ETH Zürich
Departement Informatik

Diskrete Mathematik

Ueli Maurer

Herbstsemester 2023

Vorwort

Viele Disziplinen der Wissenschaft, insbesondere der Natur- und Ingenieurwissenschaften, beruhen in einer zentralen Weise auf der Mathematik. Einerseits erlaubt die Mathematik, Sachverhalte zu modellieren und damit den Diskurs von einer intuitiven auf eine präzise und formale Stufe zu heben. Andererseits erlaubt die Mathematik, wenn Sachverhalte einmal präzise modelliert sind (z.B. die Statik als Teil der Physik), konkrete Probleme zu lösen (z.B. eine Brücke zu dimensionieren).

Welche mathematischen Disziplinen sind für die Computerwissenschaften (Informatik, Computer Science) speziell relevant? Was muss in der Informatik modelliert werden? Welche Art von Problemen möchte man verstehen und lösen können? Der gemeinsame Nenner der vielen möglichen Antworten ist, dass es in der Informatik um diskrete, meist endliche Strukturen geht. Digitale Computer haben einen endlichen Zustandsraum, d.h. der Zustand ist exakt beschreibbar als eine von endlich vielen Möglichkeiten. Zwei Zustände können nicht, wie in der Physik, beliebig ähnlich sein. Es gibt nicht das Problem, dass reellwertige Parameter (z.B. die Temperatur) nur approximativ gemessen werden können. In der Informatik im engeren Sinn gibt es keine kontinuierlichen Grössen.[1]

Das heisst natürlich nicht, dass sich die Informatik nicht mit Themen befasst, bei denen kontinuierliche Grössen wichtig sind. Die Informatik ist ja auch eine Hilfswissenschaft, z.B. für die Naturwissenschaften, wobei die Grenzen zwischen der eigentlichen Wissenschaft und der Hilfswissenschaft in einigen Bereichen verschwommener werden. In Bereichen wie Computational Biology oder Computational Chemistry werden wesentliche Beiträge direkt von der Informatik beigesteuert. In diesen Bereichen der Informatik spielen reellwertig parametrisierte Systeme eine wichtige Rolle.[2]

[1] Die Mathematik der Informatik sollte demnach einfacher verständlich sein als die kontinuierliche Mathematik (z.B. Analysis). Sollte dies dem Leser ab und zu nicht so erscheinen, so ist es vermutlich lediglich eine Frage der Gewöhnung.

[2] Die Numerik befasst sich unter anderem mit dem Thema der (in einem Computer unvermeidbaren) diskreten Approximation reeller Grössen und den daraus resultierenden Problemen wie z.B. numerische Instabilitäten.

Das Teilgebiet der Mathematik, das sich mit diskreten Strukturen befasst, heisst *diskrete Mathematik*. Der Begriff "diskret" ist zu verstehen als *endlich oder abzählbar unendlich*. Viele Teilbereiche der diskreten Mathematik sind so wichtig, dass sie vertieft in einer eigenen Vorlesung behandelt werden. Dazu gehören die Theorie der Berechnung, also die Formalisierung der Begriffe Berechnung und Algorithmus, welche in der Vorlesung "Theoretische Informatik" behandelt wird, sowie die diskrete Wahrscheinlichkeitstheorie. Eine inhaltliche Verwandtschaft besteht auch zur Vorlesung über Algorithmen und Datenstrukturen.

In dieser Lehrveranstaltung werden die wichtigsten Begriffe, Techniken und Resultate der diskreten Mathematik eingeführt. Hauptziele der Vorlesung sind nebst der Behandlung der konkreten Themen ebenso die adäquate Modellierung von Sachverhalten, sowie das Verständnis für die Wichtigkeit von Abstraktion, von Beweisen und generell der mathematisch-präzisen Denkweise, die auch beim Entwurf von Softwaresystemen enorm wichtig ist. Zudem werden einige Anwendungen diskutiert, z.B. aus der Kryptografie, der Codierungstheorie oder der Algorithmentheorie. Diskrete Mathematik ist ein sehr breites Gebiet. Entsprechend unterschiedliche Ansätze gibt es auch für den Aufbau einer Vorlesung über das Thema. Mein Ziel bei der Konzipierung dieser Lehrveranstaltung war es, speziell auf Themen einzugehen, die in der Informatik wichtig sind, sowie dem Anspruch zu genügen, keine zentralen Themen der diskreten Mathematik auszulassen. Ausnahmen sind die Kombinatorik und die Graphentheorie, die früher als Kapitel 4 und 5 dieses Skriptes erschienen, in der letzten Studienplanrevision aber in andere Vorlesungen verschoben wurden.

Die sechs Kapitel sind

1. **Introduction and Motivation**
2. **Mathematical Reasoning and Proofs**
3. **Sets, Relations, and Functions**
4. **Number Theory**
5. **Algebra**
6. **Logic**

Viele Beispiele werden nur an der Tafel oder in den Übungen behandelt. Die Vorlesung und die Übungen bilden einen integralen Bestandteil der Lehrveranstaltung und des Prüfungsstoffes. Es gibt kein einzelnes Buch, das den ganzen Stoff der Lehrveranstaltung behandelt. Aber unten folgt eine Liste guter Bücher, die als Ergänzung dienen können. Sie decken aber jeweils nur Teile der Vorlesung ab, gehen zum Teil zu wenig tief, oder sind zu fortgeschritten im Vergleich zur Vorlesung.

- N. L. Biggs, *Discrete Mathematics*, Clarendon Press.

- K. H. Rosen, *Discrete Mathematics and its Applications*, fourth edition, McGraw-Hill.
- A. Steger, *Diskrete Strukturen*, Band 1, Springer Verlag.
- M. Aigner, *Diskrete Mathematik*, Vieweg.
- J. Matousek, J. Nesetril, *Discrete Mathematics*, Clarendon Press.
- I. Anderson, *A First Course in Discrete Mathematics*, Springer Verlag.
- U. Schöning, *Logik für Informatiker*, Spektrum Verlag, 5. Auflage, 2000.
- M. Kreuzer and S. Kühling, *Logik für Informatiker*, Pearson Studium, 2006.

Das Skript ist aus verschiedenen Gründen englischsprachig verfasst, unter anderem, weil daraus eventuell einmal ein Buch entstehen soll. Wichtige Begriffe sind auf deutsch in Fussnoten angegeben. Das Skript behandelt mehr Stoff als die Vorlesung. Abschnitte, die nicht Prüfungsstoff sind und vermutlich in der Vorlesung auch nicht behandelt werden, sind mit einem Stern ($*$) markiert und in einem kleineren Font gedruckt. Im Verlauf der Vorlesung werde ich eventuell einzelne weitere Teile als *nicht prüfungsrelevant* deklarieren.

Zum Schluss einige Überlegungen und Empfehlungen für die Arbeitsweise beim Besuch dieser Lehrveranstaltung. Die Lehrveranstaltung besteht aus der Vorlesung, dem Skript, den Übungsblättern, den Musterlösungen, den Übungsstunden, und dem Selbststudium. Die verschiedenen Elemente sind aufeinander abgestimmt. Insbesondere ist die Vorlesung unter der Annahme konzipiert, dass die Studierenden das Skript zu den behandelten Teilen nach jeder Vorlesung lesen, allenfalls auch vorher als Vorbereitung. Es ist unabdingbar, dass Sie das Skript regelmässig und detailiert erarbeiten, da dies dem Konzept der Vorlesung entspricht. Ebenso ist es unabdingbar, zusätzlich zur Übungsstunde mehrere Stunden pro Woche eigenständig oder in Teamarbeit für das Lösen der Übungen aufzuwenden; ein Teil dieser Zeit soll *vor* der Übungsstunde investiert werden.

Ich danke Giovanni Deligios und David Lanzenberger für viele konstruktive Kommentare und die kritische Durchsicht des Manuskripts.

Zürich, im September 2023 Ueli Maurer

Contents

Chapter 1

Introduction and Motivation

1.1 Discrete Mathematics and Computer Science

Discrete mathematics is concerned with finite and countably infinite mathematical structures. Most areas within Computer Science make heavy use of concepts from discrete mathematics. The applications range from algorithms (design and analysis) to databases, from security to graphics, and from operating systems to program verification.[1]

There are (at least) three major reasons why discrete mathematics is of central importance in Computer Science:

1. **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 or coding theory. Many applications exploit sophisticated properties of the involved structures.

2. **Abstraction.** Abstraction is of paramount importance in Computer Science. A computer system can only be understood by considering a number of layers of abstraction, from application programs via the operating system layer down to the physical hardware. Discrete mathematics, especially the way we present it, can teach us the art of abstraction. We refer to Section 1.3 for a discussion.

3. **Mathematical derivations.** Mathematical reasoning is essential in any engineering discipline, and especially in Computer Science. In many disciplines (e.g.[2] mechanical engineering), mathematical reasoning happens in the form of *calculations* (e.g. calculating the wing profile for an airplane). In contrast, in Computer Science, mathematical reasoning often happens in the form of a *derivation* (or, more mathematically stated, a *proof*). For example, understanding a computer program means to understand it as a well-defined discrete mathematical object, and making a desirable statement about the program (e.g. that it terminates within a certain number of steps) means to prove (or derive) this statement. Similarly, the statement that a system (e.g. a block-chain system) is secure is a mathematical statement that requires a proof.

[1]We also refer to the preface to these lecture notes where the special role of mathematics for Computer Science is mentioned.

[2]"e.g.", the abbreviation of the Latin "exempli gratia" should be read as "for example".

1.2 Discrete Mathematics: A Selection of Teasers

We present a number of examples as teasers for this course. Each example is representative for one or several of the topics treated in this course.[3]

Example 1.1. Consider a $k \times k$ chess board (ignoring the black/white coloring). Prove or disprove the following statement: No matter which of the squares is marked, the remaining area of the board (consisting of $k^2 - 1$ squares) can be covered completely with (non-overlapping) L-shaped pieces of paper each consisting of three squares.

This example allows us to informally introduce a few mathematical concepts that will be discussed in detail later in this course. The above statement depends on k. For certain k it is true and for certain k it is false. Let us therefore introduce a so-called *logical predicate* P, a function from the natural numbers to $\{0, 1\}$, where 1 stands for true and 0 stands for false. Then $P(k) = 1$ means that the statement is true for k, and $P(k) = 0$ means that the statement is false for k.

The case $k = 2$ is trivial: If any square (which is a corner square) is removed from a 2×2 chess board, the remaining three squares form the given L-shape. Hence we have $P(2) = 1$.

For $k = 3$, a simple counting argument shows that $P(3) = 0$. Since $k^2 - 1 = 8$ squares should be covered three at a time (by L-shapes), two squares remain at the end. More generally, a solution can exist only if $k^2 - 1$ is divisible by 3. For which k is this the case? In our notation we will (in Chapter 4) write

$$k^2 \equiv_3 1$$

for this condition, read as "k^2 is congruent to 1 modulo 3." This condition is equivalent to

$$k \equiv_3 1 \quad \text{or} \quad k \equiv_3 2.$$

[3]The reader should not worry too much if he or she is not familiar with some of the concepts discussed in this section, for example the interpolation of a polynomial, computation modulo a number n, Euclid's algorithm for computing greatest common divisors, or matrices.

Hence we have $P(k) = 0$ for all k with $k \equiv_3 0$ (i.e.,[4] the k divisible by 3).[5]

The case $k = 4$ can be solved easily by finding a solution for each of the three types of squares (corner, edge, interior of board) that could be marked. Hence we have proved $P(4) = 1$. This proof type will later be called a *proof by case distinction*.

For the case $k = 5$ one can prove that $P(5) = 0$ by showing that there is (at least) a square which, when marked, leaves an area not coverable by L-shapes. Namely, if one marks a square next to the center square, then it is impossible to cover the remaining area by L-shapes. This proof type will later be called a *proof by counterexample*.

We have $P(6) = 0$ because 6 is divisible by 3, and hence the next interesting case is $k = 7$. The reader can prove as an exercise that $P(7) = 1$. (How many cases do you have to check?)

The question of interest is, for a general k, whether $P(k) = 1$ or $P(k) = 0$. But one can prove (explained in the lecture) that

$$P(k) = 1 \implies P(2k) = 1,$$

i.e., that if the statement is true for some k, then it is also true for two times k. This implies that $P(2^i) = 1$ for any i and also that $P(7 \cdot 2^i) = 1$ for any i. Hence we have $P(8) = 1$, and $P(9) = 0$, leaving $P(10)$ and $P(11)$ as the next open cases. One can also prove the following generalization of the above-stated fact:

$$P(k) = 1 \text{ and } P(\ell) = 1 \implies P(k\ell) = 1.$$

We point out that, already in this first example, we understand the reasoning leading to the conclusion $P(k) = 0$ or $P(k) = 1$ as a *proof*.

Example 1.2. Consider the following simple method for testing primality. Prove or disprove that an odd number n is a prime if and only if 2^{n-1} divided by n yields remainder 1, i.e., if

$$2^{n-1} \equiv_n 1.$$

One can easily check that $2^{n-1} \equiv_n 1$ holds for the primes $n = 3, 5, 7, 11, 13$ (and many more). Moreover, one can also easily check that $2^{n-1} \not\equiv_n 1$ for the first odd composite numbers $n = 9, 15, 21, 25$, etc. But is the formula a general primality test? The solution to this problem will be given in Chapter 4.

Example 1.3. The well-known cancellation law for real numbers states that if $ab = ac$ and $a \neq 0$, then $b = c$. In other words, one can divide both sides by a. How general is this law? Does it hold for the polynomials over $\mathbb{R}$, i.e., does

[4]"i.e.", the abbreviation of the Latin "id est", should be read as "that is" (German: "das heisst").

[5]The fact that the equation $k^2 \equiv_p 1$ has two solutions modulo p, for any prime p, not just for $p = 3$, will be obvious once we understand that computing modulo p is a *field* (see Chapter 5) and that every element of a field has either two square roots or none.

$a(x)b(x) = a(x)c(x)$ imply $b(x) = c(x)$ if $a(x) \neq 0$? Does it hold for the integers modulo m, i.e., does $ab \equiv_m ac$ imply $b \equiv_m c$ if $a \neq 0$? Does it hold for the permutations, when multiplication is defined as composition of permutations? What does the condition $a \neq 0$ mean in this case? Which abstraction lies behind the cancellation law? This is a typical algebraic question (see Chapter 5).

Example 1.4. It is well-known that one can interpolate a polynomial $a(x) = a_d x^d + a_{d-1}x^{d-1} + \cdots a_1 x + a_0$ of degree d with real coefficients from any $d+1$ values $a(\alpha_i)$, for distinct $\alpha_1, \ldots, \alpha_{d+1}$. Can we also construct polynomials over a finite domain (which is of more interest and use in Computer Science), keeping this interpolation property?

For example, consider computation modulo 5. There are $5^3 = 125$ polynomials $a_2x^2 + a_1x + a_0$ of degree 2 because we can freely choose three coefficients from $\{0, 1, 2, 3, 4\}$. It is straight-forward (though cumbersome) to verify that if we fix any three evaluation points (for example $0, 2$, and 3), then the polynomial is determined by the values at these points. In other words, two different polynomials p and q result in distinct lists $(p(0), p(2), p(3))$ and $(q(0), q(2), q(3))$ of polynomial values. What is the general principle explaining this? For the answer and applications, see Chapter 5.

1.3 Abstraction: Simplicity and Generality

A main theme of this course is *abstraction*. In everyday life, the term "abstract" has a negative meaning. It stands for non-intuitive and difficult-to-understand. For us, abstraction will have precisely the opposite meaning. It will stand for simplicity and generality. I hope to be able to convey the joy and importance of simplification and generalization by abstraction.

Indeed, abstraction is probably the most important principle in programming and the design of information systems. Computers and computer programs are highly (perhaps unimaginably) complex systems. For a computer system with only 1000 bits of storage, the number 2^{1000} of system states is greater than the number of atoms in the known universe. The immense complexity of software systems is usually grossly underestimated, resulting in potentially catastrophic software failures. For typical commercial software, failures are the rule rather than the exception.

In order to manage the 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. For example, the Internet communication software is divided into a number of layers, each with a dedicated set of tasks. The IP layer transports packets between computers, and the TCP layer manages reliable connections. The potential complexity of the interaction between subsystems is channeled into clearly specified interfaces. The behavior of a subsystem is described by a manageable number of rules. This is abstraction. Without abstraction, writing good software is impossible.

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.

Example 1.5. A standard Swiss chocolate consists of 6 rows of 4 pieces each. We would like to break it into its 24 pieces using the minimal number of breaking operations. The first breaking operation can break the chocolate in any of the 5 ways parallel to the short side, or in any of the 3 ways parallel to the long side. Afterwards, a breaking operation consists of taking an arbitrary piece of chocolate and breaking it along one of the marked lines. Stacking pieces is not allowed. What is the minimal number of breaking operations needed to break the chocolate into its 24 pieces? Is it better to first break the chocolate into two equal pieces or to break off one row? Is it better to first break along a short or a long line? Which abstraction explains the answer? Find a similar problem with the same abstraction.

Example 1.6. Can the shape in Figure 1.1 be cut into 9 identical pieces? If not, why? If yes, what is the abstraction that explains this? What would more general examples with the same abstraction look like? Why would it be easier to see the answer in such generalized examples?

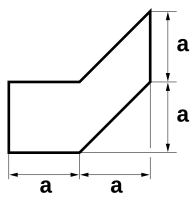

Figure 1.1: A shape to be cut into identical pieces.

Example 1.7. Extend the following sequence of numbers: $0, 1, 1, 3, 5, 11, 21, 43, 85, \ldots$. It is a natural human behavior to find a simple explanation consistent with a given observation, i.e., to abstract.[6] Which is the simplest rule that defines the sequence? There may be several answers that make sense.

Example 1.8. Euclid's well-known algorithm for computing the greatest common divisor of two positive integers a and b works as follows: In each step, the larger integer is divided by the smaller integer, and the pair of integers is replaced by the pair consisting of the smaller integer and the remainder of the division. This step is repeated until the remainder is 0. The greatest common divisor is the last non-zero remainder.

[6]Unfortunately, this also leads to over-simplifications and inappropriate generalizations.

Essentially the same algorithm works for two polynomials $a(x)$ and $b(x)$, say with integer (or real) coefficients, where the size of a polynomial is defined to be its degree. In which sense are integer numbers and polynomials similar? At which level of abstraction can they be seen as instantiations of the same abstract concept? As we will see in Chapter 5, the answer is that they are both so-called *Euclidean domains*, which is a special type of a so-called *integral domain*, which in turn is a special case of a *ring*.

Chapter 2

Mathematical Reasoning, Proofs, and a First Approach to Logic

2.1 Mathematical Statements

2.1.1 The Concept of a Mathematical Statement

People make many statements in life, like "I love you", "tomorrow it will rain", "birds can fly", or "Roger Federer is the best tennis player". By making the statement, the person making it intends to claim that it is true. However, most such statements are not sufficiently precise to be considered true or false, and often they are subjective. This is in contrast to mathematical statements.

Definition 2.1. A *mathematical statement* (also called *proposition*) is a statement that is true or false in an absolute, indisputable sense, according to the laws of mathematics.

We often simply say "statement" instead of "mathematical statement". A mathematical statement that is known to be true is often called a *theorem*, a *lemma*, or a *corollary*.[1] A mathematical statement not known (but believed) to be true or false is often called a *conjecture*. An *assumption* is a statement not known to be true but assumed to be true in a certain line of reasoning. Sometimes, before a proof of a true mathematical statement is given, it is also called

[1]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.

assertion[2] or *claim*. Examples of mathematical statements are

- 71 is a prime number.
- If p is a prime number, then $2^p - 1$ is also a prime number.
- Every natural number is the sum of at most four square numbers. (Example: $22 = 4^2 + 2^2 + 1^2 + 1^2$ and $74 = 6^2 + 5^2 + 3^2 + 2^2$.)
- Every even natural number greater than 2 can be expressed as the sum of two primes.[3] For example, $108 = 37 + 71$ and $162 = 73 + 89$.
- Any n lines $\ell_1, \ldots, \ell_n$ in the plane, no two of which are parallel, intersect in one point (see Example 2.4).
- For the chess game there exists a winning strategy for the player making the first move (playing "white").

The first statement is easily shown to be true. The second statement is false, and this can be proved by giving a counter-example: 11 is prime but $2^{11} - 1 = 2047 = 23 \cdot 89$ is not prime.[4] The third statement is true but by no means obvious (and requires a sophisticated proof). The fourth statement is not known to be true (or false). The fifth statement is false. The sixth statement is not known to be true (or false).

Example 2.1. Consider the following statement which sounds like a statement of interest in Computer Science: "There is no algorithm for factoring any n-bit integer in n^3 steps". This is not a precise mathematical statement because its truth (namely the complexity of the best algorithm) generally depends on the particular computational model one is considering. A goal of Theoretical Computer Science (TCS) is therefore to define precise models of computation and the complexity of algorithms, allowing to make such claims precise.

If one makes a statement, say S, for example in the context of these lecture notes, there can be two different meanings. The first meaning is that by stating it one claims that S is true, and the second meaning is simply to discuss the statement itself (for example as an assumption), independently of whether it is true or not. We should try to distinguish clearly between these two meanings.

2.1.2 Composition of Mathematical Statements

We can derive new mathematical statements from given statements. For example, when given three statements S, T, and U, then we can defined the following well-defined statement: Exactly two the statements S, T, and U are true. Four specific forms of derived statements are discussed below. Let S and T be mathematical statements.

[2]German: Behauptung

[3]This statement is called the Goldbach conjecture and is one of the oldest unproven conjectures in mathematics.

