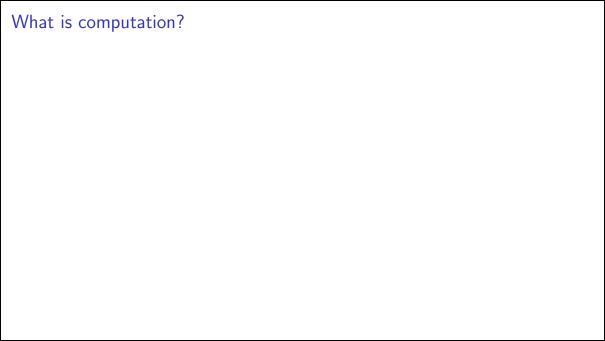
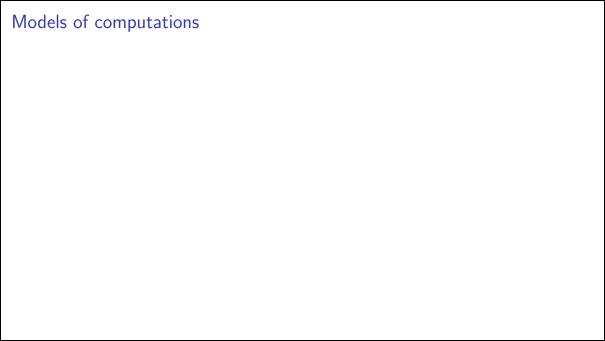
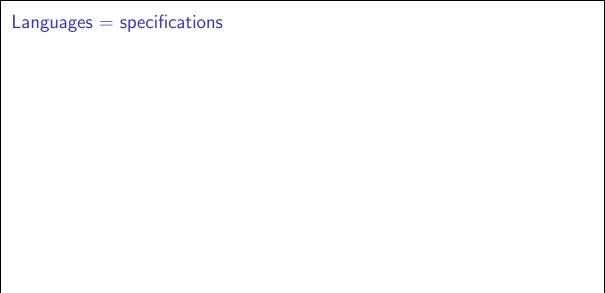
# 01204211 Discrete Mathematics Lecture 7a: Languages and regular expressions

Jittat Fakcharoenphol

August 21, 2023







## 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,  $\Sigma = \{0,1,\ldots,9\}$ ; for English string  $\Sigma = \{a,b,\ldots,z\}$ .) 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
- $ightharpoonup a \cdot x$  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

$$|w| = \left\{ \begin{array}{ll} 0 & \text{when } w = \varepsilon \\ 1 + |x| & \text{when } w = a \cdot x \end{array} \right.$$

#### Concatenation

For strings w and z, the concatenation  $w \bullet z$  is defiend recursively as

$$w \bullet z = \left\{ \begin{array}{ll} z & \text{when } w = \varepsilon \\ a \cdot (x \bullet z) & \text{when } w = a \cdot x \end{array} \right.$$

## Review: proving facts about strings

## Lemma 1

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

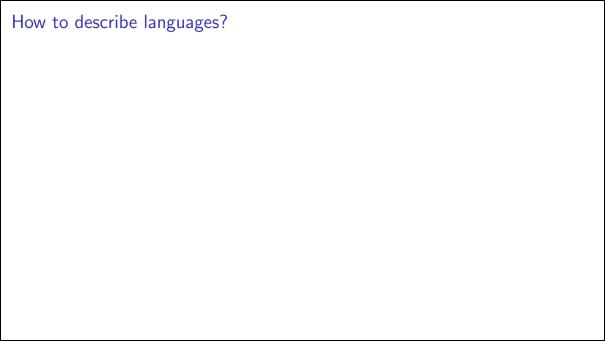
## Proof.

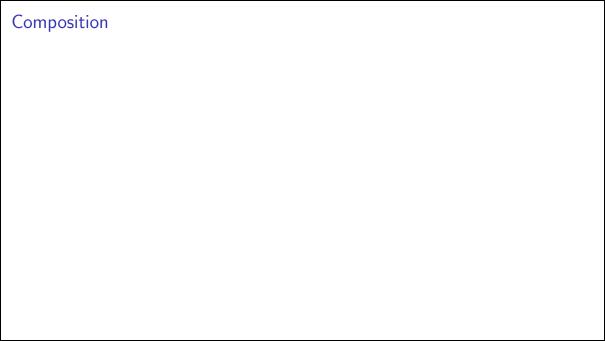
## Formal languages

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

Careful...

These are different languages:  $\emptyset, \{\varepsilon\}$  And  $\varepsilon$  is not a language.





