



# ECE-374-B: Lecture 1 - Logistics and Strings/Languages

---

Lecturer: Nickvash Kani

January 17, 2023

University of Illinois at Urbana Champaign

# Course Administration

---

# Instructional Staff

- **Instructor:**
  - Nickvash Kani
- **Teaching Assistants:**
  - Sung Woo Jeon
  - Nicholas Bampton
  - Gautham Varma
  - Sindhu Vydana
  - Sandhya Perumenki
- **Office hours:** TBD, See course webpage
- **Contacting us:** Use private notes on Piazza to reach course staff. Direct email only for sensitive or confidential information.

## Section A vs B

This semester, the two sections will be run completely **independently**.

- Different lectures.
- Different homeworks, quizzes, exams.
- Different grading policies.

## Section A vs B

This semester, the two sections will be run completely **independently**.

- Different lectures.
- Different homeworks, quizzes, exams.
- Different grading policies.

Section B will be in-person only. Recordings will be attempted but not guaranteed.

## Online resources

- **Webpage:** General information, announcements, homeworks, quizzes, course policies <https://ecealgo.com>
- **Submission(Gradescope):** Written homework submission and grading, regrade requests. Exams will be uploaded there as well.
- **Communication(Piazza):** Announcements, online questions and discussion, contacting course staff (via private notes)
- **Gradebook (Canvas):** Announcements, online questions and discussion, contacting course staff (via private notes)

See course webpage for links

**Important:** check Piazza/course web page at least once each day

# Grading Policy

---



# Grading Policy: Overview

- **Homeworks:** 24%
- There will be approximately 9 HWs with 3 questions each.
- Homeworks need to be submitted on Gradescope.
- Only the top 21 question grades will be considered for final grade calculation

# Grading Policy: Overview

- Homeworks: 24%
- Midterm/Final exams: 75% ( $3 \times 25\%$ )

Exam dates:

- Midterm 1: Thurs, Feb 16, 12:30pm–1:45pm
- Midterm 2: Tues, Apr 04, 12:30pm–1:45pm
- Midterm 3: Thurs, Apr 27, 12:30pm–1:45pm
- Final: TBD

One exam will be dropped Drop policies should eliminate need for conflict exams.

## Grading Policy: Overview

- Homeworks: 24%
- Midterm/Final exams: 75% ( $3 \times 25\%$ )
- ICES Form: 1%

Piazza is crucial for improving the course experience but only 20-35% participate.

Will ask to upload a screenshot that confirms your completion of the ICES eval before grade estimates.

## Discussion Sessions/Labs

- 50min problem solving session led by TAs
- Two times a week
- Go to your assigned discussion section
- Bring pen and paper!

## Discussion Sessions/Labs

- 50min problem solving session led by TAs
- Two times a week
- Go to your assigned discussion section
- Bring pen and paper!

Discussion sections will have questions that appear on the homework. If, you skip, you're just making more work for yourself later.

## Advice

- Attend lectures, please ask plenty of questions.
- Attend discussion sessions.
- Don't skip homework and don't copy homework solutions. Each of you should think about all the problems on the homework - do not divide and conquer.
- Start homework early! Your mind needs time to think.
- Study regularly and keep up with the course.
- This is a course on problem solving. Solve as many as you can! Books/notes have plenty.
- This is also a course on providing rigorous proofs of correctness. Refresh your 173 background on proofs.
- Ask for help promptly. Make use of office hours/Piazza.

## Miscellaneous

Please contact instructors if you need special accommodations.

Lectures are being taped (hopefully). The issue is that these recording systems are prone to failure. While I make no promises, I will try my best to record the lectures.

See course webpage for additional information.

## Over-arching course questions

---



# High-Level Questions

This course introduces three distinct fields of computer science research:

- Computational complexity.
  - Given infinite time and a certain machine, is it possible to solve a given problem.
- Algorithms
  - Given a deterministic Turing machine, how fast can we solve certain problems.
- Limits of computation.
  - Are there tasks that our computers cannot do and how do we identify these problems?

## Why not just focus on Algorithms?

When someone asks you, "How fast can you compute problem  $X$ ", they are actually asking:

- Is  $X$  solvable using the deterministic Turing machines we have at our disposal?
- If it is solvable, can we find the solution efficiently (in poly-time)?
- If it is solvable but we don't have a poly time solution, what problem(s) is it most similar too?