[4]$2^p - 1$ is prime for most primes p (e.g. for $2, 3, 5, 7, 13$ and many more), but not all.

- **Negation:** S is false.
- **And:** S and T are (both) true.
- **Or:** At least one of S and T is true.
- **Implication:** If S is true, then T is true.

Examples of such derived statements are:

- "4 is even" is false.
- 4 is an even number *and* 71 is a prime number.
- 5 is an even number *and* 71 is a prime number.
- 5 is an even number *or* 71 is a prime number.

The first statement is false because "4 is even" is true. The second statement is true because both statements "4 is an even number" and "71 is a prime number" are true. In contrast, the third statement is false because the statement "5 is an even number" is false. However, the fourth statement is again true because "71 is a prime number" is true and hence it is irrelevant whether "5 is an even number" is true or false.

For the first three types of statements, there is no particular notation used in mathematics.[5] However, interestingly, for the fourth type (implication) there exists a notation that is often used, namely

$$S \implies T.$$

One also says "S *implies* T". The statement $S \implies T$ is false if S is true and T is false, and in all other three cases, $S \implies T$ is true. In other words, the first three statements below are true while the last one is false.

- 4 is an even number $\implies$ 71 is a prime number.
- 5 is an even number $\implies$ 71 is a prime number.
- 5 is an even number $\implies$ 70 is a prime number.
- 4 is an even number $\implies$ 70 is a prime number.

We point out that $S \implies T$ does *not* express any kind of *causality* like "*because* S is true, T is also true" (but see the discussion in Section 2.2.3).

Similarly, $S \iff T$ means that S is true if and only if T is true. This can equivalently be stated as "S implies T and T implies S."

[5]Note that, when introducing logic and its symbols, we will for example use the symbol $\wedge$ for the logical "and" of two formulas, but we avoid using the symbols of logic outside of logic. Thus, in order to avoid confusion, we avoid writing something like $S \wedge T$ for "S and T."

2.1.3 Mathematical Statements in Computer Science

Many statements relevant in Computer Science are mathematical statements which one would like to prove. We give a few examples of such statements:

- Program P terminates (i.e., does not enter an infinite loop) for all inputs.
- Program P terminates within k computation steps for all inputs.
- Program P computes $f(x)$ for every input x, where f is a function of interest.
- Algorithm A solves problem S within accuracy ϵ.
- The error probability of file transmission system F in a file transmission is at most p (where p can be a function of the file length).
- The computer network C has the property that if any t links are deleted, every node is still connected with every other node.
- Encryption scheme E is secure (for a suitable definition of security).
- Cryptocurrency system C operates correctly as long as a majority of the involved nodes behave honestly, even if all the other nodes behave arbitrarily maliciously.
- Database system D provides data privacy (for a suitable definition of privacy).

Programs, algorithms, encryption schemes, etc., are (complex) discrete mathematical objects, and proving statements like those mentioned above is highly non-trivial. This course is not about programs or algorithms, let alone encryption schemes, but it provides the foundations so that later courses can reason mathematically about these objects.

2.2 The Concept of a Proof

The purpose of a proof is to demonstrate (or prove) a mathematical statement S. In this section we informally discuss the notion of a proof. We also discuss several proof strategies. In Chapter 6 about logic, the notion of a proof in a proof calculus will be formalized.

2.2.1 Examples of Proofs

We already gave examples of proofs in Chapter 1. We give one more simple example.

Example 2.2. Claim: The number $n = 2^{143} - 1$ is not a prime.

Proof: n is divisible by 2047, as one can check by a (for a computer) simple calculation.

That this is true can even be easily seen without doing a calculation, by proving a more general claim of which the above one is a special case:

Claim: n is not prime $\Longrightarrow$ $2^n - 1$ is not prime.[6]

Proof: If n is not a prime, then (by definition of prime numbers) $n = ab$ with $a > 1$ and $a < n$. Now we observe that $2^a - 1$ divides $2^{ab} - 1$:

$$2^{ab} - 1 \;=\; (2^a - 1)\sum_{i=0}^{b-1} 2^{ia} \;=\; (2^a - 1)\left(2^{(b-1)a} + 2^{(b-2)a} + \cdots + 2^a + 1\right)$$

as can easily be verified by a simple calculation. Since $2^a - 1 > 1$ and $2^a - 1 < 2^{ab} - 1$, i.e., $2^a - 1$ is a non-trivial divisor of $2^{ab} - 1$, this means that $2^{ab} - 1$ is not a prime, concluding the proof of the claim.

Let us state a warning. Recall from the previous section that

$$n \text{ is prime} \quad\Longrightarrow\quad 2^n - 1 \text{ is prime}$$

is a *false* statement, even though it may appear at first sight to follow from the above claim. However, we observe that if $S \Longrightarrow T$ is true, then generally it does not follow that if S is false, then T is false.

Example 2.3. An integer n is called a *square* if $n = m \cdot m$ for some integer m. Prove that if a and b are squares, then so is $a \cdot b$.

$$
\begin{array}{lll}
\multicolumn{3}{l}{a \text{ and } b \text{ are squares}} \\
\overset{\cdot}{\Longrightarrow} & a = u \cdot u \text{ and } b = v \cdot v \text{ for some } u \text{ and } v & \text{(def. of squares)} \\
\overset{\cdot}{\Longrightarrow} & a \cdot b = (u \cdot u) \cdot (v \cdot v) & \text{(replace } a \text{ by } u \cdot u \text{ and } b \text{ by } v \cdot v\text{)} \\
\overset{\cdot}{\Longrightarrow} & a \cdot b = (u \cdot v) \cdot (u \cdot v). & \text{(commutative and associative laws)} \\
\overset{\cdot}{\Longrightarrow} & a \cdot b \text{ is a square} & \text{(def. of squares)}
\end{array}
$$

The above proof follows a standard pattern of proofs as a sequence of implications, each step using the symbol $\overset{\cdot}{\Longrightarrow}$. Such a proof step requires that the justification for doing the step is clear. Often one justifies the proof step either in the accompanying text or as a remark on the same line as the implication statement (as in the above proof). But even more often the justification for the step is simply assumed to be understood from the context and not explicitly stated, which can sometimes make proofs hard to follow or even ambiguous.

2.2.2 Examples of False Proofs

As a next motivating example, let us prove a quite surprising assertion.[7]

[6] It is understood that this statement is meant to hold for an arbitrary n.

[7] This example is taken from the book by Matousek and Nesetril.

Example 2.4. Claim: Any n lines $\ell_1, \ldots, \ell_n$ in the plane, no two of which are parallel, intersect in one point (i.e., have one point in common).

Proof: The proof proceeds by induction.[8] The induction basis is the case $n = 2$: Any two non-parallel lines intersect in one point. The induction hypothesis is that any n lines intersect in one point. The induction step states that then this must be true for any $n+1$ lines. The proof goes as follows. By the hypothesis, the n lines $\ell_1, \ldots, \ell_n$ intersect in a point P. Similarly, the n lines $\ell_1, \ldots, \ell_{n-1}, \ell_{n+1}$ intersect in a point Q. The line ℓ_1 lies in both groups, so it contains both P and Q. The same is true for line ℓ_{n-1}. But ℓ_1 and ℓ_{n-1} intersect at a single point, hence $P = Q$. This is the common point of all lines $\ell_1, \ldots, \ell_{n+1}$.

Something must be wrong! (What?) This example illustrates that proofs must be designed with care. Heuristics and intuition, though essential in any engineering discipline as well as in mathematics, can sometimes be wrong.

Example 2.5. In the lecture we present a "proof" for the statement $2 = 1$.

2.2.3 Two Meanings of the Symbol $\Longrightarrow$

It is important to note that the symbol $\Longrightarrow$ is used in the mathematical literature for two different (but related) things:

- to express composed statements of the form $S \Longrightarrow T$ (see Section 2.1.2),
- to express a derivation step in a proof, as above.

To make this explicit and avoid confusion, we use a slightly different symbol $\overset{\cdot}{\Longrightarrow}$ for the second meaning.[9] Hence $S \overset{\cdot}{\Longrightarrow} T$ means that T can be obtained from S by a proof step, and in this case we also know that the statement $S \Longrightarrow T$ is true. However, conversely, if $S \Longrightarrow T$ is true for some statements S and T, there may not exist a proof step demonstrating this, i.e. $S \overset{\cdot}{\Longrightarrow} T$ may not hold.

An analogous comment applies to the symbol $\Longleftrightarrow$, i.e., $S \overset{\cdot}{\Longleftrightarrow} T$ can be used express that T follows from S by a simply proof step, and also S follows from T by a simply proof step.

2.2.4 Proofs Using Several Implications

Example 2.3 showed a proof of a statement of the form $S \Longrightarrow T$ using a sequence of several implications of the form $S \overset{\cdot}{\Longrightarrow} S_2$, $S_2 \overset{\cdot}{\Longrightarrow} S_3$, $S_3 \overset{\cdot}{\Longrightarrow} S_4$, and $S_4 \overset{\cdot}{\Longrightarrow} T$.

[8] Here we assume some familiarity with proofs by induction; in Section 2.6.10 we discuss them in depth.

[9] This notation is not standard and only used in these lecture notes. The symbol $\overset{\cdot}{\Longrightarrow}$ is intentionally chosen very close to the symbol $\Longrightarrow$ to allow someone not used to this to easily overlook the difference.

A proof based on several implications often has a more general form: The implications do not form a linear sequence but a more general configuration, where each implication can assume several of the already proved statements. For example, one can imagine that in order to prove a given statement T, one starts with two (known to be) true statements S_1 and S_2 and then, for some statements $S_3, \ldots, S_7$, proves the following six implications:

- $S_1 \overset{\cdot}{\Longrightarrow} S_3$,
- $S_1 \overset{\cdot}{\Longrightarrow} S_4$,
- $S_2 \overset{\cdot}{\Longrightarrow} S_5$,
- S_3 and $S_5 \overset{\cdot}{\Longrightarrow} S_6$,
- S_1 and $S_4 \overset{\cdot}{\Longrightarrow} S_7$, as well as
- S_6 and $S_7 \overset{\cdot}{\Longrightarrow} T$.

Example 2.6. In the lecture we demonstrate the proof of Example 2.2 in the above format, making every intermediate statement explicit.

2.2.5 An Informal Understanding of the Proof Concept

There is a common informal understanding of what constitutes a proof of a mathematical statement S. Informally, we could define a proof as follows:

Definition 2.2. (Informal.) A *proof* of a statement S is a sequence of simple, easily 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 results in S.

Concrete proofs vary in length and style according to

- which axioms and known facts one is assuming,
- what is considered to be easy to verify,
- how much is made explicit and how much is only implicit in the proof text, and
- to what extent one uses mathematical symbols (like $\overset{\cdot}{\Longrightarrow}$) as opposed to just writing text.

2.2.6 Informal vs. Formal Proofs

Most proofs taught in school, in textbooks, or in the scientific literature are intuitive but quite informal, often not making the axioms and the proof rules explicit. They are usually formulated in common language rather than in a rigorous mathematical language. Such proofs can be considered completely correct

if the reasoning is clear. An informal proof is often easier to read than a pedantic formal proof.

However, a proof, like every mathematical object, can be made rigorous and formally precise. This is a major goal of *logic* (see Section 2.2.7 and Chapter 6). There are at least three (related) reasons for using a more rigorous and formal type of proof.

- *Prevention of errors.* Errors are quite frequently found in the scientific literature. Most errors can be fixed, but some can not. In contrast, a completely formal proof leaves no room for interpretation and hence allows to exclude errors.
- *Proof complexity and automatic verification.* Certain proofs in Computer Science, like proving the correctness of a safety-critical program or the security of an information system, are too complex to be carried out and verified "by hand". A computer is required for the verification. A computer can only deal with rigorously formalized statements, not with semi-precise common language, hence a formal proof is required.[10]
- *Precision and deeper understanding.* Informal proofs often hide subtle steps. A formal proof requires the formalization of the arguments and can lead to a deeper understanding (also for the author of the proof).

There is a trade-off between mathematical rigor and an intuitive, easy-to-read (for humans) treatment. In this course, our goal is to do precise mathematical reasoning, but at the same time we will try to strike a reasonable balance between formal rigor and intuition. In Chapters 3 to 5, our proofs will be informal, and the Chapter 6 on logic is devoted to understanding the notion of a formal proof.

A main problem in teaching mathematical proofs (for example in this course) is that it is hard to define exactly when an informal proof is actually a valid proof. In most scientific communities there is a quite clear understanding of what constitutes a valid proof, but this understanding can vary from community to community (e.g. from physics to Computer Science). A student must learn this culture over the years, starting in high school where proof strategies like proofs by induction have probably been discussed. There is no quick and easy path to understanding exactly what constitutes a proof.

The alternative to a relatively informal treatment would be to do everything rigorously, in a formal system as discussed in Chapter 6, but this would probably turn away most students and would for the most parts simply not be manageable. A book that tries to teach discrete mathematics very rigorously is *A logical approach to discrete math* by Gries and Schneider.

[10] A crucial issue is that the translation of an informal statement to a formal statement can be error-prone.

2.2.7 The Role of Logic

Logic is the mathematical discipline laying the foundations for rigorous mathematical reasoning. Using logic, every mathematical statement as well as a proof for it (if a proof exists) can, in principle, be formalized rigorously. As mentioned above, rigorous formalization, and hence logic, is especially important in Computer Science where one sometimes wants to automate the process of proving or verifying certain statements like the correctness of a program.

Some principle tasks of logic (see Chapter 6) are to answer the following three questions:

1. What is a mathematical statement, i.e., in which language do we write statements?
2. What does it mean for a statement to be true?
3. What constitutes a proof for a statement from a given set of axioms?

Logic (see Chapter 6) defines the syntax of a 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.

2.2.8 Proofs in this Course

As mentioned above, in the literature and also in this course we will see proofs at different levels of detail. This may be a bit confusing for the reader, especially in the context of an exam question asking for a proof. We will try to be always clear about the level of detail that is expected in an exercise or in the exam. For this purpose, we distinguish between the following three levels:

- **Proof sketch** or **proof idea:** The non-obvious ideas used in the proof are described, but the proof is not spelled out in detail with explicit reference to all definitions that are used.
- **Complete proof:** The use of every definition is made explicit. Every proof step is justified by stating the rule or the definition that is applied.
- **Formal proof:** The proof is entirely phrased in a given proof calculus.

Proof sketches are often used when the proof requires some clever ideas and the main point of a task or example is to describe these ideas and how they fit together. Complete proofs are usually used when one systematically applies the definitions and certain logical proof patterns, for example in our treatments of relations and of algebra. Proofs in the resolution calculus in Chapter 6 can be considered to be formal proofs.

2.3 A First Introduction to Propositional Logic

We give a brief introduction to some elementary concepts of logic. We point out that this section is somewhat informal and that in the chapter on logic (Chapter 6) we will be more rigorous. In particular, we will there distinguish between the *syntax* of the language for describing mathematical statements (called formulas) and the *semantics*, i.e., the definition of the meaning (or validity) of a formula. In this section, the boundary between syntax and semantics is (intentionally) not made explicit.

2.3.1 Logical Constants, Operators, and Formulas

Definition 2.3. The logical values (constants) "true" and "false" are usually denoted as 1 and 0, respectively.[11]

One can define operations on logical values:

Definition 2.4.
(i) The *negation* (logical NOT) of a propositional symbol A, denoted as $\neg A$, is true if and only if A is false.
(ii) The *conjunction* (logical AND) of two propositional symbol 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 propositional symbols A and B, denoted $A \vee B$, is true if and only if A or B (or both) are true.[12]

The logical operators are functions, where $\neg$ is a function $\{0,1\} \to \{0,1\}$ and $\wedge$ and $\vee$ are functions $\{0,1\} \times \{0,1\} \to \{0,1\}$. These functions can be described by function tables, as follows:

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, the formula

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

[11] These values 1 and 0 are not meant to be the corresponding numbers, even though the same symbols are used.

[12] Sometimes $\neg A$, $A \wedge B$, and $A \vee B$ are also denoted as $\mathsf{NOT}(A)$, A AND B, and A OR B, respectively, or a similar notation.

has function table

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

A slightly more complicated example is $(A \wedge (\neg B)) \vee (B \wedge (\neg C))$ with function table

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

Definition 2.5. A correctly formed expression involving the propositional symbols $A, B, C, \ldots$ and logical operators is called a *formula* (of propositional logic).

We introduce a new, logical operator, *implication*, denoted as $A \rightarrow B$ and defined by the function table

A	B	$A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

Note that $A \rightarrow B$ is true if and only if A implies B. This means that when A is true, then also B is true. Note that $A \rightarrow B$ is false if and only if A is true and B is false, or, stated differently, if B is false despite that A is true. $A \rightarrow B$ can be understood as an alternative notation for $\neg A \vee B$, which has the same function table.

Example 2.7. Consider the following sentence: If student X reads the lecture notes every week and solves the exercises (A), then student X will get a good grade in the exam (B). This is an example of an implication $A \rightarrow B$. Saying that $A \rightarrow B$ 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 excluded that B is false when A is true. Let's hope the statement $A \rightarrow B$ is true for you :-) .

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

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

Note that $A \leftrightarrow B$ is equivalent to $(A \rightarrow B) \wedge (B \rightarrow A)$ in the sense that the two formulas have the same function table.

We now discuss a few notational simplifications. We have already seen that parentheses 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 $(A \vee B) \vee C$.

There are also precedence rules for logical operators which allow to simplify the notation, in the same sense as in algebra one can write $ab + c$ rather than $(a \cdot b) + c$ because $\cdot$ binds stronger than $+$. However, to keep things simple and avoid confusion, we will generally not make use of such rules, except that we adopt the convention that $\neg$ binds stronger than anything else. For example, we can write $\neg A \wedge B$ instead of $(\neg A) \wedge B$, or we can write $A \rightarrow \neg B$ instead of $A \rightarrow (\neg B)$.

2.3.2 Formulas as Functions

An arithmetic expression such as $(a+b) \cdot c$ can be understood as a function. If we consider as domain the natural numbers $\mathbb{N}$, the arithmetic expression $(a+b) \cdot c$ corresponds to the function $\mathbb{N}^3 \rightarrow \mathbb{N}$ assigning to every triple (a, b, c) the value $(a+b) \cdot c$, for example the value 42 to the triple $(4, 2, 7)$ (because $(4+2) \cdot 7 = 42$).

Analogously, a logical formula such as $(A \vee B) \wedge C$ can be interpreted as a *function* from the set of truth assignments for the proposition symbols A, B, and C to truth values, i.e., as a function $\{0,1\}^3 \rightarrow \{0,1\}$. For example, the function evaluates to 1 for $A = 0$, $B = 1$, and $C = 1$.

Since in propositional logic[13] the domain is finite, a function can be completely characterized by a function table. For example, the function table of the function $\{0,1\}^3 \rightarrow \{0,1\}$ corresponding to the formula $(A \wedge (\neg B)) \vee (B \wedge (\neg C))$ is shown in the previous section.

[13]but not for other logics such as predicate logic

2.3.3 Logical Equivalence and some Basic Laws

Different arithmetic expressions can correspond to the same function. For example, the expressions $(a + b) \cdot c$ and $(c \cdot a) + (b \cdot c)$ denote the same functions. Analogously, different logical formulas can correspond to the same function.

Definition 2.6. Two formulas F and G (in propositional logic) are called *equivalent*, denoted as $F \equiv G$, if they correspond to the same function, i.e., if the truth values are equal for all truth assignments to the propositional symbols appearing in F or G.

For example, it is easy to see that $\wedge$ and $\vee$ are commutative and associative, i.e.,

$$A \wedge B \;\equiv\; B \wedge A \qquad \text{and} \qquad A \vee B \;\equiv\; B \vee A$$

as well as

$$A \wedge (B \wedge C) \;\equiv\; (A \wedge B) \wedge C.$$

Because of this equivalence, we introduce the notational convention that such unnecessary parentheses can be dropped:.

$$A \wedge B \wedge C \;\equiv\; A \wedge (B \wedge C).$$

Similarly we have

$$A \vee (B \vee C) \;\equiv\; (A \vee B) \vee C$$

and can write $A \vee B \vee C$ instead, and we also have

$$\neg(\neg A) \;\equiv\; A.$$

Let us look at some equivalences involving more than one operation, which are easy to check. The operator $\vee$ can be expressed in terms of $\neg$ and $\wedge$, as follows:

$$\neg(A \vee B) \;\equiv\; \neg A \wedge \neg B,$$

which also means that $A \vee B \equiv \neg(\neg A \wedge \neg B)$. In fact, $\neg$ and $\wedge$ are sufficient to express every logical function (of propositional logic). Similarly we have

$$\neg(A \wedge B) \;\equiv\; \neg A \vee \neg B.$$

Example 2.8. $A \leftrightarrow B \;\equiv\; (A \to B) \wedge (B \to A) \;\equiv\; (A \wedge B) \vee (\neg A \wedge \neg B)$.

Example 2.9. Here is a more complicated example which the reader can verify as an exercise:

$$(A \wedge (\neg B)) \vee (B \wedge (\neg C)) \;\equiv\; (A \vee B) \wedge \neg(B \wedge C).$$

The following example shows a distributive law for $\wedge$ and $\vee$. Such laws will be discussed more systematically in Chapter 6.

Example 2.10. $(A \wedge B) \vee C \;\equiv\; (A \vee C) \wedge (B \vee C)$.

We summarize the basic equivalences of propositional logic:

Lemma 2.1.

