Capítulo III : Expressões Regulares e Linguagens Regulares

- Um Autómato Finito (AFD ou AFND) descreve uma Linguagem.
- Uma **Expressão Regular** (ER) é a formalização algébrica de um Autómato Finito.
- A Linguagem descrita por uma Expressão Regular E chama-se **Linguagem Regular** e denota-se por L(E).

- Em todos os comandos que envolvem a pesquisa de palavras (tais como o grep do Unix) o padrão é definido por uma ER que é convertida no correspondente AF cujo funcionamento é simulado sobre o(s) ficheiro(s). Geradores de Analisadores Léxicos, tais como o Lex ou o Flex constroem AFD's a partir de definições de "símbolos terminais" descritos por ER's.

Operações sobre Linguagens:

- **União** de duas Linguagens L e M,

  $\mathbf{L \cup M} = \{w \mid w \in L \vee w \in M\}$

- **Concatenação** de duas Linguagens L e M,

  $\mathbf{LM} = \{xy \mid x \in L \wedge y \in M\}$ (também denotada por **L.M**)

- **Potência** de ordem n de uma Linguagem L,

$$\mathbf{L^n} = \begin{cases} \{\varepsilon\} & \text{se } n = 0 \\ L^{n-1}L & \text{se } n > 0 \end{cases}$$

- **Fecho de Kleene** de uma Linguagem L,

  $\mathbf{L^*} = \bigcup_{i \geq 0} L^i$

$\mathbf{L^+} = \cup_{i>0} L^i$ e $L^0 = \{\varepsilon\}$. Portanto $L^+ = LL^*$ e $L^* \setminus L^+ = \{\varepsilon\}$

$\emptyset$ representa a linguagem vazia.

$\emptyset^0 = \{\varepsilon\}$ e $\emptyset^i = \emptyset$ para qualquer $i \geq 1$. Portanto $\emptyset^* = \{\varepsilon\}$

Definição de Linguagem Regular / Construção de Expressões Regulares

sobre um alfabeto Σ:

Casos Base: (Os operandos)

1. As constantes ε e $\emptyset$ são Expressões Regulares.

   $L(\varepsilon) = \{\varepsilon\}$ e $L(\emptyset) = \emptyset$

2. Qualquer símbolo $a \in \Sigma$ é uma Expressão Regular

   $L(a) = \{a\}$

3. Uma variável L representa uma Linguagem Regular

Passos Indutivos: (Os operadores)

1. Se E e F são ER's então **E+F** é uma ER.

   $L(E+F) = L(E) \cup L(F)$

2. Se E e F são ER's então **EF** é uma ER.

   $L(EF) = L(E)L(F)$

3. Se E é uma ER então **E*** é uma ER.

   $L(E^*) = (L(E))^*$

4. Se E é uma ER então **(E)** é uma ER.

   $L((E)) = L(E)$

De outra forma,

	Sintaxe	Semântica
Base	$\emptyset$	$L(\emptyset) = \emptyset$
	ε	$L(\varepsilon) = \{\varepsilon\}$
	a	$L(a) = \{a\}$
Indução	(E+F)	$L((E+F)) = L(E) \cup L(F)$
	(EF)	$L((EF)) = L(E)L(F)$
	(E*)	$L((E^*)) = L(E)^*$

Os parêntesis servem apenas para indicar a ordem pela qual as operações devem ser efectuadas. Para evitar o seu excesso, usamos as convenções:

- **Precedência** dos operadores:

1º	*
2º	.
3º	+

  por exemplo, E + F* G = (E + ((F*) G))

- **Associatividade**:

  (E(FG)) = ((EF)G) = EFG

  (E+(F+G)) = ((E+F)+G) = E+F+G

Exemplos:

$L((0 + 1)^*) = (\{0\} \cup \{1\})^* = \{0,1\}^*$
(todas as palavras possíveis no alfabeto binário)

$L(0 + 10^*) = \{0, 1, 10, 100, 1000, \ldots\}$

$L((0 + 1)^* (0 + 11))$
(palavras terminadas em 0 ou em 11)

$L(0^* + (0^* 10^* 10^* 10^*)^*)$
(palavras onde o número de 1's é divisível por 3)

$L((0 + 1)^* 001 (0 + 1)^*)$
(palavras que contêm 001)

$L((10)^* + (01)^* + 0(10)^* + 1(01)^*)$
(palavras com 0's e 1's alternados)

$L((\varepsilon + 1) (01)^* (\varepsilon + 0))$
(palavras com 0's e 1's alternados)

$L((0 + \varepsilon) (1 + 10)^*)$
(palavras sem 0's consecutivos)

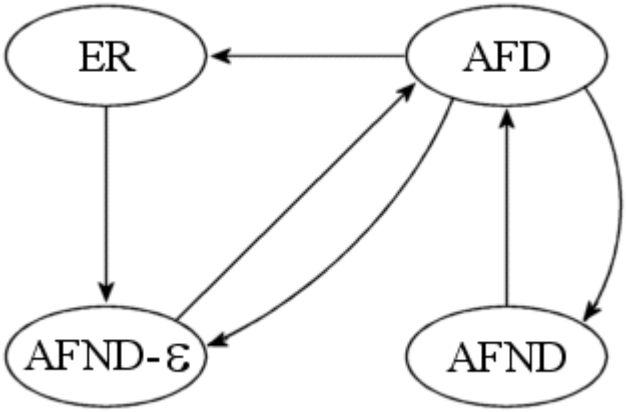
$L((1 + 01)^* (0 + \varepsilon))$
(palavras sem 0's consecutivos)

$L((0 + 1)^* 00 (0 + 1)^*)$
(palavras com, pelo menos, um par de 0's consecutivos)

$L((0 (0 + 1))^*)$
(palavras de comprimento par, com 0's nas posições ímpares)

Expressões Regulares e Autómatos Finitos

- Os Autómatos Finitos definem Linguagens.
- Existe uma equivalência entre os Autómatos Finitos Não-Deterministas (AFND e AFND-ε) e os Autómatos Finitos Deterministas (AFD).
- Essa equivalência significa que reconhecem a mesma Linguagem.
- As Expressões Regulares definem **Linguagens Regulares**.
- Resta mostrar a equivalência entre ER's e AF's.



ER → AFND-ε

Teorema: Toda a Linguagem definida por uma ER é também definida por um AFND-ε.

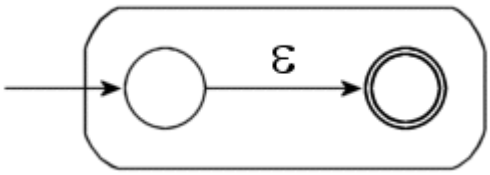
Dada uma Expressão Regular R,
construir um AFND-ε E, tal que L(R) = L(E)

Demonstração:

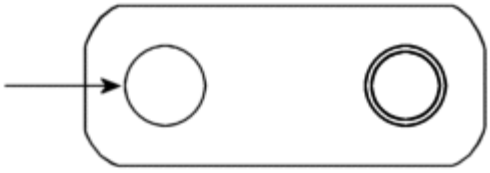
Indução sobre os operadores, de acordo com a definição.

Casos Base (reconhecimento dos operandos):

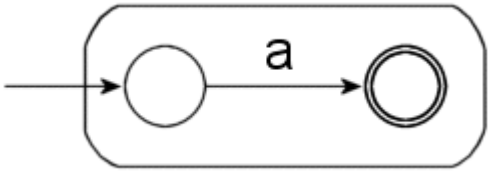
1. $L(\varepsilon) = \{\varepsilon\}$



2. $L(\emptyset) = \emptyset$

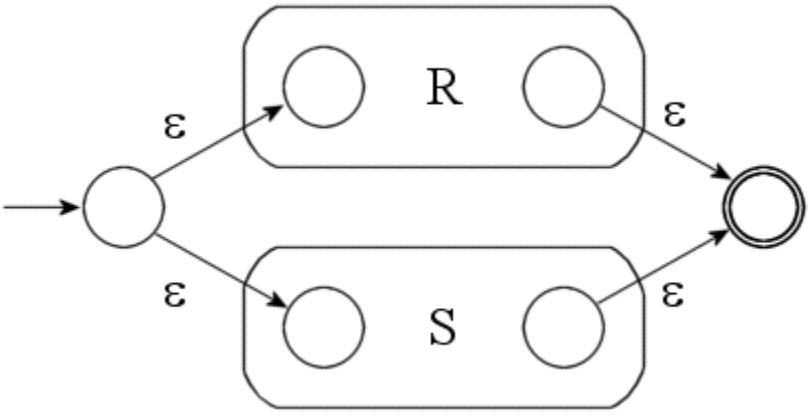


3. $L(a) = \{a\}$

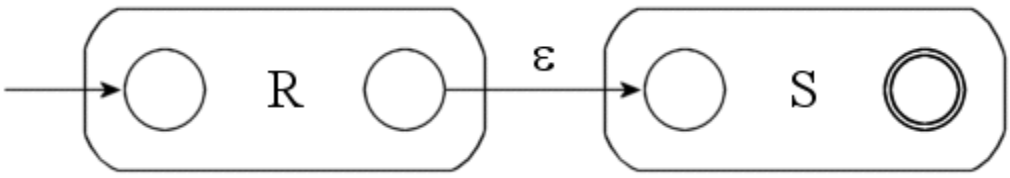


Passos Indutivos (reconhecimento dos operadores):