# Course Structure

Course divided into three parts:

- Basic automata theory: finite state machines, regular languages, hint of context free languages/grammars, Turing Machines
- Algorithms and algorithm design techniques
- Undecidability and NP-Completeness, reductions to prove intractability of problems

Week	Number Lectures	Practical Lab	Thematic Lectures	Practical Lab
Aug 20 (2)	Adaptation of cells and nerve growth Introduction to Neurobiology & Histology Sternik/Volkov, Lec. 1-5 (30 mins, video, online)	Cellular and molecular mechanisms of synaptic transmission Sternik/Volkov, Lec. 6 (30 mins, video, online)	Neurogenesis and neural migration Sternik/Volkov, Lec. 7 (30 mins, video, online)	Basal ganglia Sternik/Volkov, Lec. 8 (30 mins, video, online)
Aug 30 - Sep 1 (2)	EPSPs, inhibition, disinhibition, circuit properties Sternik/Volkov, Lec. 9-11 (30 mins, video, online)	Cellular mechanisms of synaptic plasticity Sternik/Volkov, Lec. 12 (30 mins, video, online)	Neocortical circuitry Sternik/Volkov, Lec. 13 (30 mins, video, online)	Laminar organization of the neocortex Sternik/Volkov, Lec. 14 (30 mins, video, online)
Sep 6-7 (2)	Regulation of EPSPs, EPSPs and motor responses Sternik/Volkov, Lec. 15-16 (30 mins, video, online)	Neurotransmitter release Sternik/Volkov, Lec. 17 (30 mins, video, online)	Spinal cord motor system Sternik/Volkov, Lec. 18 (30 mins, video, online)	Spinal Neuroanatomy Sternik/Volkov, Lec. 19 (30 mins, video, online)
Sept 13 (1)	Cerebral hemodynamics and cognition Sternik/Volkov, Lec. 20 (30 mins, video, online)	Cerebral blood flow Sternik/Volkov, Lec. 21 (30 mins, video, online)	Neural basis of cognition Sternik/Volkov, Lec. 22 (30 mins, video, online)	Neural Mechanisms of Cognition Sternik/Volkov, Lec. 23 (30 mins, video, online)
Sept 20 (2)	Optical Neural Networks Sternik/Volkov, Lec. 24 (30 mins, video, online)	Optical Neural Networks Sternik/Volkov, Lec. 25 (30 mins, video, online)	Machine Learning for Neuroscience Sternik/Volkov, Lec. 26 (30 mins, video, online)	No instruction
Sept 20 (2)	Behavioral Neuroscience Sternik/Volkov, Lec. 27 (30 mins, video, online)	Behavioral Neuroscience Sternik/Volkov, Lec. 28 (30 mins, video, online)	Neural basis of emotion, motivation, and decision making Sternik/Volkov, Lec. 29 (30 mins, video, online)	Emotion and Decision Making Sternik/Volkov, Lec. 30 (30 mins, video, online)
Oct 7 (1)	Neurobiology of Learning Sternik/Volkov, Lec. 31 (30 mins, video, online)	Neurobiology of Learning Sternik/Volkov, Lec. 32 (30 mins, video, online)	Neural basis of learning Sternik/Volkov, Lec. 33 (30 mins, video, online)	Learning Sternik/Volkov, Lec. 34 (30 mins, video, online)
Oct 14 (1)	Motor Dynamics, neuromuscular transmission Sternik/Volkov, Lec. 35 (30 mins, video, online)	Motor Dynamics, neuromuscular transmission Sternik/Volkov, Lec. 36 (30 mins, video, online)	Neural basis of movement Sternik/Volkov, Lec. 37 (30 mins, video, online)	Motor Dynamics Sternik/Volkov, Lec. 38 (30 mins, video, online)
Oct 21 (2)	Visual Cortex, EPSPs, EPSPs and cognition Sternik/Volkov, Lec. 39-40 (30 mins, video, online)	Visual Cortex, EPSPs, EPSPs and cognition Sternik/Volkov, Lec. 41-42 (30 mins, video, online)	Neural basis of vision Sternik/Volkov, Lec. 43 (30 mins, video, online)	Visual Cortex Sternik/Volkov, Lec. 44 (30 mins, video, online)
