

Linguagens Formais e Autômatos





# Expressões Regulares

Eduardo Furlan Miranda

Baseado em: GARCIA, A. de V.; HAEUSLER, E. H.  
Linguagens Formais e Autômatos. Londrina: EDA, 2017.

# Expressão Regular (ER)

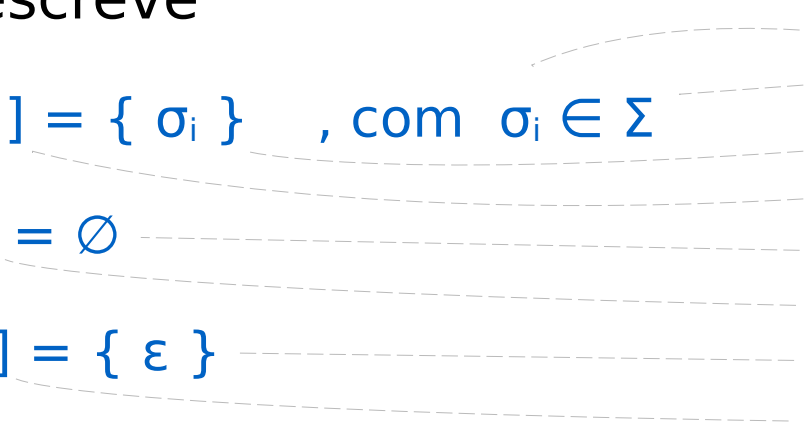
2/22

- Forma compacta de descrever **Linguagens Regulares (LR)**
  - **LR** podem ser reconhecidas por **Autômatos Finitos (AF)**
- Usa **concatenação**, **união**, e o **fecho de Kleene**
  - concatenação: “**justaposição**”, ex.:  $e_1e_2$   expressão regular
  - união: “**+**”, ex.:  $e_1+e_2$  (equivalente ao “**ou**” lógico)
  - fecho de Kleene: “**\***”, ex.:  $e^*$  (“zero ou mais repetições”)
- Outros ex. de ER:
  - linguagem vazia ( $\emptyset$ )  conjunto vazio
  - linguagem com cadeia vazia ( $\epsilon$ )  epsilon
  - somente um símbolo ( $\sigma_i$ )  sigma

- Uma ER sobre o alfabeto  $\Sigma = \{ \sigma_1, \dots, \sigma_i, \dots, \sigma_n \}$  é de uma das seguintes formas:
  - $\emptyset$  (linguagem vazia) ,  $\varepsilon$  (cadeia vazia) ,  $\sigma_i$  (símbolo)  
onde  $\sigma_i \in \Sigma$
  - Se  $e$  é uma ER, então  $e^*$  também é "e estrela" ou  
"fecho de Kleene de e"
  - Se  $e_1$  e  $e_2$  são ERs, então  $e_1e_2$  e  $e_1+e_2$  também são ERs
  - Se  $e$  é uma ER, então  $(e)$  também é
    - $( )$  agrupa partes de uma ER; define precedência
  - Nada mais é uma ER

\* : todas as combinações incluindo  $\varepsilon$

- $a$  representa a linguagem  $\{a\}$  *linguagem que só tem o símbolo  $a$*
- $\emptyset$  representa a linguagem vazia  $\{\}$
- $a + b + c$  representa a linguagem  $\{a, b, c\}$
- $\epsilon$  representa a linguagem  $\{\epsilon\}$
- $a^*$  representa a linguagem  $\{a\}^*$  *cadeias com o símbolo  $a$*
- $(a + b)^*$  : cadeias no conjunto  $\{a, b\}^*$  *todas as cadeias formadas com  $a$  e  $b$*
- $(a^* + b^*)^*$  e  $(a + b)^*$  : especificam a mesma linguagem  $\{a, b\}^*$
- $a^* + b^*$  representa cadeias formadas só com  $a$  **OU** só com  $b$  , incluindo a cadeia vazia, mas não uma mistura de ambas