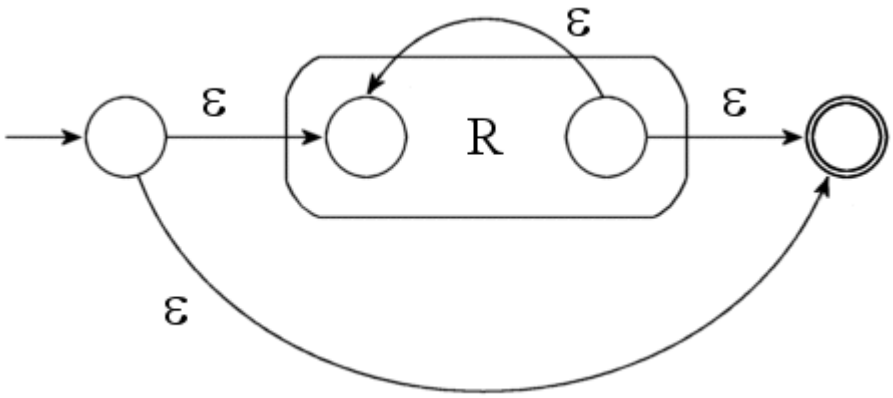
1. $L(R+S) = L(R) \cup L(S)$



2. $L(RS) = L(R)\ L(S)$



3. L(R*) = (L(R))*



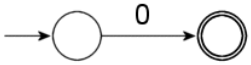
4. L((R)) = L(R)

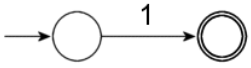
Exemplo:

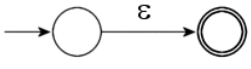
R = (1 + 01)* (0 + ε)

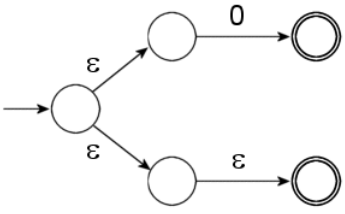
L(R) define a linguagem das palavras sem 0's consecutivos

Construção de um AFND-ε reconhecedor E:

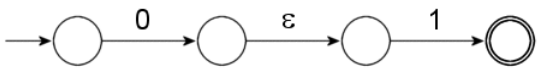
E[0] 

E[1] 

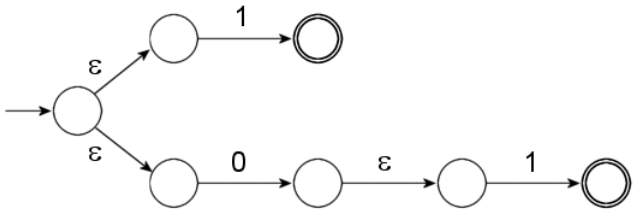
E[ε] 

E[(0 + ε)] 

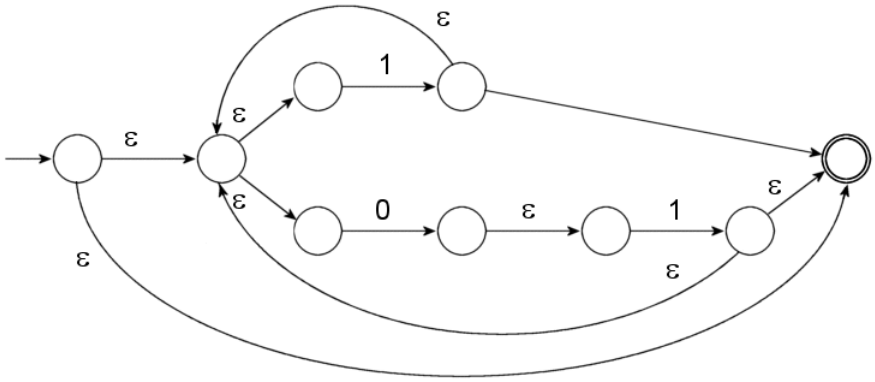
E[(01)]



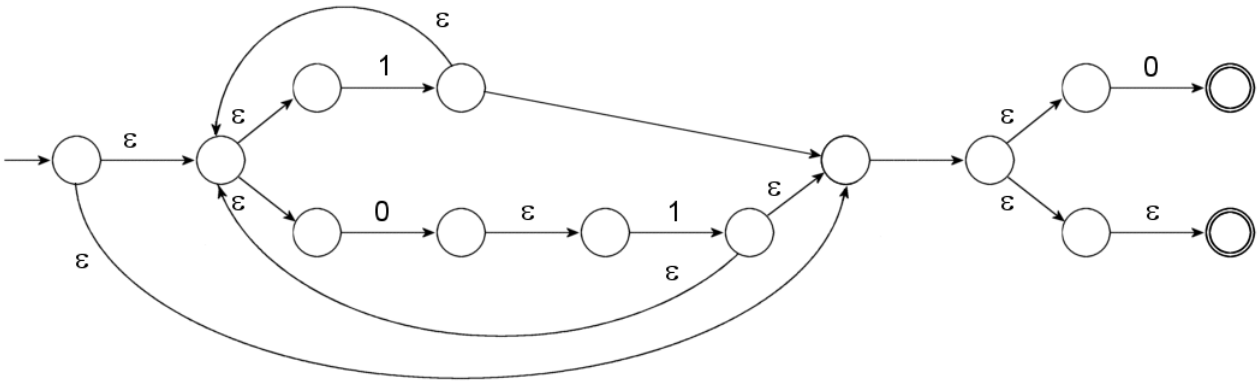
E[(1 + 01)]



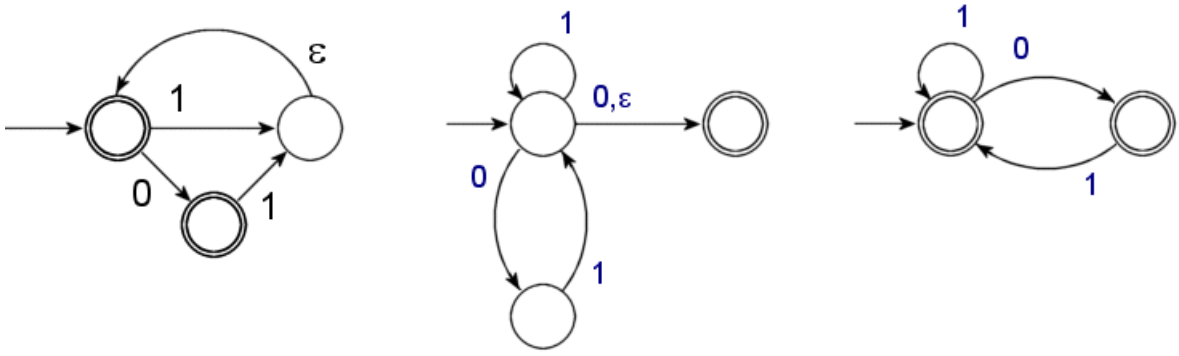
E[(1 + 01)*]



E[(1 + 01)* (0 + ε)]



Nota: Dois AFND-ε's e o AFD mínimo, reconhecedores de L(**(1 + 01)* (0 + ε)**)



AFD → ER

Teorema: Se L = L(A) é a linguagem de um AFD A,
então existe uma Expressão Regular R, tal que L = L(R).

Demonstração: Construção indutiva da Expressão Regular,
também aplicável a AFND's e AFND-ε's.

- Seja A um Autómato Finito com estados {1, 2, ..., n}
- Seja $R^{(k)}_{ij}$ uma Expressão Regular cuja linguagem é o conjunto das palavras que descrevem todos os caminhos do estado i para o estado j, em A, sem passar por nenhum estado intermédio maior que k.

por exemplo:

$R^{(0)}_{ij}$ são as palavras que representam as transições simples de i para j, sem estados intermédios.

$R^{(n)}_{ij}$ são todas as palavras que representam as transições de i para j.

Caso Base: k = 0. Não existem estados intermédios entre i e j.

1. Se **i ≠ j**, analisemos $\{a \in \Sigma \mid \delta(q_i, a) = q_j\}$:

   se não existirem transições, então $R^{(0)}_{ij} = \emptyset$

   se existir uma transição por a, então $R^{(0)}_{ij} = a$

   se existirem várias transições por $a_1, a_2, ..., a_m$,

   então $\mathbf{R^{(0)}_{ij} = a_1 + a_2 + ... + a_m}$

2. Se **i = j**, para além dos casos anteriores, pode também acontecer que não exista qualquer transição. Portanto,

   $\mathbf{R^{(0)}_{ij} = \varepsilon + a_1 + a_2 + ... + a_m}$

Passo Indutivo:

Hipótese: A proposição é verdadeira para $R^{(k-1)}_{ij}$.

Tese: $R^{(k)}_{ij} = R^{(k-1)}_{ij} + R^{(k-1)}_{ik} (R^{(k-1)}_{kk})^* R^{(k-1)}_{kj}$

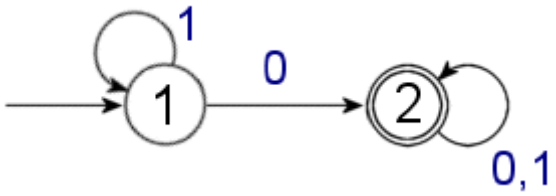
(os caminhos de i para j que passam também pelo estado k)

Demonstração:

- Se não existirem caminhos por k, então $R^{(k)}_{ij} = R^{(k-1)}_{ij}$
- $R^{(k-1)}_{ik}$ descreve os caminhos do estado i para o estado k
- $R^{(k-1)}_{kk}$ descreve os caminhos do estado k para si próprio
- $R^{(k-1)}_{kj}$ descreve os caminhos do estado k para o estado j

Por fim, a Expressão Regular pretendida é a união (+) de **todas** as fórmulas $\mathbf{R^{(n)}_{ij}}$ do Estado Inicial para todos os estados Finais do Autómato, sendo n o número total de Estados.

