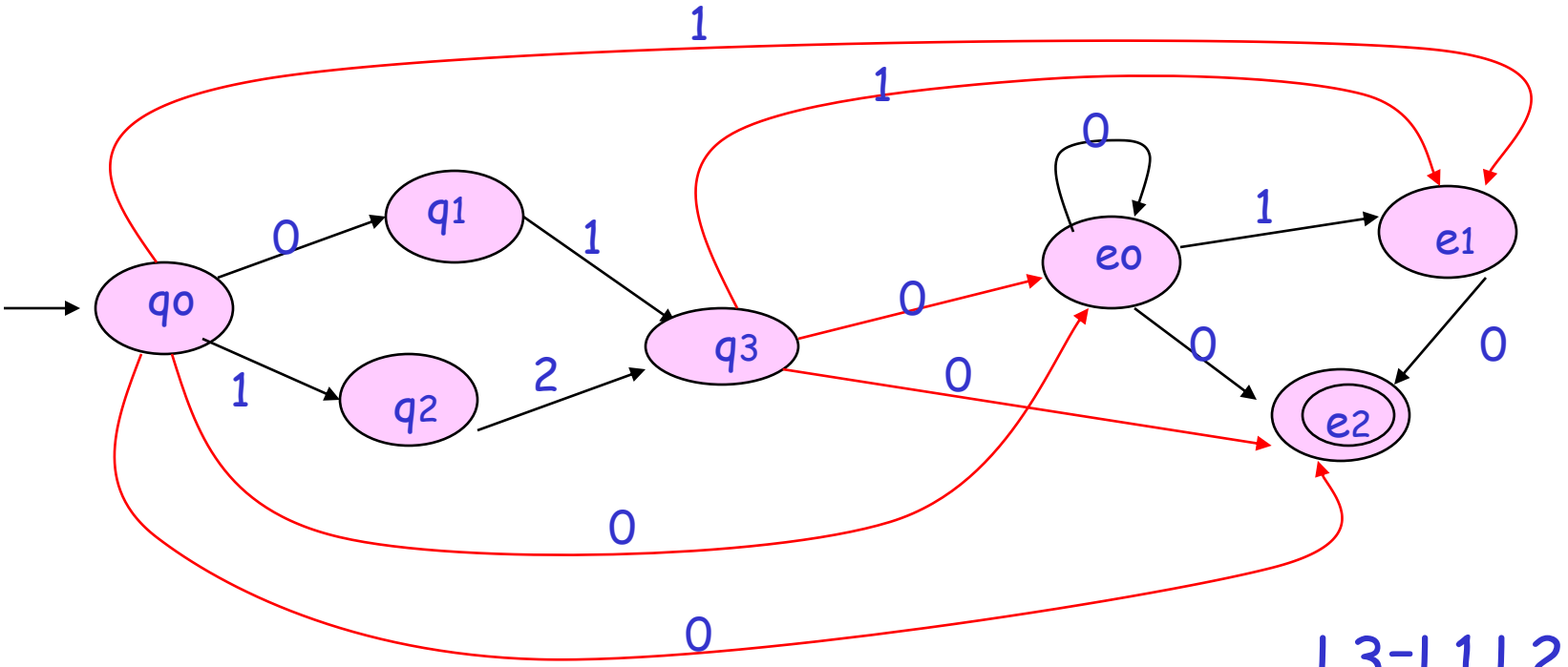
Exemplo





L1={01,12,λ}

L2= 0*(10+0)

