CSC 339 – Theory of Computation
Fall 2022-2023

5. Regular Expressions and Regular Languages

Outline

- Regular expressions
- Regular languages
- Union
- Concatenation
- Star
- Reverse
- Complement
- Intersection
- Equivalence of regular expressions with finite automata

Regular Expressions

- Algebraic way to describe languages
- Describe exactly regular languages
- If E is a regular expression, then L(E) is the regular language it defines.
- AWK, GREP in UNIX
  PERL, text editors.
- Notation: Empty string ε
- eg.
  Regular expression R: (0+1)0*;
  Language it denotes $L(R)=(\{0\}\cup\{1\})\{0\}^*$
- eg.
  R: (0+1)*; $L(R)= \{0, 1\}^*$

Formal Definition

- Definition (inductive definition)
  R is a regular expression if R is
  - $a \in \Sigma$
  - ε
  - ϕ

  The languages obtained are regular:
  - $L(R_1) \cup L(R_2)$, R_1 and R_2 are regular expressions
  - $L(R_1R_2)$, R_1 and R_2 are regular expressions
  - R: regular expression,
    R^* is a regular expression; $L(R^*)=L(R)^*$

4

Examples

- e.g., $\Sigma = \{0, 1\}$
  - 0^*10^* $\{w$: w has exactly a single 1$\}$
  - $\Sigma^*1\Sigma^*$ (w has at least 1)
  - $\Sigma^*001\Sigma^*$ (contains the substring 001)
  - $(\Sigma\Sigma)^*$ (strings of even length)
  - $01+10$ ($\{01,10\}$)
  - $0\Sigma^*0+1\Sigma^*1+0+1$ (strings starting and ending with the same symbol)
  - $(0+\varepsilon)1^*$ 01^*+1^*
  - $(0+\varepsilon)(1+\varepsilon)$ $\{\varepsilon, 0, 1, 01\}$
  - $1^*\phi$ ϕ
  - ϕ^* $\{\varepsilon\}$

    (the language is empty, the * operator can put together 0 strings, giving only the empty string)

Examples

- R: regular expression
  $L(R) \cup \phi = L(R)$
  (adding the empty language to any other language does not change it).
  $L(R\lambda) = L(R)$
- $L(R+\varepsilon)=L(R)$?
  $L(R)\phi = L(R)$?
- If R = 0 then
  - $L(R)=\{0\}$ and $L(R+\varepsilon)=\{0,\varepsilon\}$
  - $L(R)=\{0\}$ and $L(R)\phi = \phi$

Identities and Annihilators

- $\varnothing$ is a regular expression that represents the language that does not contain any strings.
- $\varnothing$ is the identity for +
  - $R+\varnothing = R$
- ε is the identity for concatenation
  - $\varepsilon R = R\varepsilon = R$
- $\varnothing$ is the annihilator for concatenation
  - $\varnothing R = R\varnothing = \varnothing$

Regular languages L_1 and L_2

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1{}^*$

Reversal: $L_1{}^R$

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

Are regular Languages

Regular languages are **closed under**

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1{}^*$

Reversal: $L_1{}^R$

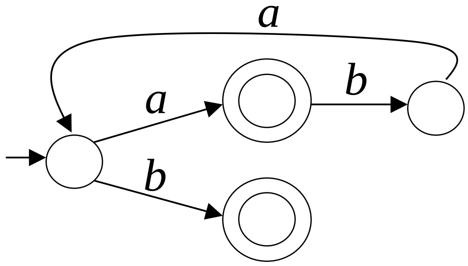
Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

$\left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\}$ Are regular Languages