- Se  $e$  é uma ER, vamos usar  $[e]$  para denotar a linguagem que  $e$  descreve
  - $[\sigma_i] = \{\sigma_i\}$  , com  $\sigma_i \in \Sigma$
  - $[\emptyset] = \emptyset$
  - $[\varepsilon] = \{\varepsilon\}$
- 
- |   |
|---|
| caractere                                     |
| alfabeto                                      |
| conjunto                                      |
| classe de caracteres aceitos                  |
| conjunto vazio                                |
| classe de caracteres vazia                    |
| conjunto que contém o símbolo $\varepsilon$   |
| classe de caracteres que contém $\varepsilon$ |

(continua)

$[e]$  é a linguagem formal descrita, especificada, pela expressão regular  $e$

Se  $e$  é uma expressão regular, então a linguagem  $[e]$  é regular

- $[ e_1 + e_2 ] = [ e_1 ] \cup [ e_2 ]$ 
  - $[e_1 + e_2]$  : classe de caracteres aceitos pelas ER  $e_1$  OU  $e_2$
  - $[e_1]$  : classe de caracteres aceita pela ER  $e_1$
  - $[e_2]$  : classe de caracteres aceita pela ER  $e_2$
  - $\cup$  : união de dois conjuntos que contém todos os elementos de ambos os conjuntos, sem duplicatas

"[]" : classe de caracteres; dentro dos colchetes são colocados os caracteres permitidos; p. ex., [a-z] representa qualquer letra minúscula de 'a' a 'z'

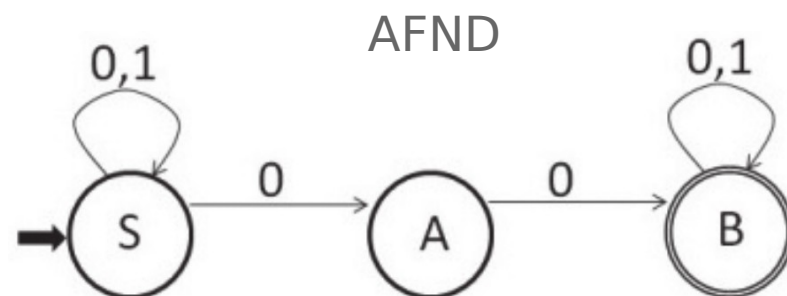
- Se tivermos:
  - $e_1 = a \rightarrow$  Aceita apenas o caractere "a"
  - $e_2 = b \rightarrow$  Aceita apenas o caractere "b"
- Então:
  - $[e_1] = \{ "a" \}$
  - $[e_2] = \{ "b" \}$
  - $[e_1 + e_2] = [e_1] \cup [e_2] = \{ "a" \} \cup \{ "b" \} = \{ "a", "b" \}$
- Isso significa que a expressão  $e_1 + e_2$  aceita qualquer cadeia composta por "a" ou "b", mas não aceita cadeias que misturem os dois

- $[ e_1 e_2 ] = [ e_1 ][ e_2 ]$
- a concatenação das linguagens definidas por  $e_1$  e  $e_2$  é equivalente à junção das duas **classes de caracteres**, ou seja, qualquer string que pode ser formada pelos caracteres de  $e_1$  seguidos pelos caracteres de  $e_2$



- $[e^*] = [e]^*$ 
  - $[e]$  : a classe de caracteres aceita pela expressão regular  $e$
  - $[e]^*$  : aplicação do fecho de Kleene à classe de caracteres  $[e]$ 
    - O fecho de Kleene é uma operação que aceita zero ou mais repetições da classe de caracteres  $[e]$
- $[e^*]$  é a classe de caracteres resultante da aplicação do fecho de Kleene diretamente à expressão regular  $e$
- $[e]^*$  é a aplicação do fecho de Kleene à classe de caracteres aceita por  $e$
- Ambas representam o mesmo conjunto: todas as strings que podem ser formadas por zero ou mais repetições da classe de caracteres  $[e]$