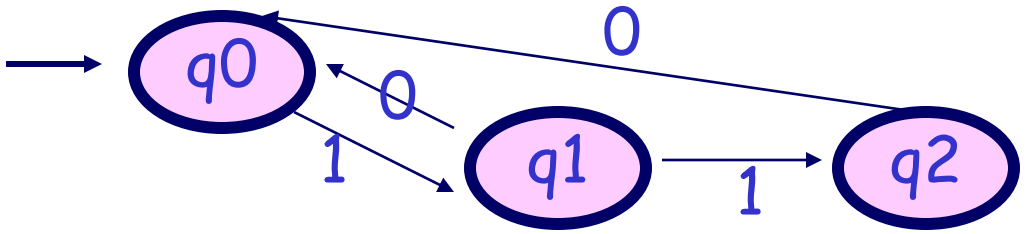


L3=L1.L2

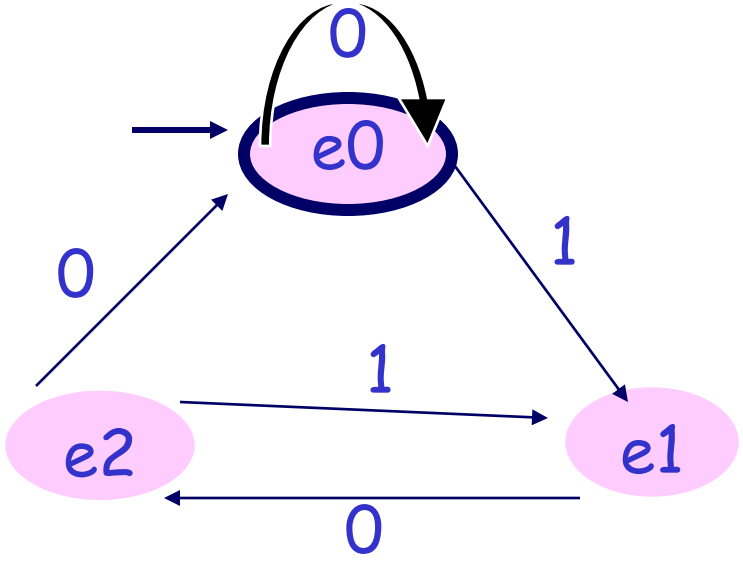
Exemplo

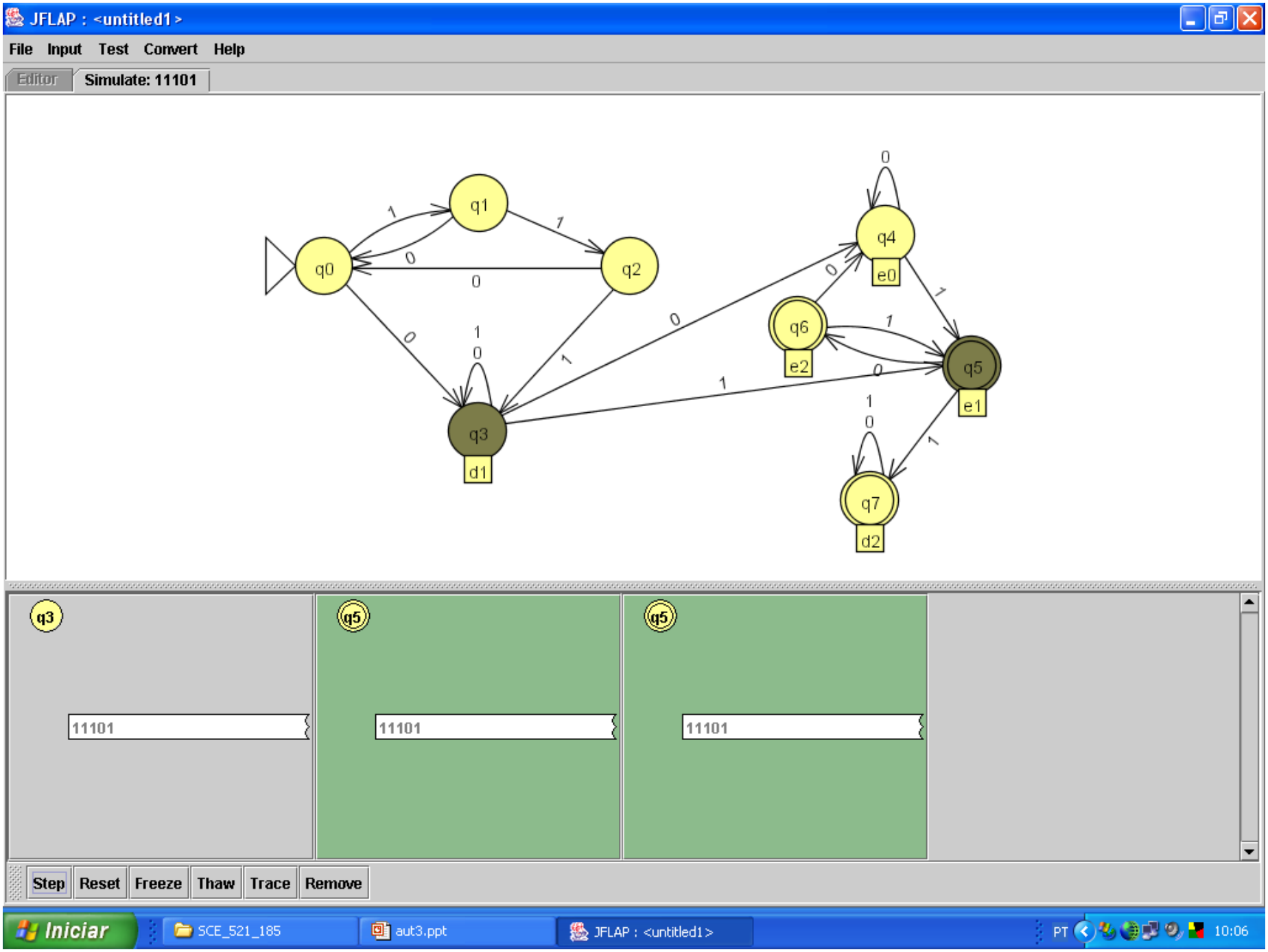
- Use o Lema 3.2 (complemento) e o Teo 3.8 (concatenação) para construir um AF M que a partir de M1 e M2 reconheça $L = \overline{L1} \,.\, \overline{L2}$