1) $A \wedge A \equiv A$ *and* $A \vee A \equiv A$ *(idempotence);*
2) $A \wedge B \equiv B \wedge A$ *and* $A \vee B \equiv B \vee A$ *(commutativity of $\wedge$ and $\vee$);*
3) $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ *and* $(A \vee B) \vee C \equiv A \vee (B \vee C)$ *(associativity);*
4) $A \wedge (A \vee B) \equiv A$ *and* $A \vee (A \wedge B) \equiv A$ *(absorption);*
5) $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ *(first distributive law);*
6) $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ *(second distributive law);*
7) $\neg\neg A \equiv A$ *(double negation);*
8) $\neg(A \wedge B) \equiv \neg A \vee \neg B$ *and* $\neg(A \vee B) \equiv \neg A \wedge \neg B$ *(de Morgan's rules).*

2.3.4 Logical Consequence (for Propositional Logic)

For arithmetic expressions one can state relations between them that are more general than equivalence. For example the relation $a+b \leq a+b+(c \cdot c)$ between the expressions $a + b$ and $a + b + (c \cdot c)$. What is meant by the relation is that *for all* values that a, b, and c can take on, the inequality holds, i.e., it holds for the functions corresponding to the expressions.

Analogously, one can state relations between formulas. The perhaps most important relation is logical consequence which is analogous to the relation $\leq$ between arithmetic expressions.

Definition 2.7. A formula G is a *logical consequence*[14] of a formula F, denoted

$$F \models G,$$

if for all truth assignments to the propositional symbols appearing in F or G, the truth value of G is 1 if the truth value of F is 1.

Intuitively, if we would interpret the truth values 0 and 1 as the numbers 0 and 1 (which we don't!), then $F \models G$ would mean $F \leq G$ (as functions).

Example 2.11. $A \wedge B \models A \vee B$.

Example 2.12. Comparing the truth tables of the two formulas $(A \wedge B) \vee (A \wedge C)$ and $\neg B \to (A \vee C)$ one can verify that

$$(A \wedge B) \vee (A \wedge C) \;\models\; \neg B \to (A \vee C).$$

[14] German: (logische) Folgerung, logische Konsequenz

Note that the two formulas are *not* equivalent.

Example 2.13. The following logical consequence, which the reader can prove as an exercise, captures a fact intuitively known to us, namely that implication is transitive:[15]

$$(A \to B) \land (B \to C) \models A \to C.$$

We point out (see also Chapter 6) that two formulas F and G are equivalent if and only if each one is a logical consequence of the other, i.e.,[16]

$$F \equiv G \quad \Longleftrightarrow \quad F \models G \text{ and } G \models F.$$

2.3.5 Lifting Equivalences and Consequences to Formulas

Logical equivalences and consequences continue to hold if the propositional symbols $A, B, C \ldots$ are replaced by other propositional symbols or by formulas $F, G, H \ldots$. At this point, we do not provide a proof of this intuitive fact. For example, because of the logical consequences stated in the previous section we have

$$F \land G \equiv G \land F \qquad \text{and} \qquad F \lor G \equiv G \lor F$$

as well as

$$F \land (G \land H) \equiv (F \land G) \land H$$

for any formulas F, G, and H.

The described lifting is analogous to the case of arithmetic expressions. For example, we have

$$(a + b) \cdot c = (a \cdot c) + (b \cdot c)$$

for any real numbers a, b, and c. Therefore, for any arithmetic expressions f, g, and h, we have

$$(f + g) \cdot h = (f \cdot h) + (g \cdot h).$$

Example 2.14. We give a more complex example of such a lifting. Because of the logical consequence stated in Example 2.13, we have

$$(F \to G) \land (G \to H) \models F \to H$$

for any formulas F, G, and H.

[15] The term "transitive" will be discussed in Chapter 3.

[16] Note that we DO NOT write $F \models G \land G \models F$ because the symbol $\land$ is used only between two formulas in order to form a new (combined) formula, and $F \models G$ and $G \models F$ are not formulas.

2.3.6 Tautologies and Satisfiability

Definition 2.8. A formula F (in propositional logic) is called a *tautology*[17] or *valid*[18] if it is true for all truth assignments of the involved propositional symbols. One often writes $\models F$ to say that F is a tautology.

Example 2.15. The formulas $A \lor (\neg A)$ and $(A \land (A \to B)) \to B$ are tautologies.

One often wants to make statements of the form that some formula F is a tautology. As stated in Definition 2.8, one also says "F is valid" instead of "F is a tautology".

Definition 2.9. A formula F (in propositional logic) is called *satisfiable*[19] if it is true for at least one truth assignment of the involved propositional symbols, and it is called *unsatisfiable* otherwise.

The symbol $\top$ is sometimes used to denote a tautology, and the symbol $\bot$ is sometimes used to denote an unsatisfiable formula. One sometimes writes $F \equiv \top$ to say that F is a tautology, and $F \equiv \bot$ to say that F is unsatisfiable. For example, for any formula F we have

$$F \lor \neg F \equiv \top \qquad \text{and} \qquad F \land \neg F \equiv \bot.$$

Example 2.16. The formula $(A \land \neg A) \land (B \lor C)$ is unsatisfiable, and the formula $A \land B$ is satisfiable.

The following lemmas state two simple facts that follow immediately from the definitions. We only prove the second one.

Lemma 2.2. *A formula F is a tautology if and only if $\neg F$ is unsatisfiable.*

Lemma 2.3. *For any formulas F and G, $F \to G$ is a tautology if and only if $F \models G$.*

Proof. The lemma has two directions which we need to prove. To prove the first direction ($\Longrightarrow$), assume that $F \to G$ is a tautology. Then, for any truth assignment to the propositional symbols, the truth values of F and G are either both 0, or 0 and 1, or both 1 (but not 1 and 0). In each of the three cases it holds that G is true if F is true, i.e., $F \models G$. To prove the other direction ($\Longleftarrow$), assume $F \models G$. This means that for any truth assignment to the propositional symbols, the truth values of G is 1 if it is 1 for F. In other words, there is no

[17] German: Tautologie

[18] German: allgemeingültig

[19] German: erfüllbar

truth assignment such that the truth value of F is 1 and that of G is 0. This means that the truth value of $F \to G$ is always 1, which means that $F \to G$ is a tautology. □

2.3.7 Logical Circuits *

A logical formula as discussed above can be represented as a tree where the leaves correspond to the propositions and each node corresponds to a logical operator. Such a tree can be implemented as a digital circuit where the operators correspond to the logical gates. This topic will be discussed in a course on the design of digital circuits[20]. The two main components of digital circuits in computers are such logical circuits and memory cells.

2.4 A First Introduction to Predicate Logic

The elements of logic we have discussed so far belong to the realm of so-called *propositional logic*[21]. Propositional logic is not sufficiently expressive to capture most statements of interest in mathematics in terms of formulas. For example, the statement *"There are infinitely many prime numbers"* cannot be expressed as a formula in propositional logic (though it can of course be expressed as a sentence in common language). We need *quantifiers*[22], *predicates*, and *functions*. The corresponding extension of propositional logic is called *predicate logic*[23] and is substantially more involved than propositional logic. Again, we refer to Chapter 6 for a more thorough discussion.

2.4.1 Predicates

Let us consider a non-empty 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 2.10. A k-ary *predicate*[24] 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).

For example, for $U = \mathbb{N}$ we can consider the unary ($k = 1$) predicate $\texttt{prime}(x)$ defined by

$$\texttt{prime}(x) = \begin{cases} 1 & \text{if } x \text{ is prime} \\ 0 & \text{else.} \end{cases}$$

[20]German: Digitaltechnik
[21]German: Aussagenlogik
[22]German: Quantoren
[23]German: Prädikatenlogik
[24]German: Prädikat

Similarly, one can naturally define the unary predicates $\texttt{even}(x)$ and $\texttt{odd}(x)$.

For any universe U with an order relation $\leq$ (e.g. $U = \mathbb{N}$ or $U = \mathbb{R}$), the binary (i.e., $k = 2$) predicate $\texttt{less}(x,y)$ can be defined as

$$\texttt{less}(x,y) = \begin{cases} 1 & \text{if } x < y \\ 0 & \text{else.} \end{cases}$$

However, in many cases we write binary predicates in a so-called "infix" notation, i.e., we simply write $x < y$ instead of $\texttt{less}(x,y)$.

For the universe of all human beings, we can define a binary predicate $\texttt{child}$ as follows: $\texttt{child}(x,y) = 1$ if and only if x is y's child. One can similarly define predicates $\texttt{parent}$, $\texttt{grandparent}$, etc.

2.4.2 Functions and Constants

In predicate logic one can also use functions on U and constants (i.e., fixed elements) in U. For example, if the universe is $U = \mathbb{N}$, we can use the functions $\texttt{add}$ addition and multiplication $\texttt{mult}$ and the constants 3 and 5. The formula

$$\texttt{less}(\texttt{add}(x,3), \texttt{add}(x,5))$$

can also be written in infix notation as

$$x + 3 < x + 5.$$

This is a true statement for *every* value x in U. In the next section we see how we can express this as a formula.

2.4.3 The Quantifiers ∃ and ∀

Definition 2.11. For a universe U and predicate $P(x)$ we define the following logical statements:[25]

$\forall x\, P(x)$	stands for:	$P(x)$ is true *for all* x in U.
$\exists x\, P(x)$	stands for:	$P(x)$ is true for *some* x in U, i.e., there *exists* an $x \in U$ for which $P(x)$ is true.

More generally, for a formula F with a variable x, which for each value $x \in U$ is either true or false, the formula $\forall x\, F$ is true if and only if F is true for all x in U, and the formula $\exists x\, F$ is true if and only if F is true for some x in U.

[25]In the literature one also finds the notations $\forall x\colon P(x)$ and $\forall x.\, P(x)$ instead of $\forall x\, P(x)$, and similarly for $\exists$.

Example 2.17. Consider the universe $U = \mathbb{N}$. Then $\forall x\ (x \geq 0)$ is true.[26] Also, $\forall x\ (x \geq 2)$ is false, and $\exists x\ (x + 5 = 3)$ is false.

The name of the variable x is irrelevant. For example, the formula $\exists x\ (x+5 = 3)$ is equivalent to the formula $\exists y\ (y + 5 = 3)$. The formula could be stated in words as: "There exists a natural number (let us call it y) which, if 5 is added to it, the result is 3." How the number is called, x or y or z, is irrelevant for the truth or falsity of the statement. (Of course the statement is false; it would be true if the universe were the integers $\mathbb{Z}$.)

Sometimes one wants to state only that a certain formula containing x is true for all x that satisfy a certain condition. For example, to state that $x^2 \geq 25$ whenever $x \geq 5$, one can write

$$\forall x\ \big((x \geq 5) \to (x^2 \geq 25)\big).$$

A different notation sometimes used to express the same statement is to state the condition on x directly after the quantifier, followed by ":"[27]

$$\forall x \geq 5:\quad (x^2 \geq 25).$$

2.4.4 Nested Quantifiers

Quantifiers can also be nested[28]. For example, if $P(x)$ and $Q(x, y)$ are predicates, then

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

is a logical formula.

Example 2.18. The formula

$$\forall x\ \exists y\ \ (y < x)$$

states that for every x there is a smaller y. In other words, it states that there is no smallest x (in the universe under consideration). This formula is true for the universe of the integers or the universe of the real numbers, but it is false for the universe $U = \mathbb{N}$.

Example 2.19. For the universe of the natural numbers, $U = \mathbb{N}$, the predicate $\texttt{prime}(x)$ can be defined as follows:[29]

$$\texttt{prime}(x)\ \stackrel{\text{def}}{\iff}\ x > 1\ \wedge\ \forall y\,\forall z\ \big((yz = x) \to ((y = 1) \vee (z = 1))\big).$$

[26]But note that $\forall x\ (x \geq 0)$ is false for the universe $U = \mathbb{R}$.

[27]We don't do this.

[28]German: verschachtelt

[29]We use the symbol "$\stackrel{\text{def}}{\iff}$" if the object on the left side is defined as being equivalent to the object on the right side.

Example 2.20. Fermat's last theorem can be stated as follows: For universe $\mathbb{N} \setminus \{0\}$,[30]

$$\neg\big(\exists\, x\ \exists y\ \exists z\ \exists n\ \ (n \geq 3\ \wedge\ x^n + y^n = z^n)\big).$$

Example 2.21. The statement "for every natural number there is a larger prime" can be phrased as

$$\forall x\ \exists y\ \big(y > x\ \wedge\ \texttt{prime}(y)\big)$$

and means that there is no largest prime and hence that there are infinitely many primes.

If the universe is $\mathbb{N}$, then one sometimes uses m, n, or k instead of x and y. The above formula could hence equivalently be written as

$$\forall m\ \exists n\ \big(n > m\ \wedge\ \texttt{prime}(n)\big).$$

Example 2.22. Let $U = \mathbb{R}$. What is the meaning of the following formula, and does it correspond to a true statement?

$$\forall x\ \big(x = 0\ \vee\ \exists y\ (xy = 1)\big)$$

Example 2.23. What is the meaning of the following formula, and for which universes is it true (or false)?

$$\forall x\ \forall y\ \big(\,(x < y) \to \exists z((x < z) \wedge (z < y))\,\big)$$

2.4.5 Interpretation of Formulas

A formula generally has some "free parts" that are left open for interpretation. To begin with, the universe is often not fixed, but it is also quite common to write formulas when the universe is understood and fixed. Next, we observe that the formula

$$\forall x\ \big(P(x) \to Q(x)\big)$$

contains the predicate symbols P and Q which can be interpreted in different ways. Depending on the choice of universe and on the interpretation of P and Q, the formula can either be true or false. For example let the universe be $\mathbb{N}$ and let $P(x)$ mean that "x is divisible by 4". Now, if $Q(x)$ is interpreted as "x is odd", then $\forall x\ (P(x) \to Q(x))$ is false, but if $Q(x)$ is interpreted as "x is even", then $\forall x\ (P(x) \to Q(x))$ is true. However, the precise definition of an interpretation is quite involved and deferred to Chapter 6.

[30]In formulas with sequences of quantifiers of the same type one sometimes omits parentheses or even multiple copies of the quantifier. For example one writes $\exists xyz$ instead of $\exists x\,\exists y\,\exists z$. We will not use such a convention in this course.

2.4.6 Tautologies and Satisfiability

The concepts interpretation, tautology, and satisfiability for predicate logic will be defined in Chapter 6.

Informally, a formula is *satisfiable* if there is an interpretation of the involved symbols that makes the formula true. Hence $\forall x\ (P(x) \rightarrow Q(x))$ is satisfiable as shown above. Moreover, a formula is a *tautology* (or *valid*) if it is true for all interpretations, i.e., for all choices of the universe and for all interpretations of the predicates.[31]

We will use the terms "tautology" and "valid" interchangeably. For example,

$$\forall x\ \big((P(x) \wedge Q(x)) \rightarrow (P(x) \vee Q(x))\big)$$

is a tautology, or valid.

2.4.7 Equivalence and Logical Consequence

One can define the equivalence of formulas and logical consequence for predicate logic analogously to propositional logic, but again the precise definition is quite involved and deferred to Chapter 6. Intuitively, two formulas are equivalent if they evaluate to the same truth value for any interpretation of the symbols in the formula.

Example 2.24. Recall Example 2.22. The formula can be written in an equivalent form, as:

$$\forall x\ \big(x = 0\ \vee\ \exists y\ (xy = 1)\big)\ \equiv\ \forall x\ \big(\neg(x = 0)\ \rightarrow\ \exists y\ (xy = 1)\big).$$

The order of identical quantifiers does not matter, i.e., we have for example:

$$\exists x \exists y\ P(x, y)\ \equiv\ \exists y \exists x\ P(x, y) \quad \text{and} \quad \forall x \forall y\ P(x, y)\ \equiv\ \forall y \forall x\ P(x, y).$$

A simple example of a logical consequence is

$$\forall x\ P(x)\ \models\ \exists x\ P(x).$$

It holds because if $P(x)$ is true for all x in the universe, then it is also true for some (actually an arbitrary) x. (Recall that the universe is non-empty.)

Some more involved examples of equivalences and logical consequences are stated in the next section.

[31] We will see in Chapter 6 that predicate logic also involves function symbols, and an interpretation also instantiates the function symbols by concrete functions.

2.4.8 Some Useful Rules

We list a few useful rules for predicate logic. This will be discussed in more detail in Chapter 6. We have

$$\forall x\ P(x)\ \wedge\ \forall x\ Q(x)\quad \equiv\quad \forall x\ \big(P(x) \wedge Q(x)\big)$$

since if $P(x)$ is true for all x and also $Q(x)$ is true for all x, then $P(x) \wedge Q(x)$ is true for all x, and vice versa. Also,[32]

$$\exists x\ \big(P(x) \wedge Q(x)\big)\quad \models\quad \exists x\ P(x)\ \wedge\ \exists x\ Q(x)$$

since, no matter what P and Q actually mean, any x that makes $P(x) \wedge Q(x)$ true (according to the left side) also makes $P(x)$ and $Q(x)$ individually true. But, in contrast, $\exists x\ (P(x) \wedge Q(x))$ is *not* a logical consequence of $\exists x\ P(x)\ \wedge\ \exists x\ Q(x)$, as the reader can verify. We can write

$$\exists x\ P(x)\ \wedge\ \exists x\ Q(x)\quad \not\models\quad \exists x\ (P(x) \wedge Q(x)).$$

We also have:

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

and

$$\neg \exists x\ P(x)\ \equiv\ \forall x\ \neg P(x).$$

The reader can prove as an exercise that

$$\exists y\ \forall x\ P(x, y)\ \models\ \forall x\ \exists y\ P(x, y)$$

but that

$$\forall x\ \exists y\ P(x, y)\ \not\models\ \exists y\ \forall x\ P(x, y).$$

2.5 Logical Formulas vs. Mathematical Statements

A logical formula is generally *not* a mathematical statement because the symbols in it can be interpreted differently, and depending on the interpretation, the formula is true or false. Without fixing an interpretation, the formula is not a mathematical statement.

2.5.1 Fixed Interpretations and Formulas as Statements

If for a formula F the interpretation (including the universe and the meaning of the predicate and function symbols) is fixed, then this can be a mathematical

[32] We point out that defining logical consequence for predicate logic is quite involved (see Chapter 6), but intuitively it should be quite clear.

statement that is either true or false. Therefore, if an interpretation is understood, we can use formulas as mathematical statements, for example in a proof with implication steps. In this case (but only if a fixed interpretation is understood) it is also meaningful to say that a formula is true or that it is false.

Example 2.25. For the universe $\mathbb{N}$ and the usual interpretation of $<$ and $>$, the formula $\exists n\ (n < 4\ \wedge\ n > 5)$ is false and the formula $\forall n\ (n > 0 \rightarrow (\exists m\ \ m < n))$ is true.

2.5.2 Mathematical Statements about Formulas

As mentioned, logical formulas are often not mathematical statements themselves, but one makes mathematical statements *about* formulas. Examples of such mathematical statements are:

- F is valid (i.e., a tautology, also written as $\models F$),
- F is unsatisfiable,
- $F \models G$.

The statement "F is valid" is a mathematical statement (*about* the formula F). Therefore we may for example write

$$F \text{ is valid} \quad\Longrightarrow\quad G \text{ is valid}, \tag{2.1}$$

as a mathematical statement about the formulas F and G. This statement is different from the statement $F \models G$. In fact, for any formulas F and G, the statement $F \models G$ implies statement (2.1), but the converse is generally false:

Lemma 2.4. *For any two formulas F and G, if $F \models G$, then (2.1) is true.*

Proof. $F \models G$ states that for every interpretation, if F is true (for that interpretation), then also G is true (for that interpretation). Therefore, if F is true for every interpretation, then also G is true for every interpretation, which is statement (2.1). □

2.6 Some Proof Patterns

In this section we discuss a few important proof patterns (which we could also call proof methods or proof techniques). Such a proof pattern can be used to prove one step within a longer proof, or sometimes also to directly prove a theorem of interest. Many proof patterns correspond to logical deduction rules. One can define a logical calculus consisting of such deduction rules, but we will defer the discussion of this topic to Chapter 6. Often, a given statement can be proved in different ways, i.e., by using different proof patterns.

2.6.1 Composition of Implications

We first explain why the composition of implications, as occurring in many proofs, is sound.

Definition 2.12. The proof step of *composing implications* is as follows: If $S \Longrightarrow T$ and $T \Longrightarrow U$ are both true, then one concludes that $S \Longrightarrow U$ is also true.

The soundness of this principle is explained by the following lemma of propositional logic which was already stated in Example 2.13.

Lemma 2.5. $(A \rightarrow B) \wedge (B \rightarrow C) \quad\models\quad A \rightarrow C$.

Proof. One writes down the truth tables of the formulas $(A \rightarrow B) \wedge (B \rightarrow C)$ and $A \rightarrow C$ and checks that whenever the first evaluates to true, then also the second evaluates to true. □

2.6.2 Direct Proof of an Implication

Many statements of interest (as intermediate steps or as the final statement of interest) are implications of the form $S \Longrightarrow T$ for some statements S and T.[33]

Definition 2.13. A *direct proof* of an implication $S \Longrightarrow T$ works by *assuming* S and then proving T under this assumption.

2.6.3 Indirect Proof of an Implication

Definition 2.14. An *indirect proof* of an implication $S \Longrightarrow T$ proceeds by assuming that T is false and proving that S is false, under this assumption.

The soundness of this principle is explained by the following simple lemma of propositional logic, where A stands for "statement S is true" and B stands for "statement T is true".

Lemma 2.6. $\neg B \rightarrow \neg A \quad\models\quad A \rightarrow B$.

