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 $A, B, C, \dots$) 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

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

that A is true and it is not excluded that B is true even if A is false, but it is 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

$(A \wedge (\neg B)) \vee (B \wedge (\neg C))$ is a formula, and corresponds to a function $\{0,1\}^3 \to \{0,1\}$.

Definition 2.7

The symbol $\top$ denote a function that is the constant 1 (true), and $\bot$ denotes the function that is constant 0 (false).

Definition 2.8

Two Formulas F and G (in propositional logic) are called *equivalent*, denoted as $F \Longleftrightarrow G$ (or also F$\equiv$ G), if they correspond to the same function (table).

For example, it is easy to see that $\wedge$ and $\vee$ are commutative and associative, i.e. for any formulas F, G, and H we have

$$F \wedge G \Longleftrightarrow G \wedge F \text{ and } F \vee G \Longleftrightarrow G \vee F$$

as well as

$$F \wedge (G \wedge H) \Longleftrightarrow (F \wedge G) \wedge H \Longleftrightarrow F \wedge G \wedge H$$

similarly

$$F \vee (G \vee H) \Longleftrightarrow (F \vee G) \vee H \Longleftrightarrow F \vee G \vee H$$

We also have

$$\neg(\neg(F)) \Longleftrightarrow F$$

[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.

2.2.4 Tautologies and Satisfiability

Definition 2.9

A formula F (in propositional logic) is called a *tautology* if it is true for all truth assignments of the involved propositional symbols. For example, the formulas $A \vee (\neg A)$ and $(A \wedge (A \to B) \to B$ are tautologies

Definition 2.10

A formula F (in propositional logic) is called *satisfiable* if it is true for at least one truth assignment of the involved propositional symbols, and it is called *unsatisfiable* otherwise

Lemma 2.1

F is a tautology if and only if $\neg F$ is unsatisfiable.

2.2.5 The Symbols $\Longrightarrow$ and $\Longleftrightarrow$

Definition 2.11

One writes $F \Longrightarrow G$ (or $F \Rightarrow G$)(The length of the arrow is completely irrelevant) to say that F implies G, i.e. that $F \to G$ is a tautology

As mentioned earlier, one writes $F \Longleftrightarrow G$ if F and G imply each other, i.e. if F and G are equivalent. $F \Longleftrightarrow G$ is also expressed as "F if and only if G".

To state $F \Longrightarrow G$ is the same as to state "$F \to G$ is a tautology". Note that $F \Longrightarrow G$ and $F \Longleftrightarrow G$ are not logical formulas since $\Longrightarrow$ and $\Longleftrightarrow$ are not allowed symbols in the syntax of logic.

Theorem 2.2

Implication is transitive: if $F \Longrightarrow G$ and $G \Longrightarrow H$, then $F \Longrightarrow H$

The theorem implies that, more generally, if one proves a statement F_1 as well as implications $F_1 \Longrightarrow F_2, F_2 \Longrightarrow F_3, \ldots F_{n-1} \Longrightarrow F_n$, then one has proved F_n.

2.3 Quantifiers and Predicate Logic

The extension of propositional logic is called *predicate logic* and is substantially more involved than propositional logic.

2.3.1 Predicates

Let us consider a set U as the universe in which we want to reason. For example, U could be the set $\mathbb{N}$ of natural numbers, the set $\mathbb{R}$ of real numbers, the set $\{0,1\}*$ of finite-length bit-strings, or a finite set like $\{0,1,2,3,4,5,6\}$.

Definition 2.12
A $k-$ary predicate P on U is a function $U^k \to \{0, 1\}$

A $k-$ary predicate P assigns to each list $(x_1, \ldots, x_k)$ of k elements of U the value $P(x_1, \ldots, x_k)$ which is either true (1) or false (0).

2.3.2 Definition of $\exists$ and $\forall$

Definition 2.13
For a universe U and predicate $P(x)$ we define the following logical statements

- $\forall x P(x)$ is the statement that $P(x)$ is true for all $x \in U$
- $\exists x P(x)$ is the statement that $P(x)$ is true for some $x \in U$, i.e. there exists an $x \in U$ for which $P(x)$ is true.

Sometimes one uses an abbreviated notation in which one specifies a condition for the variable x directly. For example, we can write $\forall x \geq 5(x^2 \geq 25)$ instead of $\forall x(x \geq 5) \to (x^2 \geq 25)$.

2.3.3 Nested Quantifiers

Quantifiers can also be nested. For example, if $P(x)$ and $Q(x, y)$ are predicates, then

$$\forall x(P(x) \vee \exists y Q(y, x))$$

is a logical formula.