A useful transformation: **use one accept state**

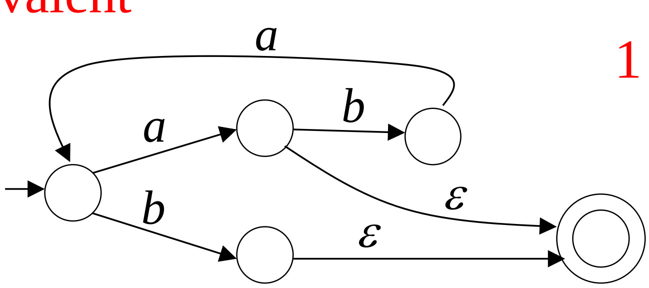
NFA



2 accept states

Equivalent

NFA

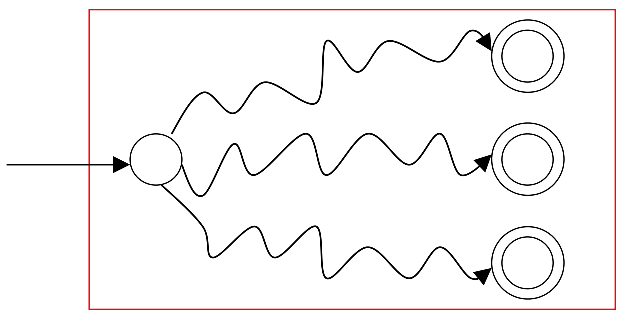


1 accept state

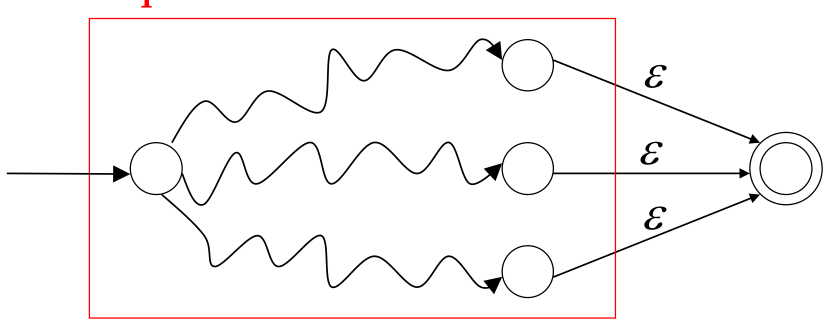
10

In General

NFA



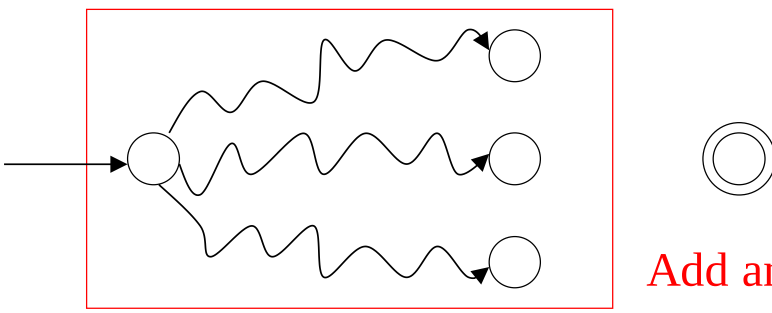
Equivalent NFA



Single accepting state

Extreme case

NFA without accepting state



Add an accepting state without transitions

Consider two languages

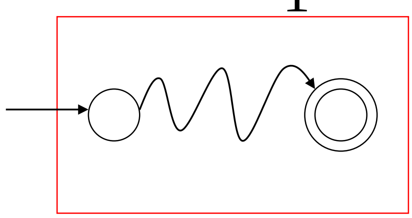
Regular language L_1

Regular language L_2

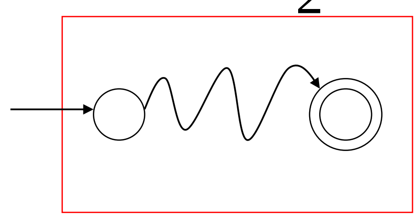
$$L(M_1) = L_1$$

$$L(M_2) = L_2$$

NFA M_1



NFA M_2



Single accepting state

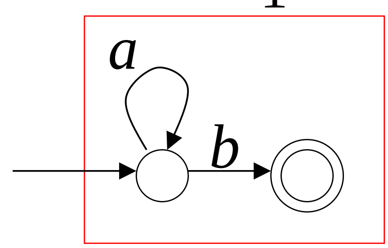
Single accepting state

13

Example:

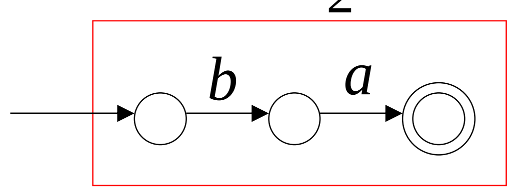
M_1

$L_1 = \{a^n b\}$
$n \geq 0$



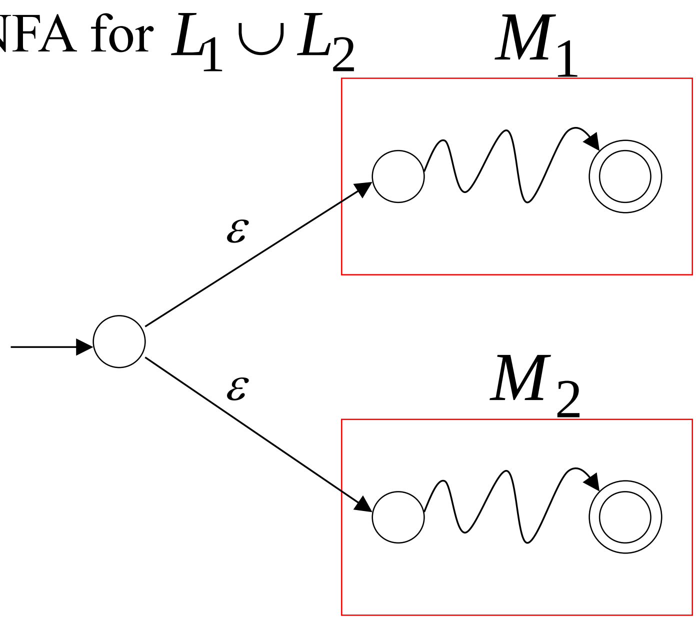
M_2

$L_2 = \{ba\}$



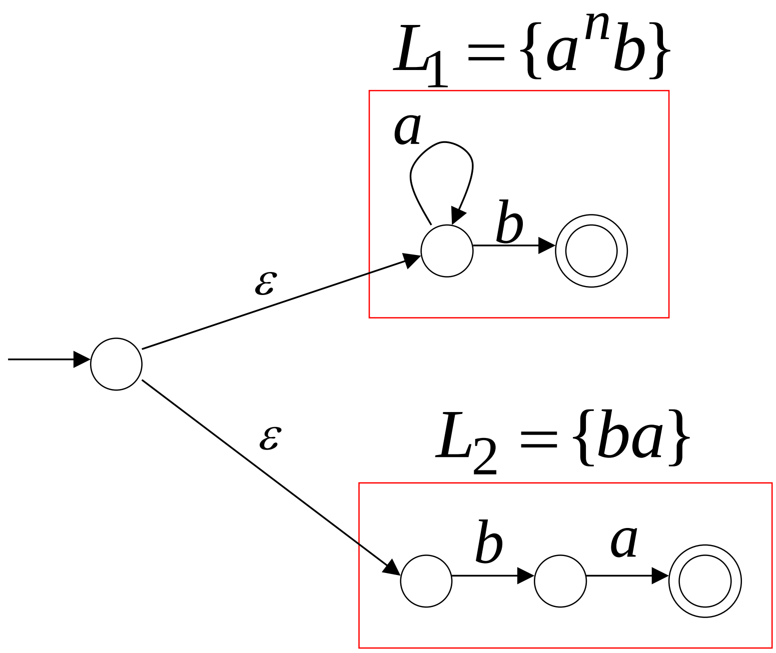
Union

NFA for $L_1 \cup L_2$



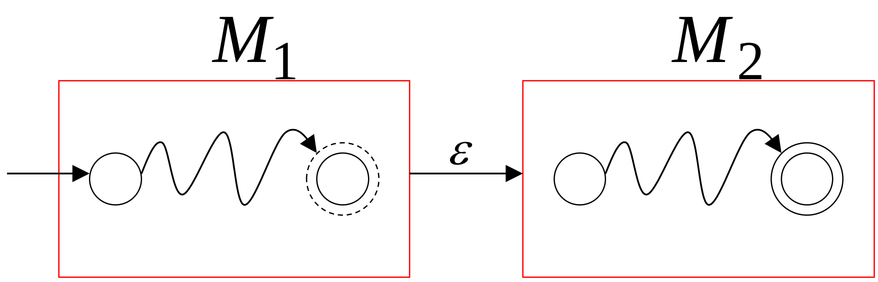
Example:

NFA for $L_1 \cup L_2 = \{a^n b\} \cup \{ba\}$



Concatenation

- NFA for L_1L_2



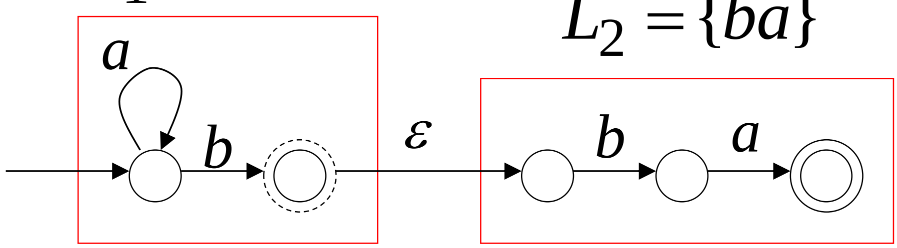
17

Example:

$$L_1 L_2 = \{a^n b\}\{ba\} = \{a^n bba\}$$

$$L_1 = \{a^n b\}$$

$$L_2 = \{ba\}$$



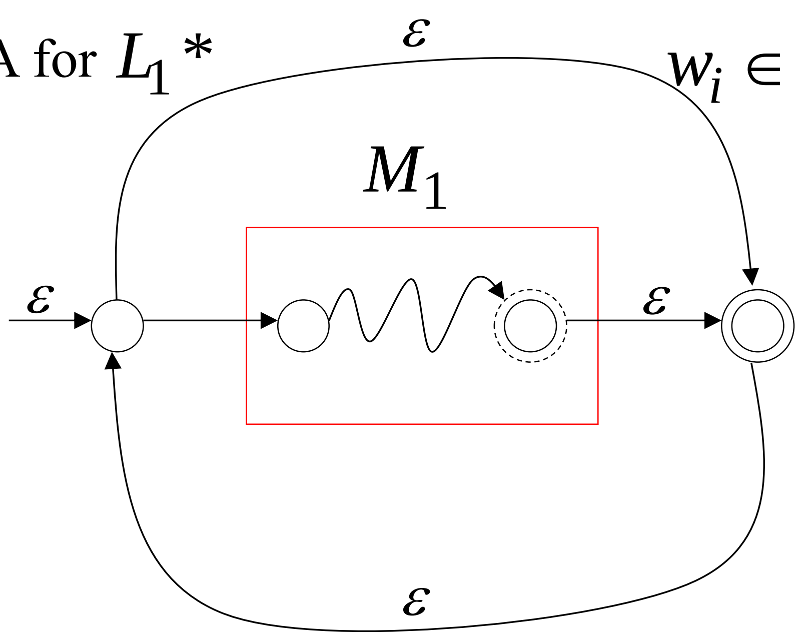
18

Star Operation

$w = w_1 w_2 \cdots w_k$

NFA for $L_1{}^*$

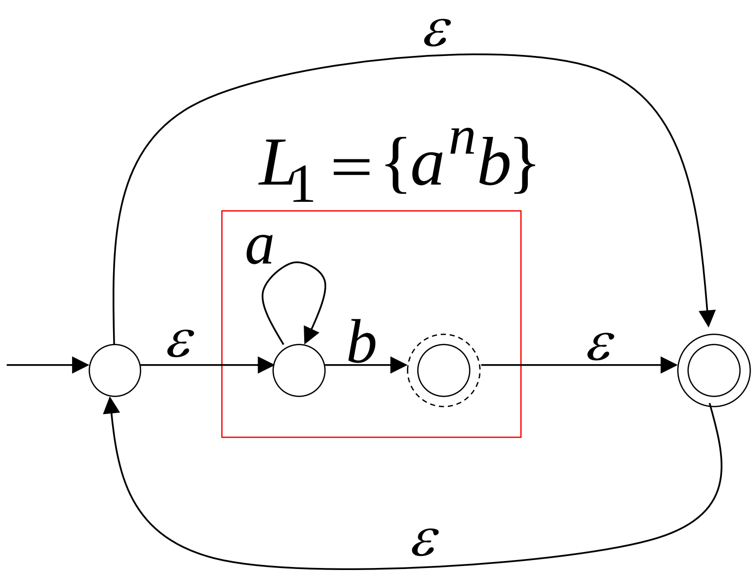
$w_i \in L_1$



$\varepsilon \in L_1^*$

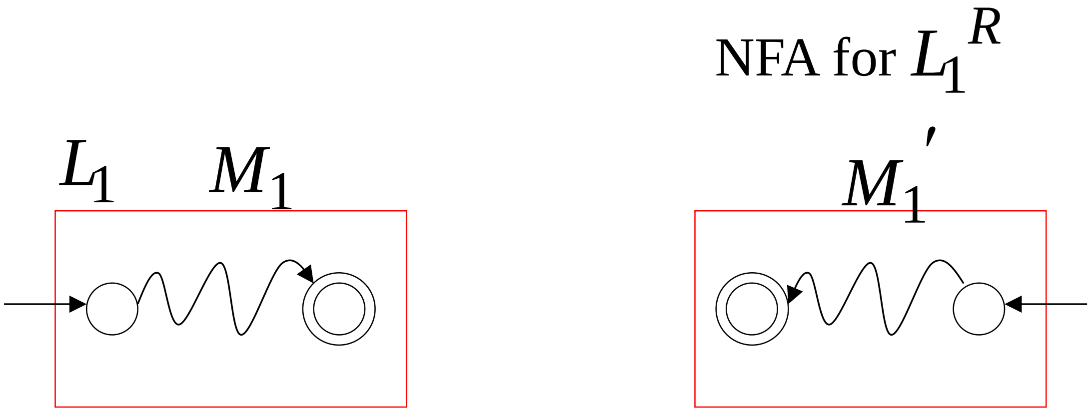
Example:

NFA for $L_1^* = \{a^n b\}^*$