Proof. One can actually prove the stronger statement, namely that $\neg B \rightarrow \neg A \ \equiv\ A \rightarrow B$, simply by examination of the truth table which is identical for both formulas $\neg B \rightarrow \neg A$ and $A \rightarrow B$. □

Example 2.26. Prove the following claim: If $x > 0$ is irrational, then also $\sqrt{x}$ is irrational. The indirect proof proceeds by assuming that $\sqrt{x}$ is not irrational and

[33]Recall Section 2.1.2.

showing that then x is also *not* irrational. Here "not irrational" means rational, i.e., we prove

$$\sqrt{x} \text{ is rational} \implies x \text{ is rational}$$

Assume hence that $\sqrt{x}$ is rational, i.e., that $\sqrt{x} = m/n$ for $m, n \in \mathbb{Z}$. This means that $x = m^2/n^2$, i.e., x is the quotient of two natural numbers (namely m^2 and n^2) and thus is rational. This completes the proof of the claim.

2.6.4 Modus Ponens

Definition 2.15. A proof of a statement S by use of the so-called *modus ponens* proceeds in three steps:

1. Find a suitable mathematical statement R.
2. Prove R.
3. Prove $R \Longrightarrow S$.

The soundness of this principle is explained by the following lemma of propositional logic. Again, the proof is by a simple comparison of truth tables.

Lemma 2.7. $A \wedge (A \to B) \quad \models \quad B.$

Examples will be discussed in the lecture and the exercises.

2.6.5 Case Distinction

Definition 2.16. A proof of a statement S by *case distinction* proceeds in three steps:

1. Find a finite list $R_1, \ldots, R_k$ of mathematical statements (the *cases*).
2. Prove that at least one of the R_i is true (at least one case occurs).
3. Prove $R_i \Longrightarrow S$ for $i = 1, \ldots, k$.

More informally, one proves for a *complete list* of cases that the statement S holds in all the cases.

The soundness of this principle is explained by the following lemma of propositional logic.

Lemma 2.8. *For every k we have*

$$(A_1 \vee \cdots \vee A_k) \wedge (A_1 \to B) \wedge \cdots \wedge (A_k \to B) \quad \models \quad B.$$

Proof. For a fixed k (say $k = 2$) one can verify the statement by examination of the truth table. The statement for general k can be proved by induction (see Section 2.6.10). □

Note that for $k = 1$ (i.e., there is only one case), case distinction corresponds to the modus ponens discussed above.

Example 2.27. Prove the following statement S: The 4th power of every natural number n, which is not divisible by 5, is one more than a multiple of 5.

To prove the statement, let $n = 5k + c$, where $1 \leq c \leq 4$. Using the usual binomial formula $(a+b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$ we obtain:

$$n^4 = (5k+c)^4 = 5^4k^4 + 4 \cdot 5^3k^3c + 6 \cdot 5^2k^2c^2 + 4 \cdot 5kc^3 + c^4.$$

Each summand is divisible by 5, except for the last term c^4. The statement S is hence equivalent to the statement that c^4 is one more than a multiple of 5, for $1 \leq c \leq 4$.

This statement S can be proved by case distinction, i.e., by considering all four choices for c. For $c = 1$ we have $c^4 = 1$, which is trivially one more than a multiple of 5. For $c = 2$ we have $c^4 = 16$, which is also one more than a multiple of 5. The same is true for $c = 3$ where $c^4 = 81$ and for $c = 4$ where $c^4 = 256$. This concludes the proof.

With a few insights from number theory and algebra we will see later that the above statement holds when 5 is replaced by any odd prime number.

2.6.6 Proofs by Contradiction

Definition 2.17. A *proof by contradiction* of a statement S proceeds in three steps:

1. Find a suitable mathematical statement T.
2. Prove that T is false.
3. Assume that S is false and prove (from this assumption) that T is true (a contradiction).

In many cases, the proof steps appear in a different order: One starts from assuming that S is false, derives statements from it until one observes that one of these statements is false (i.e., is the statement T in the above description). In this case, the fact that T is false (step 2) is obvious and requires no proof.

The soundness of this principle is explained by the following lemma of propositional logic which can again be proved by comparing the truth tables of the involved formulas.

Lemma 2.9. $(\neg A \to B) \wedge \neg B \quad \models \quad A.$

Since $\neg A \to B$ is equivalent to $A \vee B$, the principle of a proof by contradiction can alternatively be described as follows: To prove S, one proves for some statement T that either S or T is true (or both) and that T is false. This is justified because we have

$$(A \vee B) \wedge \neg B \quad \models \quad A.$$

Example 2.28. We discuss the classical proof of three statement that $\sqrt{2}$ is irrational. (This is the statement S to be proved.) Recall (from basic number theory) that a number a is rational if and only if $a = m/n$ (i.e., $m = an$) for two relatively prime[34] integers m and n (i.e., with $\gcd(m,n) = 1$).[35]

The proof by contradiction starts by assuming that S is false and deriving, from this assumption, a false statement T. In the following derivation we may use formulas as a compact way of writing statements, but the derivation itself is "normal" mathematical reasoning and is not to be understood as a formula-based logical reasoning.[36]

$$\begin{aligned} S \text{ is false} \quad &\Longleftrightarrow \quad \sqrt{2} \text{ is rational} \\ &\Longleftrightarrow \quad \exists m\, \exists n\, \big(m^2 = 2n^2 \;\wedge\; \gcd(m,n) = 1\big) \end{aligned}$$

We now consider, in isolation, the statement $m^2 = 2n^2$ appearing in the above formula, derive from it another statement (namely $\gcd(m,n) \geq 2$), and then plug this into the above formula. Each step below is easy to verify. For arbitrary m and n we have

$$\begin{aligned} m^2 = 2n^2 \quad &\Longrightarrow \quad m^2 \text{ is even} \\ &\Longrightarrow \quad m \text{ is even} \\ &\Longrightarrow \quad 4 \text{ divides } m^2 \\ &\Longrightarrow \quad 4 \text{ divides } 2n^2 \qquad \text{(also using } m^2 = 2n^2\text{)} \\ &\Longrightarrow \quad 2 \text{ divides } n^2 \\ &\Longrightarrow \quad n \text{ is even} \\ &\Longrightarrow \quad \gcd(m,n) \geq 2 \qquad \text{(also using that } m \text{ is even)} \end{aligned}$$

Hence we have

$$\exists m\, \exists n\, \big(m^2 = 2n^2 \wedge \gcd(m,n) = 1\big)$$

$$\Longrightarrow \underbrace{\exists m\, \exists n\, \big(m^2 = 2n^2 \;\wedge\; \underbrace{\gcd(m,n) \geq 2 \;\wedge\; \gcd(m,n) = 1}_{\text{false for arbitrary } m \text{ and } n}\big)}_{\text{statement } T\text{, which is false}}.$$

This concludes the proof by contradiction.

[34] German: teilerfremd

[35] $\gcd(m,n)$ denotes the greatest common divisor of m and n (see Section 4.2.3).

[36] We can write $\Longleftrightarrow$ if the implication holds in both directions, but it would be sufficient to always replace $\Longleftrightarrow$ by $\Longrightarrow$.

2.6.7 Existence Proofs

Definition 2.18. Consider a set $\mathcal{X}$ of parameters and for each $x \in \mathcal{X}$ a statement, denoted S_x. An *existence proof* is a proof of the statement that S_x is true for at least one $x \in \mathcal{X}$. An existence proof is *constructive* if it exhibits an a for which S_a is true, and otherwise it is *non-constructive*.

Example 2.29. Prove that there exists a prime[37] number n such that $n - 10$ and $n + 10$ are also primes, i.e., prove

$$\exists n\, \big(\underbrace{\mathtt{prime}(n) \wedge \mathtt{prime}(n-10) \wedge \mathtt{prime}(n+10)}_{S_n}\big).$$

A constructive proof is obtained by giving the example $n = 13$ and verifying that S_{13} is true.

Example 2.30. We prove that there are infinitely many primes by involving a non-constructive existence proof.[38] This statement can be rephrased as follows: For every number m there exists a prime p greater than m; as a formula:

$$\forall m\, \exists p\, \big(\underbrace{\mathtt{prime}(p) \;\wedge\; p > m}_{S_p}\big).$$

To prove this, consider an arbitrary but fixed number m and consider the statements S_p parameterized by p: There exists a prime p greater than m, i.e., such that $\mathtt{prime}(p) \;\wedge\; p > m$ is true.

To prove this, we use the known fact (which has been proved) that every natural number $n \geq 2$ has at least one prime divisor. We consider the specific number $m! + 1$ (where $m! = 2 \cdot 3 \cdots (m-1) \cdot m$). We observe that for all k in the range $2 \leq k \leq m$, k does not divide $m! + 1$. In particular, no prime smaller than m divides $m!+1$. Because $m!+1$ has at least one prime divisor, there exists a prime p greater than m which divides $m! + 1$. Hence there exists a prime p greater than m.[39]

2.6.8 Existence Proofs via the Pigeonhole Principle

The following simple but powerful fact is known as the *pigeonhole principle*[40]. This principle is used as a proof technique for certain existence proofs.[41]

[37] Recall that $\mathtt{prime}(n)$ is the predicate that is true if and only if n is a prime number.

[38] See also Example 2.21, where different variable names are used.

[39] Note that p is not known explicitly, it is only known to exist. In particular, p is generally not equal to $m! + 1$.

[40] German: *Schubfachprinzip*

[41] This principle is often described as follows: If there are more pigeons than pigeon holes, then there must be at least one pigeon hole with more than one pigeon in it. Hence the name of the principle.

Theorem 2.10. *If a set of n objects is partitioned into $k < n$ sets, then at least one of these sets contains at least $\lceil \frac{n}{k} \rceil$ objects.*[42]

Proof. The proof is by contradiction. Suppose that all sets in the partition have at most $\lceil \frac{n}{k} \rceil - 1$ objects. Then the total number of objects is at most $k\big(\lceil \frac{n}{k} \rceil - 1\big)$, which is smaller than n because

$$k\left(\left\lceil \frac{n}{k} \right\rceil - 1\right) < k\left(\left(\frac{n}{k} + 1\right) - 1\right) = k\left(\frac{n}{k}\right) = n.$$

□

Example 2.31. Claim: Among 100 people, there are at least nine who were born in the same month. The claim can be equivalently stated as an existence claim: Considering any 100 people, *there exists* a month in which at least nine of them have their birthday.

Proof. Set $n = 100$ and $k = 12$, and observe that $\lceil 100/12 \rceil = 9$. □

Example 2.32. Claim: In any subset A of $\{1, 2, \ldots, 2n\}$ of size $|A| = n + 1$, there exist distinct $a, b \in A$ such that $a \mid b$ (a divides b).[43]

For example, in the set $\{2, 3, 5, 7, 9, 10\}$ we see that $3 \mid 9$.

Proof. We write every $a_i \in A$ as $2^{e_i} u_i$ with u_i odd. There are only n possible values $\{1, 3, 5, \ldots, 2n - 1\}$ for u_i. Thus there must exist two numbers a_i and a_j with the same odd part ($u_i = u_j$). Therefore one of them has fewer factors 2 than the other and must hence divide it. □

Example 2.33. Let $a_1, a_2, \ldots, a_n$ be a sequence of numbers (real or integer). A subsequence of length k of this sequence is a sequence of the form $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$, where $1 \leq i_1 < i_2 < \cdots < i_k \leq n$. A sequence is called strictly increasing (decreasing) if each term is strictly greater (smaller) than the preceding one. For example, the sequence $3, 8, 2, 11, 1, 5, 7, 4, 14, 9$ contains the increasing subsequences $3, 5, 7, 9$ and $2, 5, 7, 14$ and the decreasing subsequences $3, 2, 1$ and $8, 5, 4$.

Claim: Every sequence of $m^2 + 1$ *distinct* numbers (real or integer) contains either an increasing or a decreasing subsequence of length $m + 1$. (Note that in the above example, $m = 3$ and $m^2 + 1 = 10$, and there is indeed an increasing subsequence of length 4.)

Proof. We can associate with every position ($1 \leq \ell \leq m^2 + 1$) a pair (i_ℓ, d_ℓ), where i_ℓ (d_ℓ) is the length of the longest increasing (decreasing) subsequence beginning at position ℓ. The proof is by contradiction. Suppose the claim is false, i.e., $1 \leq i_\ell \leq m$ and $1 \leq d_\ell \leq m$ for all ℓ. Then there are at most m^2 pairs

[42] In the literature, the pigeon hole principle often states only that there must be a set containing at least *two* elements.

[43] Note that this is tight. If we lower $|A|$ from $n + 1$ to n, then the set $A = \{n, n+1, \ldots, 2n-1\}$ contains no $a, b \in A$ such that $a \mid b$.

(i_ℓ, d_ℓ) that can occur. Thus the pigeonhole principle guarantees that there must be two indices $s < t$ with $(i_s, d_s) = (i_t, d_t)$. But this leads to a contradiction. Because the numbers are distinct, either $a_s < a_t$ or $a_s > a_t$. If $a_s < a_t$, then, since $i_s = i_t$, an increasing subsequence of length $i_t + 1$ can be built starting at position s, taking a_s followed by the increasing subsequence beginning at a_t. This is a contradiction. A similar contradiction is obtained if $a_s > a_t$. □

2.6.9 Proofs by Counterexample

Proofs by counterexample are a specific type of constructive existence proof, namely the proof that a counterexample exists.

Definition 2.19. Consider a set $\mathcal{X}$ of parameters and for each $x \in \mathcal{X}$ a statement, denoted S_x. A *proof by counterexample* is a proof of the statement that S_x is *not* true for all $x \in \mathcal{X}$, by exhibiting an a (called *counterexample*) such that S_a is false.

Note that a proof by counterexample corresponds to an existence proof.

Example 2.34. Prove or disprove that for every integer n, the number $n^2 - n + 41$ is prime, i.e., prove

$$\forall n \ \texttt{prime}(n^2 - n + 41).$$

One can verify the quite surprising fact that $\texttt{prime}(n^2 - n + 41)$ is true for $n = 1, 2, 3, 4, 5, 6, \ldots$, for as long as the reader has the patience to continue to do the calculation. But is it true for all n? To prove that the assertion $\forall n \ \texttt{prime}(n^2 - n + 41)$ is false, i.e., to prove

$$\neg\forall n \ \texttt{prime}(n^2 - n + 41),$$

it suffices to exhibit a counterexample, i.e., an a such that $\neg\texttt{prime}(a^2 - a + 41)$. The smallest such a is $a = 41$; note that $41^2 - 41 + 41 = 41^2$ is not a prime.

Example 2.35. Prove or disprove that every positive integer ≥ 10 can be written as the sum of at most three squares (e.g. $10 = 3^2 + 1^2$, $11 = 3^2 + 1^2 + 1^2$, $12 = 2^2 + 2^2 + 2^2$, $13 = 3^2 + 2^2$, and $14 = 3^2 + 2^2 + 1^2$.). The statement can be written as

$$\forall n \ \big(n \geq 10 \ \rightarrow \ \exists x \ \exists y \ \exists z \ \ (n = x^2 + y^2 + z^2)\big).$$

The statement is false because $n = 15$ is a counterexample.

2.6.10 Proofs by Induction

One of the most important proof technique in discrete mathematics are proofs by induction, which are used to prove statements of the form $\forall n \ P(n)$, where the universe U is the set $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ of natural numbers. Alternatively, it can also be the set $\{1, 2, 3, \ldots\}$ of positive integers, in which case $P(0)$ below must be replaced by $P(1)$. More generally, it can be the set $\{k, k+1, k+2, \ldots\}$ for some k.

A proof by induction consists of two steps:

Proof by induction:

1. *Basis step.*[44] Prove $P(0)$.
2. *Induction step.* Prove that for any arbitrary n we have $P(n) \implies P(n+1)$.

The induction step is performed by assuming $P(n)$ (for an arbitrary n) and deriving $P(n+1)$. This proof technique is justified by the following theorem.[45]

Theorem 2.11. *For the universe* $\mathbb{N}$ *and an arbitrary unary predicate* P *we have*

$$P(0) \wedge \forall n\,(P(n) \rightarrow P(n+1)) \implies \forall n\, P(n).$$

Let us discuss a few examples of proofs by induction.

Example 2.36. Prove that $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ holds for all n. To do a proof by induction, let $P(n)$ be defined by $P(n) = 1$ if and only if $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$. Step 1 is to prove $P(0)$; this holds trivially because the sum consists of the single term $2^0 = 1$ and we also have $2^{0+1} - 1 = 2 - 1 = 1$. Step 2 is to prove that for an arbitrary n, under the assumption $P(n)$, i.e., $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$, also $P(n+1)$ is true, i.e., $\sum_{i=0}^{n+1} 2^i = 2^{(n+1)+1} - 1$:

$$\sum_{i=0}^{n+1} 2^i = \sum_{i=0}^{n} 2^i + 2^{n+1} = (2^{n+1} - 1) + 2^{n+1} = 2^{(n+1)+1} - 1.$$

This concludes the proof of $\forall n\, P(n)$.

Example 2.37. Determine the set of postages you can generate using only 4-cent stamps and 5-cent stamps!

Obviously 1, 2, 3, 6, 7, and 11 cents cannot be obtained, while 4, 5, $8 = 4+4$, $9 = 4+5$, $10 = 5+5$, $12 = 4+4+4$, $13 = 4+4+5$, $14 = 4+5+5$, and $15 = 5+5+5$, can be obtained. One can prove by induction that all amounts of 15 or more cents can indeed be obtained.

Let $P(n)$ be the predicate that is true if and only if there exists a decomposition of $n+15$ into summands 4 and 5. We have just seen that $P(0)$ is true. To prove the induction step, i.e., $\forall n(P(n) \rightarrow P(n+1))$, assume that $P(n)$ is true for an arbitrary n. We distinguish two cases,[46] namely whether or not the decomposition of $n+15$ contains a 4. If this is the case, then one can replace the 4 in the decomposition by a 5, resulting in the sum $n+16$. If the decomposition of $n+15$ contains no 4, then it contains at least three times the 5. We can therefore obtain a decomposition of $n+16$ by replacing the three 5 by four 4. In both cases, $P(n+1)$ is true, hence we have proved $P(n) \rightarrow P(n+1)$ and can apply Theorem 2.11.

[44] German: Verankerung

[45] This theorem is actually one of the Peano axioms used to axiomatize the natural numbers. In this view, it is an axiom and not a theorem. However, one can also define the natural numbers from axiomatic set theory and then prove the Peano axioms as theorems. This topic is beyond the scope of this course.

[46] Note that this proof step is a proof by case distinction.

Chapter 3

Sets, Relations, and Functions

In this chapter we provide a treatment of the elementary concepts of set theory, with the goal of being able to use sets in later parts of the course, for example to define relations and functions. We will be more precise than the typical (very informal) treatment of set theory in highschool, but we will also avoid the intricacies of a full-fledged axiomatic treatment of set theory, showing only minor parts of it.

3.1 Introduction

There seems to be no simpler mathematical concept than a *set*[1], a collection of objects. Although intuitively used for a long time,[2] the formulation of a set as a mathematical concept happened as late as the end of the 19th century. For example, in 1895 Cantor proposed the following wording: "Unter einer 'Menge' verstehen wir jede Zusammenfassung M von bestimmten wohlunterschiedenen Objekten unserer Anschauung oder unseres Denkens (welche die 'Elemente' von M genannt werden) zu einem Ganzen".

3.1.1 An Intuitive Understanding of Sets

The reader is certainly familiar with statements like

- $5 \in \mathbb{N}$ (where $\mathbb{N}$ denotes the set of natural numbers),
- $-3 \notin \mathbb{N}$,
- $\{3, 5, 7\} \subseteq \mathbb{N}$, and

[1] German: Menge

[2] In fact, almost all of mathematics is based on the notion of sets.

- $\{a, b\} \cup \{b, c\} = \{a, b, c\}$,

as well as with simple definitions like the following:

Definition 3.1. (Informal.) The number of elements of a finite set A is called its *cardinality* and is denoted $|A|$.

Also, facts like

$$A \subseteq B \ \wedge \ B \subseteq C \implies A \subseteq C$$

or

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

are well-known and seem obvious if one draws a figure with intersecting circles representing sets (so-called Venn-diagrams). However, many issues do not seem to be clear mathematically, for example:

- Which objects can one use as elements of a set?
- Can a set itself be an element of a set?
- Can a set be an element of itself?
- How is set intersection or set union defined?
- How should the elements of a set be counted?
- Do the above-stated facts require a proof, or are they just "obvious" in an informal sense?

This calls for a precise mathematical treatment with clear definitions, lemmas, and proofs. The need for such a precise treatment also becomes obvious when considering Russell's paradox discussed below.

3.1.2 Russell's Paradox

The set concept introduced by Cantor and axiomatized further by Frege seemed very innocent and safe to work with. But in 1903, Bertrand Russell[3] (1872-1970) showed that set theory as understood at that point in time is inherently contradictory. This came as a shock to the mathematics community. As a consequence, set theory had to be based on much more rigorous grounds, on an axiomatic foundation, a process started by Ernst Zermelo. It is still an active area of research in mathematics which axioms can and should be used as the foundation of set theory. The most widely considered set of axioms is called Zermelo-Fraenkel (ZF) set theory. Axiomatic set theory is beyond the scope of this course.

[3] Russell was a very remarkable person. He was not only an outstanding philosopher and mathematician, but also politically active as a pacifist. Because of his protests against World War I he was dismissed from his position at Trinity College in Cambridge and imprisoned for 6 months. In 1961, at the age of 89, he was arrested again for his protests against nuclear armament. In 1950 he received the Nobel Prize for literature.

