Chapter 1: Introduction to the Theory of Computation

Dr. Yuan Yao & Dr. Qiao Lin

University of Nottingham Ningbo China (UNNC)

Learning Outcomes

Learning outcomes

At the conclusion of this chapter, the students will be able to

- Define the three basic concepts in the theory of computation.
- Solve exercises using mathematical techniques and notation learned in previous modules.
- Evaluate expressions involving operations on strings.
- Evaluate expressions involving operations on languages.
- Generate strings from simple grammars.
- Construct grammars to generate simple languages.
- Describe the essential components of an automaton.
- Design grammars to describe simple programming constructs.

Theory of Computation

Theory of Computation

In broad terms, the theory of computation is concerned with two major questions

- **Computability:**
  - Identifying the problems that computers can solve in principle, regardless of how long it may take.
  - What are “computable” problems? all mathematical problems?
- **Complexity:**
  - How long it takes to solve a problem algorithmically.
  - Identification of problems that computers can solve in a reasonable time.
  - You will learn more in the module Algorithms Data structures and Efficiency

Theory of Computation

Three basic concepts of the computation theory:

- Formal Language:
  - A set of sentences formed from a set of symbols (according to some rules).
- Grammar:
  - A set of rules for generating the sentences in a formal language.
- Automaton:
  - An abstract machine that accepts input, produce output, may have some temporary storage, and can make decisions.

Review of Mathematical Preliminaries

Sets

A **Set** is a collection of elements, without any structure other than membership.

- **Membership:** $x \in S$, $x \notin S$
- **Union ($\cup$):** $S_1 \cup S_2 = \{x : x \in S_1 \text{ or } x \in S_2\}$
- **Intersection ($\cap$):** $S_1 \cap S_2 = \{x : x \in S_1 \text{ and } x \in S_2\}$
- **Difference ($-$):** $S_1 - S_2 = \{x : x \in S_1 \text{ and } x \notin S_2\}$
- **Complementation:** $\overline{S} = \{x \in U \text{ and } x \notin S\}$, where U is the **universal set**.
- **Subset ($\subseteq$):** if $S_1 \subseteq S_2$, then $S_1 \cup S_2 = S_2$, $S_1 \cap S_2 = S_1$, $S_1 - S_2 = \emptyset$.
- **Proper subset ($\subset$):** if $S_1 \subset S_2$, then $S_1 \subseteq S_2$ and $S_2 - S_1 \neq \emptyset$.
- **Disjoint:** S_1 and S_2 are said to be **disjoint**, if $S_1 \cap S_2 = \emptyset$.
- **Powerset:** The powerset of S is denoted by 2^S which contains all subsets of S.
- **Partition:** We say $S_1, S_2, \ldots, S_n$ is a **partition** of S if the following holds:
  - $S_1, S_2, \ldots, S_n$ are mutually disjoint, non-empty subsets of S.
  - $S_1 \cup S_2 \cup \cdots \cup S_n = S$

Functions and Relations

A **function** (f) is a rule that assigns to elements of one set a unique element of another set.

$$f : S_1 \to S_2$$

Domain: the domain of f is a subset of S_1

Range: the range of f is a subset of S_2

- **Total and Partial function:** If the domain of f is S_1, then f is a **total function** on S_1. Otherwise, f is said to be a **partial function**.
- **Order of magnitude (Big-O notation):** $f(n) = O(g(n))$ if $f(n) \leq c|g(n)|$

A **relation** is more general than a function, as one elements from the domain may correspond to multiple elements in the range.

Graphs

A **Graph** consists of two finite sets:

- **Vertices:** $V = \{v_1, v_2, \ldots, v_n\}$.
- **Edges:** $E = \{e_1, e_2, \ldots, e_m\}$, where each edge $e_i = (v_j, v_k)$ is a pair of vertices.

Walk: a sequence of edges $(v_i, v_j), (v_j, v_k), \ldots, (v_m, v_n)$ is a walk from v_i to v_n.

Path: a walk in which no edge is repeated. A path is said to be **Simple** if there is no repeated vertex.

Cycle: a walk from v_i to itself with no repeated edges is a **Cycle** with **Base** v_i. A cycle is said to be simple if no vertices other than the **base** is repeated.

Loop: an edge from a vertex to itself.

Trees

A **Tree** is a directed graph that has no cycles and that has one distinct vertex, called the **Root**, such that there is exactly one path from the root to every other vertex.

- **Leaves:** vertices without outgoing edges.
- **Parent and Child:** v_i is the parent of v_j if there is an edge from v_i to v_j, and v_j is the child of v_i.
- **Level:** the number of edges in the path from the root to this vertex.
- **Height:** the largest level number of any vertices.

Proof Techniques

Two important proof technique we will use in this module:

- **Proof by Induction:** the truth of a number of statements can be inferred from the truth of a few specific instances.
  - **Prove the base cases.**
  - **Prove the Inductive Steps.**
- **Proof by Contradiction:** If we want to prove P, then we assume P is false and see if it leads to a contradiction.