- Um autômato finito com transição  $\epsilon$ ,  $AF_{\epsilon}$ , é um AFND (Autômato Finito Não Determinístico) onde a função de transição inclui a leitura da cadeia vazia
- A transição  $\epsilon$  é uma forma de realizar a transição de estados sem que sejam lidos símbolos da entrada



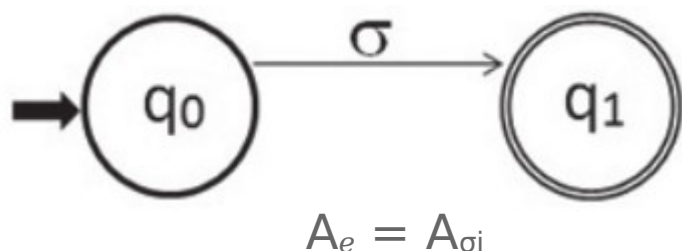
delta

- A função de transição  $\delta$  de um  $AF_\epsilon$  é tal que,
  - $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \wp(Q)$  "A função de transição  $\delta$  recebe um estado em  $Q$  e um símbolo do alfabeto  $\Sigma$  (ou a transição vazia  $\epsilon$ ) e retorna um conjunto de estados  $Q$  possíveis."
    - $Q$  é o conjunto de estados
    - $\Sigma$  é o alfabeto (conjunto de símbolos de entrada)
    - $\{\epsilon\}$  representa a transição vazia (épsilon)
    - $\wp(Q)$  é o conjunto potência de  $Q$  (o conjunto de todos os subconjuntos de  $Q$ )
 

Ex.:  $A = \{1, 2\}$  ;  $\wp(A) = \{\{\}, \{1\}, \{2\}, \{1, 2\}\}$
  - Isso significa que, para um dado estado e um símbolo de entrada (ou  $\epsilon$ ), a função de transição pode resultar em um *conjunto de estados*, o que caracteriza o não determinismo
- $Q \times (\Sigma \cup \{\epsilon\})$  representa o domínio da função, ou seja, os estados e os símbolos de entrada

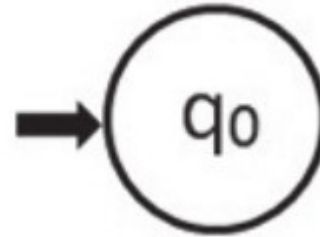
- O efeito de uma transição  $\epsilon$  é não determinístico
  - A transição  $\delta(q_1, \epsilon) = \{q_2\}$  indica que na leitura de  $\epsilon$ , estando o autômato no estado  $q_1$ , este muda para  $q_2$
  - Desde que a leitura de  $\epsilon$  é opcional, pois a cadeia vazia é subcadeia de qualquer outra cadeia, também pode-se considerar que o autômato permanece no estado  $q_1$
- Todo autômato finito com transições  $\epsilon$  (ou AFND- $\epsilon$ ) reconhece uma linguagem regular (ou LR)
  - Uma LR é um conjunto de strings (cadeias de caracteres) que pode ser reconhecido por um AF ou descrito por uma ER, que usa operações como união, concatenação e fecho de Kleene

- Seja  $e$  uma ER. O autômato finito  $A_e$  que reconhece  $e$ , é definido:
- Se  $e$  é um símbolo único  $\sigma_i$  (onde  $\sigma_i \in \Sigma$ ), então  $A_e$  é um autômato que aceita exatamente a cadeia contendo  $\sigma_i$



Se a ER  $e$  for simplesmente um único símbolo  $\sigma_i$  (ou seja, a ER mais simples possível), então o AFND  $A_e$  correspondente será o AFND que reconhece apenas esse símbolo  $\sigma_i$