Exemplo: $A = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$



$L(A) = \{w \mid w$ tem pelo menos um $0\}$

Construção da Expressão Regular R, tal que L(R) = L(A):

- $R^{(0)}_{ij}$ (caminhos sem estados intermédios)

$R^{(0)}_{11} = \varepsilon + 1$

$R^{(0)}_{12} = 0$

$R^{(0)}_{21} = \emptyset$

$R^{(0)}_{22} = \varepsilon + 0 + 1$

- $R^{(1)}_{ij} = R^{(0)}_{ij} + R^{(0)}_{i1} (R^{(0)}_{11})^* R^{(0)}_{1j}$

(caminhos sem estados intermédios passando por q_2)

$R^{(1)}_{11} = R^{(0)}_{11} + R^{(0)}_{11} (R^{(0)}_{11})^* R^{(0)}_{11}$

$= (\varepsilon + 1) + (\varepsilon + 1) (\varepsilon + 1)^* (\varepsilon + 1)$

como $(\varepsilon + R)^* = R^*$

$(\varepsilon + R) R^* = R^* (\varepsilon + R) = R^*$

$= (\varepsilon + 1) + 1^*$

$= \varepsilon + 1 + 1^* = 1^*$

$$R^{(1)}_{12} = R^{(0)}_{12} + R^{(0)}_{11} (R^{(0)}_{11})^* R^{(0)}_{12}$$
$$= 0 + (\varepsilon + 1)(\varepsilon + 1)^* 0$$
$$= 0 + 1^* 0 \quad = 1^* 0$$

$$R^{(1)}_{21} = R^{(0)}_{21} + R^{(0)}_{21} (R^{(0)}_{11})^* R^{(0)}_{11}$$
$$R^{(1)}_{21} = \emptyset + \emptyset (\varepsilon + 1)^* (\varepsilon + 1)$$

como $\emptyset R = R \emptyset = \emptyset$

$\emptyset + R = R + \emptyset = R$

$$= \emptyset$$

$$R^{(1)}_{22} = (\varepsilon + 0 + 1) + \emptyset (\varepsilon + 1)^* 0$$
$$= \varepsilon + 0 + 1$$

- $R^{(2)}_{ij} = R^{(1)}_{ij} + R^{(1)}_{i2} (R^{(1)}_{22})^* R^{(1)}_{2j}$

$R^{(2)}_{11} = 1^* + 1^* 0 (\varepsilon + 0 + 1)^* \emptyset = 1^*$

$R^{(2)}_{12} = 1^* 0 + 1^* 0 (\varepsilon + 0 + 1)^* (\varepsilon + 0 + 1) = 1^* 0 (0+1)^*$

$R^{(2)}_{21} = \emptyset + (\varepsilon + 0 + 1)(\varepsilon + 0 + 1)^* \emptyset = \emptyset$

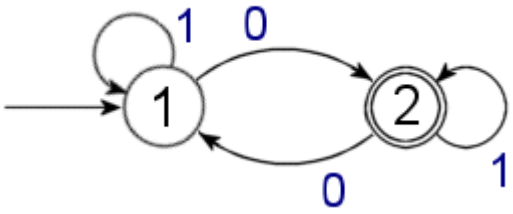
$R^{(2)}_{22} = (\varepsilon + 0 + 1) + (\varepsilon + 0 + 1)(\varepsilon + 0 + 1)^* (\varepsilon + 0 + 1) = (0+1)^*$

Finalmente, resta construir a união de todas as expressões $R^{(k)}_{ij}$, onde k=2, i=1 (estado inicial) e j=2 (único estado final). Neste caso:

$$R = R^{(2)}_{12} = 1^* 0 (0 + 1)^*$$

e, efectivamente, L(R) é a linguagem das palavras com, pelo menos, um 0.

Exemplo: $A = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$



$L(A) = \{w \mid w$ tem um número ímpar de 0's$\}$

Construção da Expressão Regular R, tal que L(R) = L(A):

(Tal como no exemplo anterior, só $R^{(2)}_{12}$ vai ser necessária)

- $R^{(0)}_{ij}$

$R^{(0)}_{11} = 1 + \varepsilon$ $\quad$ $R^{(0)}_{21} = 0$

$R^{(0)}_{12} = 0$ $\quad$ $R^{(0)}_{22} = 1 + \varepsilon$

- $R^{(1)}_{ij}$

$$R^{(1)}_{12} = R^{(0)}_{12} + R^{(0)}_{11} (R^{(0)}_{11})^* R^{(0)}_{12}$$
$$= 0 + (1 + \varepsilon) (1 + \varepsilon)^* 0$$
$$= 0 + (1 + \varepsilon)^+ 0$$

$$R^{(1)}_{22} = R^{(0)}_{22} + R^{(0)}_{21} (R^{(0)}_{11})^* R^{(0)}_{12}$$
$$= (1 + \varepsilon) + 0 (1 + \varepsilon)^* 0$$

- $R^{(2)}_{12} = R^{(1)}_{12} + R^{(1)}_{12} (R^{(1)}_{22})^* R^{(1)}_{22}$
$= (0 + (1 + \varepsilon)^+ 0) + (0 + (1 + \varepsilon)^+ 0) ((1 + \varepsilon) + 0 (1 + \varepsilon)^* 0)^+$
$= (0 + (1 + \varepsilon)^+ 0) (1 + \varepsilon + 0 (1 + \varepsilon)^* 0)^*$
$= (1 + \varepsilon)^* 0 (1 + \varepsilon + 0\ 1^* 0)^*$
$= 1^* 0 (1 + 0\ 1^* 0)^*$

$R = 1^* 0 (1 + 0\ 1^* 0)^*$

Comentários ao Algoritmo anterior:

- O método funciona sempre, mesmo para AFND's e AFND-ε's.
- Para um AF com n estados é necessário construir $O(n^3)$ expressões.
- Cada expressão pode ter comprimento $O(4^n)$.
- Existem demasiadas repetições de cálculos. Por exemplo, para cada expressão da forma $R^{(k+1)}_{ij}$ ocorre $(R^{(k)}_{kk})^*$ repetida n^2 vezes.

AFD → ER — Método de Conversão por Eliminação de Estados:

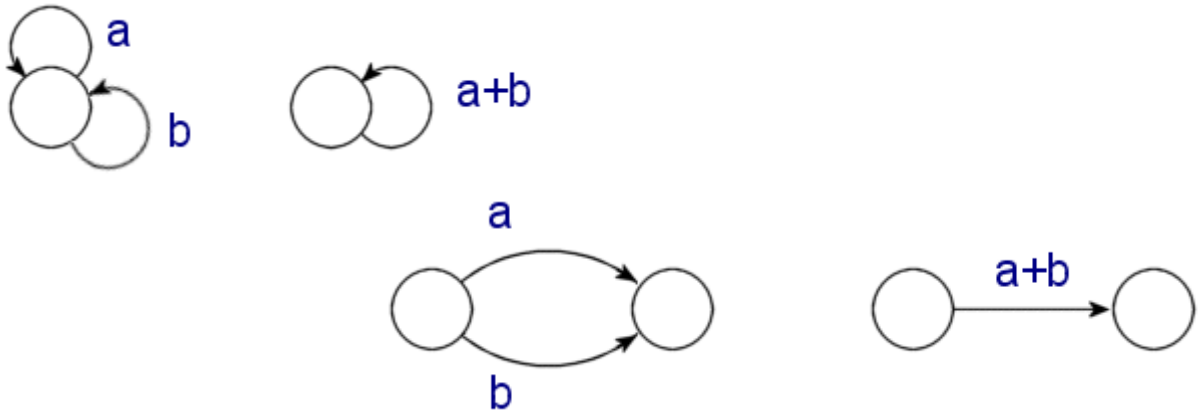
A ideia:

Cada símbolo a ∈ Σ pode ser encarado como uma ER, cuja Linguagem é {a}. No caso de uma transição por ε, a Linguagem é {ε}.

Cada caminho no grafo (AF) corresponde a uma ER formada pela concatenação dos sucessivos símbolos (etiquetas nos arcos).
A ER pretendida corresponde à união de todos os possíveis caminhos, desde o estado inicial, até todos os estados finais.

Se um estado for removido, os símbolos nos arcos incidentes terão de ser incluídos nos arcos entre os estados adjacentes.
Deste modo, as etiquetas nos arcos passam a ser ER's.