The problem with Cantor's intuitive set theory is that, because it was not clearly axiomatized, it makes the following apparently innocent (yet false) assumption. Whenever one specifies a precise condition (i.e., a logical predicate P), allowing to distinguish between objects that satisfy the predicate and objects that don't, then $\{x \mid P(x)\}$, the set of objects satisfying the predicate is well-defined. Russell proposed the set

$$R = \{A \mid A \notin A\}$$

of sets that are not elements of themselves. Note that there seems to be nothing wrong with a set being an element of itself. For example, the set of sets containing at least 10 elements seems to be an element of itself, as it contains more than 10 elements. Similarly, the set of sets containing *at most* 10 elements is not an element of itself.

Either $R \in R$ or $R \notin R$. If $R \in R$, then by the definition of R, this implies $R \notin R$, a contradiction. Thus the only alternative is $R \notin R$. But again, by the definition of R, this implies $R \in R$, again a contradiction. In other words, $R \in R$ if and only if $R \notin R$, a paradox that requires an explanation.

The problem, subsequently addressed by Zermelo's axiomatization, is the following: While for any set B and predicate P, $\{x \in B \mid P(x)\}$ is actually a well-defined set, $\{x \mid P(x)\}$ is not. We must have a set to begin with before being able to create new sets by selecting the elements satisfying a certain condition. In other words, the universe $\mathcal{U}$ of objects one has in mind when writing $\{x \mid P(x)\}$ is not itself a set.[4]

3.2 Sets and Operations on Sets

3.2.1 The Set Concept

In set theory one postulates that there is a universe of possible sets and a universe of objects which can be elements of sets. Nothing prevents us from thinking that the two universes are the same, i.e., the elements of sets are also sets. We further postulate a binary predicate E to be given, and if $E(x, y) = 1$ we say that *x is an element of y*. We can call E the elementhood predicate. Instead of the predicate E we use an infix notation and write $x \in y$ rather than $E(x, y) = 1$. We also use the short-hand $x \notin y$ for $\neg(x \in y)$, i.e., if x is not an element of y.

Now we can postulate certain properties that the elementhood predicate E should satisfy, capturing the essence of set theory. This makes explicit that E is not some arbitrary predicate, but that it really captures natural properties of sets. In a systematic mathematical approach, one carefully chooses a list of axioms

[4]In fact, in Zermelo-Fraenkel (ZF) set theory, the axioms exclude that a set can be an element of itself.

and develops a theory (set theory) based on these axioms. There are indeed several different (but related) axiom systems for set theory, and it is beyond the scope of this course to discuss set theory in a formal sense.[5] However, we will informally introduce some of these properties/axioms in order to arrive at a sufficiently precise treatment of sets.

When writing formulas, it will often be convenient to not only use the usual logical variable symbols x, y, etc., but to use in the same formula symbols like A, B, etc. This is convenient because it makes the formulas look closer to how set theory is usually informally discussed. However, whether we use the symbol x or A for a set is not mathematically relevant.

3.2.2 Set Equality and Constructing Sets From Sets

A set is completely specified by its elements, regardless of how it is described.[6] There is no other relevant information about a set than what its elements are. In other words, two sets A and B are equal ($A = B$) if (and only if) they contain the same elements, independently of how A and B are described. In other words, there can not be two different sets A and B which contain exactly the same elements. This is called the *axiom of extensionality* in set theory. Since we are not aiming for an axiomatic treatment of set theory, we state this simply as a definition.

Definition 3.2. $A = B \overset{\text{def}}{\iff} \forall x\, (x \in A \leftrightarrow x \in B)$.

We postulate[7] that if a is a set, then the set containing exactly (and only) a exists, and is usually denoted as $\{a\}$. Similarly, for any finite list of sets, say a, b, and c, the set containing exactly these elements exists and is usually denoted as $\{a, b, c\}$.

Since a set is specified by its elements, we can conclude that if two sets, each containing a single element, are equal, then the elements are equal. This can be stated as a lemma (in set theory), and it actually requires a proof.

Lemma 3.1. *For any (sets) a and b, $\{a\} = \{b\} \implies a = b$.*

Proof. Consider any fixed a and b. The statement is an implication, which we prove indirectly. Assume that $a \neq b$. Then $\{a\} \neq \{b\}$ because there exists an

[5]Indeed, mathematicians are still working on fundamental questions regarding the theory of sets (but not questions relevant to us).

[6]For example, the set containing exactly the three natural numbers 1, 2, and 3 has many different descriptions, including $\{1, 2, 3\}$, $\{3, 1, 2\}$, $\{1, 1+1, 1+1+1\}$, etc. All these descriptions refer to the *same* set.

[7]In axiomatic set theory this is guaranteed by appropriate axioms.

element, namely a, that is contained in the first set, but not in the second. Thus we have proved that $a \neq b \implies \{a\} \neq \{b\}$. According to Definition 2.14, this proves $\{a\} = \{b\} \implies a = b$. □

Note that, in contrast, $\{a, b\} = \{c, d\}$ neither implies that $a = c$ nor that $b = d$. In a set, say $\{a, b\}$, there is no order of the elements, i.e.,

$$\{a, b\} = \{b, a\}.$$

However, in mathematics one wants to also define the concept of an (ordered) list of objects. Let us consider the special case of *ordered pairs*. For the operation of forming an ordered pair of two objects a and b, denoted (a, b), we define

$$(a, b) = (c, d) \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad a = c \wedge b = d.$$

Example 3.1. This example shows that one can model ordered pairs by using only (unordered) sets?[8] This means that the sets corresponding to two ordered pairs must be equal if and only if the ordered pairs are equal. A first approach is to define $(a, b) \stackrel{\text{def}}{=} \{a, \{b\}\}$. However, this definition of an ordered pair fails because one could not distinguish whether the set $\{\{b\}, \{c\}\}$ denotes the ordered pair $(\{b\}, c)$ or the ordered pair $(\{c\}, b)$. The reader can verify as an exercise that the following definition is correct:

$$(a, b) \stackrel{\text{def}}{=} \{\{a\}, \{a, b\}\}.$$

3.2.3 Subsets

Definition 3.3. The set A is a subset of the set B, denoted $A \subseteq B$, if every element of A is also an element of B, i.e.,

$$A \subseteq B \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad \forall x\, (x \in A \to x \in B).$$

The following lemma states an alternative way for capturing the equality of sets, via the subset relation. In fact, this is often the best way to prove that two sets are equal.

Lemma 3.2. $A = B \iff (A \subseteq B) \,\wedge\, (B \subseteq A)$.

Proof. The proof first makes use (twice) of Definition 3.3, then uses the fact from predicate logic that $\forall F \wedge \forall G \equiv \forall (F \wedge G)$, then uses the fact from propositional logic that $(C \to D) \wedge (D \to C) \equiv C \leftrightarrow D$,[9] and then makes use of Definitions 3.2. For any sets A and B we have the following equivalences of statements about A and B:

$$\begin{aligned}
(A \subseteq B) \,\wedge\, (B \subseteq A) \quad &\Longleftrightarrow \quad \forall x\, (x \in A \to x \in B) \,\wedge\, \forall x\, (x \in B \to x \in A) \\
&\Longleftrightarrow \quad \forall x\, \big((x \in A \to x \in B) \,\wedge\, (x \in B \to x \in A)\big) \\
&\Longleftrightarrow \quad \forall x\, (x \in A \leftrightarrow x \in B) \\
&\Longleftrightarrow \quad A = B
\end{aligned}$$

□

The next lemma states that the subset relation is transitive (a term discussed later). The proof is left as an exercise.

Lemma 3.3. *For any sets A, B, and C,*

$$A \subseteq B \,\wedge\, B \subseteq C \implies A \subseteq C.$$

3.2.4 Union and Intersection

Let us discuss a few well-known operations on sets and the laws for these operations.

Definition 3.4. The *union* of two sets A and B is defined as

$$A \cup B \stackrel{\text{def}}{=} \{x \mid x \in A \vee x \in B\},$$

and their *intersection* is defined as

$$A \cap B \stackrel{\text{def}}{=} \{x \mid x \in A \wedge x \in B\}.$$

The above definition can be extended from two to several sets, i.e., to a set (or collection) of sets. Let $\mathcal{A}$ be a non-empty set of sets, with finite or infinite cardinality. The only restriction on $\mathcal{A}$ is that its elements must be sets. Then we define the union of all sets in $\mathcal{A}$ as the set of all x that are an element of at least one of the sets in $\mathcal{A}$:

$$\bigcup \mathcal{A} \stackrel{\text{def}}{=} \{x \mid x \in A \text{ for some } A \in \mathcal{A}\}.$$

Similarly, we define the intersection of all sets in $\mathcal{A}$ as the set of all x that are an element of every set in $\mathcal{A}$:

$$\bigcap \mathcal{A} \stackrel{\text{def}}{=} \{x \mid x \in A \text{ for all } A \in \mathcal{A}\}.$$

[8] We briefly address this question, although we will not make use of this later and will continue to think about ordered pairs and lists in a conventional sense and with conventional notation.

[9] Here we use C and D rather than A and B to avoid confusion because A and B are used here to denotes sets.

Example 3.2. Consider the set of sets

$$\mathcal{A} = \{\{a,b,c,d\},\{a,c,e\},\{a,b,c,f\},\{a,c,d\}\}.$$

Then we have $\bigcup \mathcal{A} = \{a,b,c,d,e,f\}$ and $\bigcap \mathcal{A} = \{a,c\}$.

Typically, the sets (elements) in a set $\mathcal{A}$ of sets are indexed by some index set I: $\mathcal{A} = \{A_i |\, i \in I\}$. In this case, one also writes $\{A_i\}_{i\in I}$, and for the intersection and union one writes $\bigcap_{i\in I} A_i$ and $\bigcup_{i\in I} A_i$, respectively.

Definition 3.5. The *difference* of sets B and A, denoted $B\backslash A$ is the set of elements of B without those that are elements of A:

$$B \setminus A \stackrel{\text{def}}{=} \{x \in B |\, x \notin A\}.$$

Since union, intersection, and complement are defined by logical operations on set membership expressions (e.g. $a \in A$), that these set operations satisfy the corresponding statements of Lemma 2.1, stated as a theorem.

Theorem 3.4. *For any sets A, B, and C, the following laws hold:*

Idempotence:	$A \cap A = A$;
	$A \cup A = A$;
Commutativity:	$A \cap B = B \cap A$;
	$A \cup B = B \cup A$;
Associativity:	$A \cap (B \cap C) = (A \cap B) \cap C$;
	$A \cup (B \cup C) = (A \cup B) \cup C$;
Absorption:	$A \cap (A \cup B) = A$;
	$A \cup (A \cap B) = A$;
Distributivity:	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$;
	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$;
Consistency:	$A \subseteq B \iff A \cap B = A \iff A \cup B = B$.

Proof. The proof is straight-forward and exploits the related laws for logical operations. For example, the two associative laws are implied by the associativity of the logical AND and OR, respectively. The proof is left as an exercise. □

3.2.5 The Empty Set

Definition 3.6. A set A is called *empty* if it contains no elements, i.e., if $\forall x\, \neg(x \in A)$.

Lemma 3.5. *There is only one empty set (which is often denoted as $\varnothing$ or $\{\}$).*[10]

Proof. Let $\varnothing$ and $\varnothing'$ both be arbitrary empty sets. Since both are empty, every element that is in $\varnothing$ (namely none) is also in $\varnothing'$, and vice versa. This means according to Definition 3.2 that $\varnothing = \varnothing'$, which means that there is only one empty set. □

Lemma 3.6. *The empty set is a subset of every set, i.e., $\forall A\, (\varnothing \subseteq A)$*

Proof. The proof is by contradiction: Assume that there is a set A for which $\varnothing \not\subseteq A$. This means that there exists an x for which $x \in \varnothing$ but $x \notin A$. But such an x cannot exist because $\varnothing$ contains no element, which is a contradiction.

The above is a valid proof. Just to illustrate (as an example) that the same proof could be made more formal and more precise we can write the proof as follows, making use of logical transformation rules for formulas with quantifiers. Let A be an arbitrary (but fixed) set. The proof is by contradiction (see Definition 2.17), where the statement S to be proved is $\varnothing \subseteq A$ and as the statement T we choose $\neg\forall x\, (x \notin \varnothing)$, which is false because it is the negation of the definition of $\varnothing$. The proof that the negation of S implies T (step 3 in Definition 2.17) is as follows:

$$\begin{array}{llll}
\neg(\varnothing \subseteq A) & \iff & \neg\forall x\, (x \in \varnothing \to x \in A) & (\text{def. of } \varnothing \subseteq A) \\
& \iff & \exists x\, \neg(x \in \varnothing \to x \in A) & (\neg\forall x\, F \equiv \exists x\, \neg F) \\
& \iff & \exists x\, \neg(\neg(x \in \varnothing) \vee x \in A) & (\text{def. of } \to) \\
& \iff & \exists x\, (x \in \varnothing \wedge \neg(x \in A)) & (\text{de Morgan's rule}) \\
& \Longrightarrow & \exists x\, (x \in \varnothing) \wedge \exists x\, \neg(x \in A) & (\exists x\, (F \wedge G) \models (\exists x F) \wedge (\exists x G)) \\
& \Longrightarrow & \exists x\, (x \in \varnothing) & (F \wedge G \text{ implies } F) \\
& \iff & \neg\forall x\, \neg(x \in \varnothing). & (\neg\forall x\, F \equiv \exists x\, \neg F) \\
& \iff & \varnothing \text{ is not the empty set} & (\text{Definition 3.6})
\end{array}$$

which is false, and hence we have arrived at a contradiction. □

3.2.6 Constructing Sets from the Empty Set

At this point, the only set we know to exist, because we have postulated it, is the empty set. We can hence construct new sets $\varnothing$. The set $\{\varnothing\}$ is a set with a single element (namely $\varnothing$). It is important to note that $\{\varnothing\}$ is *not* the empty set

[10] We take it for granted that $\varnothing$ is actually a set. But in an axiomatic treatment of set theory, this must be stated as an axiom.

$\varnothing$, i.e., $\{\varnothing\} \neq \varnothing$. Note that $|\{\varnothing\}| = 1$ while $|\varnothing| = 0$. One can thus define a whole sequence of such sets:

$$\varnothing,\ \{\varnothing\},\ \{\{\varnothing\}\},\ \{\{\{\varnothing\}\}\}, \ldots .$$

Note that, except for the empty set, all these sets have cardinality 1.

Example 3.3. A few other sets constructed from the empty set are:
$A = \{\varnothing, \{\varnothing\}\}$,
$B = \{\{\varnothing, \{\varnothing\}\}\}$, and
$C = \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}$.
Their cardinalities are $|A| = 2$, $|B| = 1$, and $|C| = 3$. Also, $A \subseteq C$ and $B \subseteq C$.

Example 3.4. We have considered three relations between sets: $\in$, $=$, and $\subseteq$. Which of these relations hold for the following sets?
$A = \{\{\varnothing\}\}$,
$B = \{\{\varnothing\}, \{\varnothing, \varnothing\}\}$,
$C = \{\varnothing, \{\varnothing\}\}$, and
$D = \{\varnothing, \{\varnothing, \{\varnothing\}\}\}$.
The answer is: $B = A \subseteq C \in D$.

3.2.7 A Construction of the Natural Numbers

We briefly discuss a way to define the natural numbers from basic set theory. We use bold-face font to denote objects that we define here as special sets, and then can observe that they can be seen as corresponding to the natural numbers with the same symbol (but written in non-bold font). We define the sets

$$\mathbf{0} \stackrel{\text{def}}{=} \varnothing, \quad \mathbf{1} \stackrel{\text{def}}{=} \{\varnothing\}, \quad \mathbf{2} \stackrel{\text{def}}{=} \{\varnothing, \{\varnothing\}\}, \quad \mathbf{3} \stackrel{\text{def}}{=} \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}, \ldots$$

The successor of a set $\mathbf{n}$, which we can denote by $s(\mathbf{n})$, is defined as

$$s(\mathbf{n}) \stackrel{\text{def}}{=} \mathbf{n} \cup \{\mathbf{n}\}.$$

For example, we have $\mathbf{1} = s(\mathbf{0})$ and $\mathbf{2} = s(\mathbf{1})$. We note that $|\mathbf{0}| = 0$, $|\mathbf{1}| = 1$, $|\mathbf{2}| = 2$, $|\mathbf{3}| = 3, \ldots$, and, more generally, that $|s(\mathbf{n})| = |\mathbf{n}| + 1$.

An operation **+** on these sets $\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots$, which corresponds to addition of numbers, can be defined recursively as follows:

$$\mathbf{m}\boldsymbol{+}\mathbf{0} \stackrel{\text{def}}{=} \mathbf{m} \qquad \text{and} \qquad \mathbf{m}\boldsymbol{+}s(\mathbf{n}) \stackrel{\text{def}}{=} s(\mathbf{m}\boldsymbol{+}\mathbf{n}).$$

One can also define a multiplication operation and prove that these operations satisfy the usual laws of the natural numbers (commutative, associative, and distributive laws).

3.2.8 The Power Set of a Set

Definition 3.7. The *power set* of a set A, denoted $\mathcal{P}(A)$, is the set of all subsets of A:[11]

$$\mathcal{P}(A) \stackrel{\text{def}}{=} \{S |\ S \subseteq A\}.$$

For a finite set with cardinality k, the power set has cardinality 2^k (hence the name 'power set' and the alternative notation 2^A).

Example 3.5. $\mathcal{P}(\{a, b, c\}) = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$ and $|\mathcal{P}(\{a, b, c\})| = 8$.

Example 3.6. We have
$\mathcal{P}(\varnothing) = \{\varnothing\}$,
$\mathcal{P}(\{\varnothing\}) = \{\varnothing, \{\varnothing\}\}$,
$\mathcal{P}(\{\varnothing, \{\varnothing\}\}) = \{\varnothing, \{\varnothing\}, \{\{\varnothing\}\}, \{\varnothing, \{\varnothing\}\}\}$,
$\{1, 7, 9\} \in \mathcal{P}(\mathbb{N})$.

3.2.9 The Cartesian Product of Sets

Recall that two ordered pairs are equal if and only if both components agree, i.e.,

$$(a, b) = (c, d) \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad a = c \ \wedge\ b = d.$$

More generally, we denote an (ordered) list of k objects $a_1, \ldots, a_k$ as $(a_1, \ldots, a_k)$. Two lists of the same length are equal if they agree in every component.

Definition 3.8. The *Cartesian product* $A \times B$ of two sets A and B is the set of all ordered pairs with the first component from A and the second component from B:

$$A \times B = \big\{(a, b)\ \big|\ a \in A \ \wedge\ b \in B\big\}.$$

For finite sets, the cardinality of the Cartesian product of some sets is the product of their cardinalities: $|A \times B| = |A| \cdot |B|$.

Example 3.7. Prove or disprove the following statements:

(i) $\varnothing \times A = \varnothing$.

(ii) $A \times B = B \times A$.

More generally, the Cartesian product of k sets $A_1, \ldots, A_k$ is the set of all lists of length k (also called k-tuples) with the i-th component from A_i:

$$\bigtimes_{i=1}^{k} A_i = \{(a_1, \ldots, a_k)|\ a_i \in A_i \ \text{ for } 1 \leq i \leq k\}$$

[11] In axiomatic set theory, the existence of the power set of every set must be postulated as an axiom.

We point out that the Cartesian product is *not* associative, and in particular

$$\bigtimes_{i=1}^{3} A_i \neq (A_1 \times A_2) \times A_3.$$

3.3 Relations

Relations are a fundamental concept in discrete mathematics and Computer Science. Many special types of relations (e.g., equivalence relations, order relations, and lattices) capture concepts naturally arising in applications. Functions are also a special type of relation.

3.3.1 The Relation Concept

Definition 3.9. A *(binary) relation ρ from a set A to a set B* (also called an (A, B)-*relation*) is a subset of $A \times B$. If $A = B$, then ρ is called a relation *on* A.

Instead of $(a, b) \in \rho$ one usually writes

$$a \;\rho\; b,$$

and sometimes we write $a \not\rho\, b$ if $(a, b) \notin \rho$.

Example 3.8. Let S be the set of students at ETH and let C be the set of courses taught at ETH. Then a natural relation from S to C is the "takes" relation. If $s \in S$ is a student and $c \in C$ is a course, then (s, c) is in the relation if and only if s takes course c. If we denote the relation by `takes`, we can write $(s, c) \in$ `takes` or s `takes` y.[12] We can also consider the set P of professors at ETH and the natural relation from P to C.

Example 3.9. Let H be the set of all human beings (alive or dead). Then "child of" is a relation on H. If we denote the relation by `childof`, then $(x, y) \in$ `childof` (or equivalently x `childof` y) means that x is y's child. Other relations on H are "parent of", "grandparent of", "cousin of", "ancestor of", "married to", etc.

Example 3.10. On the integers $\mathbb{Z}$ we already know a number of very natural relations: $=, \neq, \leq, \geq, <, >, |$ (the 'divides' relation), and $\not|$ (does not divide).

Example 3.11. The relation $\equiv_m$ on $\mathbb{Z}$ is defined as follows:

$$a \equiv_m b \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad a - b = km \text{ for some } k,$$

i.e., $a \equiv_m b$ if and only if a and b have the same remainder when divided by m. (See Section 4.2.)

[12]Note that the relation `takes` can change over time, and in such an example we consider the relation at a certain point in time.

Example 3.12. The relation $\{(x, y) |\; x^2 + y^2 = 1\}$ on $\mathbb{R}$ is the set of points on the unit circle, which is a subset of $\mathbb{R} \times \mathbb{R}$.

Example 3.13. For any set S, the subset relation ($\subseteq$) is a relation on $\mathcal{P}(S)$.

Example 3.14. Two special relations from A to B are the empty relation (i.e., the empty set $\varnothing$) and the complete relation $A \times B$ consisting of all pairs (a, b).

Definition 3.10. For any set A, the *identity relation* on A, denoted id_A (or simply id), is the relation $\mathsf{id}_A = \{(a, a) |\; a \in A\}$.