- Se  $e = \emptyset$  , então  $A_e = A_{\emptyset}$

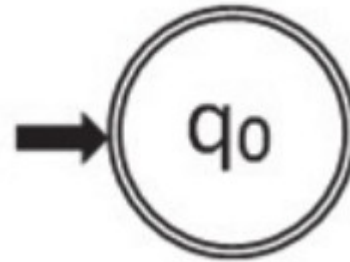


- Se a expressão regular  $e$  representa o conjunto vazio (ou seja, não reconhece nenhuma string), então o autômato finito  $A_e$  construído a partir dessa expressão regular é o autômato  $A_{\emptyset}$  , que também reconhece o conjunto vazio (ou seja, não aceita nenhuma string)

$A_e$ : "A índice  $\varepsilon$ " ou "A associado a  $\varepsilon$ " ( $\varepsilon$  sendo a expressão regular em questão)

$A_{\emptyset}$ : "A índice vazio" ou "A associado ao conjunto vazio"

- Se  $e = \varepsilon$ , então  $A_e = A_\varepsilon$

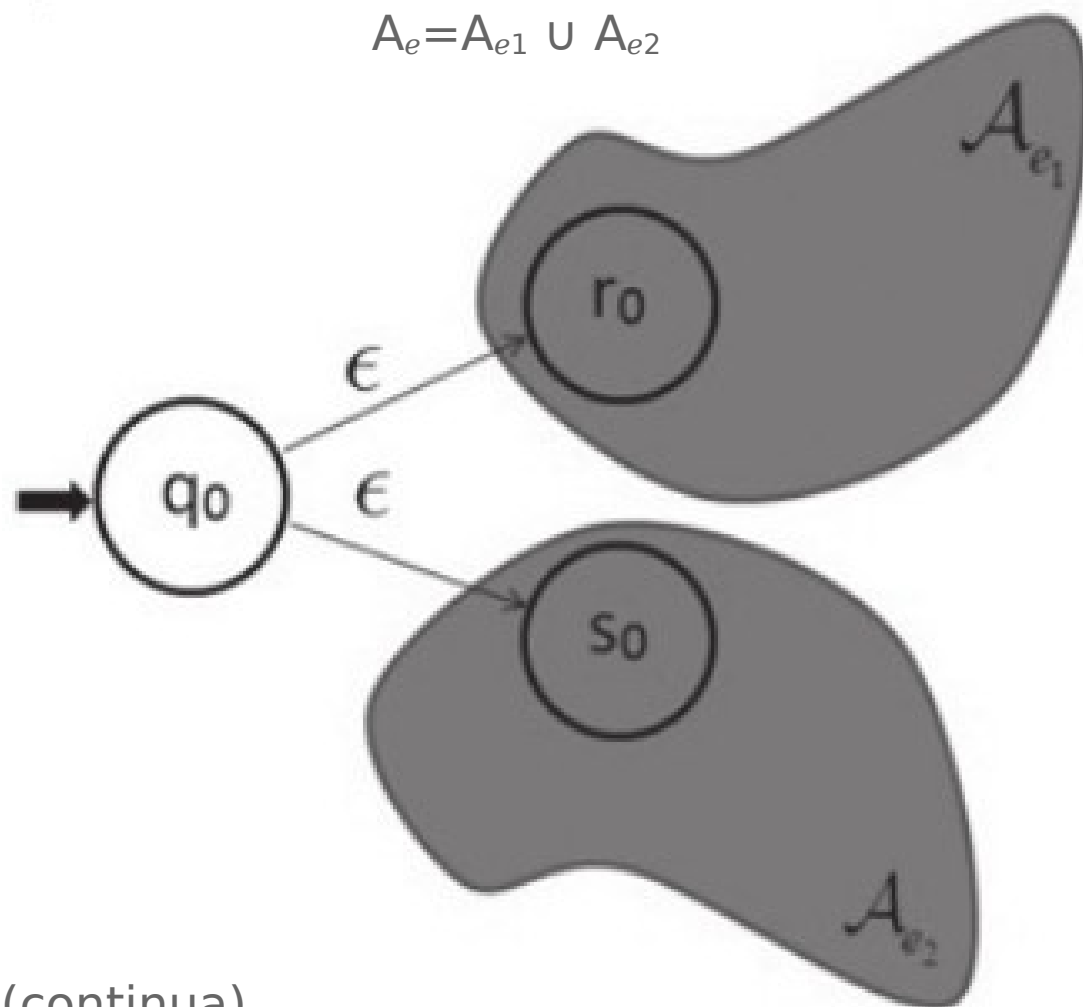