20

Reverse

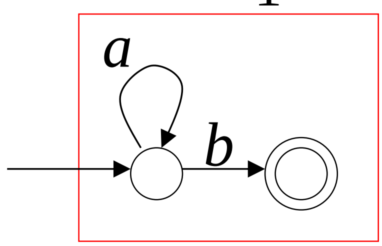


1. Reverse all transitions.
2. Make the initial state an accepting state and vice versa.

Example:

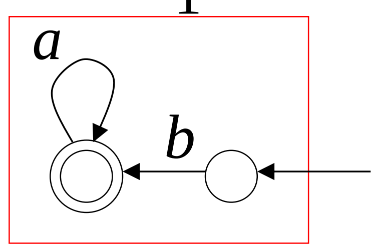
M_1

$L_1 = \{a^n b\}$

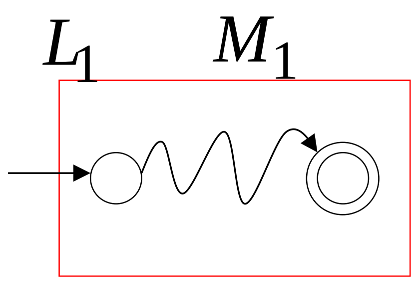


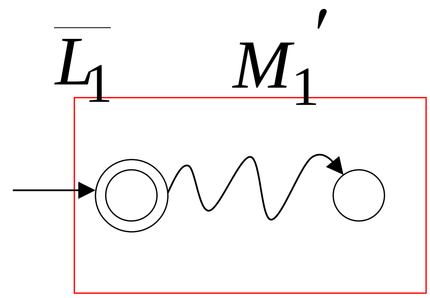
M_1'

$L_1^R = \{ba^n\}$



Complement





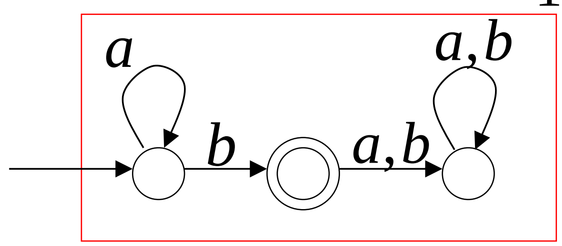
1. Take the DFA that accepts L_1
2. Make accepting states non-final,
   and vice-versa.