Oct 28 (2)	Behavioral Neuroscience, Neuroanatomy of Cognition Sternik/Volkov, Lec. 45-46 (30 mins, video, online)	Behavioral Neuroscience, Neuroanatomy of Cognition Sternik/Volkov, Lec. 47-48 (30 mins, video, online)	Neural basis of cognition Sternik/Volkov, Lec. 49 (30 mins, video, online)	Neural Mechanisms of Cognition Sternik/Volkov, Lec. 50 (30 mins, video, online)
Nov 4 (1)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 51 (30 mins, video, online)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 52 (30 mins, video, online)	Neural basis of movement Sternik/Volkov, Lec. 53 (30 mins, video, online)	Motor Dynamics Sternik/Volkov, Lec. 54 (30 mins, video, online)
Nov 11 (2)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 55-56 (30 mins, video, online)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 57-58 (30 mins, video, online)	Neural basis of movement Sternik/Volkov, Lec. 59 (30 mins, video, online)	Motor Dynamics Sternik/Volkov, Lec. 60 (30 mins, video, online)
Nov 18 (2)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 61-62 (30 mins, video, online)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 63-64 (30 mins, video, online)	Neural basis of movement Sternik/Volkov, Lec. 65 (30 mins, video, online)	Motor Dynamics Sternik/Volkov, Lec. 66 (30 mins, video, online)
Nov 25 (2)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 67-68 (30 mins, video, online)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 69-70 (30 mins, video, online)	Neural basis of movement Sternik/Volkov, Lec. 71 (30 mins, video, online)	Motor Dynamics Sternik/Volkov, Lec. 72 (30 mins, video, online)
Dec 2 (2)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 73-74 (30 mins, video, online)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 75-76 (30 mins, video, online)	Neural basis of movement Sternik/Volkov, Lec. 77 (30 mins, video, online)	Motor Dynamics Sternik/Volkov, Lec. 78 (30 mins, video, online)
Dec 9 (2)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 79-80 (30 mins, video, online)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 81-82 (30 mins, video, online)	Neural basis of movement Sternik/Volkov, Lec. 83 (30 mins, video, online)	Motor Dynamics Sternik/Volkov, Lec. 84 (30 mins, video, online)
Dec 16 (2)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 85-86 (30 mins, video, online)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 87-88 (30 mins, video, online)	Neural basis of movement Sternik/Volkov, Lec. 89 (30 mins, video, online)	Motor Dynamics Sternik/Volkov, Lec. 90 (30 mins, video, online)
Dec 23 (2)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 91-92 (30 mins, video, online)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 93-94 (30 mins, video, online)	Neural basis of movement Sternik/Volkov, Lec. 95 (30 mins, video, online)	Motor Dynamics Sternik/Volkov, Lec. 96 (30 mins, video, online)
Dec 30 (2)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 97-98 (30 mins, video, online)	Motor Dynamics, Neuromuscular Transmission Sternik/Volkov, Lec. 99-100 (30 mins, video, online)	Neural basis of movement Sternik/Volkov, Lec. 101 (30 mins, video, online)	Motor Dynamics Sternik/Volkov, Lec. 102 (30 mins, video, online)

