

Automatos Finitos

Operações Fechadas sobre LR's Aplicações

(H&U, 1969), (H&U, 1979),
(H;M;U, 2001) e (Menezes, 2002)
(Sipser, 1997)

a^n



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 ($a_1a_2\dots a_n \rightarrow a_n\dots a_2a_1$).
- 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:
 - $L(M) = \{ x \in \{0,1\}^* \mid \text{o nro de 1's em } x \text{ é múltiplo de 3 OU de 4} \}$
 - 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.

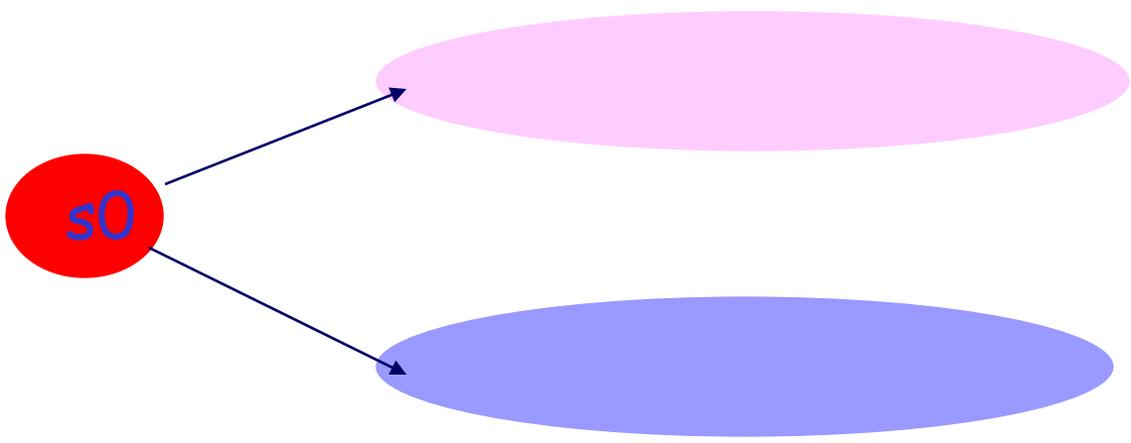
- PROVA: Sejam $L1$ e $L2$ reconhecidas por AF $M1 = (Q1, \Sigma1, \delta1, q0, F1)$ e $M2 = (Q2, \Sigma2, \delta2, r0, F2)$. Suponha $Q1 \cap Q2 = \emptyset$ e $s0 \notin Q1; s0 \notin Q2$.

$M3 = (Q1 \cup Q2 \cup \{s0\}, \Sigma1 \cup \Sigma2, \delta3, s0, F)$
é um AFND onde:

- Se $\lambda \in L1$ ou $L2$ então $F = F1 \cup F2 \cup \{s0\}$ cc.
 $F = F1 \cup F2$