Relations on a finite set are of special interest. There are 2^{n^2} different relations on a set of cardinality n. (Why?)

The relation concept can be generalized from binary to k-ary relations for given sets $A_1, \ldots, A_k$. A k-ary relation is a subset of $A_1 \times \cdots \times A_k$. Such relations play an important role in modeling relational databases. Here we only consider binary relations.

3.3.2 Representations of Relations

For finite sets A and B, a (binary) relation ρ from A to B can be represented as a Boolean $|A| \times |B|$ matrix M^ρ with the rows and columns labeled by the elements of A and B, respectively. For $a \in A$ and $b \in B$, the matrix entry $M^\rho_{a,b}$ is 1 if $a \;\rho\; b$, and 0 otherwise.

Example 3.15. Let $A = \{a, b, c, d\}$ and $B = \{q, r, s, t, u\}$, and consider the relation $\rho = \{(a, r), (a, s), (a, u), (b, q), (b, s), (c, r), (c, t), (c, u), (d, s), (d, u)\}$. The matrix representation is

$$M^\rho \;=\; \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{ccccc} q & r & s & t & u \end{array} \\ \left[\begin{array}{ccccc} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{array}\right] \end{array}$$

where the rows and columns are labeled by the elements of A and B, respectively.

For relations *on* a set A, the matrix is an $|A| \times |A|$ square matrix.

Example 3.16. For the set $A = \{1, 2, 3, 4, 5\}$, the relations $=$, $\geq$, and $\leq$ correspond to the identity matrix,[13] the lower triangular matrix, and the upper triangular matrix, respectively.

[13]The identity relation ($=$) on any finite set corresponds to the identity matrix.

An alternative representation of a relation ρ from A to B is by a directed graph with $|A|+|B|$ vertices[14] labeled by the elements of A and B. The graph contains the edge[15] from a to b if and only if $a\ \rho\ b$. For a relation *on* a set A, the graph contains only $|A|$ vertices, but it can contain loops (edges from a vertex to itself).

3.3.3 Set Operations on Relations

Relations from A to B are sets, and therefore we can apply any operation defined on sets: union, intersection, and complement. In the matrix representation of relations, these operations correspond to the position-wise logical OR, AND, and negation, respectively. A relation can also be a subset of another relation.

Example 3.17. On the set $\mathbb{Z}$, the relation $\leq \cup \geq$ is the complete relation, $\leq \cap \geq$ is the identity relation, and the complement of $\leq$ is the relation $>$. Moreover, we have $< \subseteq \leq$ and $= \subseteq \geq$.

Example 3.18. For any relatively prime integers m and n, the relation $\equiv_m \cap \equiv_n$ is $\equiv_{mn}$, as will be shown in Chapter 4. More generally, For general m and n, the relation $\equiv_m \cap \equiv_n$ is $\equiv_{\mathrm{lcm}(m,n)}$, where $\mathrm{lcm}(m,n)$ denotes the least common multiple of m and n.

3.3.4 The Inverse of a Relation

Definition 3.11. The *inverse* of a relation ρ from A to B is the relation $\widehat{\rho}$ from B to A defined by

$$\widehat{\rho} \ \stackrel{\text{def}}{=}\ \{(b,a) \mid (a,b) \in \rho\}.$$

Note that for all a and b we have $b\ \widehat{\rho}\ a \iff a\ \rho\ b$. An alternative notation for the inverse of ρ is ρ^{-1}.

Example 3.19. Let H be the set of people, O the set of objects, and π the ownership relation from H to O. The inverse $\widehat{\pi}$ is the "owned by" relation determining who owns an object.

Example 3.20. If ϕ is the parenthood relation on the set H of humans (i.e., $a\ \phi\ b$ if a is a parent of b), then the inverse relation $\widehat{\phi}$ is the childhood relation.

Example 3.21. On the set $\mathbb{Z}$, the inverse of the relation $\leq$ is the relation $\geq$. The inverse of id is again id.

In the matrix representation, taking the inverse of a relation corresponds to the transposition of the matrix. In the graph representation, taking the inverse corresponds to inverting the direction of all edges.

[14] German: Knoten
[15] German: Kante

3.3.5 Composition of Relations

Definition 3.12. Let ρ be a relation from A to B and let σ be a relation from B to C. Then the *composition* of ρ and σ, denoted $\rho \circ \sigma$ (or also $\rho\sigma$), is the relation from A to C defined by

$$\rho \circ \sigma \ \stackrel{\text{def}}{=}\ \{(a,c) \mid \exists b\,\big((a,b) \in \rho \ \wedge\ (b,c) \in \sigma\big)\}.$$

The n-fold composition of a relation ρ on a set A with itself is denoted ρ^n.

Lemma 3.7. *The composition of relations is associative, i.e., we have* $\rho \circ (\sigma \circ \phi) = (\rho \circ \sigma) \circ \phi$.

Proof. We use the short notation $\rho\sigma$ instead of $\rho \circ \sigma$. The claim of the lemma, $\rho(\sigma\phi) = (\rho\sigma)\phi$, states an equality of sets, which can be proved by proving that each set is contained in the other (see Section 3.2.3). We prove $\rho(\sigma\phi) \subseteq (\rho\sigma)\phi$; the other inclusion is proved analogously. Suppose that $(a,d) \in \rho(\sigma\phi)$. We need to prove that $(a,d) \in (\rho\sigma)\phi$. For illustrative purposes, We provide two formulations of this proof, first as a text and then in logical formulas.

Because $(a,d) \in \rho(\sigma\phi)$, there exists b such that $(a,b) \in \rho$ and $(b,d) \in \sigma\phi$. The latter implies that there exists c such that $(b,c) \in \sigma$ and $(c,d) \in \phi$. Now, $(a,b) \in \rho$ and $(b,c) \in \sigma$ imply that $(a,c) \in \rho\sigma$, which together with $(c,d) \in \phi$ implies $(a,d) \in (\rho\sigma)\phi$.

Now comes the more formal version of the same proof, where the justification for each step is omitted:[16]

$$
\begin{aligned}
(a,d) \in \rho(\sigma\phi) \quad &\stackrel{\cdot}{\Longrightarrow} \quad \exists b\,\big((a,b) \in \rho \wedge (b,d) \in \sigma\phi\big) \\
&\stackrel{\cdot}{\Longrightarrow} \quad \exists b\,\big((a,b) \in \rho \wedge \exists c\,\big((b,c) \in \sigma \wedge (c,d) \in \phi\big)\big) \\
&\stackrel{\cdot}{\Longrightarrow} \quad \exists b \exists c\,\big((a,b) \in \rho \wedge \big((b,c) \in \sigma \wedge (c,d) \in \phi\big)\big) \\
&\stackrel{\cdot}{\Longrightarrow} \quad \exists b \exists c\,\big(\big((a,b) \in \rho \wedge (b,c) \in \sigma\big) \wedge (c,d) \in \phi\big) \\
&\stackrel{\cdot}{\Longrightarrow} \quad \exists c \exists b\,\big(\big((a,b) \in \rho \wedge (b,c) \in \sigma\big) \wedge (c,d) \in \phi\big) \\
&\stackrel{\cdot}{\Longrightarrow} \quad \exists c\,\big(\exists b\,\big((a,b) \in \rho \wedge (b,c) \in \sigma\big) \wedge (c,d) \in \phi\big) \\
&\stackrel{\cdot}{\Longrightarrow} \quad \exists c\,\big((a,c) \in \rho\sigma \wedge (c,d) \in \phi\big) \\
&\stackrel{\cdot}{\Longrightarrow} \quad (a,d) \in (\rho\sigma)\phi.
\end{aligned}
$$

□

[16] The justifications should be obvious, except perhaps for the following fact from predicate logic (explained in Chapter 6) used several times in the proof: $\exists x(F \wedge G) \equiv F \wedge \exists x G$ if x does not appear in F.

Example 3.22. Consider the ownership relation π and the parenthood relation ϕ as above. Then the relation $\phi\pi$ from H to O can be interpreted as follows: $a\ \phi\pi\ b$ if and only if person a has a child who owns object b.

Example 3.23. If ϕ is the parenthood relation on the set H of humans, then the relation ϕ^2 is the grand-parenthood relation.[17]

In the matrix representation, composing two relations corresponds to a special type of matrix multiplication. If the matrices are considered as integer matrices (with 0 and 1 entries), then after the multiplication all entries > 1 are set to 1.[18] In the graph representation the composition corresponds to the natural composition of the graphs, where $a\ \rho\sigma\ c$ if and only if there is a path from a (over some b) to c.

The proof of the following lemma is left as an exercise.

Lemma 3.8. *Let ρ be a relation from A to B and let σ be a relation from B to C. Then the inverse $\widehat{\rho\sigma}$ of $\rho\sigma$ is the relation $\widehat{\sigma}\widehat{\rho}$.*

3.3.6 Special Properties of Relations

Definition 3.13. A relation ρ on a set A is called *reflexive* if

$$a\ \rho\ a$$

is true for all $a \in A$, i.e., if

$$\mathsf{id} \subseteq \rho.$$

In other words, a relation is reflexive if it contains the identity relation id. In the matrix representation of relations, reflexive means that the diagonal is all 1. In a graph representation, reflexive means that every vertex has a loop (an edge from the vertex to itself).

Example 3.24. The relations $\leq$, $\geq$, and $|$ (divides) on $\mathbb{Z} \setminus \{0\}$ are reflexive, but the relations $<$ and $>$ are not.

Definition 3.14. A relation ρ on a set A is called *irreflexive* if $a \not\rho\ a$ for all $a \in A$, i.e., if $\rho \cap \mathrm{id} = \varnothing$.[19]

[17] Note that the notation ϕ^2 is actually ambiguous; it could also denote the Cartesian product $\phi \times \phi$. But in these lecture notes no ambiguity will arise.

[18] If the matrices are considered as Boolean matrices, then for multiplying two matrices one takes the OR of all product terms contributing to an entry in the product matrix.

[19] Note that irreflexive is not the negation of reflexive, i.e., a relation that is not reflexive is not necessarily irreflexive.

Definition 3.15. A relation ρ on a set A is called *symmetric* if

$$a\ \rho\ b \iff b\ \rho\ a$$

is true for all $a, b \in A$, i.e., if

$$\rho = \widehat{\rho}.$$

In the matrix representation of relations, symmetric means that the matrix is symmetric (with respect to the diagonal).

A symmetric relation ρ on a set A can be represented as an undirected graph, possibly with loops from a node to itself.

Example 3.25. The relation $\equiv_m$ on $\mathbb{Z}$ is symmetric.

Example 3.26. The "married to" relation on the set H of humans is symmetric.

Definition 3.16. A relation ρ on a set A is called *antisymmetric* if

$$a\ \rho\ b\ \wedge\ b\ \rho\ a \implies a = b$$

is true for all $a, b \in A$, i.e., if

$$\rho \cap \widehat{\rho} \subseteq \mathsf{id}.$$

In a graph representation, antisymmetric means that there is no cycle of length 2, i.e., no distinct vertices a and b with edges both from a to b and from b to a. Note that antisymmetric is *not* the negation of symmetric.

Example 3.27. The relations $\leq$ and $\geq$ are antisymmetric, and so is the division relation $|$ on $\mathbb{N}$: If $a \mid b$ and $b \mid a$, then $a = b$. But note that the division relation $|$ on $\mathbb{Z}$ is *not* antisymmetric. Why?

Definition 3.17. A relation ρ on a set A is called *transitive* if

$$a\ \rho\ b\ \wedge\ b\ \rho\ c \implies a\ \rho\ c$$

is true for all $a, b, c \in A$.

Example 3.28. The relations $\leq$, $\geq$, $<$, $>$, $|$, and $\equiv_m$ on $\mathbb{Z}$ are transitive.

Example 3.29. Let ρ be the ancestor relation on the set H of humans, i.e., $a\ \rho\ b$ if a is an ancestor of b. This relation is transitive.

Lemma 3.9. *A relation ρ is transitive if and only if $\rho^2 \subseteq \rho$.*

Proof. The "if" part of the theorem ($\Longleftarrow$) follows from the definition of composition: If $a\ \rho\ b$ and $b\ \rho\ c$, then $a\ \rho^2\ c$. Therefore also $a\ \rho\ c$ since $\rho^2 \subseteq \rho$.[20] This

[20] In set-theoretic notation: $(a, c) \in \rho^2 \wedge \rho^2 \subseteq \rho \implies (a, c) \in \rho$.

means transitivity.

Proof of the "only if" part ($\Longrightarrow$): Assume that ρ is transitive. To show that $\rho^2 \subseteq \rho$, assume that $a\ \rho^2\ b$ for some a and b. We must prove that $a\ \rho\ b$. The definition of $a\ \rho^2\ b$ states that there exists c such that $a\ \rho\ c$ and $c\ \rho\ b$. Transitivity of ρ thus implies that $a\ \rho\ b$, which concludes the proof. □

3.3.7 Transitive Closure

The reader can verify as an exercise that for a transitive relation ρ we have $\rho^n \subseteq \rho$ for all $n > 1$.

Definition 3.18. The *transitive closure* of a relation ρ on a set A, denoted ρ^*, is

$$\rho^* = \bigcup_{n \in \mathbb{N}\setminus\{0\}} \rho^n.$$

In the graph representation of a relation ρ on A, we have $a\ \rho^k\ b$ if and only if there is a walk of length k from a to b in the graph, where a walk may visit a node multiple times. The transitive closure is the reachability relation, i.e., $a\ \rho^*\ b$ if and only if there is a path (of arbitrary finite length) from a to b.

Example 3.30. Consider the set P of all convex polygons. We can think of them as being given as pieces of paper. By cutting a piece into two pieces with a straight cut one can obtain new polygons. Let $\succeq$ be the relation defined as follows: $a \succeq b$ if and only if with a single straight-line cut (or no cut) one can obtain b from a. Moreover, consider the covering relation $\sqsupseteq$, where $a \sqsupseteq b$ if and only if a can completely cover b (if appropriately positioned). It is easy to see that $\sqsupseteq$ is reflexive, anti-symmetric, and transitive[21] whereas $\succeq$ is only reflexive and antisymmetric. Note that $\sqsupseteq$ is the transitive closure of $\succeq$.

3.4 Equivalence Relations

3.4.1 Definition of Equivalence Relations

Definition 3.19. An *equivalence relation* is a relation on a set A that is reflexive, symmetric, and transitive.

Example 3.31. The relation $\equiv_m$ is an equivalence relation on $\mathbb{Z}$.

[21]Such a relation will be defined below as a partial order relation.

Definition 3.20. For an equivalence relation θ on a set A and for $a \in A$, the set of elements of A that are equivalent to a is called the *equivalence class of* a and is denoted as $[a]_\theta$:[22]

$$[a]_\theta \stackrel{\text{def}}{=} \{b \in A |\ b\ \theta\ a\}.$$

Two trivial equivalence relations on a set A are the complete relation $A \times A$, for which there is only one equivalence class A, and the identity relation for which the equivalence classes are all singletons[23] $\{a\}$ for $a \in A$.

Example 3.32. The equivalence classes of the relation $\equiv_3$ are the sets

$$[0] = \{\ldots, -6, -3, 0, 3, 6, \ldots\},$$
$$[1] = \{\ldots, -5, -2, 1, 4, 7, \ldots\},$$
$$[2] = \{\ldots, -4, -1, 2, 5, 8, \ldots\}.$$

Example 3.33. Consider the set $\mathbb{R}^2$ of points (x, y) in the plane, and consider the relation ρ defined by $(x, y)\ \rho\ (x', y') \iff x + y = x' + y'$. Clearly, this relation is reflexive, symmetric, and transitive. The equivalence classes are the set of lines in the plane parallel to the diagonal of the second quadrant.

The proof of the following theorem is left as an exercise.

Lemma 3.10. *The intersection of two equivalence relations (on the same set) is an equivalence relation.*

Example 3.34. The intersection of $\equiv_5$ and $\equiv_3$ is $\equiv_{15}$.

3.4.2 Equivalence Classes Form a Partition

Definition 3.21. A *partition* of a set A is a set of mutually disjoint subsets of A that cover A, i.e., a set $\{S_i | i \in \mathcal{I}\}$ of sets S_i (for some index set $\mathcal{I}$) satisfying

$$S_i \cap S_j = \varnothing \quad \text{for } i \neq j \qquad \text{and} \qquad \bigcup_{i \in \mathcal{I}} S_i = A.$$

Consider any partition of a set A and define the relation $\equiv$ such that two elements are $\equiv$-related if and only if they are in the same set of the partition. It is easy to see that this relation is an equivalence relation. The following theorem states that the converse also holds. In other words, partitions and equivalence relations capture the same (simple) abstraction.

[22]When the relation θ is understood, we can drop the subscript θ.

[23]A singleton is a set with one element.

Definition 3.22. The set of equivalence classes of an equivalence relation θ, denoted by

$$A/\theta \stackrel{\text{def}}{=} \{[a]_\theta \mid a \in A\},$$

is called the *quotient set of A by θ*, or simply *A modulo θ*, or $A \bmod \theta$.

Theorem 3.11. *The set A/θ of equivalence classes of an equivalence relation θ on A is a partition of A.*

Proof. Since $a \in [a]$ for all $a \in A$ (reflexivity of θ), the union of all equivalence classes is equal to A. It remains to prove that equivalence classes are disjoint. This is proved by proving, for any fixed a and b, that

$$a \;\theta\; b \implies [a] = [b]$$

and

$$a \;\not\theta\; b \implies [a] \cap [b] = \varnothing.$$

To prove the first statement we consider an arbitrary $c \in [a]$ and observe that

$$\begin{array}{rcll} c \in [a] & \stackrel{\cdot}{\iff} & c \;\theta\; a & \text{(def. of } [a]) \\ & \stackrel{\cdot}{\implies} & c \;\theta\; b & \text{(use } a \;\theta\; b \text{ and transitivity)} \\ & \stackrel{\cdot}{\iff} & c \in [b] & \text{(def. of } [b].) \end{array}$$

Note that $c \in [a] \implies c \in [b]$ (for all $c \in A$) is the definition of $[a] \subseteq [b]$. The statement $[b] \subseteq [a]$ is proved analogously but additionally requires the application of symmetry. (This is an exercise.) Together this implies $[a] = [b]$.

The second statement is proved by contradiction. Suppose it is false[24], i.e., $a \not\theta\, b$ and $[a] \cap [b] \neq \varnothing$, i.e., there exists some $c \in [a] \cap [b]$, which means that $c \;\theta\; a$ and $c \;\theta\; b$. By symmetry we have $a \;\theta\; c$ and thus, by transitivity, we have $a \;\theta\; b$, a contradiction. (As an exercise, the reader can write this proof as a sequence of implications.) □

3.4.3 Example: Definition of the Rational Numbers

We consider the set $A = \mathbb{Z} \times (\mathbb{Z}\backslash\{0\})$ and define the equivalence relation $\sim$ on A as follows:

$$(a, b) \sim (c, d) \stackrel{\text{def}}{\iff} ad = bc.$$

This relation is reflexive ($(a, b) \sim (a, b)$ since $ab = ba$), symmetric (since $ad = bc \implies cb = da$), and transitive. For the latter, assume $(a, b) \sim (c, d)$ and

[24]Recall that $\neg(A \to B) \equiv A \wedge \neg B$.

$(c, d) \sim (e, f)$. Then $ad = bc$ and $cf = de$, and thus $adcf = bcde$. Canceling d (which is $\neq 0$) gives

$$acf = bce.$$

We have to consider the cases $c \neq 0$ and $c = 0$ separately. If $c \neq 0$, then c can be canceled, giving $af = be$. If $c = 0$, then $a = 0$ since $d \neq 0$ but $ad = bc$. Similarly, $e = 0$, and thus we also have $af = be$. Therefore $\sim$ is transitive and hence an equivalence relation.

To every equivalence class $[(a, b)]$ we can associate the rational number a/b ($b \neq 0$). It is easy to see that all pairs $(u, v) \in [(a, b)]$ correspond to the same rational number, and two distinct rational numbers correspond to distinct equivalence classes. Thus[25]

$$\mathbb{Q} \stackrel{\text{def}}{=} (\mathbb{Z} \times (\mathbb{Z}\backslash\{0\})) \,/ \sim .$$

3.5 Partial Order Relations

3.5.1 Definition

Taking the definition of an equivalence relation and simply replacing the symmetry condition by the anti-symmetry condition results in a completely different, but even more interesting type of relation.

Definition 3.23. A *partial order* (or simply an *order relation*[26]) on a set A is a relation that is reflexive, antisymmetric, and transitive. A set A together with a partial order $\preceq$ on A is called a *partially ordered set* (or simply *poset*) and is denoted as $(A; \preceq)$.[27]

In a graph representation of relations, a partial order has no cycles (but this is of course not a complete characterization).

Example 3.35. The relations $\leq$ and $\geq$ are partial orders on $\mathbb{Z}$, $\mathbb{Q}$, or $\mathbb{R}$. The relations $<$ and $>$ are not partial orders because they are not reflexive (though they are both transitive and, in fact, also antisymmetric because $a < b \wedge b < a$ is never true, i.e., $< \cap \widehat{<} = \varnothing$).

Example 3.36. The division relation ($|$) is a partial order on $\mathbb{N} \setminus \{0\}$ or any subset of $\mathbb{N} \setminus \{0\}$.

Example 3.37. The subset relation on the power set of a set A is a partial order. In other words, for any set A, $(\mathcal{P}(A); \subseteq)$ is a poset.

[25]This is a more fancy way of saying that two rational numbers a/b and c/d are the same number if and only if the ratio is the same. But actually, this *is* the definition of the rational numbers. If the reader is surprised, he or she is challenged to come up with a simpler definition.

