


# 01204211 Discrete Mathematics

## Lecture 7a: Languages and regular expressions<sup>1</sup>

Jittat Fakcharoenphol

August 19, 2025

---

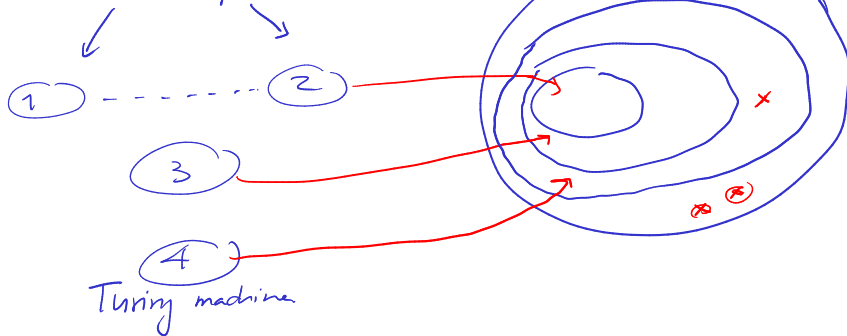
<sup>1</sup>Based on lecture notes of *Models of Computation* course by Jeff Erickson. 

What is computation? "ชีวิตเรา"

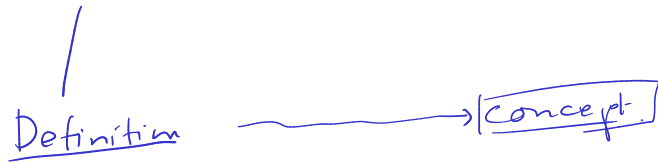
โมเดลการคำนวณ

โปรแกรม

↳ ปัญหาที่คอมพิวเตอร์  
แก้ได้บ้าง



# Models of computations



Languages = specifications

↳ set of strings

$\{ \text{"hello"}, \text{"ekitike"} \}$

$\{ \text{"1+1=2"}, \text{"15+32=47"}, \dots \}$

$\{ \text{"2"}, \text{"3"}, \text{"5"}, \text{"7"}, \text{"11"}, \dots \}$  set of natural numbers:

$\{ 0, 00, 000, 0000, \dots \}$

$\{ (a, b, c) \mid a, b, c \in \mathbb{N}, a+b=c \}$



y - output  
N - input

Strings

• binary symbols  $\Sigma = \{0, 1\}$

①  $\Sigma^*$  is the set of strings over  $\Sigma$

② a string  $x$  is a sequence of symbols in  $\Sigma$

def:  $a \in \Sigma$

## Formal definition: strings

Intuitively, a string is a *finite* sequence of symbols. However, to be able to formally prove properties of strings we need a precise definition.

Let a finite set  $\Sigma$  be the alphabet. (E.g., for bit strings,  $\Sigma = \{0, 1\}$ ; for digits,  $\Sigma = \{0, 1, \dots, 9\}$ ; for English string  $\Sigma = \{a, b, \dots, z\}$ .)

The following is a recursive definition of strings.

### Recursive definition of strings

A **string**  $w$  over alphabet  $\Sigma$  is either

- ▶ the empty string  $\varepsilon$ , or
- ▶  $a \cdot x$  where  $a \in \Sigma$  and  $x$  is a string.

The set of all strings over alphabet  $\Sigma$  is denoted by  $\Sigma^*$ .

## Review: more recursive definitions

### Lengths

For a string  $w$ , let  $|w|$  be the length of  $w$  defined as

$$|w| = \begin{cases} 0 & \text{when } w = \varepsilon \\ 1 + |x| & \text{when } w = a \cdot x \end{cases}$$

### Concatenation

For strings  $w$  and  $z$ , the concatenation  $w \cdot z$  is defined recursively as

$$w \cdot z = \begin{cases} z & \text{when } w = \varepsilon \\ a \cdot (x \cdot z) & \text{when } w = a \cdot x \end{cases}$$

# Review: proving facts about strings

## Lemma 1

For strings  $w$  and  $x$ ,  $|w \cdot x| = |w| + |x|$ .

