Contents

Chapter 1

Introduction and Motivation

1.1 Discrete Mathematics and Computer Science

Discrete Mathematics is concerned with finite and countably infinite mathematical structures. There are at least three major reasons why discrete mathematics is of central importance in computer science

- **Discrete structures**
  Many objects studied in computer science are discrete mathematical objects, for example a graph modeling a computer network or an algebraic group used in cryptography. Many applications exploit sophisticated properties of the involved structures
- **Abstraction**
  A computer system can only be understood by considering a number of layers of abstraction, from application programs to the physical hardware
- **Programs as mathematicals objects**
  Understanding a computer program means to understand it as a discrete mathematical object

1.2 Discrete Mathematics: A Selection of Teasers

This section contains 8 examples, worth taking a look at.

1.3 Abstraction: Simplicity and Generality

In Discrete Mathematics, abstraction stands for simplicity and generality. In order to manage complexity, software systems are divided into components (called modules, layers, objects or abstract data types) that interact with each other and with the environment in a well-defined manner.

Abstraction means *simplification*. By an abstraction one ignores all aspects of a system that are not relevant for the problem at hand, concentrating on the properties that matter

Abstraction also means *generalization*. If one proves a property of a system described at an abstract level, then this property holds for any system with the same abstraction, independently of any details.

1.4 Programs as Discrete Mathematical Objects

A computer program is a discrete mathematical object. The correctness of a program is a (discrete) mathematical statement. The proof that a program is correct (i.e., that it satisfies its specification) is a mathematical proof, not a heuristic argument.

Chapter 2

Mathematical Reasoning, Proofs, and a First Approach to Logic

2.1 What is a Proof?

2.1.2 The concept of a proof

Definition 2.1 (Informal)
A *proof* of a statement S is a sequence of simple, easy verifiable consecutive steps. The proof starts from a set of axioms (things postulated to be true) and known (previously proved) facts. Each step corresponds to the application of a derivation rule to a few already proven statements, resulting in a newly proved statement, until the final step proves S.

2.1.3 Informal vs. Formal Proofs

Formal Proof: Uses mathematical symbols and language.
Informal Proof: Explained using common, everyday language.

There are at least three (related) reasons for using a more rigurous and formal type of proof:

- **Prevention of errors**
  A completely formal proof leaves no room for interpretation, and hence allows to exclude errors
- **Proof complexity and automatic verification**
  Certain proofs are too complex to be carried out and verified "by hand". A computer is required for the verification, which can only deal with rigurously formalized statements, not with semi-precise common language.
- **Precision and deeper understanding**
  A formal proof requires the formalization of the arguments and can lead to a deeper understanding (also for the author of the proof).

2.1.4 The Role of Logic

Logic defines the syntax of language for expressing statements and the semantics of such a language, defining which statements are true and which are false. A logical calculus allows to express and verify proofs in a purely syntactic fashion, for example by a computer.

2.2 Propositions and Logical Formulas

2.2.1 Propositions, True and False

Definition 2.2
A proposition is a (mathematical) statement that is either *true* or *false*. Alternative terms for "proposition" are assertion, claim or simply statement.

Statements like "Bob loves Carol", "it is raining", and "birds can fly" are NOT (mathematical) propositions, unless we assume that the validity of these statements is defined (outside of mathematics, e.g. by common sense) to be a fixed, constant value (true or false).

Definition 2.3
A true proposition is often called a *theorem*, a *lemma*, or a *corollary*

There is no fixed naming convention, but the term "theorem" is usually used for an important result, whereas a lemma is an intermediate, often technical result, possibly used in several subsequent proofs. A corollary is a simple consequence (e.g. a special case) of a theorem or lemma.

2.2.2 Logical constants, Operators and Formulas

Definition 2.4
The Logical values (constants) "true" and "false" are usually denoted as 1 and 0, respectively.

Definition 2.5

i) The *negation* (Logical NOT) of a proposition A, denoted as $\neg A$, is true if and only if A is false.

ii) The *conjunction* (Logical AND) of two propositions A and B, denoted $A \wedge B$, is true if and only if both A and B are true.

iii) The *disjunction* (Logical OR) of two propositions A and B, denoted $A \vee B$, is true if and only if A or B (or both) are true

A	$\neg A$
0	1
1	0

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Logical operators can also be combined, in the usual way of combining functions. For example, if A, B and C are propositions, then

$$A \vee (B \wedge C)$$

is also a proposition. Its function table is

A	B	C	$A \vee (B \wedge C)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Definition 2.6
A correctly formed expression involving propositional symbols (like $A, B, C, \ldots$) and logical operators is called a *formula* (of propositional logic)

We introduce a new, derived logical operator: *implication*, denoted as $A \to B$ and defined by[1]

$$A \to B :\Longleftrightarrow \neg A \vee B$$

A	B	$A \to B$
0	0	1
0	1	1
1	0	0
1	1	1

Example 2.2

Consider the following sentence: If you read the lecture note every week and do the exercises (A), then you will get a good grade in the exam (B). This is an example of an implication $A \to B$. Saying that $A \to B$ is true does not mean that A is true and it is not excluded that B is true even if A is false, but it is

[1]The symbol $:\Longleftrightarrow$ simply means that the left side (here $A \to B$) is defined to mean the right side (here $\neg A \vee B$)

excluded that B is false when A is true.

Two-sided implication, denoted $A \leftrightarrow B$, is defined as follows[2]:

$$A \leftrightarrow B :\Longleftrightarrow (A \to B) \wedge (B \to A)$$

A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

Parenthesis can sometimes be dropped in a formula without changing its meaning, for example we can write $A \vee B \vee C$ instead of $A \vee (B \vee C)$ or $(A \vee B) \vee C$. Namely, $\wedge$ and $\vee$ bind stronger than $\to$ and $\leftrightarrow$. Also, $\neg$ binds stronger than $\wedge$ and $\vee$. For example, we can write $A \vee \neg B \to B \wedge C$ instead of $(A \vee (\neg B)) \to (B \wedge C)$

2.2.3 Formulas as Functions and Logical Equivalence

[2]Note that $A \equiv B$ holds if and only if $A \leftrightarrow B$ is true, but $A \equiv B$ itself is not a logical formula.