As transições múltiplas podem também ser removidas:



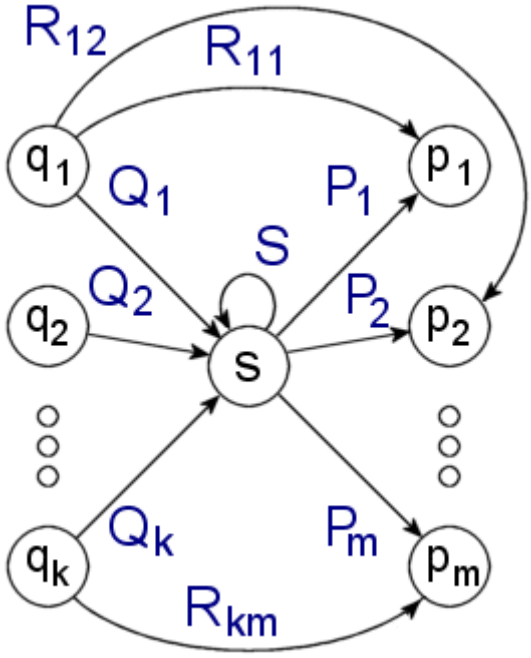
A estratégia:

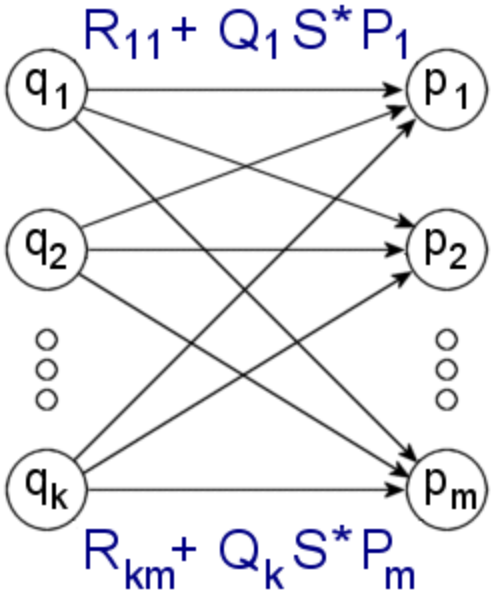
Pretendemos eliminar o estado s.

$\{q_1, q_2, ..., q_k\}$ são os estados predecessores a s e $\{p_1, p_2, ..., p_m\}$ os estados sucessores.

$\{Q_1, Q_2, ..., Q_k\}$ são as expressões nos arcos de $\{q_1, q_2, ..., q_k\}$ para s, $\{P_1, P_2, ..., P_m\}$ nos arcos de s para $\{p_1, p_2, ..., p_m\}$.

R_{ij} é cada expressão de q_i para p_j.





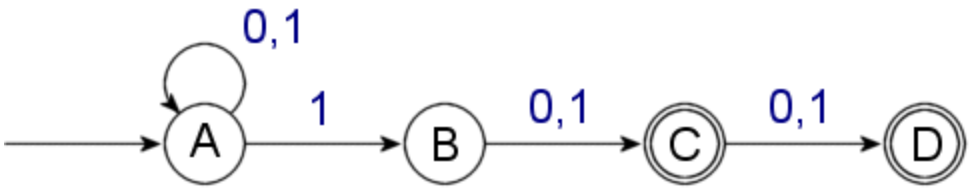
Após a remoção de S,
os arcos de $\{q_1, q_2, ..., q_k\}$ para $\{p_1, p_2, ..., p_m\}$, serão etiquetados com expressões da forma geral,

$$R_{ij} + Q_i S^* P_j$$

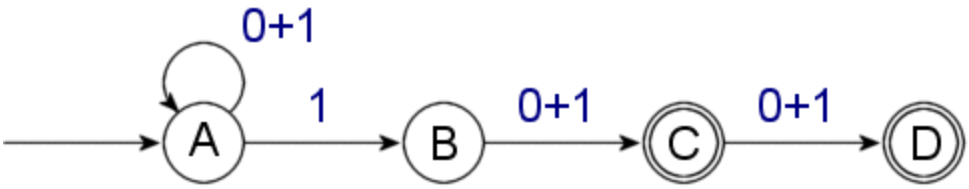
O Método:

Eliminar sucessivamente todos os estados, começando pelos estados internos.

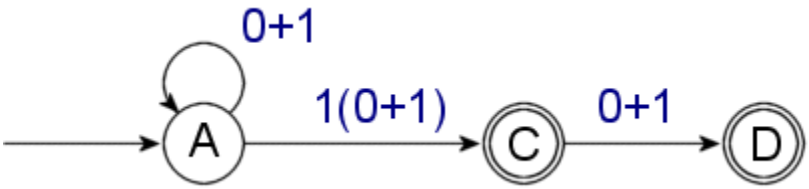
Exemplo: AFND reconhecedor das cadeias binárias com 1 na penúltima ou na ante-penúltima posição:



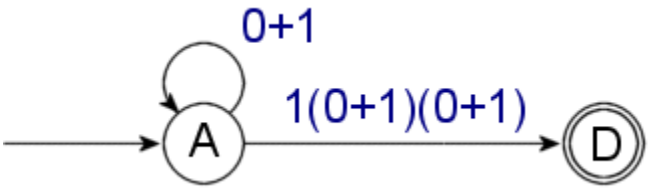
Conversão das transições múltiplas em ER's,

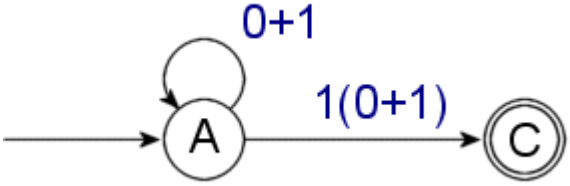


Eliminação do estado interno B,



Eliminação dos estados finais C e D,





$$R = (0 + 1)^* \, 1 \, (0 + 1) + (0 + 1)^* \, 1 \, (0 + 1) \, (0 + 1)$$

Aplicações das Expressões Regulares

- As ER's têm um (cada vez mais) vasto campo de aplicações práticas: Sistemas Operativos, construção de Compiladores, pesquisas em Bases de Dados, ...
- No sistema Unix foi, pela primeira vez, definido um conjunto de operadores que estabelecem uma extensão do conceito teórico de ER. Comandos como o grep (Global search for Regular Expression and Print) permitem a pesquisa de um padrão num conjunto de ficheiros.
- Sistemas como o Lex ou o Flex permitem a geração automática de Analisadores Léxicos. A partir da definição da ER pretendida, é automaticamente gerado (em C ou em Pascal) o AFD que constitui a fase da Análise Léxica de um Compilador.

Alguns Operadores da extensão Unix de ER's:

*	(zero ou mais ocorrências de ...)
+	(uma ou mais ocorrências de ...)
?	(zero ou uma ocorrência de ...)
.	(qualquer caracter)
|	(ou)
{ }	(número de ocorrências de ...)
[ ]	(classe de caracteres: uma ocorrência de ...)
-	(sub-domínio)

Exemplos:

aaa	a{3}
aa ou aaa ou aaaa	a{2,4}
bababa	(ba){3}
um inteiro sem sinal	$[0\text{-}9]^+$
um identificador	[A-Za-z] [a-zA-Z0-9]*
um número em Pascal	$[+\text{-}]?\ [0\text{-}9]^+\ (\.[0\text{-}9]^+)?\ ((e\|E)\ [+\text{-}]?\ [0\text{-}9]^+)?$

Regras Algébricas das Expressões Regulares

R = S significa que são Expressões **Equivalentes**, isto é, que L(R) = L(S).
(Não têm necessariamente a mesma forma algébrica)

- Associatividade e Comutatividade

  $(R\ (S\ T)) = ((R\ S)\ T) = R\ S\ T$

  $(R + (S + T)) = ((R + S) + T) = R + S + T$

  $R + S = S + R$ (mas $R\ S \neq S\ R$)

- Distributividade

  $R\ (S + T) = R\ S + R\ T$

  $(R + S)\ T = R\ T + S\ T$

- Elementos Neutro e Absorvente

  $\varepsilon\ R = R\ \varepsilon = R$

  $\emptyset + R = R + \emptyset = R$

  $\emptyset R = R\emptyset = \emptyset$

- Idempotência

  $R + R = R$

- Leis do Fecho

  $(R^*)^* = R^*$

  $\varepsilon^* = \varepsilon$

  $(\emptyset)^* = \varepsilon$

  $R^+ = R\ R^* = R^*\ R$

  $R^* = R^+ + \varepsilon$

… claro que têm de ser todas demonstradas.

Um exemplo de Demonstração de uma Regra Algébrica

Teorema: $R\,(S + T) = R\,S + RT$ (distributividade à esquerda)

Demonstração: Trata-se de provar que geram a mesma Linguagem, ou que

$L(R\,(S + T)) = L(R\,S + RT)$

ou, pela definição de ER, que

$L(R)\,L(S + T) = L(R\,S) \cup L(RT)$

$L(R)\,(L(S) \cup L(T))) = L(R\,S) \cup L(RT)$

$L(R)\,(L(S) \cup L(T))) = (L(R)\,L(\,S)) \cup (L(R)\,L(T))$

a) $L(R)\,(L(S) \cup L(T))) \subseteq (L(R)\,L(\,S)) \cup (L(R)\,L(T))$

Seja $w \in L(R)\,(L(S) \cup L(T)))$

então $w = xy$, com $x \in L(R) \wedge (y \in L(S) \vee y \in L(T))$

ou $(x \in L(R) \wedge y \in L(S)) \vee (x \in L(R) \wedge y \in L(T))$

e portanto $w \in (L(R)\,L(\,S)) \cup (L(R)\,L(T))$

b) $L(R)\,(L(S) \cup L(T))) \supseteq (L(R)\,L(\,S)) \cup (L(R)\,L(T))$

Seja $w \in (L(R)\,L(\,S)) \cup (L(R)\,L(T))$

então $w \in (L(R)\,L(\,S)) \vee w \in (L(R)\,L(T))$

Se $w = xy$ com $x \in L(R)$ então $y \in L(S) \vee y \in L(T)$

e portanto $w \in L(R)\,(L(S) \cup L(T)))$ □

Existem muitas outras regras algébricas das ER's, por exemplo:

Exercício: Provar que $(R + S)^* = (R^*\,S^*)^*$

Propriedades das Linguagens Regulares

- Todas as Linguagens finitas (com um número finito de palavras) são Regulares. Bastaria escrever uma ER (construir um AF reconhecedor) para cada uma das palavras da Linguagem.