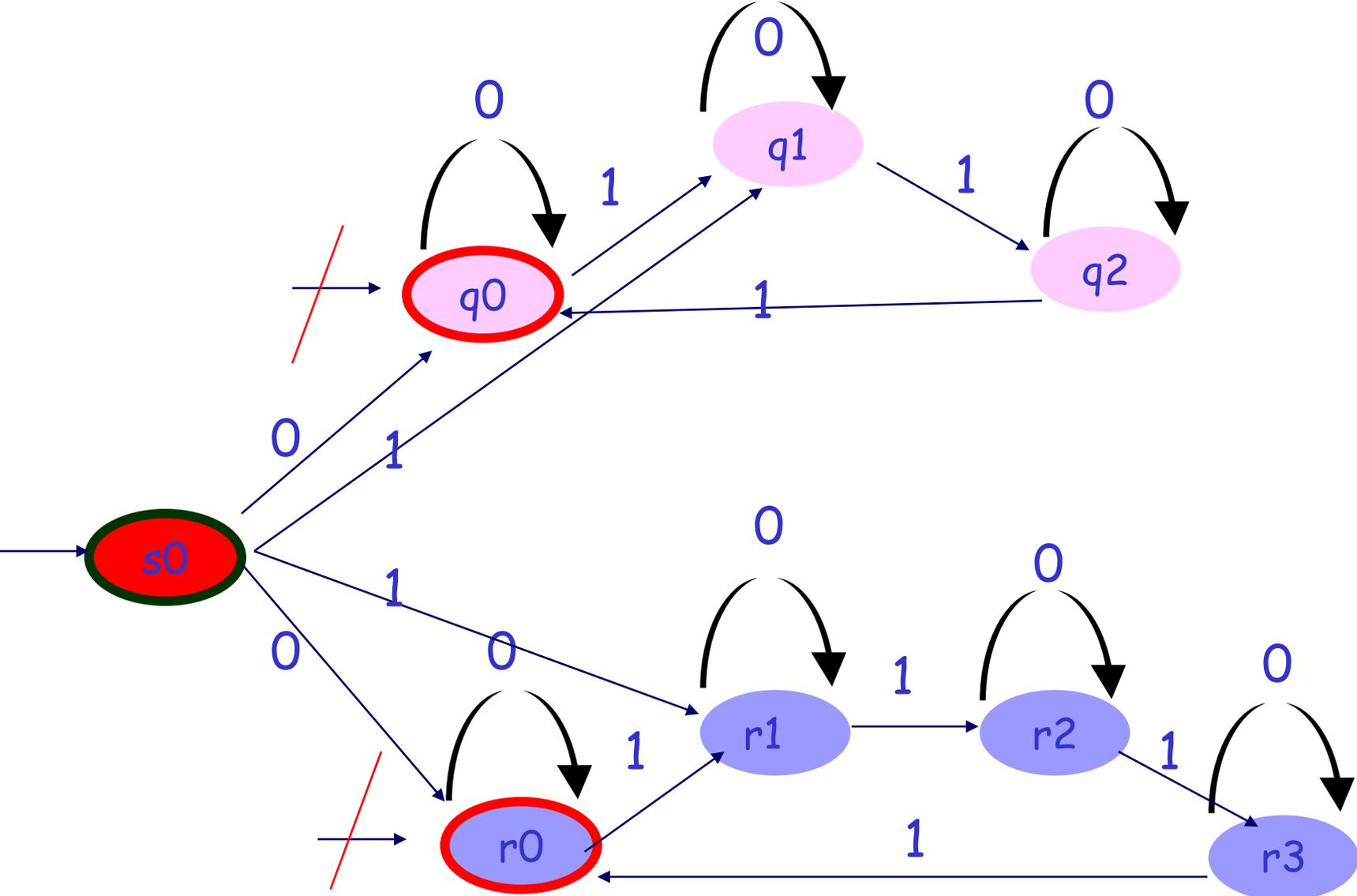
- δ_3 {
 - 1) $\delta(s0, a) = \{\delta(q0, a), \delta(r0, a)\}$ para todo $a \in \Sigma1 \cup \Sigma2$
 - 2) $\delta(q, a) = \delta1(q, a)$ para todo $q \in Q1$ e $a \in \Sigma1$
 - 3) $\delta(q, a) = \delta2(q, a)$ para todo $q \in Q2$ e $a \in \Sigma2$
 } Continuum os mesmos

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



Exemplo

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



$M_3 = ?$

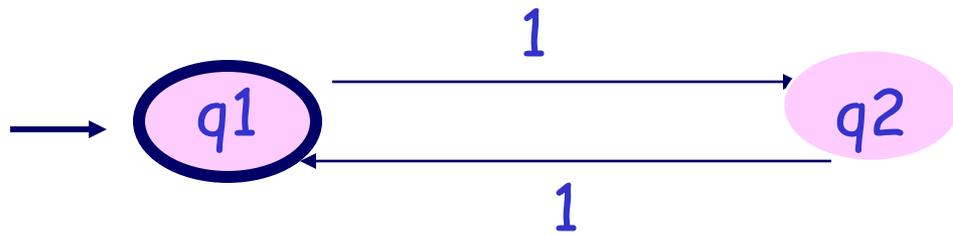
Outro Teorema para União de AF's

- No livro do Sipser, pag 59 a união é feita através de um AF com movimentos nulos.