- Se a expressão regular  $e$  representa a string vazia ( $\varepsilon$ ), então o autômato finito  $A_e$  construído a partir dessa expressão regular é o autômato  $A_\varepsilon$ , que reconhece apenas a string vazia

- Se  $e = e_1 + e_2$  então  $A_e = A_{e_1} \cup A_{e_2}$ , onde é criado um novo estado inicial que tem uma transição  $\epsilon$  indo para cada um dos estados iniciais de  $A_{e_1}$  e  $A_{e_2}$

Ex.: seja  $e_1 = a$  e  $e_2 = b$

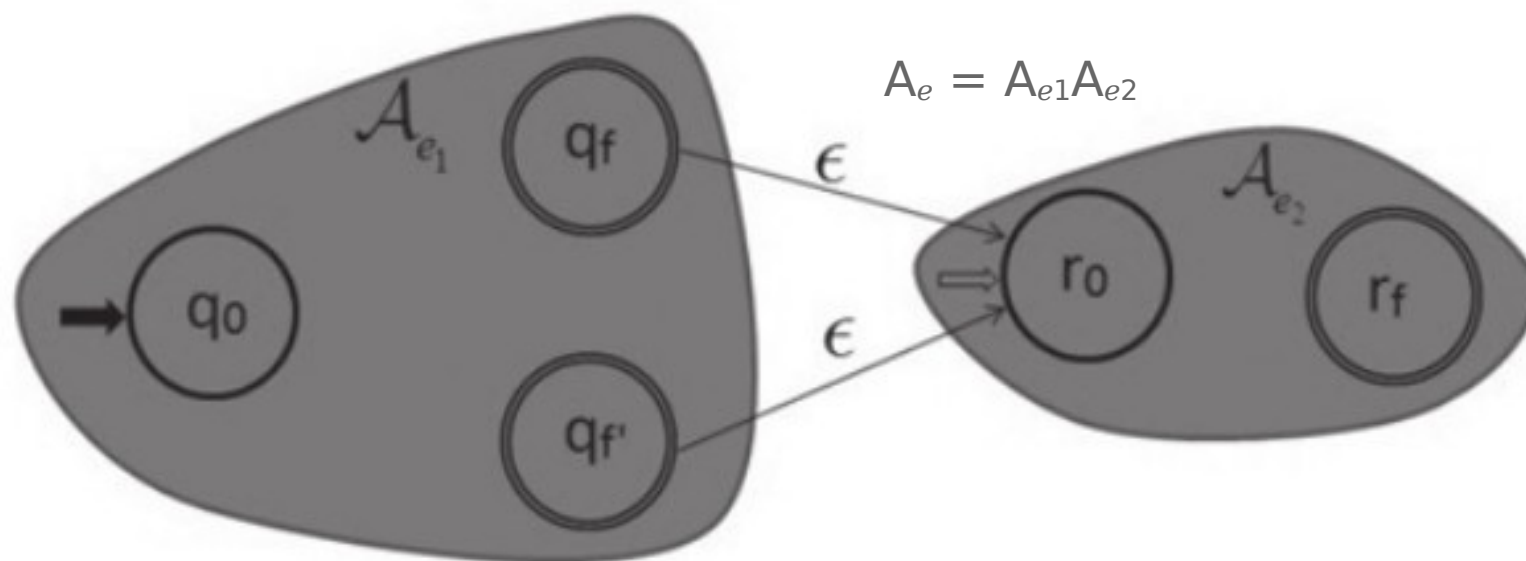
- $A_{e_1}$  seria um autômato com um estado inicial e um estado final, com uma transição  $a$  entre eles
- $A_{e_2}$  seria um autômato similar, mas com uma transição  $b$
- $A_e$  (para  $e = a + b$ ) teria um novo estado inicial  $q_0$  com uma transição  $\epsilon$  para o estado inicial de  $A_{e_1}$  e outra transição  $\epsilon$  para o estado inicial de  $A_{e_2}$