- Nem todas as Linguagens são Regulares (reconhecíveis por um AF). Por exemplo:
  C = {$w \in \{0, 1\}^*$ | w tem um número igual de 0's e de 1's}
  não é uma Linguagem Regular, enquanto que:
  D = {$w \in \{0, 1\}^*$ | w tem um número igual de sequências 01 e 10}
  é uma Linguagem Regular.

- Como verificar se uma dada Linguagem é ou não Regular?

Um exemplo:

$L = \{a^n b^n \mid n \geq 0\}$ será uma Linguagem Regular?

Suponhamos que é Regular. Então existe um AFD,
$A = \{Q, \{a, b\}, \delta, q_0, F\}$ capaz de reconhecê-la.
Calculemos $\delta^\wedge(q_0, a^i)$ para i = 1, 2, 3, ...
Como existe um número infinito de i's mas um número finito de estados, existirá pelo menos um estado q, tal que,

$$\delta^\wedge(q_0, a^m) = \delta^\wedge(q_0, a^n) = q \quad \text{com } m \neq n .$$

Mas para o Autómato reconhecer $a^n b^n$, deverá:

$$\delta^\wedge(q, b^n) = q_f \in F$$

Donde se conclui que:

$$\delta^\wedge(q_0, a^m b^n) = \delta^\wedge(\delta^\wedge(q_0, a^m), b^n) = \delta^\wedge(q, b^n) = q_f$$

Ou seja, o AFD aceitaria também palavras da forma $a^m b^n$ com $m \neq n$.
Não existe portanto um AFD capaz de reconhecer apenas palavras da forma $a^n b^n$, ou seja, a Linguagem **não** é Regular.

O Lema da Bombagem

Teorema: (*Lema da Bombagem para Linguagens Regulares*)

Seja L uma Linguagem Regular. Então existe um inteiro positivo $\underline{n}$ tal que, para toda a palavra $w \in L$ com $|w| \geq n$, existem $x, y, z \in \Sigma^*$ tais que:

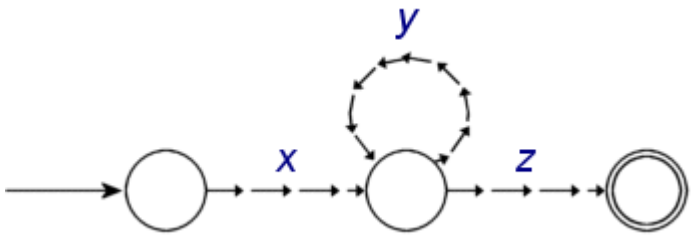
- $w = xyz$
- $y \neq \varepsilon$
- $|xy| \leq n$
- $\forall\, k \geq 0,\ xy^k z \in L$

de modo mais formal:

$$\forall\ L \text{ Linguagem Regular}, \exists\ n \in \mathbb{N} : \forall\ w \in L \text{ com } |w| \geq n, \exists\ x, y, z \in \Sigma^* : w = xyz \text{ com } y \neq \varepsilon \text{ e } |xy| \leq n : \forall\ k \in \mathbb{N}_0,\ xy^k z \in L$$

ou seja:

Toda a palavra (suficientemente longa) de L pode ser dividida em três partes de modo a que, com um número qualquer de repetições da parte central, continue a ser uma palavra de L.
Por isso se diz que a parte central é **bombeada**.



Demonstração:

Se L é uma Linguagem Regular, então existe um AFD A tal que L(A) = L.

Seja n o número de estados de A = {{q_0, q_1, ..., q_{n-1}}, Σ, δ, q_0, F}} e tomemos esse n para o n do próprio Lema.

Consideremos uma palavra $w \in L$, $w = a_1a_2 \ldots a_m$, com $|w| = m \geq n$.
O reconhecimento de w por A corresponde a um caminho,

$$q_o \rightarrow \delta(q_o, a_1) \rightarrow \delta\hat{}(q_o, a_1a_2) \rightarrow \ldots \rightarrow \delta\hat{}(q_o, a_1a_2 \ldots a_m) \in F$$

Como existem apenas n estados diferentes e este caminho tem comprimento maior que n, então ocorrem repetições de estados. Isto é, existem i e j, com $0 \leq i < j \leq n$ tais que $\delta\hat{}(q_o, a_1a_2 \ldots a_i) = \delta\hat{}(q_o, a_1a_2 \ldots a_j)$.

Para qualquer $k \geq 0$, mostremos que,

$\delta\hat{}(q_o, a_1a_2 \ldots a_i (a_{i+1} \ldots a_j)^k) = \delta\hat{}(q_o, a_1a_2 \ldots a_j)$

Caso base: (k = 0)
$\delta\hat{}(q_o, a_1a_2 \ldots a_i) = \delta\hat{}(q_o, a_1a_2 \ldots a_j)$

Prova do passo indutivo:

$$\begin{aligned}\delta\hat{}(q_o, a_1a_2 \ldots a_i (a_{i+1} \ldots a_j)^{k+1}) &= \delta\hat{}(\, \delta\hat{}(q_o, a_1a_2 \ldots a_i (a_{i+1} \ldots a_j)^k), a_{i+1} \ldots a_j) \\ &= \delta\hat{}(\, \delta\hat{}(q_o, a_1a_2 \ldots a_j), a_{i+1} \ldots a_j) \\ &= \delta\hat{}(\, \delta\hat{}(q_o, a_1a_2 \ldots a_i), a_{i+1} \ldots a_j) \\ &= \delta\hat{}(\, q_o, a_1a_2 \ldots a_i\, a_{i+1} \ldots a_j)\end{aligned}$$

Sejam $x = a_1a_2 \ldots a_i$, $y = a_{i+1} \ldots a_j$ e $z = a_{j+1} \ldots a_m$, tais que $w = xzy$.

Como $i \neq j$, $0 \leq i < j \leq n$, então $y \neq \varepsilon$ e $|xy| \leq n$

Para todo o $k \geq 0$, $\delta\hat{}(q_o, xy^k) = \delta\hat{}(q_o, xy)$

$\delta\hat{}(q_o, xy^kz) = \delta\hat{}(q_o, xyz)$

e portanto $xy^kz \in L$ □

L é uma Linguagem Regular $\Longrightarrow$ L satisfaz o Lema da Bombagem

L não satisfaz o Lema da Bombagem $\Longrightarrow$ L não é uma Linguagem Regular

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

Suponhamos que L é uma Linguagem Regular.
Então existe um $n \in \mathbb{N}$ para o qual todas as palavras, suficientemente grandes, de L satisfazem as condições do Lema da Bombagem.

Tomemos $w = a^n b^n$.

Se w for suficientemente longa, existem x, y e z tais que $w = xyz$, com $y \neq \varepsilon$, $|xy| \leq n$ e $xy^k z \in L$ para todo o $k \in \mathbb{N}_0$.

Para que se verifique $|xy| \leq n$ com $|y| \geq 1$, tanto x como y terão de ser sequências de a's, isto é,

$$x = a^r,\ y = a^s \text{ e } z = a^t b^n \text{ com } r + s + t = n,\ r \geq 0,\ s \geq 1 \text{ e } t \geq 0.$$

Ora, pelo Lema da Bombagem com k = 0,

$$xy^0z = a^r \varepsilon a^t b^n = a^{r+t} b^n$$

mas como $r + t < n$, então a palavra $xy^0z \notin L$, o que contradiz o Lema da Bombagem.

Portanto $L = \{a^m b^m \mid m \geq 0\}$ **não é** uma Linguagem Regular.

Exemplo: $L = \{ 0^i \mid \underline{i} \text{ é um Número Primo}\}$

Suponhamos que L é uma Linguagem Regular. Então existe um $n \in \mathbb{N}$ para o qual todas as palavras $w \in L$, $|w| \geq n$ (suficientemente longas) satisfazem as condições do Lema da Bombagem.

Consideremos $w = 0^m \in L$, onde $m \geq n + 2$ com $\underline{m}$ primo.

Se w for suficientemente longa, existem x, y e z tais que $w = xyz$, com $y \neq \varepsilon$, $|xy| \leq n$ de modo que $xy^kz \in L$, para todo o $k \geq 0$.

Seja $w = xyz$. Para que se verifique $|xy| \leq n$ com $y \neq \varepsilon$,

$$x = 0^r,\ y = 0^s \text{ e } z = 0^t \text{ com } r \geq 0,\ s \geq 1 \text{ e } t \geq 0.$$

Ora pelo Lema da Bombagem, tomando $k = r + t$,

$$xy^{r+t}z = 0^r (0^s)^{r+t} 0^t = 0^{r + s(r+t) + t}$$

mas como $r + s(r + t) + t = (s + 1)(r + t)$, então a palavra $xy^{r+t}z \notin L$, o que contradiz o Lema da Bombagem.
Portanto L não é uma Linguagem Regular.