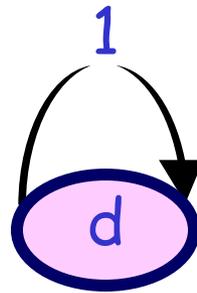
Lema 3.2 (H&U, 69) - Complemento

- A classe dos conjuntos aceitos por um AF é fechada sobre o complemento (Se L é uma LR então \bar{L} também é).
- Prova: $M_1 = (Q, \Sigma_1, \delta_1, q_0, F)$ é um AF que aceita um conjunto S_1 . Seja Σ_2 um alfabeto contendo Σ_1 (pode ser o mesmo) e d um estado $\notin Q$. M_2 aceita $\Sigma_2^* - S_1$.
- $M_2 = (Q \cup \{d\}, \Sigma_2, \delta_2, q_0, (Q - F) \cup \{d\})$

- $\delta_2(q,a) = \delta_1(q,a)$ 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 $L(M_1) = (x \in \{1\}^* \mid x \text{ possui um nro impar de 1's})$.
- Faça para $\Sigma_2 = \Sigma_1 = \{1\}$ e para $\Sigma_2 \subset \Sigma_1$, isto é, $\Sigma_2 = \{0\dots 9\}$

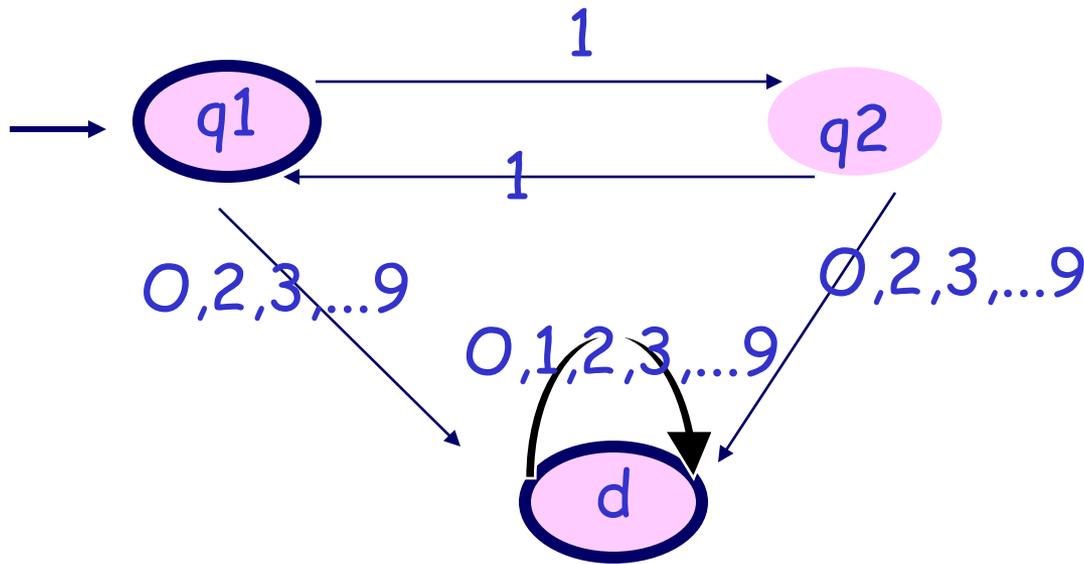


Reconhece um
nro par de 1's



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

$\overline{M} = ?$



Só não
reconhece nro
impar de 1's

$\overline{M} = ?$

Teo 3.6 (H&U, 69) - Intersecção

- A classe dos conjuntos aceites por um AF forma uma Álgebra de Boole. Isto é, é uma coleção de conjuntos fechados sobre a união, complemento e intersecção.
- Prova: dos Lemas 3.1 e 3.2 e da Lei de Morgan, desde que a própria **intersecção** usa operações de união e complemento.
- $L1 \cap L2 = \overline{\overline{L1} \cup \overline{L2}}$