(continua)



- Se  $e = e_1 e_2$ , então  $A_e = A_{e_1} A_{e_2}$ , onde cada estado final de  $A_{e_1}$  tem uma transição  $\epsilon$  indo para o estado inicial de  $A_{e_2}$

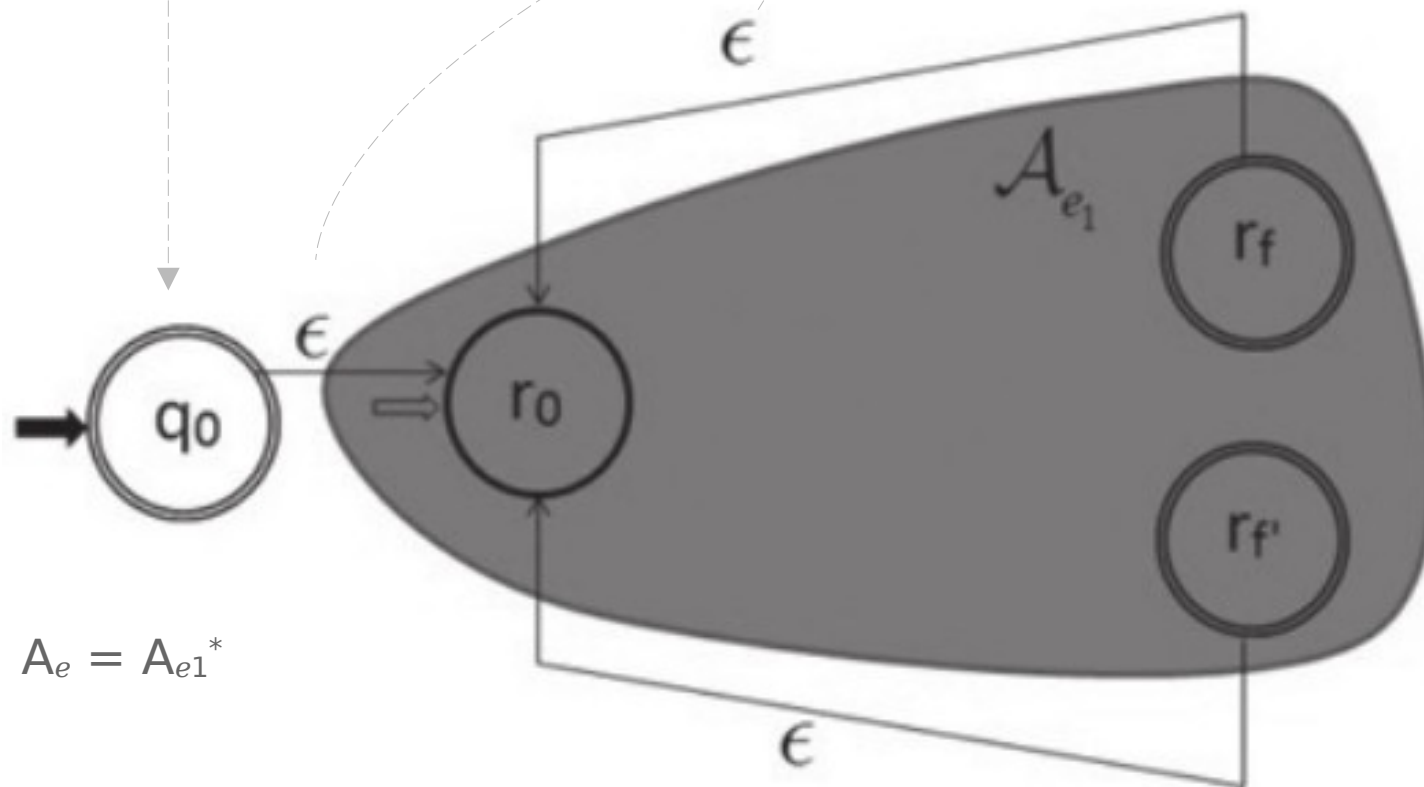


- Ex.: seja  $e_1 = a + b$  e  $e_2 = c$
- $A_{e_1}$  seria um autômato com um estado inicial e dois estados finais diferentes para representar "a" e "b"
- $A_{e_2}$  seria um autômato com um estado inicial e um estado final, com uma transição "c" entre eles
- $A_e$  (para  $e = ab$ ) seria construído conectando os estados finais de  $A_{e_1}$  ao estado inicial de  $A_{e_2}$  com transições  $\epsilon$

(continua)

se uma ER  $e$  é o fecho de Kleene de  $e_1$ , então o autômato  $A_e$  correspondente pode ser construído com base no autômato  $A_{e_1}$  a partir da aplicação do fecho de Kleene

- Se  $e = e_1^*$ , então  $A_e = A_{e_1}^*$ , onde é criado um novo estado inicial/final  $q_0$ , com uma transição  $\epsilon$  indo para o estado inicial de  $A_{e_1}$  e são incluídas transições  $\epsilon$  dos estados finais para o inicial de  $A_{e_1}$



- Para qualquer Expressão Regular (ER)  $e$ , o **autômato finito com transições  $\epsilon$**  ( $AF_\epsilon A_e$ ) reconhece a linguagem definida por  $[e]$
- Ao eliminar as transições  $\epsilon$  desse autômato, obtemos um AFND equivalente que também reconhece a linguagem definida por  $[e]$