## Proof.

Induction Hypothesis: Given string  $y$  s.t.  $|y| < |w|$ ,  
 $|y \cdot x| = |y| + |x|$ .

Case 1:  $w = \epsilon$ .

Case 2:  $w = a \cdot y$  for string  $y$  s.t.  $|y| < |w|$

$$\begin{aligned} \text{Hence } |w \cdot x| &= |a \cdot y \cdot x| \quad \text{rather} \\ &= 1 + |y \cdot x| \quad \text{rather vs length} \\ &= 1 + |y| + |x| \quad \boxed{\text{IH}} = |w| + |x| \quad \text{rather } w \end{aligned}$$



# Formal languages

A formal language is a set of strings over some finite alphabet  $\Sigma$ .

Examples:

$\{ \epsilon, 10, 100, 1000, 10000, 100000, \dots \}$

$\{ 1, 11, 101, 111, 1001, 1011, 1101, 1111, \dots \}$



## Careful...

These are different languages:  $\emptyset, \{\varepsilon\}$

And  $\varepsilon$  is not a language.



How to describe languages?

$$A = \{\text{hello}\}$$

$$Z = \phi$$

$$B = \{a, an, the\}$$

$$C = \{hunsen, thaksin, prayuth, ink, \}$$

$$A \cup B \cup C$$

$$B \cdot C = \{ahunsen, anhuse, \dots, \\ atun, an, \dots\}$$

$$\{0^n \mid n \geq 0\}$$

$$\{\epsilon, 0, 00, 000, 0000, \dots\}$$

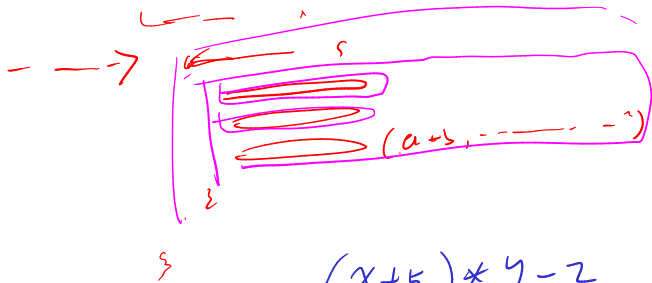
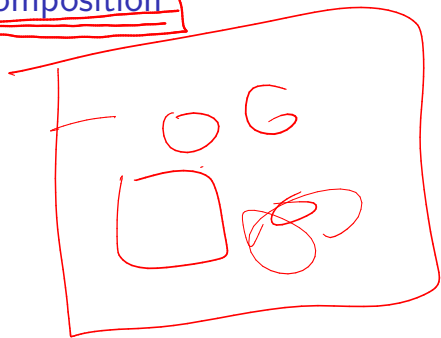
$$\{0^n 1^n \mid n \geq 0\}$$

$$\{\epsilon, 01, 0011, 000111, \dots\}$$

$$\{0^p \mid p \text{ is a prime number}\}$$

ภาษาเรกูลาร์

# Composition



$$\underline{(x+5) * y - 2}$$

decomposition

LU.

## Combining languages

If  $A$  and  $B$  are languages over alphabet  $\Sigma$ .

- ▶ Basic set operations:  $\underline{A \cup B}$ ,  $\underline{A \cap B}$ ,  $\underline{\bar{A}} = \Sigma^* \setminus A$ .
- ▶ Concatenation:  $\underline{A \cdot B}$ .

$$A \cdot B = \{xy \mid x \in A, y \in B\}$$

- ▶ Kleene closure or Kleene star:  $A^*$ .

String  $w \in A^*$  iff

1.  $w = \varepsilon$ .