Exercise : Proof by induction and proof by contradiction

- Prove that $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$.
- Prove that $\sqrt{2}$ is irrational.

Exercise 1: Proof by induction

Prove that $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$.

- **Base case:** when $n = 1$, $1 = \frac{1\times(1+1)}{2}$, checks.
- Assume the above equation holds for $n = k$: $1 + 2 + \cdots + k = \frac{k(k+1)}{2}$ (Induction hypothesis)
- **Inductive Step:** show that $n = k + 1$ holds

$$1 + 2 + \cdots + k + (k + 1) = \frac{(k + 1)((k + 1) + 1)}{2}$$

Start from the left side of the equation $1 + 2 + \cdots + k + (k + 1)$
= $\frac{k(k+1)}{2} + (k + 1)$ by Induction hypothesis
= $\frac{k(k+1)+2(k+1)}{2}$ by $\frac{2}{2} = 1$ and distribution of division over addition
= $\frac{(k+2)(k+1)}{2}$ by distribution of multiplication over addition
= $\frac{(k+1)(k+2)}{2}$ by commutativity.
QED.

Exercise 2: Proof by contradiction

Prove that $\sqrt{2}$ is irrational.

- **Proof:** We assume $\sqrt{2}$ is a rational number written as

$$\sqrt{2} = \frac{n}{m}$$

where m and n are non-zero integers without a common factor.
- We could then have

$$2m^2 = n^2$$

- This equation implies n^2 is an even number, and thus n is also an even number. We could rewrite n as $2k$:

$$m^2 = 2k^2$$

- Therefore, m is even as well. But from the assumption we made, n and m do not have common factor. **Contradiction**.
- QED.

More exercises

Prove that the followings are irrational.

- $\sqrt{8}$
- $2 - \sqrt{2}$
- $\sqrt{3}$

Formal Language

Languages

- What does the word “language” mean?
  - Informally, a system suitable for the expression of ideas, facts or concepts, including a set of symbols and rules for their manipulation.
- What are natural languages?
- What are formal languages?
- What are their differences?
- We focus on **formal language** in this module.

Basic Concepts

- **Alphabet:** a finite, nonempty set of symbols, e.g., $\Sigma = \{a, b\}$.
- **String (Sentence):** a finite sequence of symbols from Σ, e.g., $v = aba$, $w = abaaa$.
- **Concatenation:** the concatenation of two strings v and w is obtained by appending the symbols of w to the right end of v, e.g., $vw = abaabaaa$.
- **Reverse:** the reverse of a string is obtained by writing the symbols in reverse order, e.g., $w^R = aaaba$.
- **Length:** the length of a string is the number of symbols, denoted by $|$|, e.g., $|w| = 5$.
- λ: empty string, i.e., $|\lambda| = 0$.

Basic Concepts

- **Substring:** Any string of consecutive symbols in some string w is said to be the substring of w.

If $w = vu$, then

- **Prefix:** the substring v is the prefix of w.
- **Suffix:** the substring u is the prefix of w.
- $|vu| = |v| + |u|$

Suppose $w = abaaa$, what are the prefixes of w?

- **Repetition:** w^n stands for the string obtained by repeating w n times, e.g., $w^2 = abaaaabaaa$

Definition

$\Sigma = \{a, b\} \rightarrow \Sigma^* = \{\lambda, a, b, aa, bb, ab, ba, aaa, abab, etc\}$

$\Sigma^+ = \{a, b, aa, bb, ab, ba, aaa, abab, etc\}$

- Σ^*: the set of all strings formed by concatenating zero or more symbols in Σ.
- Σ^+: the set of all non-empty strings formed by concatenating symbols in Σ, i.e.,

$$\Sigma^+ = \Sigma^* - \{\lambda\}$$

- A formal language is any subset of Σ^*.
  - $L_1 = \{a^n b^n : n \geq 0\}$
  - $L_2 = \{ab, aa\}$
  - $L_3 = \emptyset$ 空集
  - $L_4 = \{\lambda\}$ 包含空字符串 λ.

Set Operation

- A language is a set. Therefore, set operations can be directly used on languages.
- Suppose, $L_1 = \{a^n b^n : n \geq 0\}$ and $L_2 = \{ab, aa\}$. Find out the followings:
  - $L_1 \cup L_2 = \{aa, \lambda, ab, aabb, aaabbb \ldots\}$
  - $L_1 \cap L_2 = \{ab\}$
  - $L_1 - L_2 = \{\lambda, aabb, aaabbb \ldots\}$
  - $L_2 - L_1 = \{aa\}$
  - $\overline{L_2} = \Sigma^* - \{ab, aa\}$

Other Operations

- Suppose, $L_1 = \{a^n b^n : n \geq 0\}$ and $L_2 = \{ab, aa\}$.
  - **Reverse:** reversing all strings in a language
    - $L_2^R = \{ba, aa\}$
  - **Concatenation:** concatenating strings from two languages
    - $L_1 L_2 = \{ab, aa, abab, abaa, aabbab, aabbaa, \dots\}$
    - $L_2 L_2$ or $L_2^2 = \{abab, abaa, aaab, aaaa\}$
    - What is $L_1 \emptyset$?
  - **Star-Closure:** concatenating strings from the same language
    - $L_2^* = L_2^0 \cup L_2^1 \cup L_2^2 \cup \dots$
  - **Positive-Closure:**
    - $L_2^+ = L_2^1 \cup L_2^2 \cup \dots$
  - For every language L, the language L^+ is infinite (True or False?)