Ambos os autômatos, o  $AF_\epsilon$  original e o AFND resultante da eliminação das transições  $\epsilon$ , aceitam exatamente as mesmas strings, ex.:  $\{ab, ac\}$ . Isso demonstra que, para qualquer ER, é possível construir um  $AF_\epsilon$  que reconhece a mesma linguagem, e esse  $AF_\epsilon$  pode ser convertido para um AFND equivalente sem transições  $\epsilon$ , comprovando a equivalência entre eles

$AF_\epsilon A_e$  : autômato finito não determinístico com transições  $\epsilon$  que reconhece a linguagem descrita pela expressão regular  $e$ .  
É um  $AF_\epsilon$  construído para reconhecer uma linguagem particular  $e$ .

- Gramáticas regulares
- Via autômatos finitos (determinísticos ou não)
- Via expressões regulares

- Afirma que se conhecemos as linguagens  $L_1$  e  $L_2$  e definimos a linguagem  $L$  pela igualdade:

$$L = L_1 L + L_2 \quad (\text{recursiva, } L \text{ aparece nos 2 lados})$$

- Então a menor linguagem que atende à igualdade é dada por:

$$L = L_1^* L_2 \quad (\text{ER direta, sem recursão})$$

- A solução é única quando o símbolo vazio  $\varepsilon$  não pertence a  $L_1$ 
  - Isso significa que  $L_1$  não deve conter a palavra vazia, caso contrário, haveria outras soluções possíveis

- Considere as LR :  $L_1 = \{ a \}$  ;  $L_2 = \{ b \}$
- Vamos definir a linguagem  $L$  tal que:  $L = L_1 L + L_2$
- Usando o Lema de Arden, sabemos que:  $L = L_1^* L_2$
- Logo:  $L_1^* = \{ \epsilon , a , aa , aaa , \dots \}$
- Portanto, a menor linguagem  $L$  que satisfaz a equação é:  
$$L = \{ a \}^* \{ b \} = \{ b , ab , aab , aaab , \dots \}$$
- $L$  é formada por todas as cadeias que terminam em  $b$  e que podem ter qualquer número de  $a$  antes do  $b$