Check out multiple examples on page 19 about nested quantifiers

2.3.4 Tautologies and Satisfiability

A formula is *satisfiable* if, informally there is an interpretation that makes the formula true. Moreover, a formula is a *tautology* if it is true for all interpretations, i.e. for all choices of the universe and for all interpretations of the predicates. For example:

$$\forall x((P(x) \wedge Q(x)) \to (P(x) \vee Q(x)))$$

is a tautology. As mentioned in section 2.2.5, we write $F \implies G$ to mean that $F \to G$ is a tautology.

If the universe is fixed and there are no unspecified parts like predicate symbols, then a formula can trivially only be a tautology or unsatisfiable.

2.3.5 Some Rules

We list a few useful rules for predicate logic

- $\forall x P(x) \wedge \forall x Q(x) \iff \forall x (P(x) \wedge Q(x))$
- $\exists x (P(x) \wedge Q(x)) \implies \exists x P(x) \wedge \exists x Q(x)$
- $(\exists x P(x) \wedge \exists x Q(x)) \rightarrow (\exists x (P(x) \wedge Q(x)))$ is not a tautology
- $\neg \forall x P(x) \iff \exists x \neg P(x)$
- $\neg \exists x P(x) \iff \forall x \neg P(x)$

2.4 Some Proof Patterns and Techniques

What is to be proven (e.g. a statement called a theorem) is given by a logical formula F, or by a sentence in natural language that could be made precise as (and hence stands for) a logical formula F. Typically F is a proposition, i.e. it is either true or false, and for formulas in predicate logic the universe is understood.

2.4.1 Modus Ponens

The following theorem states than in order to prove G one can try to identify a formula F such that one can prove both F as well as $F \rightarrow G$. The proof is similar to the proof of theorem 2.2, using the fact that the propositional formula

$$((A \wedge (A \rightarrow B))) \rightarrow B$$

is a tautology.

Theorem 2.3

If F and $F \rightarrow G$ are tautologies, then G is also a tautology.

2.4.2 Direct Proof of an Implication

Definition 2.14

A *direct proof* of an implication $F \rightarrow G$ works by assuming F and then deriving G (from F), where the derivation can possibly involve several proof steps.

2.4.3 Indirect Proof of an Implication

Definition 2.15

An *indirect proof* of an implication $F \rightarrow G$ works by assuming $\neg G$ and deriving $\neg F$, i.e. by proving $\neg G \rightarrow \neg F$

This is correct because $F \rightarrow G$ id logically equivalent to $\neg G \rightarrow \neg F$, i.e.

$$F \rightarrow G \iff \neg G \rightarrow \neg F$$

2.4.4 Proofs by Contradiction

The following theorem states that in order to prove F one can try to identify a formula G such that one can prove both $\neg G$ as well as $\neg F \to G$. The proof is similar to the proof of Theorem 2.2, using the fact that the propositional formula

$$((\neg A \to B) \wedge \neg B) \to A$$

is a tautology.

Theorem 2.4

If $\neg F \to G$ and $\neg G$ are tautologies, then F is also a tautology (or: If $\neg F \to \bot$ is a tautology, then F is also a tautology.

Such proof usually works by assuming $\neg F$, and deriving from this assumption a formula H and also its negation $\neg H$, which is a contradiction. In other words, one proves $\neg F \to H$ as well as $\neg F \to \neg H$, and hence also $\neg F \to (H \wedge \neg H)$, i.e. the formula G of the theorem is $G = H \wedge \neg H$

2.4.5 Existence Proofs

Definition 2.16
An existence proof is the proof of a statement of the form $\exists x P(x)$

There are constructive existence proofs, which demonstrate an a for which $P(a)$ is true, as well as non-constructive existence proofs which simply show the existence of an a for which $P(a)$ is true, without exhibiting such an a.

2.4.6 Inexistence Proofs

Definition 2.17
An inexistence proof is a proof of a statement of the form $\neg\exists x P(x)$

To prove the inexistence, one can use

$$\neg\exists x P(x) \Longleftrightarrow \forall x \neg P(x)$$

2.4.7 Proofs by Counterexample

Definition 2.18
A proof by counterexample is a proof of a statement of the form $\neg\forall x P(x)$ for some fixed predicate P, using

$$\neg\forall x P(x) \Longleftrightarrow \exists x \neg P(x)$$

An a for which $\neg P(a)$ is true is called a counterexample

2.4.8 Proofs by Induction

A proof by induction consists in two steps:

1. **Basis Step**: Prove $P(0)$
2. **Induction Step**: Prove $\forall n(P(n) \to P(n+1))$

Theorem 2.5

For every predicate P on $\mathbb{N}$ we have

$$P(0) \wedge \forall n(P(n) \to P(n+1)) \Longrightarrow \forall n P(n)$$

Fact 2.1

The natural numbers are well-ordered, i.e. every nonempty set of natural numbers has a least element[3]

[3]The well-ordering principle could be stated more formally as

$$\exists n P(n) \Longrightarrow \exists m\,(P(m) \wedge \forall k < m \neg P(k))$$

but we will not need this formula.

Chapter 3

Sets, Relations and Functions