Grammar

Formal Languages

- For every language L, the language L^+ is infinite (True or False?)
  - **False.** With the following counter-examples:
    - If $L = \{\lambda\}$, then $L^+ = \{\lambda\}$, which is finite.
    - If $L = \emptyset$ then $L^+ = \emptyset$.
  - In general, for any language L, we have $L^0 = \{\lambda\}$.
  - **Why we need it at all?** To make sure the algebraic law $L^{m+n} = L^m L^n$ holds for all values of m and n, i.e., we want $L^{0+n} = L^0 L^n$.
  - Therefore, we have:
    - $\{\lambda\}^* = \{\lambda\}$
    - $\{\lambda\}^+ = \{\lambda\}$
    - $\emptyset^* = \{\lambda\}$
    - $\emptyset^+ = \emptyset$

Definition

- How to precisely describe a language?
- **Grammar:** a precise mechanism to describe the strings in a language.
- A grammar G is defined as a 4-tuples (V, T, S, P), where:
  - V is a finite set of variable (or non-terminal) symbols.
  - T is a finite set of terminal symbols.
  - $S \in V$ is a special variable, called the start symbol.
  - P is a finite set of productions, each of the form $\alpha \to \beta$, in which:

$$\alpha \in (V \cup T)^* V (V \cup T)^*, \beta \in (V \cup T)^*$$

  - Example: In the following grammar, S is the start symbol:
    - $V = \{S\}$
    - $T = \{a, b\}$
    - $P = \{S \to aSb, S \to \lambda\}$

String Derivation

- Beginning with the start symbol, strings are derived by repeatedly replacing variable symbols with the expression on the right-hand side of any applicable production.
- Any applicable production can be used, in arbitrary order, until the string contains no variable symbols.
- Example: In the following grammar, S is the start symbol:
  - $V = \{S\}$
  - $T = \{a, b\}$
  - $P = \{S \to aSb, S \to \lambda\}$
- Sample derivation using the above grammar:

$$
\begin{aligned}
S &\Rightarrow aSb \\
&\Rightarrow aaSbb \\
&\Rightarrow aabb
\end{aligned}
$$

The Language Generated by a Grammar

- For a given grammar G, the language $L(G)$ generated by G is the set of all strings derived from the start symbol.
  - Let $G = (V, T, S, P)$ be a grammar. Then the language $L(G)$ is defined as follows:

$$L(G) = \{w \in T^* : S \overset{*}{\Rightarrow} w\}$$

    Where $S \overset{*}{\Rightarrow} w$ means w can be drived from S in an unspecified number of steps.
- Given the following grammar $G = (V, T, S, P)$:
  - $V = \{S\}$
  - $T = \{a, b\}$
  - $P = \{S \to aSb, S \to \lambda\}$
- What is $L(G)$?

$S \Rightarrow aSb$
$\Rightarrow aaSbb$
$\Rightarrow aabb$

Equivalence of Grammars

- It is not difficult to find out that

$$L(G) = \{a^n b^n : n \geq 0\}$$

- However, to show that $L(G)$ is the language generated by G, we need to:
  - Show that every string in L can be generated by G.
  - Show that every string generated by G is in L.
- Two grammars are equivalent if they generate the same language.
- Can you design a different grammar for $L(G)$?

Equivalence of Grammars

- $G_1 = (V, T, S, P)$ where:
  - $V = \{S, A\}$
  - $T = \{a, b\}$
  - $P = \{S \to aAb|\lambda,$ $A \to aAb|\lambda\}$
- What is the language $L(G_1)$?

Grammars for Programming Languages

- They syntax of constructs in a programming languages is commonly described with grammars.
- The rules for variable identifiers in C are:
  - An identifier is a sequence of letters, digits, and underscore.
  - An identifier must start with a letter or an underscore.
  - Identifiers allow both upper- and lower-case letters.
- **Practise:** provide a grammar for the variable identifiers in C.

Automaton

Automaton

- An automaton is an abstract model of a digital computer.
- An automaton consists of:
  - An input mechanism
  - A control unit
  - Possibly, a storage mechanism
  - Possibly, an output mechanism
- Control unit can be in any number of internal states, as determined by a next-state or transition function.

Automaton