2.  $w = \underline{x} \cdot \underline{y}$  iff  $x \in A$  and  $y \in A^*$

$$A = \{a, an, the\}$$

$$A^* = \{\varepsilon, a, an, the, aa, ana, thea, \underline{anan, aan, thean}, \dots\}$$

Also  $\boxed{A^+} = A \cdot A^*$

## Examples

$$\Sigma^1 = \{0, 1\}$$

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \dots\}$$

$$(\Sigma^*) \cdot \{0\}$$

# Regular languages

$$\{\} \circ \{\text{hello}\} = \emptyset$$

$$\{a\} \circ \{c, d, a\} \\ = \{ac, ad, aa\}$$

## Definition: regular languages

A language  $L$  is **regular** if and only if it satisfies one of the following conditions:

- ▶  $L$  is empty;
- ▶  $L$  contains one string (can be the empty string  $\varepsilon$ );
- ▶  $L$  is a union of two regular languages;
- ▶  $L$  is the concatenation of two regular languages; or
- ▶  $L$  is the Kleene closure of a regular language.

$W$  is regular expr  
a'1

- ①  $W = \emptyset$
- ②  $W = x$  where  $x$  is a string
- ③  $W = S + T$  where  $S$  &  $T$  are regular expressions
- ④  $W = S \circ T$  where  $S$  &  $T$  are regular expressions
- ⑤  $W = S^*$

# Examples

$$\Sigma^+ = \{0, 1\}^*$$

$$\{ \epsilon \}$$

$$\{ 000 \}$$

$$\left( \{ 000 \} \cup \{ 11 \} \right) \cdot \{ \epsilon \} =$$

$$\{ 000, 11 \} \cdot \{ \epsilon \} = \{ \epsilon \} \quad \times$$

$$\{ 0, 1 \}^*$$

$$\downarrow$$

$$(\{ \epsilon \} \cup \{ 1 \})^*$$

$$\{ 00 \}^* = \{ \epsilon, 0000, 006000, \dots \}$$

$$\{ 000, 11 \} = \{ 000, 11 \} \cdot \{ \epsilon \}$$

# Regular expressions

Let  $\Sigma = \{0, 1\}$ . Consider

$$\begin{aligned} & ((\underbrace{\{01\}} \cup (\underbrace{\{1\}} \cdot (\underbrace{\{0\}} \cup \underbrace{\{10\}}))) \cup (\underbrace{\{00\}} \cdot (\underbrace{\{1\}})^*)) \cdot ((\underbrace{\{0\}} \cdot \underbrace{\{0\}}) \cdot \underbrace{\{1\}}) \\ & \underbrace{\hspace{1.5cm}} \\ & \underbrace{\hspace{2.5cm}} \\ & \underbrace{\hspace{4.5cm}} \end{aligned}$$



# Regular expressions

→ offer (describe) regular language

Regular language

set  $\rightarrow ((\{01\} \cup (\{1\} \cdot (\{0\} \cup \{10\}))) \cup (\{00\} \cdot (\{1\})^*)) \cdot ((\{0\} \cdot \{0\}) \cdot \{1\})$

is represented as

$(01 + 1(0 + 10) + 00(1)^*)001$

$001 + 10 + 00^*$

Regular expressions

- ▶ omit braces around one-string sets
- ▶ use  $+$  instead of  $\cup$
- ▶ omit  $\cdot$
- ▶ follow the precedence: Kleene star operator  $*$ ,  $\cdot$  (implicitly), and  $+$ .

*Remark:  $+$  and  $\cdot$  are associative, i.e.,  $(A + B) + C = A + (B + C)$  and  $(A \cdot B) \cdot C = A \cdot (B \cdot C)$ .*