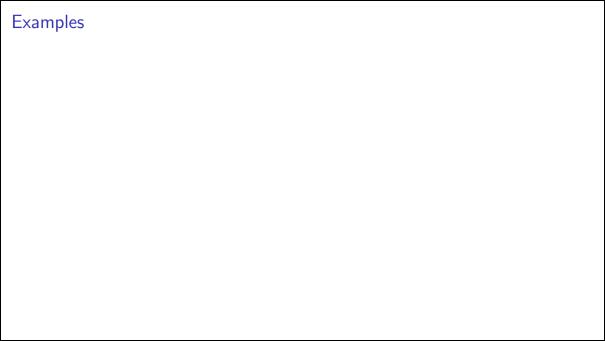
# Combining languages

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

- $\blacktriangleright \ \, \text{Basic set operations:} \ \, A \cup B \text{, } A \cap B \text{, } \bar{A} = \Sigma^* \setminus A.$
- ► Concatenation:  $A \bullet B$ .

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

Also 
$$A^+ = A \bullet A^*$$

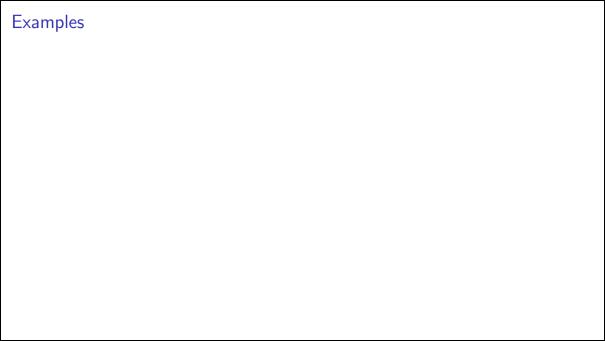


## Regular languages

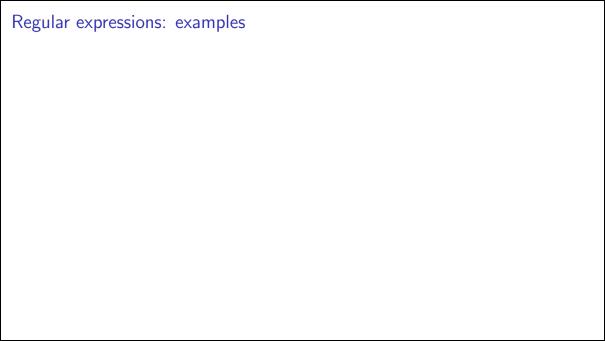
### Definition: regular languages

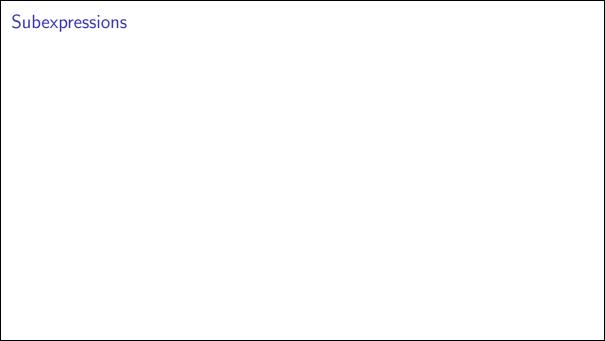
A language L is regular if and only if it satisfies one of the following conditions:

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

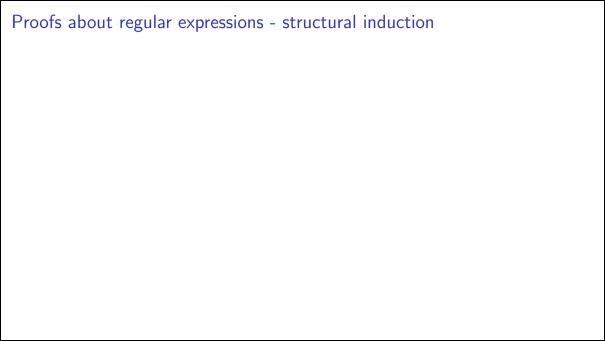












#### Lemma 2

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:  $R = \emptyset$ .

Case 2: R is a single string.

# **Proof.** (cont.2/4) Case 3: R = S + T for some regular expressions S and T.

# **Proof.** (cont.3/4)Case 4: $R = S \bullet T$ for some regular expressions S and T.

**Proof.** (cont.4/4) Case 5:  $R = S^*$  for some regular expression S.

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

#### 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.:** 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?

- $ightharpoonup R = \emptyset$
- ightharpoonup R is a single string.
- ightharpoonup R = S + T for some regular expressions S and T.
- ightharpoonup R = S ullet T for some regular expressions S and T.
- $ightharpoonup R = S^*$  for some regular expression S.