# Goals

- Algorithmic thinking
- Learn/remember some basic tricks, algorithms, problems, ideas
- Understand/appreciate limits of computation (intractability)
- Appreciate the importance of algorithms in computer science and beyond (engineering, mathematics, natural sciences, social sciences, ...)

## **Formal languages and complexity (The Blue Weeks!)**

---

# Why Languages?

First 5 weeks devoted to language theory.

# Why Languages?

First 5 weeks devoted to language theory.

But why study languages?

# Multiplying Numbers

Consider the following problem:

**Problem** Given two  $n$ -digit numbers  $x$  and  $y$ , compute their product.

## Grade School Multiplication

Compute “partial product” by multiplying each digit of  $y$  with  $x$  and adding the partial products.

$$\begin{array}{r} 3141 \\ \times 2718 \\ \hline 25128 \\ 3141\phantom{0} \\ 21987\phantom{00} \\ 6282\phantom{000} \\ \hline 8537238 \end{array}$$



## Time analysis of grade school multiplication

- Each partial product:  $\Theta(n)$  time
- Number of partial products:  $\leq n$
- Adding partial products:  $n$  additions each  $\Theta(n)$  (Why?)
- Total time:  $\Theta(n^2)$
- Is there a faster way?

# Fast Multiplication

- $O(n^{1.58})$  time [Karatsuba 1960] disproving Kolmogorov's belief that  $\Omega(n^2)$  is best possible
- $O(n \log n \log \log n)$  [Schönhage-Strassen 1971].  
**Conjecture:**  $O(n \log n)$  time possible
- $O(n \log n \cdot 2^{O(\log^* n)})$  time [Furer 2008]
- $O(n \log n)$  [Harvey-van der Hoeven 2019]

Can we achieve  $O(n)$ ? No lower bound beyond trivial one!

## Equivalent Complexity

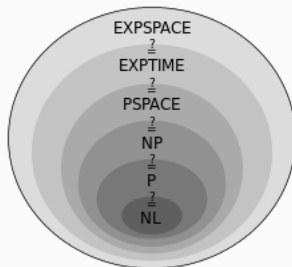
Does this mean multiplication is as complex as another problem that has a  $O(n \log n)$  algorithm like sorting/QuickSort?

## Equivalent Complexity

Does this mean multiplication is as complex as another problem that has a  $O(n \log n)$  algorithm like sorting/QuickSort? How do we compare? The two problems have:

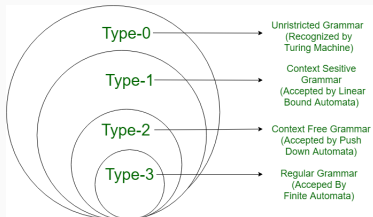
- Different inputs (two numbers vs n-element array)
- Different outputs (a number vs n-element array)
- Different entropy characteristics (from a information theory perspective)

An algorithm has a runtime complexity.



# Languages, Problems and Algorithms ... oh my! III

A problem has a complexity class!



Problems do not have run-time since a problem  $\neq$  the algorithm used to solve it. *Complexity classes are defined differently.*

How do we compare problems? What if we just want to know if a problem is "computable".

## Definition

1. An **algorithm** is a step-by-step way to solve a problem.
2. A **problem** is some question that we'd like answered given some input. It should be a decision problem of the form "Does a given input fulfill property X."
3. A **Language** is a set of strings. Given a alphabet,  $\Sigma$  a language is a subset of  $\Sigma^*$

## Definition

1. An **algorithm** is a step-by-step way to solve a problem.
2. A **problem** is some question that we'd like answered given some input. It should be a decision problem of the form "Does a given input fulfill property X."
3. A **Language** is a set of strings. Given a alphabet,  $\Sigma$  a language is a subset of  $\Sigma^*$  A language is a formal realization of this problem. For problem X, the corresponding language is:

$$L = \{w \mid w \text{ is the encoding of an input } y \text{ to problem } X \text{ and the answer to input } y \text{ for a problem } X \text{ is "YES"} \}$$

A decision problem X is "YES" is the string is in the language.



# Language of multiplication

How do we define the multiplication problem as a language?

Define  $L$  as language where inputs are separated by comma and output is separated by  $|$ .

Machine accepts a  $x*y=z$  if " $x*y|z$ " is in  $L$ . Rejects otherwise.

# Language of multiplication

How do we define the multiplication problem as a language?

Define L as language where inputs are separated by comma and output is separated by |.

Machine accepts a  $x*y=z$  if " $x*y|z$ " is in L. Rejects otherwise.

$$L_{MULT2} = \left\{ \begin{array}{ccc} 1 \times 1|1, & 1 \times 2|2, & 1 \times 3|3, \dots \\ 2 \times 1|2, & 2 \times 2|4, & 2 \times 3|6, \dots \\ \vdots & \vdots & \vdots \\ n \times 1|n, & n \times 2|2n, & n \times 3|3n, \dots \end{array} \right\} \quad (1)$$

## Language of sorting

We do the same thing for sorting.

Define  $L$  as language where inputs are separated by comma and output is separated by  $|$ .

Machine accepts a  $[i_1, i_2, \dots] = \text{sort}(\{i_1, i_2, \dots\})$  if " $x[]|z[]$ " is in  $L$ .  
Rejects otherwise.

## Language of sorting

We do the same thing for sorting.

Define L as language where inputs are separated by comma and output is separated by |.