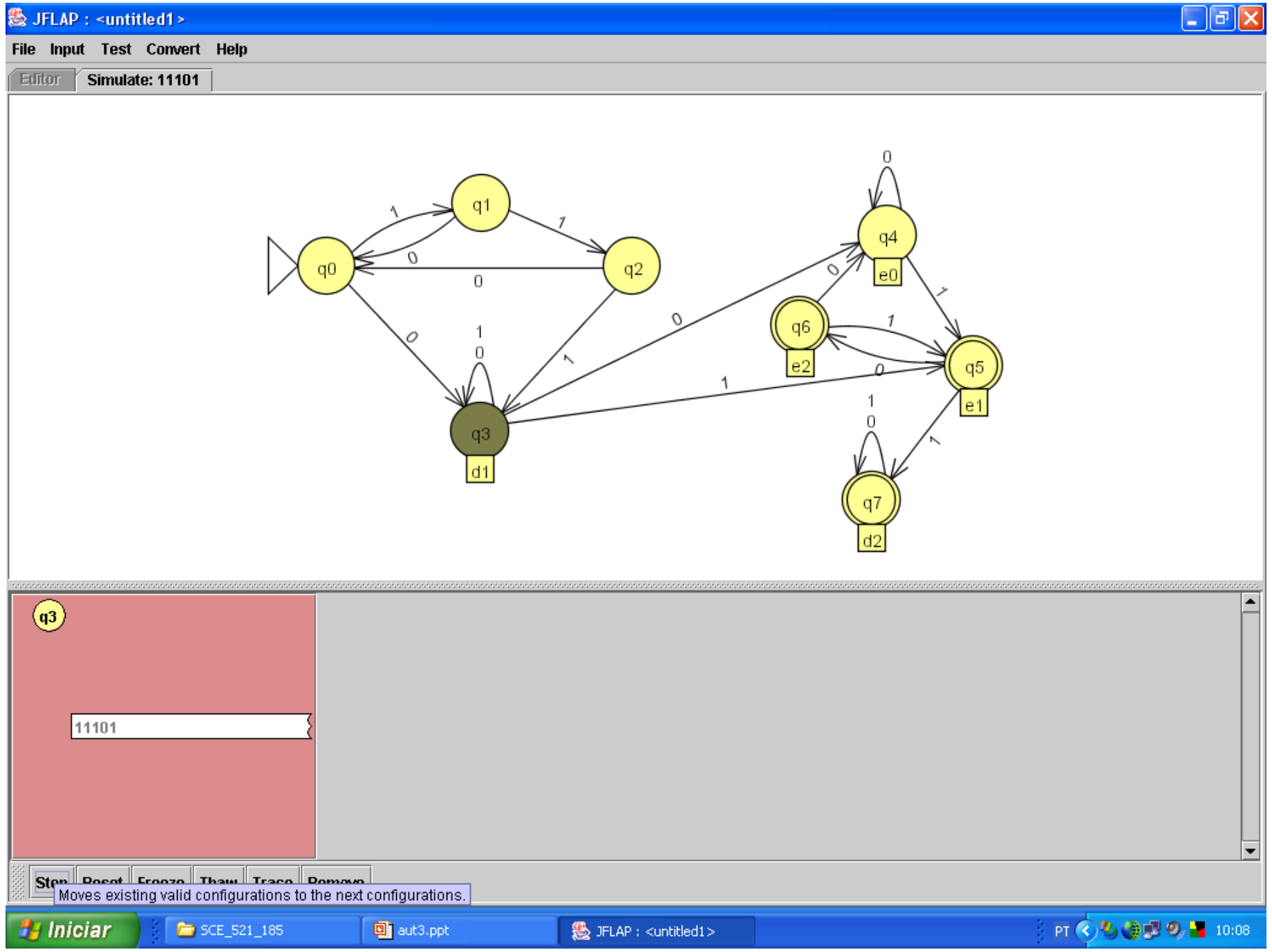
M1



M2




JFLAP : <untitled1>
File Input Test Convert Help
Editor
Simulate: 11101
q0
q1
q2
q3
d1
q4
e0
q5
e1
q6
e2
q7
d2
0
1
q3
q5
q5
11101
11101
11101
Step
Reset
Freeze
Thaw
Trace
Remove
Iniciar
SCE_521_185
aut3.ppt
JFLAP : <untitled1>
PT
10:06


JFLAP : <untitled1>
File Input Test Convert Help
Editor
Simulate: 11101
q0
q1
q2
q3
d1
q4
e0
q5
e1
q6
e2
q7
d2
q3
11101
Stop
Reset
Freeze
Thaw
Trace
Remove
Moves existing valid configurations to the next configurations.
Iniciar
SCE_521_185
aut3.ppt
JFLAP : <untitled1>
PT
10:08

Teo 3.9 (H&U, 69) - Fechamento

- A classe dos conjuntos aceitos por um AF é fechada sobre o fechamento. Se L é LR então L* é LR.
- Prova: Seja M = (K, Σ, δ, qo,F) um AF que aceita L. M´ = (K $\cup$ {qo´}, Σ, δ´ , qo´ , F $\cup$ {qo´}) aceita L*.

⇒ qo´ é também final pois $\lambda \in$ao fecho

$$\delta'(q,a) = \begin{cases} \{\delta(q,a), qo)\} \text{ se } \delta(q,a) \in F \\ \{\delta(q,a)\} \text{ c.c. para } \forall\ q \in K \end{cases}$$