[26]German: Ordnungsrelation

[27]Partial orders are often denoted by $\leq$ or by a similar symbol like $\preceq$ or $\sqsubseteq$.

Example 3.38. The covering relation on convex polygons (see Example 3.30) is a partial order.

For a partial order relation $\preceq$ we can define the relation $a \prec b$ similar to how the relation $<$ is obtained from $\leq$:

$$a \prec b \quad \overset{\text{def}}{\Longleftrightarrow} \quad a \preceq b \ \wedge \ a \neq b.$$

Definition 3.24. For a poset $(A; \preceq)$, two elements a and b are called *comparable*[28] if $a \preceq b$ or $b \preceq a$; otherwise they are called *incomparable*.

Definition 3.25. If any two elements of a poset $(A; \preceq)$ are comparable, then A is called *totally ordered* (or *linearly ordered*) by $\preceq$.

Example 3.39. $(\mathbb{Z}; \leq)$ and $(\mathbb{Z}; \geq)$ are totally ordered posets (or simply totally ordered sets), and so is any subset of $\mathbb{Z}$ with respect to $\leq$ or $\geq$. For instance, $(\{2, 5, 7, 10\}; \leq)$ is a totally ordered set.

Example 3.40. The poset $(\mathcal{P}(A); \subseteq)$ is not totally ordered if $|A| \geq 2$, nor is the poset $(\mathbb{N}; |)$.

3.5.2 Hasse Diagrams

Definition 3.26. In a poset $(A; \preceq)$ an element b is said to *cover*[29] an element a if $a \prec b$ and there exists no c with $a \prec c$ and $c \prec b$ (i.e., between a and b).

Example 3.41. In a hierarchy (say of a company), if $a \prec b$ means that b is superior to a, then b covers a means that b is the direct superior of a.

Definition 3.27. The *Hasse diagram* of a (finite) poset $(A; \preceq)$ is the directed graph whose vertices are labeled with the elements of A and where there is an edge from a to b if and only if b covers a.

The Hasse diagram is a graph with directed edges. It is usually drawn such that whenever $a \prec b$, then b is placed higher than a. This means that all arrows are directed upwards and therefore can be omitted.

Example 3.42. The Hasse diagram of the poset $(\{2, 3, 4, 5, 6, 7, 8, 9\}; |)$ is shown in Figure 3.1 on the left.

[28]German: vergleichbar
[29]German: überdecken

Example 3.43. A nicer diagram is obtained when A is the set of all divisors of an integer n. The Hasse diagram of the poset $(\{1, 2, 3, 4, 6, 8, 12, 24\}; |)$ is shown in Figure 3.1 in the middle.

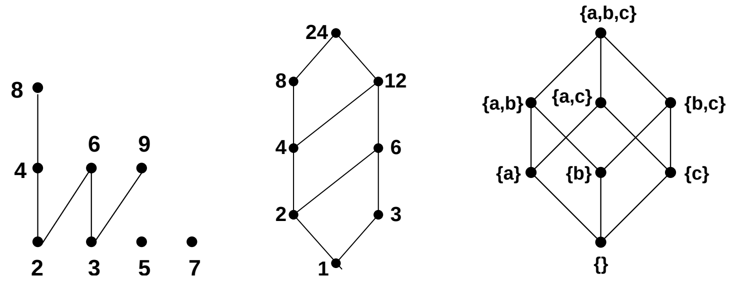

Figure 3.1: The Hasse diagrams of the posets $(\{2, 3, 4, 5, 6, 7, 8, 9\}; |)$, $(\{1, 2, 3, 4, 6, 8, 12, 24\}; |)$, and $(\mathcal{P}(\{a, b, c\}); \subseteq)$.

Example 3.44. The Hasse diagram of the poset $(\mathcal{P}(\{a, b, c\}); \subseteq)$ is shown in Figure 3.1 on the right.

Example 3.45. For the two Hasse diagrams in Figure 3.2, give a realization as the divisibility poset of a set of integers.

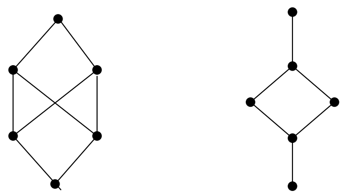

Figure 3.2: Two Hasse diagrams.

Example 3.46. Consider the covering[30] relation on the convex polygons discussed in Example 3.30. A polygon a is covered by a polygon b if b can be placed on top of a such that a disappears completely. Are there sets of six polygons resulting in a poset with the left (or right) Hasse diagram in Figure 3.2?

[30]The term "cover" is used here in a physical sense, not in the sense of Definition 3.26.

3.5.3 Combinations of Posets and the Lexicographic Order

Definition 3.28. The *direct product* of posets $(A;\preceq)$ and $(B;\sqsubseteq)$, denoted $(A;\preceq) \times (B;\sqsubseteq)$, is the set $A \times B$ with the relation $\leq$ (on $A \times B$) defined by

$$(a_1, b_1) \leq (a_2, b_2) \quad \overset{\text{def}}{\Longleftrightarrow} \quad a_1 \preceq a_2 \ \wedge \ b_1 \sqsubseteq b_2.$$

Theorem 3.12. $(A;\preceq) \times (B;\sqsubseteq)$ *is a partially ordered set.*

The proof is left as an exercise. The reader can verify that when replacing $\wedge$ by $\vee$, the resulting relation is in general *not* a partial order relation.

A more interesting order relation on $A \times B$ is defined in the following theorem, whose proof is left as an exercise.

Theorem 3.13. *For given posets $(A;\preceq)$ and $(B;\sqsubseteq)$, the relation $\leq_{\text{lex}}$ defined on $A \times B$ by*

$$(a_1, b_1) \leq_{\text{lex}} (a_2, b_2) \quad \overset{\text{def}}{\Longleftrightarrow} \quad a_1 \prec a_2 \ \vee \ (a_1 = a_2 \wedge b_1 \sqsubseteq b_2)$$

is a partial order relation.[31]

The relation $\leq_{\text{lex}}$ is the well-known *lexicographic order* of pairs, usually considered when both posets are identical. The lexicographic order $\leq_{\text{lex}}$ is useful because if both $(A;\preceq)$ and $(B;\sqsubseteq)$ are totally ordered (e.g. the alphabetical order of the letters), then so is the lexicographic order on $A \times B$ (prove this!).

The lexicographic order can easily be generalized to the k-tuples over some alphabet Σ (denoted Σ^k) and more generally to the set Σ^* of finite-length strings over Σ. The fact that a total order on the alphabet results in a total order on Σ^* is well-known: The telephone book has a total order on all entries.

3.5.4 Special Elements in Posets

We define a few types of special elements that a poset can have.

Definition 3.29. Let $(A;\preceq)$ be a poset, and let $S \subseteq A$ be some subset of A. Then

1. $a \in A$ is a *minimal (maximal) element* of A if there exists no $b \in A$ with $b \prec a$ $(b \succ a)$.[32]
2. $a \in A$ is the *least (greatest) element* of A if $a \preceq b \quad (a \succeq b)$ for all $b \in A$.[33]
3. $a \in A$ is a *lower (upper) bound*[34] of S if $a \preceq b \quad (a \succeq b)$ for all $b \in S$.[35]
4. $a \in A$ is the *greatest lower bound (least upper bound)* of S if a is the greatest (least) element of the set of all lower (upper) bounds of S.[36]

[31] Recall that for a partial order $\preceq$ we can define the relation $\prec$ as $a \prec b \overset{\text{def}}{\Longleftrightarrow} a \preceq b \wedge a \neq b$.

[32] The relations $\succeq$ and $\succ$ are defined naturally by $a \succeq b \overset{\text{def}}{\Longleftrightarrow} b \preceq a$ and $a \succ b \overset{\text{def}}{\Longleftrightarrow} b \prec a$.

Minimal, maximal, least, and greatest elements are easily identified in a Hasse diagram.

The greatest lower bound and the least upper bound of a set S are sometimes denoted as $\text{glb}(S)$ and $\text{lub}(S)$, respectively.

Example 3.47. Consider the poset $(\{2,3,4,5,6,7,8,9\};\,|)$ shown in Figure 3.1. It has no least or greatest elements, but 2, 3, 5, and 7 are minimal elements, and 5, 6, 7, 8 and 9 are maximal elements. The number 2 is a lower bound (actually the greatest lower bound) of the subset $\{4,6,8\}$, and the subset $\{4,9\}$ has no lower (nor upper) bound.

Example 3.48. The poset $(\{1,2,3,4,6,8,12,24\};\,|)$ shown in Figure 3.1 has both a least (1) and a greatest (24) element. The subset $\{8,12\}$ has the three lower bounds $1, 2$, and 4, and 4 is the greatest lower bound of $\{8,12\}$. Actually, this poset is special in that any set of elements has a greatest lower bound and a least upper bound. How can $\text{glb}(S)$ and $\text{lub}(S)$ be defined?

Example 3.49. The poset $(\mathcal{P}(\{a,b,c\});\subseteq)$ shown in Figure 3.1 has both a least element, namely $\varnothing$, and a greatest element, namely $\{a,b,c\}$.

Example 3.50. In the poset $(\mathbb{Z}^+;\,|)$, 1 is a least element but there is no greatest element.

Definition 3.30. A poset $(A;\preceq)$ is *well-ordered*[37] if it is totally ordered and if every non-empty subset of A has a least element.[38]

Note that every totally ordered finite poset is well-ordered. The property of being well-ordered is of interest only for infinite posets. The natural numbers $\mathbb{N}$ are well-ordered by $\leq$. Any subset of the natural numbers is also well-ordered. More generally, any subset of a well-ordered set is well-ordered (by the same order relation).

[33] Note that a least or a greatest element need not exist. However, there can be at most one least element, as suggested by the word "the" in the definition. This follows directly from the antisymmetry of $\preceq$. If there were two least elements, they would be mutually comparable, and hence must be equal.

[34] German: untere (obere) Schranke

[35] Note that the definitions of the least element and of a lower bound differ only in that a lower bound can be outside of the considered subset S (and therefore need not be unique).

[36] Note that for a poset $(A;\preceq)$ and a subset $S \subseteq A$, restricting $\preceq$ to S results in a poset $(S;\preceq)$.

[37] German: wohlgeordnet

[38] The least element is defined naturally (see Definition 3.29).

3.5.5 Meet, Join, and Lattices

Definition 3.31. Let $(A;\preceq)$ be a poset. If a and b (i.e., the set $\{a,b\}\subseteq A$) have a greatest lower bound, then it is called the *meet* of a and b, often denoted $a\wedge b$. If a and b have a least upper bound, then it is called the *join* of a and b, often denoted $a\vee b$.

Definition 3.32. A poset $(A;\preceq)$ in which every pair of elements has a meet and a join is called a *lattice*[39].

Example 3.51. The posets $(\mathbb{N};\leq)$, $(\mathbb{N}\setminus\{0\};\ |\ )$, and $(\mathcal{P}(S);\subseteq)$ are lattices, as the reader can verify.

Example 3.52. The poset $(\{1,2,3,4,6,8,12,24\};|)$ shown in Figure 3.1 is a lattice. The meet of two elements is their greatest common divisor, and their join is the least common multiple. For example, $6\wedge 8=2$, $6\vee 8=24$, $3\wedge 4=1$, and $3\vee 4=12$. In contrast, the poset $(\{2,3,4,5,6,7,8,9\};|)$ is not a lattice.

3.6 Functions

The concept of a function is perhaps the second most fundamental concept in mathematics (after the concept of a set). We discuss functions only now, after having introduced relations, because functions are a special type of relation, and several concepts defined for relations (e.g. inversion and composition) apply to functions as well.

Definition 3.33. A *function* $f: A\to B$ from a *domain*[40] A to a *codomain*[41] B is a relation from A to B with the special properties (using the relation notation $a\ f\ b$):[42]

1. $\forall a\in A\quad \exists b\in B\qquad a\ f\ b$ (f is totally defined),
2. $\forall a\in A\ \ \forall b,b'\in B\ \ \big(a\ f\ b\ \wedge\ a\ f\ b'\ \to\ b=b'\big)$ (f is well-defined).

As the reader certainly knows, a function f can be understood as a mapping from A to B, assigning to every $a\in A$ a unique element in B, usually denoted as $f(a)$. One writes $f:\ A\to B$ to indicate the domain and codomain of f, and

$$f:\ a\mapsto \text{“expression in } a\text{”}$$

(e.g. $f:\ a\mapsto a^2$ or, equivalently, $f:\ x\mapsto x^2$) to define the function.

[39] German: Verband
[40] German: Definitionsbereich
[41] German: Bildbereich, Wertebereich
[42] Here we use the convenient notation $\forall a\in A$ and $\exists b\in B$.

Definition 3.34. The set of all functions $A\to B$ is denoted as B^A. [43]

One can generalize the function concept by dropping the first condition (totally defined), i.e., allowing that there can exist elements $a\in A$ for which $f(a)$ is not defined.

Definition 3.35. A *partial function* $A\to B$ is a relation from A to B such that condition 2. above holds.

Two (partial) functions with common domain A and codomain B are *equal* if they are equal as relations (i.e., as sets). $f=g$ is equivalent to saying that the function values of f and g agree for all arguments (including, in case of partial functions, whether or not it is defined).

Definition 3.36. For a function $f: A\to B$ and a subset S of A, the *image*[44] of S under f, denoted $f(S)$, is the set

$$f(S)\ \stackrel{\text{def}}{=}\ \{f(a)\,|\,a\in S\}.$$

Definition 3.37. The subset $f(A)$ of B is called the *image* (or *range*) of f and is also denoted $\text{Im}(f)$.

Example 3.53. Consider the function $f:\mathbb{R}\to\mathbb{R}$ defined by $f(x)=x^2$. The image of the interval $[2,3]$ is the interval $[4,9]$. The range of f is the set $\mathbb{R}^{\geq 0}$ of non-negative real numbers.

Definition 3.38. For a subset T of B, the *preimage*[45] of T, denoted $f^{-1}(T)$, is the set of values in A that map into T:

$$f^{-1}(T)\ \stackrel{\text{def}}{=}\ \{a\in A|\ f(a)\in T\}.$$

Example 3.54. Consider again the function $f(x)=x^2$. The preimage of the interval $[4,9]$ is $[-3,-2]\cup[2,3]$.

[43] This notation is motivated by the fact that if A and B are finite, then there are $|B|^{|A|}$ such functions.
[44] German: Bild
[45] German: Urbild

Definition 3.39. A function $f : A \to B$ is called

1. *injective* (or *one-to-one* or an *injection*) if for $a \neq a'$ we have $f(a) \neq f(a')$, i.e., no two distinct values are mapped to the same function value (there are no "collisions").
2. *surjective* (or *onto*) if $f(A) = B$, i.e., if for every $b \in B$, $b = f(a)$ for some $a \in A$ (every value in the codomain is taken on for some argument).
3. *bijective* (or a *bijection*) if it is both injective and surjective.

Definition 3.40. For a bijective function f, the *inverse* (as a relation, see Definition 3.11) is called the inverse function[46] of f, usually denoted as f^{-1}.

Definition 3.41. The *composition* of a function $f : A \to B$ and a function $g : B \to C$, denoted by $g \circ f$ or simply gf, is defined by $(g \circ f)(a) = g(f(a))$.[47]

Example 3.55. Consider again the function $f(x) = x^3 + 3$ and $g(x) = 2x^2 + x$. Then $g \circ f(x) = 2(f(x))^2 + f(x) = 2x^6 + 13x^3 + 21$.

Lemma 3.14. *Function composition is associative, i.e.,* $(h \circ g) \circ f = h \circ (g \circ f)$.

Proof. This is a direct consequence of the fact that relation composition is associative (see Lemma 3.7). □

3.7 Countable and Uncountable Sets

3.7.1 Countability of Sets

Countability is an important concept in Computer Science. A set that is countable can be enumerated by a program (even though this would take unbounded time), while an uncountable set can, in principle, not be enumerated.

[46] It is easy to see that this is a function

[47] Note that the composition of functions is the same as the composition of relations. However, unfortunately, different notation is used: The composition of relations f and g is denoted $f \circ g$ while, if considered as functions, the same resulting composition is denoted as $g \circ f$. (The reason is that one thinks of functions as mapping "from right to left".) Because of this ambiguity one must make explicit whether the symbol $\circ$ refers to function or relation composition.

Definition 3.42.

(i) Two sets A and B *equinumerous*[48], denoted $A \sim B$, if there exists a bijection $A \to B$.

(ii) The set B *dominates* the set A, denoted $A \preceq B$, if $A \sim C$ for some subset $C \subseteq B$ or, equivalently, if there exists an injective function $A \to B$.

(iii) A set A is called *countable*[49] if $A \preceq \mathbb{N}$, and *uncountable*[50] otherwise.[51]

Example 3.56. The set $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ of integers is countable, and $\mathbb{Z} \sim \mathbb{N}$. A bijection $f : \mathbb{N} \to \mathbb{Z}$ is given by $f(n) = (-1)^n \lceil n/2 \rceil$.

Lemma 3.15. [52]

(i) The relation $\sim$ is an equivalence relation.

(ii) The relation $\preceq$ is transitive: $A \preceq B \wedge B \preceq C \implies A \preceq C$.

(iii) $A \subseteq B \implies A \preceq B$.

Proof. Proof of (i). Assume $A \sim B$ and $B \sim C$, i.e., there exist bijections $f : A \to B$ and $g : B \to C$. Then $g \circ f$ is a bijection $A \to C$ and hence we have $A \sim C$.

Proof of (ii). If there is an injection from A to B and also an injection from B to C, then their composition is an injection from A to C. (We omit the proof of this statement.)

Proof of (iii). If $A \subseteq B$, then the identity function on A is an injection from A to B. □

A non-trivial theorem, called the Bernstein-Schröder theorem, is stated without proof.[53] It is not needed in this course.

Theorem 3.16. $A \preceq B \wedge B \preceq A \implies A \sim B$.

3.7.2 Between Finite and Countably Infinite

For finite sets A and B, we have $A \sim B$ if and only if $|A| = |B|$. A finite set has never the same cardinality as one of its proper subsets. Somewhat surprisingly, for infinite sets this is possible.

[48] German: gleich mächtig

[49] German: abzählbar

[50] German: überabzählbar

[51] Recall that $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.

[52] Here $\sim$ and $\preceq$ should be understood as relations on a given set of sets.

[53] An elegant proof of this theorem is given in *Proofs from THE BOOK* by M. Aigner and G. Ziegler.

Example 3.57. Let $\mathbf{O} = \{1, 3, 5, \ldots\}$ be the set of odd natural numbers. Of course, $\mathbf{O}$ is countable since the identity function is a (trivial) injection from $\mathbf{O}$ to $\mathbb{N}$. Actually, there is even a *bijection* $f : \mathbb{N} \to \mathbf{O}$, namely $f(n) = 2n + 1$. Indeed, Theorem 3.17 below states a more general fact.

Theorem 3.17. *A set A is countable if and only if it is finite or if $A \sim \mathbb{N}$.*

The theorem can be restated as follows: There is no cardinality level between finite and countably infinite.

Proof. A statement of the form "if and only if" has two directions. To prove the direction $\Longleftarrow$, note that if A is finite, then it is countable, and also if $A \sim \mathbb{N}$, then A is countable.

To prove the other direction ($\Longrightarrow$), we prove that if A is countable and infinite, then $A \sim \mathbb{N}$. According to the definition, $A \preceq \mathbb{N}$ means that there is a bijection $f : A \to C$ for a set $C \subseteq \mathbb{N}$. For any infinite subset of $\mathbb{N}$, say C, one can define a bijection $g : C \to \mathbb{N}$ as follows. According to the well-ordering principle, there exists a least element of C, say c_0. Define $g(c_0) = 0$. Define $C_1 = C \setminus \{c_0\}$. Again, according to the well-ordering principle, there exists a least element of C_1, say c_1. Define $g(c_1) = 1$. This process can be continued, defining inductively a bijection $g : C \to \mathbb{N}$. Now $g \circ f$ is a bijection $A \to \mathbb{N}$, which proves $A \sim \mathbb{N}$. □

3.7.3 Important Countable Sets

Theorem 3.18. *The set $\{0,1\}^* \stackrel{\text{def}}{=} \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$ of finite binary sequences is countable.*[54]

Proof. We could give an enumeration of the set $\{0,1\}^*$, i.e., a bijection between $\{0,1\}^*$ and $\mathbb{N}$, but to prove the theorem it suffices to provide an injection $\{0,1\}^* \to \mathbb{N}$, which we define as follows. We put a "1" at the beginning of the string and then interpret it as an natural number using the usual binary representation of the natural numbers. For example, the string 0010 is mapped to the number 18.[55] □

Theorem 3.19. *The set $\mathbb{N} \times \mathbb{N}$ $(= \mathbb{N}^2)$ of ordered pairs of natural numbers is countable.*

Proof. A possible bijection $f : \mathbb{N} \to \mathbb{N}^2$ is given by $f(n) = (k, m)$, where k and m are determined using the following equations: $k + m = t - 1$ and

[54] Here ϵ denotes the empty string.

[55] Note that without prepending a 1, different strings (e.g. 0010 and 00010) would result in the same integer and hence the mapping would not be an injection.