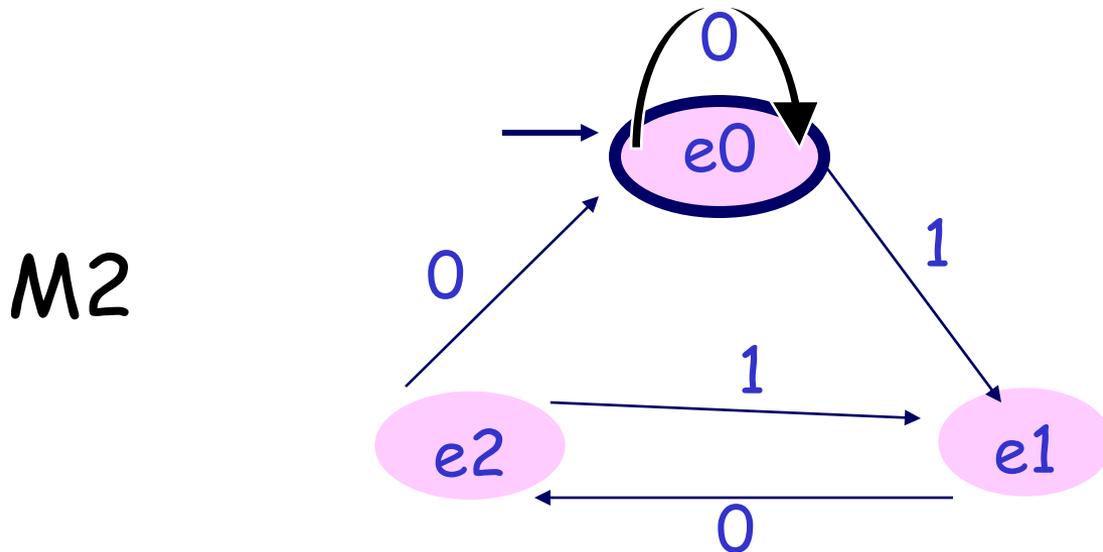
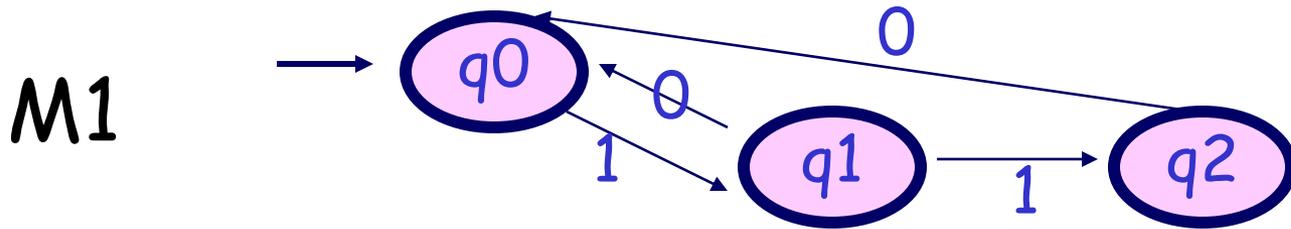
Teo 3.8 (H&U, 69) - Concatenação

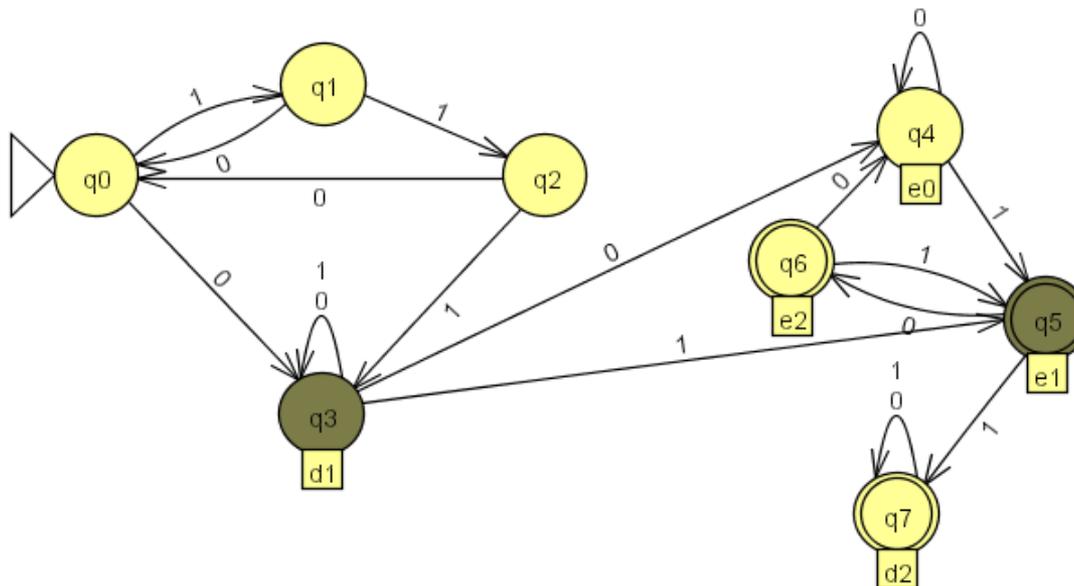
- A classe dos conjuntos aceitos por um AF é fechada sobre o produto (ou concatenação).
- Prova: Seja $M1 = (Q1, \Sigma1, \delta1, q1, F1)$ aceitando $L1$ e $M2 = (Q2, \Sigma2, \delta2, q2, F2)$ aceitando $L2$.
- Assumimos $Q1$ e $Q2$ disjuntos e $\Sigma = \Sigma1 = \Sigma2$.
- $M3 = (Q1 \cup Q2, \Sigma, \delta3, q1, F)$ é um AFND onde:

- $\delta_3(q,a) = \{\delta_1(q,a)\}$ para cada $q \in (Q_1 - F_1)$. M_3 age como M_1 para o começo da cadeia (possivelmente vazia)
- $\delta_3(q,a) = \{\delta_1(q,a), \delta_2(q_2,a)\}$ para cada $q \in F_1$. M_3 continua em M_1 ou vai para M_2
- $\delta_3(q,a) = \{\delta_2(q,a)\}$ para cada $q \in Q_2$. M_3 age como M_2 depois que a cadeia de entrada pertence a L_2 .
- Se $\lambda \notin L_2$ então $F = F_2$ (vai aceitar em M_2)
- Se $\lambda \in L_2$ então $F = F_1 \cup F_2$ (pode ser que não tenha nada em M_2).

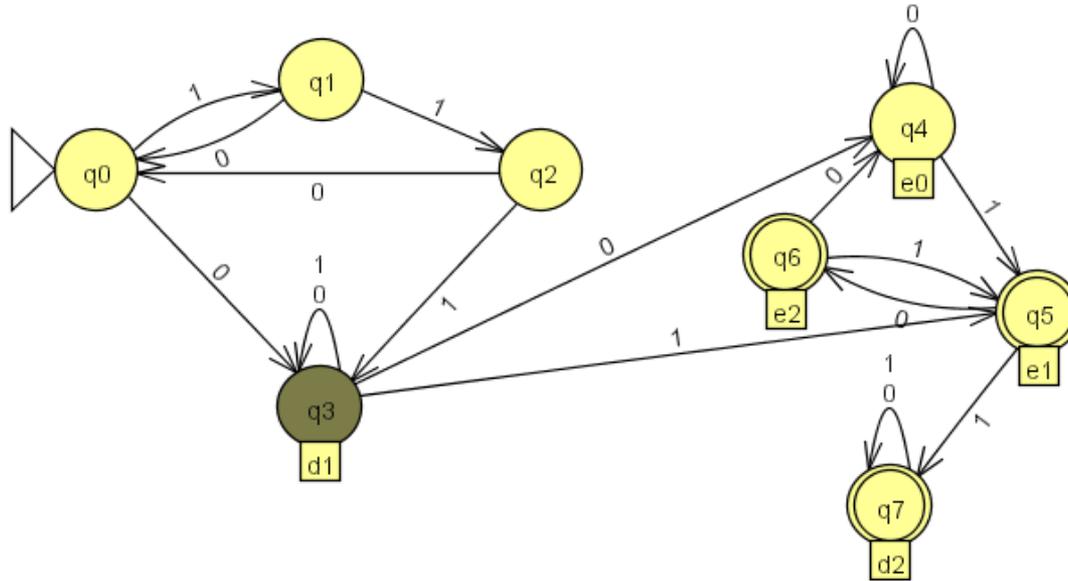
Exemplo

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





<p>q3</p> <p>11101</p>	<p>q5</p> <p>11101</p>	<p>q5</p> <p>11101</p>	
------------------------	------------------------	------------------------	--



q3

11101

Step Reset Freeze Thaw Trace Remove

Moves existing valid configurations to the next configurations.

