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
Strings and Languages

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

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

- $\Sigma_3 = \{ 0 \}$  
  *ok*
- $\Sigma_4 = \{ 0,1,a,b,c,d \}$  
  *ok*
- $\Sigma_5 = \emptyset$  
  *not ok - empty*
- $\Sigma_6 = \{ 0,1,2,3,4,5,6,\ldots \}$  
  *not ok - infinite*
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.

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

$$w_3 = \$210 \text{ over } \Gamma \quad w_6 = \varepsilon \text{ the empty string}$$

$$w_4 = 0 \text{ over } \Sigma_1 \quad \text{OK, a string}$$
$$w_4 = 0 \text{ over } \Sigma_2 \quad \times, \text{not over } \Sigma_2$$

$$w_5 = \ldots \text{not a string, infinite}$$
**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 $|w|$.  

$$|w_1| = 5, \quad |w_2| = 6, \quad |\varepsilon| = 0$$

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

Operations on strings (let $w = w_1w_2...w_n$):

- reverse: $w^R = w_nw_{n-1}...w_1$  
  $w^R = 543210$

- substring: $w_iw_{i+1}...w_j$  
  $w_2w_3w_4 = 123$

- concatenation of $w$ with a string $z = z_1z_2...z_m$:  
  $z = abcde$
  $wz = w_1w_2...wnz_1z_2...z_m$  
  $wz = 012345abcde$

- $w^k$ means concatenation of $k$ copies of $w$  
  $w^3 = 012345012345012345$
  $w^1 = w$  
  $w^0 = \epsilon$  
  $w^k \cdot w^l = w^{k+l}$  
  $k, l \geq 0$
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 \( \Sigma = \{0,1\} \):
  \[ \varepsilon, 0, 1, 00, 01, 10, 11, 000, ... \]
Language: a set of strings over an alphabet $\Sigma$, e.g.

$L_1 = \{ a, ab, bab \}$ \hspace{1cm} $|L_1| = 3$

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

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

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

$1 \in L_4$

$101 \in L_4$

$00 \notin L_4$

$01 \in L_4$

$\varepsilon \in L_4$
Operations on languages:

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

$L_1 = \{a, ba, baba\}$
$L_1 \cup L_2 = L_1$
$L_2 = \emptyset$
$L_2 \cap L_3 = \emptyset$
$L_3 = \{\varepsilon\}$
$L_1 \cap L_3 = \emptyset$
Strings and Languages

Operations on languages:

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

- concatenation: $L_1.L_2 = \{ w_1w_2 \mid w_1 \in L_1, w_2 \in L_2 \}$

$L_1 \cdot \emptyset = \emptyset$

$L_1 \cdot \{ \epsilon \} = L_1$

$L_1 = \{ a, bb \} \quad |L_1|=2$

$L_2 = \{ cd, c \} \quad |L_2|=2$

$L_1L_2 = \{ acd, bbc, ac, bcbd \} \quad |L_1L_2|=4$

$|L_1L_2| \leq |L_1| \cdot |L_2|$

not necessarily equal, e.g.:

$L_1 = \{ \epsilon, a, c \}$

$L_2 = \{ c, ac \}$
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: $L^* = \bigcup_{k=0}^{\infty} L^k$

$L^* = \text{the set of all strings over } \{a,b\}$

\[
\bigcup_{k=0}^{\infty} L^k = L^0 \cup L^1 \cup L^2 \cup L^3 \cup L^4 \cup \ldots
\]

\[
L^0 = \{ \epsilon \} \quad L = \{a, b\}
\]

\[
L^1 = \{a, b\} \quad L^2 = \{aa, ab, ba, bb\}
\]

\[
L^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}
\]

$L^k = \text{all strings over } \{a, b\} \text{ of length } k$
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: $L_1L_2 = \{ w_1w_2 | w_1 \in L_1, w_2 \in L_2 \}$
- Kleene's star: $L^* = \bigcup_{k=0}^{\infty} L^k$
- reverse: $L^R = \{ w^R | w \in L \}$

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

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