Machine accepts a  $[i_1, i_2, \dots] = \text{sort}(\{i_1, i_2, \dots\})$  if " $x[]|z[]$ " is in L.  
Rejects otherwise.

$$L_{\text{Sort2}} = \left\{ \begin{array}{ccc} 1, 1|1, 1 & 1, 2|1, 2 & 1, 3|1, 3, \dots \\ 2, 1|1, 2, & 2, 2|2, 2, & 2, 3|2, 3, \dots \\ \vdots & \vdots & \vdots \\ n, 1|1, n, & n, 2|2, n, & n, 3|3, n, \dots \end{array} \right\} \quad (2)$$

## Language of sorting

We do the same thing for sorting.

Define  $L$  as language where inputs are separated by comma and output is separated by  $|$ .

Machine accepts a  $[i_1, i_2, \dots] = \text{sort}(\{i_1, i_2, \dots\})$  if " $x[]|z[]$ " is in  $L$ .  
Rejects otherwise.

$$L_{\text{Sort2}} = \left\{ \begin{array}{ccc} 1, 1|1, 1 & 1, 2|1, 2 & 1, 3|1, 3, \dots \\ 2, 1|1, 2, & 2, 2|2, 2, & 2, 3|2, 3, \dots \\ \vdots & \vdots & \vdots \\ n, 1|1, n, & n, 2|2, n, & n, 3|3, n, \dots \end{array} \right\} \quad (2)$$

If the same type of machine can recognize both languages, then that gives us an upperbound to their hardness.

**How do we formulate languages?**

---

# Strings

---

# Alphabet

An **alphabet** is a **finite** set of symbols.

Examples of alphabets:

- $\Sigma = \{0, 1\},$
- $\Sigma = \{a, b, c, \dots, z\},$
- ASCII.
- UTF8.
- $\Sigma = \{\langle \text{moveforward} \rangle, \langle \text{moveback} \rangle, \langle \text{moveleft} \rangle, \langle \text{moveright} \rangle\}$



# String Definition

## Definition

1. A **string/word** over  $\Sigma$  is a **finite sequence** of symbols over  $\Sigma$ .  
For example, '0101001', '*string*', ' $\langle \text{moveback} \rangle \langle \text{rotate90} \rangle$ '
2.  $x \cdot y \equiv xy$  is the concatenation of two strings
3. The **length** of a string  $w$  (denoted by  $|w|$ ) is the number of symbols in  $w$ . For example,  $|101| = 3$ ,  $|\epsilon| = 0$
4. For integer  $n \geq 0$ ,  $\Sigma^n$  is set of all strings over  $\Sigma$  of length  $n$ .  
 $\Sigma^*$  is the set of all strings over  $\Sigma$ .
5.  $\Sigma^*$  set of all strings of all lengths including empty string.

**Question:**  $\{ 'a', 'c' \}^* =$

# Emptiness

- $\epsilon$  is a **string** containing no symbols. It is not a set
- $\{\epsilon\}$  is a **set** containing one string: the empty string. It is a set, not a string.
- $\emptyset$  is the **empty set**. It contains no strings.

**Question:** What is  $\{\emptyset\}$

# Concatenation and properties

- If  $x$  and  $y$  are strings then  $xy$  denotes their concatenation.
- **Concatenation** defined recursively :
  - $xy = y$  if  $x = \epsilon$
  - $xy = a(wy)$  if  $x = aw$
- $xy$  sometimes written as  $x \bullet y$ .
- concatenation is **associative**:  $(uv)w = u(vw)$  hence write  $uvw \equiv (uv)w = u(vw)$
- **not** commutative:  $uv$  not necessarily equal to  $vu$
- The identity element is the empty string  $\epsilon$ :

$$\epsilon u = u\epsilon = u.$$

## Definition

$v$  is **substring** of  $w \iff$  there exist strings  $x, y$  such that  $w = xvy$ .

- If  $x = \epsilon$  then  $v$  is a **prefix** of  $w$
- If  $y = \epsilon$  then  $v$  is a **suffix** of  $w$

# Subsequence

A subsequence of a string  $w[1\dots n]$  is either a subsequence of  $w[2\dots n]$  or  $w[1]$  followed by a subsequence of  $w[2\dots n]$ .

## Example

*kapa* is a subsequence of *knapsack*

# Subsequence

A subsequence of a string  $w[1\dots n]$  is either a subsequence of  $w[2\dots n]$  or  $w[1]$  followed by a subsequence of  $w[2\dots n]$ .