$001 + [10 + 00^*]$   
 $[001 + 10] + 00^*$

## Regular expressions: examples 1

Strings ที่นำหน้าด้วย 0000000000

$\{0,1\}^*$

$(0+1)^*0000000000$

substring ~~~ 010 ~~~

subsequence

~~~~~ 0 ~~~ 1 ~~~ 0 ~~~

(b) strings the  
end 010

$(0+1)^*010(0+1)^*$

(c)

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

$(0+1)^*0$

$(0+1)^*1$

## Regular expressions: examples 2

0101

All strings over  $\{0, 1\}$  except 010

$$0^* + 1^* + 1(0+1)^* + 00(0+1)^* \\ + 011(0+1)^* + 010(0+1)(0+1)^*$$

models

(d) :-

@

$$0^* + 1^* (0^* + 00^* 1^*)$$

grammar

• regular expressions

machine

• finite state automata.

# Subexpressions

01+111

regular expression  $w$  for  $\gamma$

which  $\emptyset$  is a language which is non empty.

If  $w \in \text{Regular expression}$ ,  $(w \text{ is } \emptyset) \Rightarrow (w \text{ is a language which is empty})$

**I.H.** Suppose  $w$  is a subexpression  $S$  of  $w$ , if  $S$  is  $\emptyset$ ,  $S$  is a non empty language.

Case 1:  $w = \emptyset$  — 😊

Case 2:  $w = x$  where  $x$  is a string

Case 3:  $w = S + T$  where  $S$  &  $T$  are regex

Case 4:  $w = S \cdot T$  — — —

Case 5:  $w = S^*$  where  $S$  is a regex

# Regex is everywhere

# Proofs about regular expressions - structural induction

## Lemma 2

*Every regular expression that does not use the symbol  $\emptyset$  represents a non-empty language.*

**Proof.**

## Lemma 2

*Every regular expression that does not use the symbol  $\emptyset$  represents a non-empty language.*

### **Proof.**

Let  $R$  be a regular expression that does not use the symbol  $\emptyset$ . We prove by (structural) induction that  $R$  represents a non-empty language.



## Lemma 2

Every regular expression that does not use the symbol  $\emptyset$  represents a non-empty language.

$P \Rightarrow Q$

**Proof.**

Let  $R$  be a regular expression that does not use the symbol  $\emptyset$ . We prove by (structural) induction that  $R$  represents a non-empty language.

**Induction hypothesis:** Every subexpression of  $R$  that does not use the symbol  $\emptyset$  represents a non-empty language.

## Lemma 2

*Every regular expression that does not use the symbol  $\emptyset$  represents a non-empty language.*

### **Proof.**

Let  $R$  be a regular expression that does not use the symbol  $\emptyset$ . We prove by (structural) induction that  $R$  represents a non-empty language.

**Induction hypothesis:** Every subexpression of  $R$  that does not use the symbol  $\emptyset$  represents a non-empty language.

Case 1:  $R = \emptyset$ . *X is it not?*

*במסגרת assumption ה'  $R \neq \emptyset$ ,  $R$  איננו יכול להיות  $\emptyset$  וזה בדיוק מה שצריך להוכיח*

## Lemma 2

*Every regular expression that does not use the symbol  $\emptyset$  represents a non-empty language.*

### **Proof.**

Let  $R$  be a regular expression that does not use the symbol  $\emptyset$ . We prove by (structural) induction that  $R$  represents a non-empty language.

**Induction hypothesis:** Every subexpression of  $R$  that does not use the symbol  $\emptyset$  represents a non-empty language.

*Case 1:*  $R = \emptyset$ .

*Case 2:*  $R$  is a single string.

Quoted,  $R$  is a nonempty language if string  $\neq \epsilon$

## Lemma 2

Every **regular expression** that does not use the symbol  $\emptyset$  represents a non-empty language.

### Proof.

Let  $R$  be a regular expression that does not use the symbol  $\emptyset$ . We prove by (structural) induction that  $R$  represents a non-empty language.  $L(R) \neq \emptyset$