Exercícios: Mostrar que **não** são Linguagens Regulares:

$L = \{ 0^i \mid \underline{i} \text{ é um Quadrado Perfeito}\}$

$L = \{ 0^{n!} \mid n \geq 0\}$

$L = \{ a^m b^n \mid m, n \geq 0 \text{ e } m \neq n\}$

$L = \{ a^m b^n c^{m+n} \mid m, n \geq 0\}$

Estas demonstrações tornam-se mais simples, por aplicação directa de propriedades relativas ao Fecho das Linguagens Regulares, para certos Operadores.

Operações sobre Linguagens Regulares

Observação: Quando uma linguagem L é definida sobre um alfabeto Σ, está também, definida sobre qualquer alfabeto $\supseteq \Sigma$.
Assim duas dadas linguagens, L_1 definida sobre Σ_1 e L_2 definida sobre Σ_2, estão também definidas sobre $\Sigma_1 \cup \Sigma_2$.

Propriedades do Fecho:

- **A Classe das Linguagens Regulares é fechada para os Operadores de uma Álgebra de Boole.**

  Dadas $L_1 = L(R_1)$ e $L_2 = L(R_2)$,

  União: $L_1 \cup L_2$ é uma Linguagem Regular,
  pois $L_1 \cup L_2 = L(R_1 + R_2)$ que é Regular, por definição.

  Complementar: $\overline{L} = \Sigma^* \setminus L$ é uma Linguagem Regular.
  Seja $A = (Q, \Sigma, \delta, q_0, F)$ o AFD reconhecedor de L. Então o AFD $B = (Q, \Sigma, \delta, q_0, Q \setminus F)$ reconhecerá $\overline{L}$.

  Intersecção: $L_1 \cap L_2$ é uma Linguagem Regular,
  pois $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

- **Também é fechada para muitos outros Operadores.**

  Diferença: $L_1 \setminus L_2$ é uma Linguagem Regular,
  pois $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$.

  Concatenação: L_1L_2 é uma Linguagem Regular.

  Fecho de Kleene: L^* é uma Linguagem Regular.

  Inversão: L^{-1} (ou L^R) é uma Linguagem Regular.

O Teorema da Substituição:

Seja L uma **Linguagem Regular** sobre um Alfabeto Σ.
(A cada símbolo $a \in \Sigma$ associamos uma Linguagem Regular L_a)

Seja S a substituição: $S(a) = L_a$, $\forall\ a \in \Sigma$.

Construímos a extensão de S para palavras $w = a_1\ a_2\ ...\ a_n$ como sendo a concatenação das Linguagens associadas:

$S(a_1\ a_2\ ...\ a_n) = L_{a_1}\ L_{a_2}\ ...\ L_{a_n}$

Construímos a extensão de S para qualquer linguagem M:

$S(M) = \cup_{w \in M}\ S(w)$

Teorema: $S(L)$ é uma Linguagem Regular.

Demonstração: Seja R uma Expressão Regular geradora de L.
$\forall\ a \in \Sigma$, seja R_a uma ER geradora de $S(a) = L_a$,
A partir de R, construímos E, substituindo cada símbolo a pela correspondente expressão R_a.
Resta provar que **$L(E) = S(L)$** .

Casos Base: Se R = a, então $E = R_a = a$ e $L(E) = \{a\}$
$S(L) = S(\{a\}) = L(R_a) = \{a\}$
(Verificar os casos $R = \emptyset$ e $R = \varepsilon$)

Passos Indutivos: Há três casos a considerar,
$R = R_1 + R_2$, $R = R_1\ R_2$ e $R = R_1^*$

As provas são idênticas e simples. Vejamos apenas $R = R_1 R_2$

Seja $L = L_1L_2$, onde $L_1 = L(R_1)$ e $L_2 = L(R_2)$,

sejam E_1 a expressão de R_1 com cada símbolo $\underline{a}$ substituído por R_a

e E_2 a expressão de R_2 com cada símbolo $\underline{a}$ substituído por R_a

Por Hipótese de Indução, $L(E_1) = S(L_1)$ e $L(E_2) = S(L_2)$.

Então $L(E) = L(E_1)\, L(E_2) = S(L_1)\, S(L_2) = S(L)$. □

A utilização do Teorema da Substituição pode simplificar bastante a verificação de diversas propriedades das Linguagens Regulares.

Por exemplo, se L_1 e L_2 forem Linguagens Regulares:

- L_1L_2 é uma Linguagem Regular:
  Basta considerar a LR {ab} e substituir $\underline{a}$ por L_1 e $\underline{b}$ por L_2.

- $L_1 \cup L_2$ é uma Linguagem Regular:
  Substituir em {a, b}.

- L_1^* é uma Linguagem Regular:
  Substituir em {a*}.

Um importante caso particular consiste na substituição de cada símbolo por uma palavra.

Definição: Sejam Σ e Γ dois alfabetos. A função

$$h : \Sigma \longrightarrow \Gamma^*$$

chama-se **homomorfismo** de palavras.

Homomorfismos:

Exemplos:

- Seja L = {0* 1*} e $h : \{0, 1\} \longrightarrow \{a\}^*$
  definido por h(0) = aa e h(1) = ε.

  Então h(L) = {(aa)*} = {palavras com um número par de a's}.

- Seja L = $\{0^n 1^n \mid n \geq 0\}$ e $h : \{0, 1\} \longrightarrow \{a, b\}^*$
  definido por h(0) = ab e h(1) = ba.

  Então h(L) = $\{(ab)^n (ba)^n \mid n \geq 0\}$.

Notas:

- O resultado da aplicação de um homomorfismo a uma palavra $w = a_1a_2...a_n \in \Sigma^*$ é uma concatenação de palavras de Γ^*, $h(w) = h(a_1)h(a_2)...h(a_n) \in \Gamma^*$.

- A aplicação de um homomorfismo a uma ER produz uma ER (Provar!). Portanto, a classe das Expressões Regulares é **fechada** para o Homomorfismo de palavras.

- Se L for uma Linguagem $\subseteq \Sigma^*$, então $h(L) = \{h(w) \in \Gamma^* \mid w \in L\}$.

- A classe das Linguagens Regulares é **fechada** para o Homomorfismo de palavras. Pelo Teorema da Substituição, se L for uma Linguagem Regular então a Linguagem h(L) também é Regular.

- Mas se uma Linguagem L não for Regular, h(L) pode ser Regular.

  Por exemplo, para a Linguagem L = $\{0^n 1^n \mid n \geq 0\}$ com h(0) = a e h(1) = ε, obtemos h(L) = {a*} que é Regular.

Homomorfismo Inverso:

Dado um Homomorfismo $h : \Sigma^* \longrightarrow \Gamma^*$ e uma Linguagem $L \subseteq \Gamma^*$, define-se,

$$h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$$

o conjunto das palavras cuja imagem homomórfica pertence a L.

Exemplo:

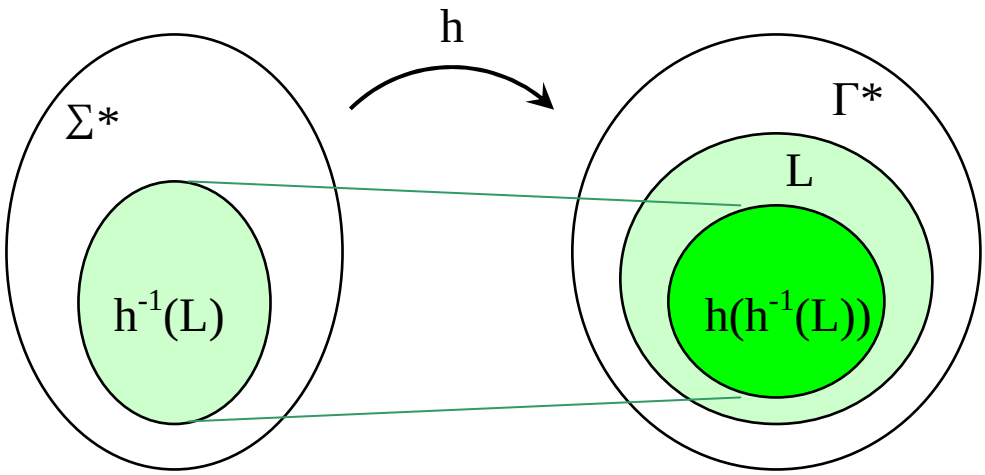
Seja $L = \{(00 + 1)^*\}$ com $h(a) = 01$ e $h(b) = 10$.

Como exemplos, $h^{-1}(1001) = \{ba\}$ e $h^{-1}(010110) = \{aab\}$.

Prova-se que $h^{-1}(L) = \{(ba)^*\}$.

E então $h(h^{-1}(L)) = \{(1001)^*\} \subset L$.

Nota: $\mathbf{h(h^{-1}(L)) \subseteq L}$ onde a igualdade não é necessariamente verdadeira.



... do mesmo modo que $\mathbf{L \subseteq h^{-1}(h(L))}$.

