Introduction to CS Theory (Honors & Traditional):
- formalization of computation
- various models of computation (increasing difficulty/power)
- what can / cannot be done?

Why a theory course?
- relevant to practice (*grammars* for programming languages, *finite automata & regular expressions* for pattern matching of strings, *NP-completeness* to determine required time complexity – e.g. for cryptography)
- problem solving skills independent of current technology (specific programming languages, etc.), ability to express ideas clearly, succinctly, and correctly
Welcome to Honors Intro to CS Theory

Honors vs. Traditional Intro to CS Theory
- Book: M. Sipser, Introduction to the Theory of Computation
- faster speed through introductory topics and simpler models, more time for advanced topics (decidability, complexity)
- more challenging and non-traditional homeworks

You should:
- be very comfortable with discrete math
- have fun (a course full of puzzles! 😊)
Introduction

Automata Theory
- mathematical models of computation

Computability Theory
- what can be computed?

Complexity Theory
- which problems are computationally hard / easy?

Need math background
- review Chapter 0
- discrete math quiz next class
Strings and Languages

**Alphabet** - non-empty finite set of **symbols**, typically denoted by $\Sigma$ or $\Gamma$, e.g.

\[ \Sigma_1 = \{ 0,1 \}, \quad \Sigma_2 = \{ a,b,c,d \}, \quad \Gamma = \{ \#,\$,0,1,2 \} \]

**String** over an alphabet - a finite sequence of symbols from the alphabet, e.g.

\[ w_1 = 00101 \text{ over } \Sigma_1, \quad w_2 = \text{badcab} \text{ over } \Sigma_2 \]

The **length** of a string $w$ over $\Sigma$ (the number of symbols in $w$) is denoted $|w|$.

The string with no symbols is called the **empty string** and denoted $\epsilon$. [in other books]
Operations on strings (let \( w = w_1w_2\ldots w_n \)): \(|w| = n \\
- reverse: \quad w^R = w_nw_{n-1}\ldots w_1 \\
- substring: \quad w_iw_{i+1}\ldots w_j \\
- concatenation of \( w \) with a string \( z = z_1z_2\ldots z_m \): \(|z| = m \\
\quad wz = w_1w_2\ldots w_nz_1z_2\ldots z_m \\
- \( w^k \) means concatenation of \( k \) copies of \( w \) \\
- lexicographic ordering of strings: first by length, then "alphabetically," e.g for \( \Sigma = \{0,1\} \):

\[ \epsilon,0,1,00,01,10,11,000,\ldots \]

strings over \( \{0,1\} \) that have 1 in the second place:

\[ 01, 10, 001, 1001, 0100, 1010, \ldots \]
Strings and Languages

Language: a set of strings over an alphabet $\Sigma$, e.g.

$\{ a, ab, bab \}$, $\emptyset$, $\{ \varepsilon \}$,

$\{ w \text{ over } \{0,1\} \mid w \text{ contains more 1's than 0's} \}$

Operations on languages:
- typical set operations: $\cup$, $\cap$, etc.
- concatenation: $L_1.L_2 = \{ w_1w_2 \mid w_1 \in L_1, w_2 \in L_2 \}$
- Kleene's star: $L^* = \bigcup_{k=0}^{\infty} L^k$
- reverse: $L^R = \{ w^R \mid w \in L \}$

Note: a language: $L \subseteq \Sigma^*$
Strings and Languages

Language: a set of strings over an alphabet \( \Sigma \), e.g.

\[
\{ a, ab, bab \}, \emptyset, \{ \varepsilon \}, \\
\{ w \text{ over } \{0,1\} \mid w \text{ contains more 1's than 0's} \}
\]

Operations on languages:

- typical set operations: \( \cup, \cap, \text{etc.} \)
- concatenation: \( L_1.L_2 = \{ w_1w_2 \mid w_1 \in L_1, w_2 \in L_2 \} \)
- Kleene's star: \( L^* = \bigcup_{k=0}^{\infty} L^k \)
- reverse: \( L^R = \{ w^R \mid w \in L \} \)

Note: a language: \( L \subseteq \Sigma^* \)
Strings and Languages

Language: a set of strings over an alphabet $\Sigma$, e.g.

$\{ a, ab, bab \}, \emptyset, \{ \varepsilon \},$

$\{ w \text{ over } \{0,1\} \mid w \text{ contains more 1's than 0's} \}$

Operations on languages:
- typical set operations: $\cup$, $\cap$, etc.
- concatenation: $L_1.L_2 = \{ w_1w_2 \mid w_1 \in L_1, w_2 \in L_2 \}$
- Kleene’s star: $L^* = \bigcup_{k=0}^{\infty} L^k$
- reverse: $L^R = \{ w^R \mid w \in L \}$

Note: a language: $L \subseteq \Sigma^*$

[throughout the book, e.g. page 44]