**Induction hypothesis:** Every **subexpression** of  $R$  that does not use the symbol  $\emptyset$  represents a non-empty language.  $(L(S) \neq \emptyset)$

Case 1:  $R = \emptyset$ .

Case 2:  $R$  is a single string.

Goal:  $R$  is non-empty language.

**Proof.** (cont.2/4)

Case 3:  $R = S + T$  for some regular expressions  $S$  and  $T$ .

| given                        | known                                                                          |
|------------------------------|--------------------------------------------------------------------------------|
| 1. $S$ is $\emptyset$        | because $R$ is $\emptyset$ and $S$ is a subexpression of $R$                   |
| 2. $S$ is non-empty language | I.H. case ①                                                                    |
| 3. $T$ is $\emptyset$        |                                                                                |
| 4. $T$ is non-empty language |                                                                                |
| 5. $R$ is non-empty language | because $R$ is the union of language of $S$ and $T$ and $S$ is $\emptyset$ , ② |

for  $L(R)$  is language of  $R$

- because  $S$  is  $\emptyset$ ;  $T$  is subexpression of  $R$  and  $R$  is  $\emptyset$  and  $S$  is  $\emptyset$  and  $T$  is  $\emptyset$  and

- an I.H. assumption [  $S$  is  $\emptyset$ ;  $T$  is non empty language ]

$$L(S) \neq \emptyset, L(T) \neq \emptyset$$

- because  $R$  is non empty language and union of language of  $S$  and  $T$  and  $S$  is  $\emptyset$  and  $T$  is non empty language and  $R$  is non empty language and

$$\text{because } L(R) = L(S) \cup L(T), \text{ and } L(S) \neq \emptyset, L(T) \neq \emptyset \Rightarrow L(R) \neq \emptyset$$

**Proof.** (cont.3/4)

Case 4:  $R = S \cdot T$  for some regular expressions  $S$  and  $T$ .

- בסיסית  $S$  ו- $T$  הן subexpr של  $R$ ,  $S \neq \emptyset$  ו- $T \neq \emptyset$

- נניח  $L(S) \neq \emptyset$ ,  $L(T) \neq \emptyset$

- נניח  $L(R) = L(S) \cdot L(T)$

נניח  $x \in L(S)$  ו- $y \in L(T)$

אז  $xy \in L(S) \cdot L(T)$  ולכן

$L(R) \neq \emptyset$

**Proof.** (cont.4/4)

Case 5:  $R = S^*$  for some regular expression  $S$ .

မည်သည့်  $\varepsilon \in S^*$  ,

လည်း  $L(R) \neq \emptyset$  .

လည်း နံပါတ်တိုင်း regular expr  $R$  အတွက်  $\emptyset$  ,  $L(R) \neq \emptyset$  . 

**Proof.** (cont.4/4)

Case 5:  $R = S^*$  for some regular expression  $S$ .

In every case, the language  $L(R)$  is non-empty.



### Lemma 3

Every non-empty regular language is represented by a regular expression that does not use the symbol  $\emptyset$ .

Given regular language  $(R)$  <sup>(set)</sup> for

I.H. Given non subset  $r \in R$ .

ex  $(\emptyset \cup \{10\})^* \cup (\emptyset \cdot \{1\}^*)^*$

$\downarrow$   
 $(\emptyset + 10)^* + (\emptyset \cdot 1^*)^*$   
 $\emptyset$

$(10)^* + \{\epsilon\}$

### Lemma 3

*Every non-empty regular language is represented by a regular expression that does not use the symbol  $\emptyset$ .*

Let  $R$  be a regular expression.

### Lemma 3

*Every non-empty regular language is represented by a regular expression that does not use the symbol  $\emptyset$ .*

Let  $R$  be a regular expression. We prove that if  $L(R) \neq \emptyset$ , then there exists a regular expression  $R'$  such that  $L(R) = L(R')$  and  $R'$  does not contain  $\emptyset$ .