## Example

*kapa* is a subsequence of *knapsack*

**Question:** How many sub-sequences are there in a string  $|w| = 5$ ?

### Definition

If  $w$  is a string then  $w^n$  is defined inductively as follows:

$$w^n = \epsilon \text{ if } n = 0$$

$$w^n = ww^{n-1} \text{ if } n > 0$$

**Question:**  $(\textit{blah})^3 =$ .

## Rapid-fire questions -strings

Answer the following questions taking  $\Sigma = \{0, 1\}$ .

1. What is  $\Sigma^0$ ?
2. How many elements are there in  $\Sigma^n$ ?
3. If  $|u| = 2$  and  $|v| = 3$  then what is  $|u \bullet v|$ ?
4. Let  $u$  be an arbitrary string in  $\Sigma^*$ . What is  $\epsilon u$ ? What is  $u \epsilon$ ?



# Languages

---

**Definition**

A **language**  $L$  is a set of strings over  $\Sigma$ . In other words  $L \subseteq \Sigma^*$ .

## Definition

A **language**  $L$  is a set of strings over  $\Sigma$ . In other words  $L \subseteq \Sigma^*$ .

Standard set operations apply to languages.

- For languages  $A, B$  the **concatenation** of  $A, B$  is  $AB = \{xy \mid x \in A, y \in B\}$ .
- For languages  $A, B$ , their **union** is  $A \cup B$ , **intersection** is  $A \cap B$ , and **difference** is  $A \setminus B$  (also written as  $A - B$ ).
- For language  $A \subseteq \Sigma^*$  the **complement** of  $A$  is  $\bar{A} = \Sigma^* \setminus A$ .

# Set Concatenation

## Definition

Given two sets  $X$  and  $Y$  of strings (over some common alphabet  $\Sigma$ ) the **concatenation** of  $X$  and  $Y$  is

$$XY = \{xy \mid x \in X, y \in Y\} \quad (3)$$

**Question:**  $X = \{fido, rover, spot\}$ ,  $Y = \{fluffy, tabby\} \implies XY = .$

## Definition

1.  $\Sigma^n$  is the set of all strings of length  $n$ . Defined inductively:  
 $\Sigma^n = \{\epsilon\}$  if  $n = 0$   
 $\Sigma^n = \Sigma\Sigma^{n-1}$  if  $n > 0$
2.  $\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$  is the set of all finite length strings
3.  $\Sigma^+ = \bigcup_{n \geq 1} \Sigma^n$  is the set of non-empty strings.

## Definition

A **language**  $L$  is a set of strings over  $\Sigma$ . In other words  $L \subseteq \Sigma^*$ .

**Question:** Does  $\Sigma^*$  have strings of infinite length?

## Rapid-Fire questions - Languages

### Problem

Consider languages over  $\Sigma = \{0, 1\}$ .

1. What is  $\emptyset^0$ ?
2. If  $|L| = 2$ , then what is  $|L^4|$ ?
3. What is  $\emptyset^*$ ,  $\{\epsilon\}^*$ ,  $\epsilon^*$ ?
4. For what  $L$  is  $L^*$  finite?
5. What is  $\emptyset^+$ ?
6. What is  $\{\epsilon\}^+$ ,  $\epsilon^+$ ?

# Terminology Review

Let's review what we learned.

- A **character**( $a, b, c, x$ ) is a unit of information represented by a symbol: (letters, digits, whitespace)
- A **alphabet**( $\Sigma$ ) is a set of characters
- A **string**( $w$ ) is a sequence of characters
- A **language**( $A, B, C, L$ ) is a set of strings

# Terminology Review

Let's review what we learned.

- A **character**( $a, b, c, x$ ) is a unit of information represented by a symbol: (letters, digits, whitespace)
- A **alphabet**( $\Sigma$ ) is a set of characters
- A **string**( $w$ ) is a sequence of characters
- A **language**( $A, B, C, L$ ) is a set of strings
- A **grammar**( $G$ ) is a set of rules that defines the strings that belong to a language



# Languages: easiest, easy, hard, really hard, really<sup>n</sup> hard

- Regular languages.
  - Regular expressions.
  - DFA: Deterministic finite automata.
  - NFA: Non-deterministic finite automata.
  - Languages that are not regular.
- Context free languages (stack).
- Turing machines: Decidable languages.
- TM Undecidable/unrecognizable languages (halting theorem).