- Prova-se que, se L for uma Linguagem Regular então $h^{-1}(L)$ também é Regular. Assim, a classe das Linguagens Regulares também é **fechada** para o Homomorfismo Inverso.

Inversão (no sentido de Reversão):

- A **Palavra Inversa** (Reversa) de $w = a_1a_2...a_n$ é a palavra $w^R = a_na_{n-1}...a_1$.

- Para uma dada Linguagem L, a **Linguagem Inversa** L^R é o conjunto das Inversas das palavras de L.

- A Linguagem Inversa de uma Linguagem Regular é também Regular. Ou seja, a classe das Linguagens Regulares também é **fechada** para a Inversão. Uma demonstração simples, consiste em considerar a Expressão Regular E geradora de L e provar que $L(E^R) = (L(E))^R$, por Indução.

- Esta propriedade pode também ser demonstrada, considerando um Autómato Finito A (AFD, AFND ou AFND-ε) tal que L = L(A). Para construir um Autómato Finito A^R, capaz de reconhecer as inversas da palavras de L basta:

    1. Inverter o sentido de todas as Transições em A;
    2. Fazer do Estado Inicial de A o (único) Estado Final de A^R;
    3. Criar um novo Estado (Inicial de A^R) com Transições-ε para todos os Estados Finais de A.

Conclusão:

- **A Classe das Linguagens Regulares é fechada para uma grande variedade de Operações.**

Equivalência de Expressões Regulares e Minimização de Autómatos Finitos Deterministas

- ✓ Cada AFD define uma única Linguagem Regular, mas o inverso não é verdadeiro;
- ✓ Para cada Linguagem Regular, existe um **AFD mínimo** que é único a menos de um isomorfismo;
- ✓ O Algoritmo para a construção do AFD mínimo pode ser utilizado para testar a **Equivalência de Expressões Regulares.**

Definição: Seja $A = (Q, \Sigma, \delta, q_0, F)$ um Autómato Finito Determinista.

Dois **estados** $p, q \in Q$ são **indistinguíveis** ($p \equiv q$) quando,

$$\forall w \in \Sigma^*,\ \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$$

$$\{\ \forall w \in \Sigma^*,\ \hat{\delta}(p, w) \in F \Rightarrow \hat{\delta}(q, w) \in F \quad \hat{\delta}(p, w) \notin F \Rightarrow \hat{\delta}(q, w) \notin F\ \}$$

Dois estados $p, q \in Q$ são **distinguíveis** ($p \not\equiv q$) quando,

$$\exists w \in \Sigma^* :\ \hat{\delta}(p, w) \in F \text{ e } \hat{\delta}(q, w) \notin F \text{ (ou vice-versa)}$$

$$\{\ \exists w \in \Sigma^*: (\hat{\delta}(p, w) \in F \wedge \hat{\delta}(q, w) \notin F) \vee (\hat{\delta}(p, w) \notin F \wedge \hat{\delta}(q, w) \in F)\ \}$$

Teorema: A relação (**p ≡ q**) definida entre estados de um AFD por,

$\forall\, w \in \Sigma^*,\ \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$

é uma **Relação de Equivalência**.

Demonstração:

Reflexiva: **p ≡ p** , $\forall\, p \in Q$

$\forall\, w \in \Sigma^*$, $\hat{\delta}(p, w)$ é um único estado.

Simétrica: **p ≡ q** ⇒ **q ≡ p** , $\forall\, p, q \in Q$

$\forall\, w \in \Sigma^*$, $\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$

Transitiva: **p ≡ q** ∧ **q ≡ r** ⇒ **p ≡ r** , $\forall\, p, q, r \in Q$

Suponhamos que **p ≡ q** e **q ≡ r**, mas que **p ≢ r**.
Nesse caso, com,

$\forall\, w \in \Sigma^*,\ \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$
$\forall\, w \in \Sigma^*,\ \hat{\delta}(q, w) \in F \Leftrightarrow \hat{\delta}(r, w) \in F$

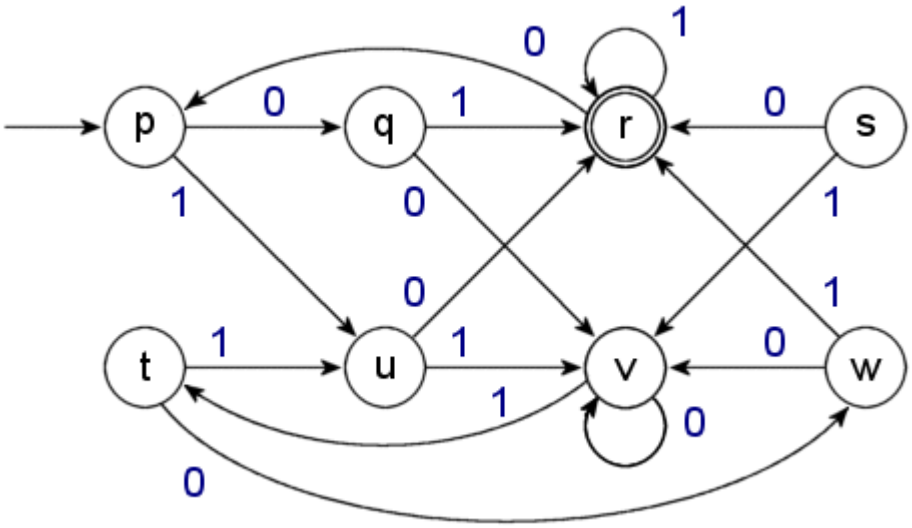
dois casos poderiam acontecer:

se $\exists\, w \in \Sigma^* : \hat{\delta}(p, w) \in F \wedge \hat{\delta}(r, w) \notin F$
então, como **p ≡ q** ⇒ $\hat{\delta}(q, w) \in F$
e como **q ≡ r** ⇒ $\hat{\delta}(r, w) \in F$

se $\exists\, w \in \Sigma^* : \hat{\delta}(p, w) \notin F \wedge \hat{\delta}(r, w) \in F$
então, como **p ≡ q** ⇒ $\hat{\delta}(q, w) \notin F$
e como **q ≡ r** ⇒ $\hat{\delta}(r, w) \notin F$

Portanto **p ≡ r**.

Exemplo: ({p, q, r, s, t, u, v, w}, {0, 1}, δ, p, {r})



Analisemos a equivalência de alguns pares de estados:

(r, v) ? $\quad \hat{\delta}(r, \varepsilon) \in F$

$\hat{\delta}(v, \varepsilon) \notin F \quad \Rightarrow$ **r ≢ v**

{ Todo o estado final é distinguível de todo o estado não final. }

(s, w) ? $\quad \hat{\delta}(s, 0) = r \in F$

$\hat{\delta}(w, 0) = t \notin F \quad \Rightarrow$ **s ≢ w**

(p, t) ? $\quad \hat{\delta}(p, \varepsilon) = p \notin F \ \wedge \ \hat{\delta}(t, \varepsilon) = t \notin F$

$\hat{\delta}(p, 1) = \hat{\delta}(t, 1) = u$

$\Rightarrow \ \hat{\delta}(p, 1x) = \hat{\delta}(t, 1x) = \hat{\delta}(u, x)$, $\forall x \in \Sigma^*$

$\hat{\delta}(p, 00) = \hat{\delta}(t, 00) = v$

$\Rightarrow \ \hat{\delta}(p, 00x) = \hat{\delta}(t, 00x) = \hat{\delta}(v, x)$, $\forall x \in \Sigma^*$

$\hat{\delta}(p, 01) = \hat{\delta}(t, 01) = r$

$\Rightarrow \ \hat{\delta}(p, 01x) = \hat{\delta}(t, 01x) = \hat{\delta}(r, x)$, $\forall x \in \Sigma^*$

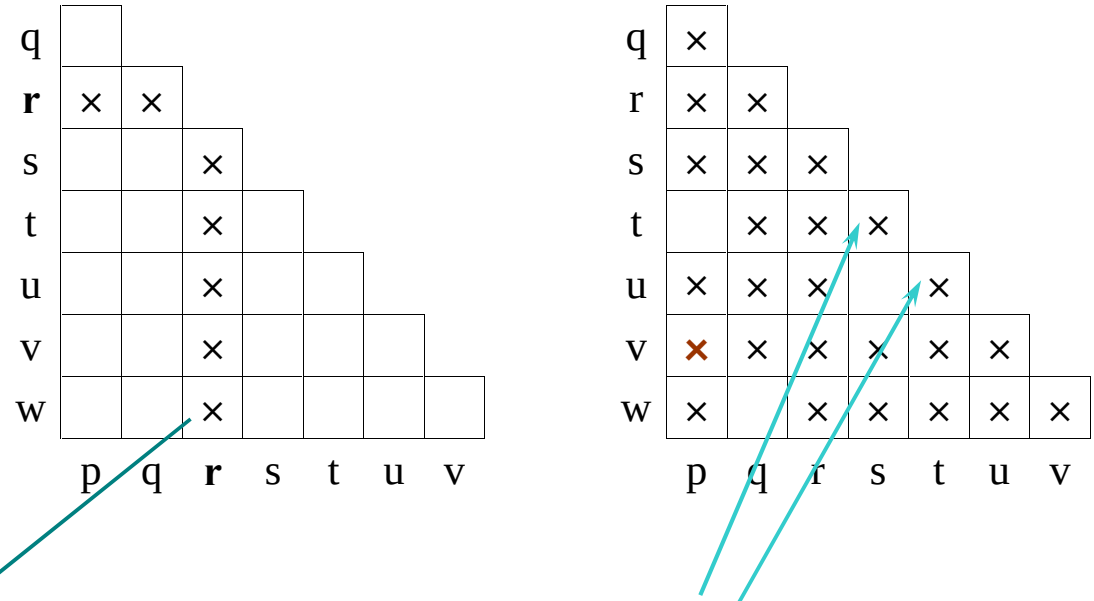
Portanto **p ≡ t**

Algoritmo para a identificação de Pares Distinguíveis:

Caso Base: se $p \in F$ e $q \notin F$, então $p \not\equiv q$.