### Lemma 3

*Every non-empty regular language is represented by a regular expression that does not use the symbol  $\emptyset$ .*

Let  $R$  be a regular expression. We prove that if  $L(R) \neq \emptyset$ , then there exists a regular expression  $R'$  such that  $L(R) = L(R')$  and  $R'$  does not contain  $\emptyset$ .

We prove by induction. What should the induction hypothesis be?

I.H. សំរាប់ ៗ subexpression  $S$  ក្នុង  $R$ , if  $L(S) \neq \emptyset$ ,  
ទើប  $S'$  រក  $L(S') = L(S)$  ហើយ  $S'$  មិនមាន  $\emptyset$ .

**I.H.:** For every subexpression  $S$  of  $R$ , if  $L(S) \neq \emptyset$ , there exists an  $\emptyset$ -free regular expression  $S'$  such that  $L(S) = L(S')$ .

**I.H.:** For every subexpression  $S$  of  $R$ , if  $L(S) \neq \emptyset$ , there exists an  $\emptyset$ -free regular expression  $S'$  such that  $L(S) = L(S')$ .

What are the cases that we have to consider?

$$\triangleright R = \emptyset$$

$$\triangleright R = x \quad \text{or} \quad R = S \mid L$$

$$\triangleright R = S + T$$

$$\triangleright R = S \cdot T$$

$$\triangleright R = S^*$$

**I.H.:** For every subexpression  $S$  of  $R$ , if  $L(S) \neq \emptyset$ , there exists an  $\emptyset$ -free regular expression  $S'$  such that  $L(S) = L(S')$ .

What are the cases that we have to consider?

- ▶  $R = \emptyset$ , when  $L(R) \neq \emptyset$ , no.
- ▶  $R$  is a single string. ✓
- ▶  $R = \underline{S} + \underline{T}$  for some regular expressions  $\underline{S}$  and  $\underline{T}$ .
- ▶  $R = S \cdot T$  for some regular expressions  $S$  and  $T$ .
- ▶  $R = S^*$  for some regular expression  $S$ .

Case 1:  $L(S) \neq \emptyset, L(T) \neq \emptyset$   
 if  $S' \text{ bbn; } T' \text{ n' / n' } \neq \emptyset$   
 bbn:  $L(S') = L(S)$  &  
 $L(T) = L(T)$   
 bn  $R' = S' + T'$ .

Case 2:  $L(S) = \emptyset, L(T) = \emptyset$ ;  
 q: n' n'  $L(R) = L(S) \cup L(T) = \emptyset$  n'  
 n' n' n' n' (q: n' n' n')

Case 3:  $L(S) = \emptyset$  bbn:  $L(T) \neq \emptyset$  n'

$$L = \{10, 11, 111, 100\}$$

$$L^R = \{01, 11, 111, 001\}$$

(E-ex1-6) For string  $w$ , the reversal  $w^R$  is defined recursively as follows:

$$w^R = \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ \underline{x^R} \cdot a & \text{if } w = ax \text{ for some symbol } a \text{ and some string } x \end{cases}$$

For a language  $L$ , the reversal of  $L$  is defined as

$$\underline{L^R} = \{\underline{w^R} \mid \underline{w} \in L\}.$$

$$L = \{\text{hello, world, good}\}$$

$$L^R = \{\text{olleh, dlrow, doog}\}$$

You may assume the following facts.

- ▶  $L^* \cdot L^* = L^*$  for every language  $L$ .
- ▶  $(w^R)^R = w$  for every string  $w$ .
- ▶  $(x \cdot y)^R = y^R \cdot x^R$  for all strings  $x$  and  $y$ .

$$\begin{aligned} (\text{love!you})^R &= \underline{\text{you}^R} \cdot \underline{\text{love}^R} \\ &= \text{uoY evo!} \end{aligned}$$



~~Prove that  $(L^R)^* \subseteq (L^*)^R$ .~~

$$|x \cdot y| = |x| + |y|$$

Given string  $w$ ,  $|w| = |w^R|$ .

Proof: Given string  $w$  for

I.H. Given any substring  $x$  for  $w$  if  $|x| < |w|$ ,  $|x| = |x^R|$ .

Case:  $w = \epsilon$ , trivially reverse  $w^R = \epsilon$  and  $|w| = 0 = |w^R|$  min. ✓

Case:  $w = a \cdot x$  for  $x$  for string

reverse  $R$ ,  $w^R = (a \cdot x)^R = x^R \cdot a$

$$|w| = |a \cdot x| = 1 + |x|$$

$$= 1 + |x^R|$$

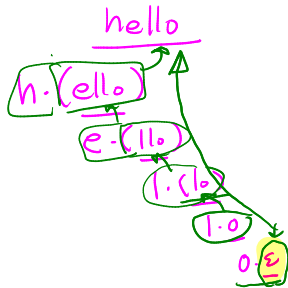
$$= |x^R| + 1$$

use reverse length

in I.H.

$$= |x^R \cdot a| = |w^R|$$

concat  $|a|=1$  reverse



Given any string  $w$