Welcome to Intro to CS Theory

Introduction to CS Theory:
- 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
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
Alphabet - non-empty finite set of symbols, typically denoted by $\Sigma$ or $\Gamma$, e.g.

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

$$\Sigma_3 = \{ 0, 1, 2, 3, \ldots, 255 \}$$

$$\Gamma = \{ a \}$$

$$\exists \Delta = \emptyset$$

$$\Sigma = \{ 0, 1, 2, 3, \ldots \} \quad \text{not finite}$$
**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.

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

$$w_3 = 0 \text{ over } \Sigma_1$$

over $\Sigma_2 \notin \text{ over } \Sigma_2$

$$w_4 = aaaaa \ldots \notin \text{ finite}$$

$\varepsilon \to \text{ the empty string}$
**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 \( |w| \). \( |w_1| = 5 \), \( |w_2| = 6 \), \( |\varepsilon| = 0 \)

The string with no symbols is called the *empty string* and denoted \( \varepsilon \).
Strings and Languages

Operations on strings (let \( w = w_1w_2\ldots w_n \)):

- reverse: \( w^R = w_nw_{n-1}\ldots w_1 \)
- substring: \( w_iw_{i+1}\ldots w_j \)
- concatenation of \( w \) with a string \( z = z_1z_2\ldots z_m \):
  \[
  wz = w_1w_2\ldots w_nz_1z_2\ldots z_m
  \]
- \( w^k \) means concatenation of \( k \) copies of \( w \)
  \[
  w^3 = 012340123401234 = www
  \]
  \[
  w^1 = 01234 = w
  \]
  \[
  w^0 = \epsilon
  \]
  \[
  w^k \cdot w^l = w^{k+l} \quad \text{for some } k, l \geq 0
  \]

\( w = 01234 \)
\( w = 43210 \)
\( w_2w_3\ldots w_5 = 1234 \)
\( z = abcde1 \)
\( wz = 01234abcde1 \)
\( z \cdot \epsilon = z \quad \text{true for any string} \)
\( \epsilon \cdot z = z \)
\( \epsilon \cdot \epsilon = \epsilon \)
Operations on strings (let $w = w_1w_2...w_n$):

- reverse: $w^R = w_nw_{n-1}...w_1$
- substring: $w_iw_{i+1}...w_j$
- concatenation of $w$ with a string $z = z_1z_2...z_m$:
  
  $$wz = w_1w_2...w_nz_1z_2...z_m$$
- $w^k$ means concatenation of $k$ copies of $w$
- lexicographic ordering of strings: first by length, then “alphabetically,” e.g. for $\Sigma = \{0,1\}$:
  
  $$\varepsilon, 0, 1, 00, 01, 10, 11, 000, ...$$

$\Sigma = \{0,1\}$ guarantees to get to every string over $\Sigma$ in a finite amount of time

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

$$\varepsilon, 0, 1, 00, 01, 01, 011, 100, 100, 100, ...$$
Language: a set of strings over an alphabet $\Sigma$, e.g.

$L_1 = \{ a, ab, bab \}$

$L_2 = \emptyset$ \hspace{1cm} $|L_2| = 0$

$L_3 = \{ \varepsilon \}$ \hspace{1cm} $|L_3| = 1$

$L_4 = \{ w \text{ over } \{0,1\} \mid w \text{ contains more 1's than 0's} \}$

$011 \in L_4$

$0110101 \in L_4$

$100 \notin L_4$

$\varepsilon \notin L_4$
Strings and Languages

Operations on languages:
- typical set operations: $\cup$, $\cap$, etc.

$L_1 = \{a, ab, bca\}$
$L_2 = \emptyset$
$L_3 = \{c\}$

$L_1 \cup L_2 = L_1$
$L_1 \cap \{\varepsilon, ba, babc, b\} = \{bab\}$
Operations on languages:

- typical set operations: $\cup$, $\cap$, etc.

- concatenation: $L_1 \cdot L_2 = \{ w_1 w_2 | w_1 \in L_1, w_2 \in L_2 \}$

$L_1 = \{ a, ab, bab \}$
$L_2 = \{ c, dd \}$
$L_1 \cdot L_2 = \{ ac, add, abc, abdd, babc, babdd \}$

$|L_1| = 3$ \hspace{1cm} $|L_2| = 2$ \hspace{1cm} $|L_1 \cdot L_2| = 6$

$L_3 = \{ \epsilon \}$
$L_1 \cdot L_3 = \{ a, ab, bab \} = L_1$

$L_1 = \{ a, ac \}$
$L_2 = \{ \epsilon, c \}$
$L_1 \cdot L_2 = \{ a, ac, acc \}$

$|L_1 \cdot L_2| \neq |L_1| \cdot |L_2|$

$L_1 \cdot \emptyset = \emptyset = \emptyset \cdot L_1$
Strings and Languages

Operations on languages:

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

$L^k = \text{the set of all strings over \{0,1\} of length } k$
$L^0 = \{ \varepsilon \}$
Strings and Languages

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: $L^* = \bigcup_{k=0}^{\infty} L^k$

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

Operations on languages:

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

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