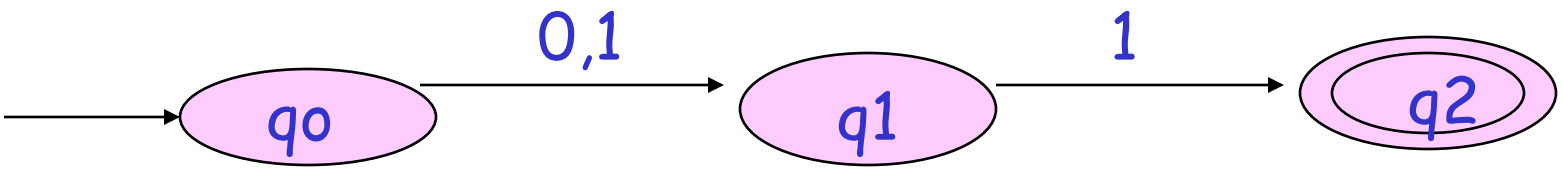
(todas as transições que estavam chegando no final, ganham uma cópia para voltar ao início)

$$\delta'(qo',a) = \begin{cases} \{\delta(qo,a),qo\} \text{ se } \delta(qo,a) \in F \\ \{\delta(qo,a)\} \text{ cc} \end{cases}$$

Exemplo

- Use o Teo 3.9 para construir um AF que reconheça o fecho de L = {01,11}

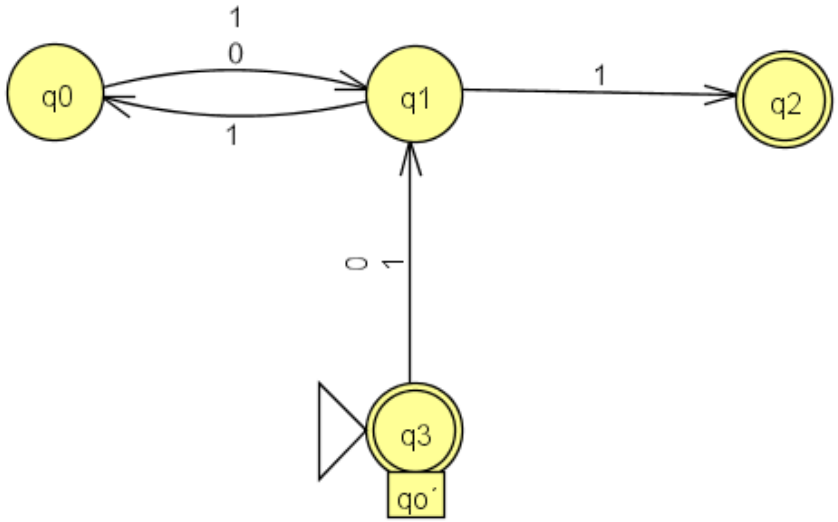
- L* = {λ, 01,11,0101,0111,1101,1111, 010101, ...}



JFLAP : <untitled1>

File Input Test Convert Help

Editor | Multiple Inputs



Input	Result
	Accept
01	Accept
11	Accept
0101	Accept
0111	Accept
1101	Accept
1111	Accept
010101	Accept

Run Inputs | Clear | Enter Lambda

Iniciar | Microsoft PowerPoint ... | SCE_521_185 | JFLAP : <untitled1> | PT | 19:54

Teo 4.10 (H,M,U,2001) Diferença

- Se L e M são linguagens regulares então L – M também é.

- Prova: $L - M = L \cap \overline{M}$. Como o complemento e a intersecção de linguagens regulares também são regulares, L – M também é.

Como testar a pertinência em uma LR

Dadas uma cadeia w e uma linguagem regular L, $w \in L$?

- Se L é representada por um AFD:
  - simule o autômato e verifique se um estado final é alcançado. Se $|w|=n$ e o AFD é representado por um estrutura de dados adequada (matriz de transição), então cada transição exigirá um tempo constante e a verificação levará tempo $O(n)$. (veja que o tamanho da entrada é dado pelo comprimento da cadeia)

Como testar a equivalência de LRs, ou se descritores distintos definem a mesma linguagem

- Há uma forma para minimizar um AFD. Ou seja, sempre é possível encontrar um AFD equivalente que tenha o número mínimo de estados.
- Esse autômato mínimo é **único**: dados dois AFDs quaisquer, com número mínimo de estados, sempre podemos renomear os estados de modo que os dois AFDs se tornem iguais.