Passo Indutivo: Se $\exists\, a \in \Sigma$: $\delta(p, a) \not\equiv \delta(p, a)$, então $p \not\equiv q$.

Para o exemplo anterior, começando por marcar os pares (final, não final),



w ≢ r $\quad \delta(t, 0) = w\,,\ \delta(s, 0) = r \quad \Rightarrow$ **t ≢ s**
$\quad \delta(t, 0) = w\,,\ \delta(u, 0) = r \quad \Rightarrow$ **t ≢ u**

v ≢ r $\quad \delta(w, 0) = v\,,\ \delta(s, 0) = r \quad \Rightarrow$ **w ≢ s**
$\quad \delta(w, 0) = v\,,\ \delta(u, 0) = r \quad \Rightarrow$ **w ≢ u**

… quase todos os outros pares podem ser marcados na tabela, por análise de transições simples.

Falta apenas o par (v, p) que também é distinguível a partir de (t, u):

t ≢ u $\quad \delta(v, 1) = t\,,\ \delta(p, 1) = u \quad \Rightarrow$ **v ≢ p**

Teorema: Se p, q ∈ Q não são distinguíveis pelo algoritmo anterior, então p ≡ q.

Demonstração (como exercício)

Teste de Equivalência de Expressões Regulares

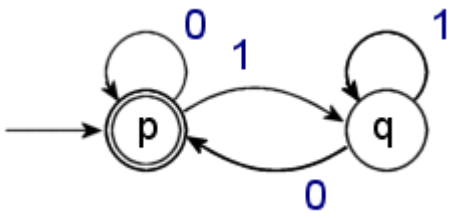
Sejam R e S duas Expressões Regulares.
Para testar a sua equivalência, isto é, L(R) = L(S):

1. Converter R e S em Autómatos Finitos;
2. Juntar os dois Autómatos na mesma tabela;
3. Usando o Algoritmo anterior, as Expressões são equivalentes, se e só se os dois estados iniciais forem equivalentes.

Exemplo: Consideremos a Linguagem,

$$L = \{ w \in \{0, 1\}^* \mid w \text{ é vazia ou acaba em } 0\}$$

Podemos construir o AFD,

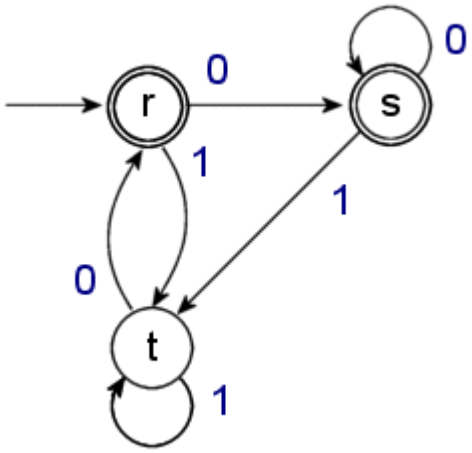


que, pelo Método de Eliminação de Estados,
corresponde à Expressão Regular:

$$(0 + 11^*0)^* = (1^*0)^*$$

Por outro lado, a Expressão Regular $\varepsilon + (0+1)^*0$
gera obviamente a Linguagem pretendida.

Construindo o AFND-ε correspondente
e efectuando a conversão para AFD, obtemos,



Serão Equivalentes?

Juntemos todos os estados dos dois autómatos numa mesma tabela.

q	×			
r		×		
s		×		
t	×		×	×
	p	q	**r**	**s**

Neste caso, bastou marcar os pares (final, não final).

Como os dois estados iniciais são Equivalentes, **p ≡ r**,
os dois AFD's são também Equivalentes.

Então, as Expressões Regulares **(1*0)*** e **ε + (0+1)*0** geram a mesma Linguagem, sendo portanto **Equivalentes**.

$$(1^*0)^* = \varepsilon + (0+1)^*0$$

Minimização de Autómatos Finitos Deterministas

- A Relação de Equivalência definida, subdivide o conjunto de Estados em **Classes de Equivalência**, que são os Estados do **Autómato Mínimo**.

Voltando ao outro exemplo (pág. 33),

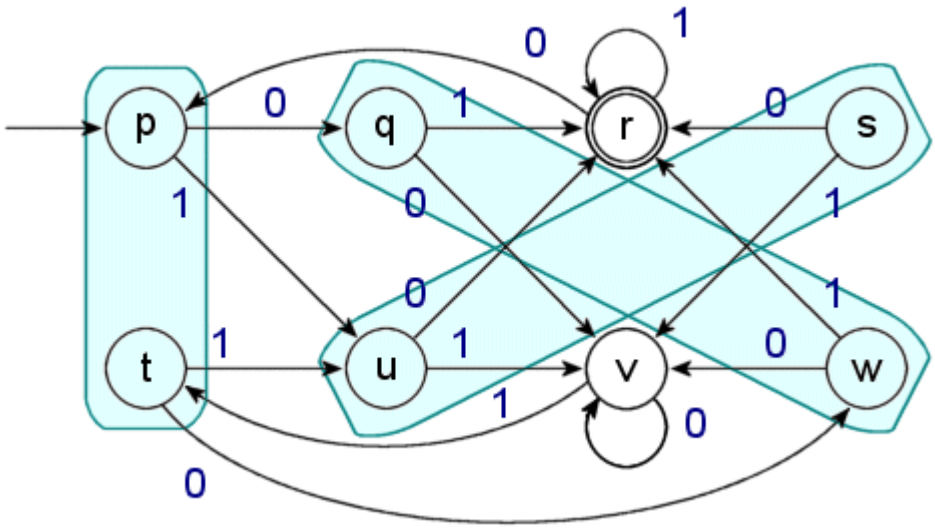
q	×						
r	×	×					
s	×	×	×				
t		×	×	×			
u	×	×	×		×		
v	×	×	×	×	×	×	
w	×		×	×	×	×	×
	p	q	r	s	t	u	v

Restam três pares de estados não distinguíveis (**equivalentes**):

p ≡ t, q ≡ w, s ≡ u

O conjunto de Estados foi dividido nas Classes de Equivalência:

Q = { {p, t}, {q, w}, {s, u}, {r}, {v} }



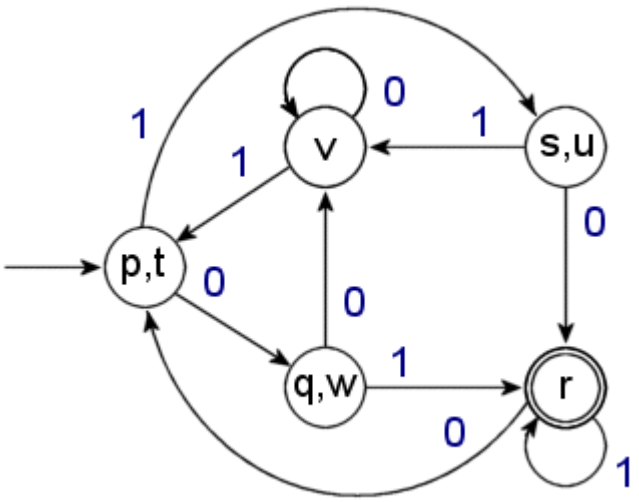
Essas Classes de Equivalência são os estados do **Autómato Mínimo**:

Calculando as transições,

		0	1
⟶	p	q	u
	q	v	r
*	r	p	r
	s	r	v
	t	w	u
	u	r	v
	v	v	t
	w	v	r

		0	1
⟶	{p,t}	{q,w}	{s,u}
	{q,w}	{v}	{r}
	{s,u}	{r}	{v}
	{v}	{v}	{p,t}
*	{r}	{p,t}	{r}

Obtemos o Autómato Mínimo:



Teorema: O Autómato assim obtido é mesmo Mínimo, a menos de um Isomorfismo.

Demonstrações (como exercício)

Problemas de Decisão sobre Linguagens Regulares:

- **Dada uma Linguagem Regular, representada por um Autómato Finito ou por uma Expressão Regular, que questões se podem pôr acerca de L, e como respondê-las?**

  Como existe uma equivalência entre as duas representações possíveis, podemos sempre escolher a que for mais conveniente.

- ? A Linguagem L é uma Linguagem **Regular**?
- ? A Linguagem L é uma Linguagem **Não-Regular**?
- ? Uma dada palavra *w* **pertence** a uma dada Linguagem Regular L?
- ? A Linguagem Regular L é **vazia**?
- ? A Linguagem Regular L é **finita**?
- ? Serão duas dadas Linguagens Regulares L1 e L2 **iguais**?
- ? Existirão **Relações de Equivalência** entre Linguagens Regulares?
- ? Existirão **Isomorfismos** entre Linguagens Regulares?
- ? ...