23

Example:

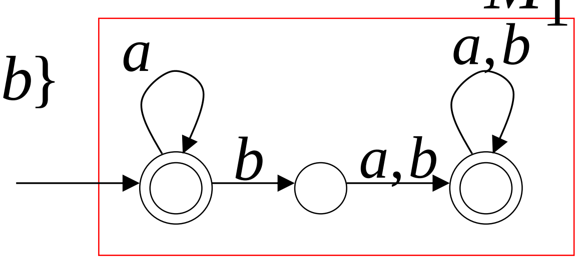
M_1

$$L_1 = \{a^n b\}$$



M_1'

$$\overline{L_1} = \{a,b\}^* - \{a^n b\}$$



24

Intersection

L_1 regular

L_2 regular

We show

$L_1 \cap L_2$

regular

De Morgan's Law: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

$L_1 , \ L_2$ regular

$\Rightarrow$ $\overline{L_1} , \ \overline{L_2}$ regular

$\Rightarrow$ $\overline{L_1} \cup \overline{L_2}$ regular

$\Rightarrow$ $\overline{\overline{L_1} \cup \overline{L_2}}$ regular

$\Rightarrow$ $L_1 \cap L_2$ regular

Example:

$L_1 = \{a^n b\}$ regular

$L_2 = \{ab, ba\}$ regular

$\Rightarrow L_1 \cap L_2 = \{ab\}$

regular

27

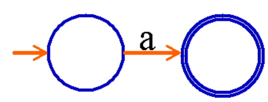
Equivalence of RE with Finite Automata

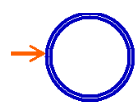
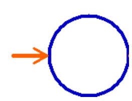
- Theorem:
  A language is regular if and only if some regular expression describes it.

- Lemma:
  If a language is described by a regular expression then it is regular.

Equivalence of RE with Finite Automata

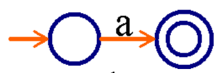
- Proof:
  Let R be a regular expression describing some language A
  Goal: Convert R into an NFA N.

  - $R=a \in \Sigma$, $L(R)=\{a\}$ 

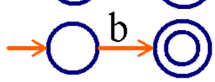
  - $R=\varepsilon$, $L(R)=\{\varepsilon\}$ 
  - $R=\phi$, $L(R)=\phi$ 
  - $R = R_1+R_2$
  - $R = R_1R_2$
  - $R = R_1{}^*$

Equivalence of RE with Finite Automata

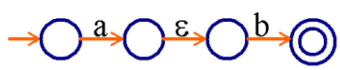
- e.g., (ab + a)*
  - a



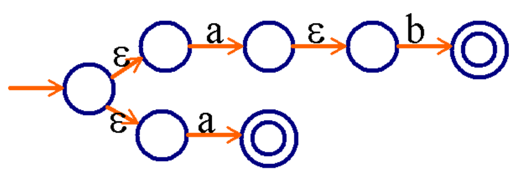
  - b



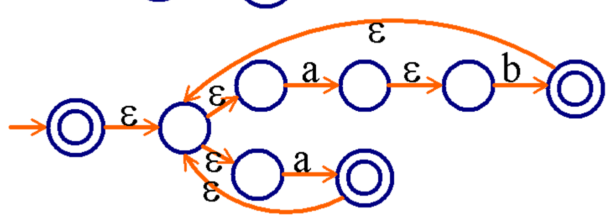
  - ab



  - ab + a



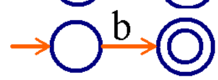
  - (ab + a)*



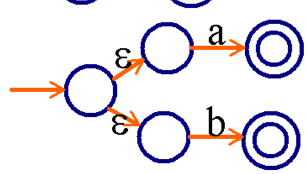
Equivalence of RE with Finite Automata

- eg. (a + b)*aba
  - a

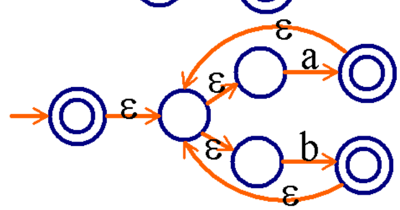
  - b



  - a + b



  - (a + b)*



  - aba

Equivalence of RE with Finite Automata

– (a + b)*aba