Teo 3.9 (H&U, 69) - Fechamento

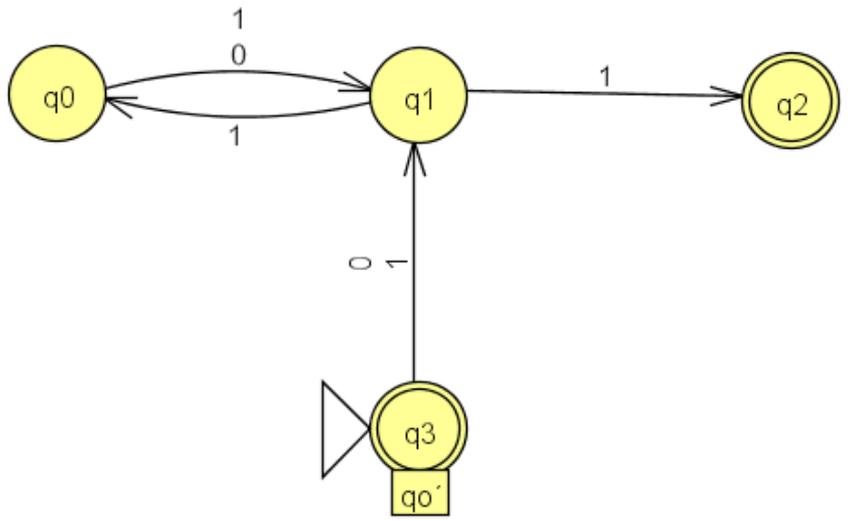
- A classe dos conjuntos aceitos por um AF é fechada sobre o fechamento.
 - Prova: Seja $M = (K, \Sigma, \delta, q_0, F)$ um AF que aceita L . $M' = (K \cup \{q_0'\}, \Sigma, \delta', q_0', F \cup \{q_0'\})$ aceita L^* .
 -  q_0' é também final pois $\lambda \in a_0$ fecho
- $$\delta'(q, a) = \begin{cases} \{\delta(q, a), q_0\} & \text{se } \delta(q, a) \in F \\ \{\delta(q, a)\} & \text{c.c. para } \forall q \in K \end{cases}$$

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

$$\delta'(q_0', a) = \begin{cases} \{\delta(q_0, a), q_0\} & \text{se } \delta(q_0, a) \in F \\ \{\delta(q_0, a)\} & \text{cc} \end{cases}$$

Exemplo

- Use o Teo 3.9 para construir um AF que reconheça o fecho de $L = \{01,11\}$
- $L^* = \{\lambda, 01,11,0101,0111,1101,1111, 010101, \dots\}$



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

Teo 4.10 (H,M,U,2001) Diferença

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

Aplicações: Analisadores Léxicos

- **Analisadores léxicos (AL) de compiladores.** O AL é a interface entre o programa fonte e o resto do compilador. Ele é responsável principalmente por:
 - “empacotar” os caracteres do programa e lhes dar um rótulo que será usado pelo analisador sintático montar a árvore sintática.
 - Os rótulos são:
 - identificadores,
 - os nomes dos símbolos simples (<, =, [,), etc.),
 - os nomes dos símbolos compostos (:= , <>, <=, .., etc.),
 - constantes inteiras,
 - constantes reais,
 - constantes literais (cadeias e caracteres)
 - constantes lógicas (true/false),
 - o nome das palavras-chaves.

AL são modelados por AF para depois serem programados em uma linguagem de programação.

Outra opção é utilizar um gerador de analisadores léxicos como o Lex, Flex.

A entrada para esses geradores é uma **expressão regular** e a saída é um programa que gerencia uma enorme tabela de transição de estados.

- Já fizemos um AF para identificadores, inteiros do Pascal, e operadores relacionais
 - Como exercício façam para os outros elementos do Vt.
- Atentem que os números reais, por exemplo, **não** são iguais em todas as linguagens. Algol e Fortran permitem **5.** e **.5** quando Pascal, por exemplo, **não**!!
- Para a modelagem de um AL podemos unir todos os AF's em um único que reconhecerá todo o Vt da linguagem escolhida.
- Nele também representamos o tratamento dado aos símbolos br/CR/LF/tab e comentários
 - porém **NÃO** podemos reconhecer/aceitar esses elementos.

- Observem, entretanto que a modelagem com AF mostra **o que** o Analisador Léxico deve reconhecer **MAS não mostra como**.
 - Por exemplo, nada diz sobre o que fazer quando uma cadeia pode ter 2 análises como é o caso de:

2.3 (real **ou** inteiro seguido de ponto seguido de real)

Ou

<= (menor seguido de igual **ou** menor igual)

OU

Program (identificador **ou** palavra reservada program)

Regras de Desambiguação

- Assim, precisamos de regras para desambiguar esses casos.
- Usamos as regras:
 - escolha a maior cadeia
 - Dê preferência para a formação de:
 - palavras-reservadas em detrimento de identificadores
 - Estas regras estão embutidas em geradores de Analisadores Léxico, como o LEX.