$m = n - \binom{t}{2}$, where $t > 0$ is the smallest integer such that $\binom{t+1}{2} > n$. This corresponds to the enumeration of the pairs (k, m) along diagonals with constant sum $k + m$. More concretely, we enumerate the pairs as follows: $(0,0)$, $(1,0)$, $(0,1)$, $(2,0)$, $(1,1)$, $(0,2)$, $(3,0)$, $(2,1)$, $(1,2)$, $(0,3)$, $(4,0)$, $(3,1)$, $\cdots$. It is easy to see that this is a bijection $\mathbb{N} \to \mathbb{N}^2$, hence $\mathbb{N}^2$ is countable. □

An alternative proof works as follows. We have $\mathbb{N} \sim \{0,1\}^*$ and hence $\mathbb{N} \times \mathbb{N} \sim \{0,1\}^* \times \{0,1\}^*$. Hence it suffices to show a bijection $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. The concatenation of the two sequences is not a bijection because a bit-string can be split in many ways into two strings. One possibility to map a pair (a, b) of bit-strings to a single bit-string is as follows:

$$(a, b) \;\mapsto\; 0^{|a|}||1||a||b,$$

where we first encode the length $|a|$ of a and then append a and b.

Corollary 3.20. *The Cartesian product $A \times B$ of two countable sets A and B is countable, i.e., $A \preceq \mathbb{N} \wedge B \preceq \mathbb{N} \implies A \times B \preceq \mathbb{N}$.*

Proof. We first prove $A \times B \preceq \mathbb{N} \times \mathbb{N}$ by exhibiting an injection $g :\; A \times B \to \mathbb{N} \times \mathbb{N}$, namely $g(a, b) = (f_1(a), f_2(b))$. That g is an injection can be proved as follows:

$$\begin{array}{rcll}
(a, b) \neq (a', b') & \Longrightarrow & a \neq a' \vee b \neq b' & \text{(definition of pairs)} \\
& \Longrightarrow & f_1(a) \neq f_1(a') \vee f_2(b) \neq f_2(b') & (f_1 \text{ and } f_2 \text{ are injections}) \\
& \Longrightarrow & \big(f_1(a), f_2(b)\big) \neq \big(f_1(a'), f_2(b')\big) & \text{(definition of pairs).}
\end{array}$$

Using $A \times B \preceq \mathbb{N} \times \mathbb{N}$ (just proved) and $\mathbb{N} \times \mathbb{N} \preceq \mathbb{N}$ (Theorem 3.19) now gives $A \times B \preceq \mathbb{N}$ because $\preceq$ is transitive (Lemma 3.15(i)). □

Corollary 3.21. *The rational numbers $\mathbb{Q}$ are countable.*

Proof. Every rational number can be represented uniquely as a pair (m, n) where $m \in \mathbb{Z}$, $n \in \mathbb{N} \setminus \{0\}$, and where m and n are relatively prime. Hence $\mathbb{Q} \preceq \mathbb{Z} \times \mathbb{N}$. According to Example 3.56, $\mathbb{Z}$ is countable, i.e., $\mathbb{Z} \preceq \mathbb{N}$. Thus, according to Corollary 3.20, $\mathbb{Z} \times \mathbb{N} \preceq \mathbb{N}$. Hence, using transitivity of $\preceq$, we have $\mathbb{Q} \preceq \mathbb{N}$ (i.e., $\mathbb{Q}$ is countable). □

The next theorem provides some other important sets that are countable.

Theorem 3.22. *Let A and A_i for $i \in \mathbb{N}$ be countable sets.*

(i) For any $n \in \mathbb{N}$, the set A^n of n-tuples over A is countable.

(ii) The union $\cup_{i \in \mathbb{N}} A_i$ of a countable list $A_0, A_1, A_2, \ldots$ of countable sets is countable.

(iii) The set A^* of finite sequences of elements from A is countable.

Proof. Statement (i) can be proved by induction. The (trivial) induction basis is that $A^1 = A$ is countable. The induction step shows that if A^n is countable, then also $A^{n+1} \sim A^n \times A$ is countable. This follows from Corollary 3.20 because both A^n and A are countable.

We omit the proof of (ii).

We now prove (iii), which implies (i), and hence gives an alternative proof for (i). We define an injection $A^* \to \{0,1\}^*$. This is achieved by using an arbitrary injection $f: A \to \{0,1\}^*$ and defining the natural injection $g: A^* \to (\{0,1\}^*)^*$ as follows: For a sequence of length n in A^*, say $(a_1, \ldots, a_n)$, we let

$$g(a_1, \ldots, a_n) = \big(f(a_1), \ldots, f(a_n)\big),$$

i.e., each element in the sequence is mapped separately using f. Now it only remains to demonstrate an injection $(\{0,1\}^*)^* \to \{0,1\}^*$, which can be achieved as follows.[56] We replace every 0-bit in a sequence by 00 and every 1-bit by 01, which defines a (length-doubling) injection $\{0,1\}^* \to \{0,1\}^*$. Then we concatenate all obtained expanded sequences, always separated by 11. This is an injection because the separator symbols 11 can be detected and removed and the extra 0's can also be removed. Hence a given sequence can be uniquely decomposed into the component sequences, and hence no two sequences of binary (component) sequences can result in the same concatenated sequence. □

Example 3.58. We illustrate the above injection $(\{0,1\}^*)^* \to \{0,1\}^*$ by an example. Consider the sequence $(0100, 10111, 01, 1)$ of bit-sequences. Now 0100 is mapped to 00010000, 10111 is mapped to 0100010101, etc. and the final concatenated sequence is 000100001101000101011100011101, which can uniquely be decomposed into the original four sequences.

3.7.4 Uncountability of $\{0,1\}^\infty$

We now consider semi-infinite binary sequences $(s_0, s_1, s_2, s_3, \ldots)$. One can interpret such a binary sequence as specifying a subset of $\mathbb{N}$: If $s_i = 1$, then i is in the set, otherwise it is not. Equivalently, we can understand a semi-infinite sequence $(s_0, s_1, s_2, s_3, \ldots)$ as a function $\mathbb{N} \to \{0,1\}$, i.e., as a predicate on $\mathbb{N}$. For example, the primality predicate $\texttt{prime}: \mathbb{N} \to \{0,1\}$ (where $\texttt{prime}(n) = 1$ if and only if n is prime) corresponds to the semi-infinite sequence 001101010001010001010001000001010000001 . . ..

Definition 3.43. Let $\{0,1\}^\infty$ denote the set of semi-infinite binary sequences or, equivalently, the set of functions $\mathbb{N} \to \{0,1\}$.

[56] Note that a simple concatenation of the sequences does not work because the concatenated sequences can not uniquely be decomposed into the original sequences, i.e., this is not an injection.

Theorem 3.23. *The set $\{0,1\}^\infty$ is uncountable.*

Proof. This is a proof by contradiction. To arrive at a contradiction, assume that a bijection $f: \mathbb{N} \to \{0,1\}^\infty$ exists.[57] Let $\beta_{i,j}$ be the jth bit in the i-th sequence $f(i)$, where for convenience we begin numbering the bits with $j = 0$:

$$f(i) \stackrel{\text{def}}{=} \beta_{i,0}, \beta_{i,1}, \beta_{i,2}, \beta_{i,3}, \ldots.$$

Let $\overline{b}$ be the complement of a bit $b \in \{0,1\}$. We define a new semi-infinite binary sequence α as follows:

$$\alpha \stackrel{\text{def}}{=} \overline{\beta_{0,0}}, \overline{\beta_{1,1}}, \overline{\beta_{2,2}}, \overline{\beta_{3,3}}, \ldots.$$

Obviously, $\alpha \in \{0,1\}^\infty$, but there is no $n \in \mathbb{N}$ such that $\alpha = f(n)$ since α is constructed so as to disagree in at least one bit (actually the nth bit) with every sequence $f(n)$ for $n \in \mathbb{N}$. This shows that f cannot be a bijection, which concludes the proof. □

This proof technique is known as Cantor's *diagonalization argument*; it has also other applications.

By interpreting the elements of $\{0,1\}^\infty$ as the binary expansion of a real number in the interval $[0,1]$, and vice versa, one can show that the interval $[0,1]$ (and hence $\mathbb{R}$ itself), is uncountable.[58]

3.7.5 Existence of Uncomputable Functions

The above theorem states that there are uncountably many functions $\mathbb{N} \to \{0,1\}$. On the other hand, every computer program, regardless of the programming language it is written in, corresponds to a *finite* string of symbols. Without loss of generality, one can think of a program as a finite binary sequence $p \in \{0,1\}^*$). Hence the set of programs is countable, whereas the set of functions $\mathbb{N} \to \{0,1\}$ is uncountable. If every program computes at most one function, there must be functions $\mathbb{N} \to \{0,1\}$ not computed by a program. This for Computer Science fundamental consequence of Theorem 3.23 is stated below.

Definition 3.44. A function $f: \mathbb{N} \to \{0,1\}$ is called *computable* if there is a program that, for every $n \in \mathbb{N}$, when given n as input, outputs $f(n)$.

[57] Here we make use of Theorem 3.17 which implies that $\{0,1\}^\infty$ is countable if and only if such a bijection exists.

[58] A subtlety, which is not a problem in the proof, is that some real numbers have two representations as bit-strings. For example, the number 0.5 has representations $10000000\cdots$ and $0111111\cdots$.

Corollary 3.24. *There are uncomputable functions $\mathbb{N} \to \{0,1\}$.*

In fact, essentially all such functions are uncomputable. Those that are computable are rare exceptions. For example, the function `prime` $: \mathbb{N} \to \{0,1\}$ is computable.

Is there a specific uncomputable function? A prominent example is the so-called *Halting problem* defined as follows: Given as input a program (encoded as a bit-string or natural number) together with an input (to the program), determine whether the program will eventually stop (function value 1) or loop forever (function value 0) on that input. This function is uncomputable. This is usually stated as: *The Halting problem is undecidable.*

This theorem can also be proved by a diagonalization argument similar to the one above. The theorem has far-reaching consequences in theoretical and practical Computer Science. It implies, for example, that it is impossible to write a program that can verify (in the most general case) whether a given program satisfies its specification, or whether a given program contains malicious parts.

Chapter 4

Number Theory

4.1 Introduction

Number theory is one of the most intriguing branches of mathematics. For a long time, number theory was considered one of the purest areas of mathematics, in the sense of being most remote from having applications. However, since the 1970's number theory has turned out to have intriguing and unexpected applications, primarily in cryptography.

In this course we discuss only some basic number-theoretic topics that have applications in Computer Science. In addition to the rich applications, a second reason for discussing the basics of number theory in a course in Computer Science is as a preparation for the chapter on algebra (Chapter 5).

4.1.1 Number Theory as a Mathematical Discipline

Number theory (in a strict sense[1]) is the mathematical theory of the natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$ or, more generally, of the integers, $\mathbb{Z}$. The laws of the integers are so natural, simple, and well-known to us that it is amazing how apparently simple questions about the integers turn out to be extremely difficult and have resisted all attacks by the brightest mathematicians.

Example 4.1. A simple conjecture unproven to date is that there are infinitely many prime pairs, i.e., primes p such that $p + 2$ is also prime. The first prime pairs are $(3, 5)$, $(5, 7)$, $(11, 13)$, and $(17, 19)$.

Example 4.2. Can one find a triangle with a $90°$ angle whose three sides a, b, and c have integer lengths? An equivalent question is whether there exist positive

[1] In a more comprehensive understanding, number theory refers to a richer mathematical theory which also includes topics like, for instance, algebraic extensions of the rational numbers.

integers a, b, and c such that $a^2 + b^2 = c^2$. The answer is yes. Examples are $3^2 + 4^2 = 5^2$ and $12^2 + 5^2 = 13^2$. A straight-forward generalization of this question is whether there exist positive integers a, b, c, and $n \geq 3$ such that $a^n + b^n = c^n$. The answer (no such integers exist) is known as *Fermat's last theorem*, which remained one of the most famous open conjectures until Andrew Wiles settled the question some years ago, using highly sophisticated mathematics.

Example 4.3. The recent proof of the *Catalan conjecture* by Preda Mihailescu, who worked at ETH Zürich, is another break-through in number theory. This theorem states that the equation $a^m - b^n = 1$ has no other integer solutions but $3^2 - 2^3 = 1$ (for $m, n \geq 2$).

4.1.2 What are the Integers?

In this course we are trying to present a rigorous mathematical treatment of the material. Consequently, in order to present number theory, it appears that we would first have to define the integers, so we know what we are talking about, in contrast to the intuitive understanding of numbers acquired since the early years at school. However, such a formal, axiomatic treatment of the integers is beyond the scope of the course.

In this chapter we take the usual approach where we assume that we know what numbers and operations on numbers are and that we also know the basic facts about numbers (e.g. the commutative, associative and distributive laws, etc.) which we can use to prove statements. But we should point out that in such an informal approach it is difficult (if not impossible) to draw the dividing line between facts that are well-known and facts that require a proof. For example, why is there no integer between 0 and 1, why is $-0 = 0$, and why is $a^2 \geq 0$ for all $a \in \mathbb{Z}$? What is the complete list of facts we consider known, and which facts require a proof? The answer is not clear unless one states a list of axioms. For example, we will show an interesting proof of the fact that every number can be factored uniquely into primes. This is definitely a theorem that requires a proof, even though, after many years of mathematical education, the reader may consider it a well-known basic fact.

The integers are a special case of a mathematical structure called a *ring*, which will be discussed in Chapter 5. In this chapter we mention in a few places that concepts like divisors, greatest common divisors, ideals, etc. can be defined for any ring, not just for the integers.

4.2 Divisors and Division

4.2.1 Divisors

Definition 4.1. For integers a and b we say that a *divides* b, denoted $a \mid b$, if there exists an integer c such that $b = ac$. In this case, a is called a *divisor*[2] of b, and b is called a *multiple*[3] of a. If $a \neq 0$ and a divisor c exists it is called the[4] *quotient* when b is divided by a, and we write $c = \frac{b}{a}$ or $c = b/a$. We write $a \nmid b$ if a does not divide b.

Note that every non-zero integer is a divisor of 0. Moreover, 1 and -1 are divisors of every integer.

4.2.2 Division with Remainders

In the previous section we defined division of a by d for any divisor d of a. In this section we generalize division to the case where d is not a divisor of a and hence division yields a remainder[5]. The following theorem was proved by Euclid around 300 B.C.

Theorem 4.1 (Euclid). *For all integers a and $d \neq 0$ there exist unique integers q and r satisfying*

$$a = dq + r \qquad \textit{and} \qquad 0 \leq r < |d|.$$

Here a is called the *dividend*, d is called the *divisor*, q is called the *quotient*, and r is called the *remainder*. The remainder r is often denoted as $R_d(a)$ or sometimes as $a \bmod d$.

Proof. We carry out this proof in a detailed manner, also to serve as an example of a systematic proof.

We define S to be the set of possible nonnegative remainders:

$$S \stackrel{\text{def}}{=} \{s \mid s \geq 0 \ \text{ and } \ a = dt + s \ \text{ for some } t \in \mathbb{Z}\}.$$

We prove the following three claims by first proving 1), then proving that 1) implies 2), and then proving that 2) implies 3).

1) S is not empty.
2) S contains an $r < |d|$.
3) The r of claim 2) is unique.

[2] German: Teiler
[3] German: Vielfaches
[4] One can prove that it is unique.
[5] German: Rest

Proof of 1): We use case distinction and prove the statement for three cases (one of which is always satisfied):
Case 1: $a \geq 0$. Then $a = d0 + a$ and hence $a \in S$.
Case 2: $a < 0$ and $d > 0$. Then $a = da + (1-d)a$ and thus $(1-d)a \in S$ since $(1-d)a \geq 0$ because both $(1-d)$ and a are ≤ 0.
Case 3: $a < 0$ and $d < 0$. Then $a = d(-a) + (1+d)a$ and thus $(1+d)a \in S$ since $(1+d)a \geq 0$ because both $(1+d)$ and a are ≤ 0.

Proof that 1) implies 2): Because S is not empty, it has a smallest element (due to the well-ordering principle), which we denote by r. We now prove that $r < |d|$, by contradiction, i.e., assuming $r \geq |d|$. By definition of S we have $a = dq + r$ for some q. We make a case distinction: $d > 0$ and $d < 0$. If $d > 0$, then

$$a = d(q+1) + (r - |d|),$$

hence $r - |d| \geq 0$ and therefore $r - |d| \in S$, which means that r is not the smallest element of S, a contradiction. If $d < 0$, then $a = d(q-1) + (r - |d|)$, and the same argument as above shows that $r - |d| \in S$, a contradiction.

Proof that 2) implies 3): It remains to prove that r is unique. We give a proof only for $d > 0$; the case $d < 0$ is analogous and is left as an exercise. The proof is by contradiction. Suppose that there also exist $r' \neq r$ with $0 \leq r' < |d|$ and such that $a = dq' + r'$ for some q'. We distinguish the three cases $q' = q$, $q' < q$, and $q' > q$. If $q' = q$, then $r' = a - dq' = a - dq = r$, a contradiction since we assumed $r' \neq r$. If $q' < q$, then $q - q' \geq 1$, so

$$r' = a - dq' = (a - dq) + d(q - q') \ \geq \ r + d.$$

Since $r' \geq r + d \geq d$, the condition $0 \leq r' < |d|$ is violated, which is a contradiction. A symmetric argument shows that $q' > q$ also results in a contradiction, □

4.2.3 Greatest Common Divisors

Definition 4.2. For integers a and b (not both 0), an integer d is called a *greatest common divisor*[6] of a and b if d divides both a and b and if every common divisor of a and b divides d, i.e., if

$$d \mid a \ \wedge \ d \mid b \ \wedge \ \forall c \left((c \mid a \ \wedge \ c \mid b) \ \rightarrow \ c \mid d \right).$$

The concept of a greatest common divisor applies not only to $\mathbb{Z}$, but to more general structures (e.g. polynomial rings). If d and d' are both greatest common divisors of a and b, then $d \mid d'$ and $d' \mid d$. For the integers $\mathbb{Z}$, this means that $d' = \pm d$, i.e., there are two greatest common divisors. (But for more general structures there can be more than two greatest common divisors.)

[6]Note that the term "greatest" does not refer to the order relation $\leq$ but to the divisibility relation.

Definition 4.3. For $a, b \in \mathbb{Z}$ (not both 0) one denotes the unique positive greatest common divisor by $\gcd(a,b)$ and usually calls it *the* greatest common divisor. If $\gcd(a,b) = 1$, then a and b are called *relatively prime*[7].

Lemma 4.2. *For any integers m, n and q, we have*

$$\gcd(m, n - qm) = \gcd(m, n).$$

Proof. It is easy to prove (as an exercise) that every common divisor of m and $n - qm$ (and therefore also the greatest) is also a common divisor of m and n, and vice versa. □

This lemma implies in particular that

$$\gcd(m, R_m(n)) = \gcd(m, n),$$

which is the basis for Euclid's well-known $\gcd$-algorithm: Start with $m < n$ and repeatedly replace the pair (m, n) by the pair $(R_m(n), m)$ until the remainder is 0, at which point the last non-zero number is equal to $\gcd(m, n)$.

Definition 4.4. For $a, b \in \mathbb{Z}$, the *ideal generated by a and b*[8], denoted (a, b), is the set

$$(a, b) \stackrel{\text{def}}{=} \{ua + vb \mid u, v \in \mathbb{Z}\}.$$

Similarly, the ideal generated by a single integer a is

$$(a) \stackrel{\text{def}}{=} \{ua \mid u \in \mathbb{Z}\}.$$

The following lemma implies that every ideal in $\mathbb{Z}$ can be generated by a single integer.

Lemma 4.3. *For $a, b \in \mathbb{Z}$ there exists $d \in \mathbb{Z}$ such that $(a, b) = (d)$.*

Proof. If $a = b = 0$, then $d = 0$. Assume now that at least one of the numbers is non-zero. Then (a, b) contains some positive numbers, so (by the well-ordering principle) let d be the smallest positive element in (a, b). Clearly $(d) \subseteq (a, b)$ since every multiple of d is also in (a, b). It remains to prove $(a, b) \subseteq (d)$. For any $c \in (a, b)$ there exist q and r with $0 \leq r < d$ such that $c = qd + r$. Since both c and d are in (a, b), so is $r = c - qd$. Since $0 \leq r < d$ and d is (by assumption) the smallest positive element in (a, b), we must have $r = 0$. Thus $c = qd \in (d)$. □

Lemma 4.4. *Let $a, b \in \mathbb{Z}$ (not both 0). If $(a, b) = (d)$, then d is a greatest common divisor of a and b.*

[7]German: teilerfremd
[8]German: durch a und b erzeugtes Ideal

Proof. d is a common divisor of a and b since $a \in (d)$ and $b \in (d)$. To show that d is a greatest common divisor, i.e., that every common divisor c of a and b divides d, note that c divides every integer of the form $ua + vb$, in particular d. □

The following corollary follows from Lemmas 4.3 and 4.4.

Corollary 4.5. *For $a, b \in \mathbb{Z}$ (not both 0), there exist $u, v \in \mathbb{Z}$ such that*

$$\gcd(a, b) = ua + vb.$$

Example 4.4. For $a = 26$ and $b = 18$ we have

$$\gcd(26, 18) \;=\; 2 \;=\; (-2) \cdot 26 + 3 \cdot 18.$$

Also, for $a = 17$ and $b = 13$ we have

$$\gcd(17, 13) \;=\; 1 \;=\; (-3) \cdot 17 + 4 \cdot 13.$$

An extension of Euclid's well-known gcd-algorithm allows to efficiently compute not only $\gcd(a, b)$, but also u and v such that $\gcd(a, b) = ua + vb$.

4.2.4 Least Common Multiples

The least common multiple is a dual concept of the greatest common divisor.

Definition 4.5. The *least common multiple* l of two positive integers a and b, denoted $l = \mathrm{lcm}(a, b)$, is the common multiple of a and b which divides every common multiple of a and b, i.e.,

$$a \mid l \;\wedge\; b \mid l \;\wedge\; \forall m \big((a \mid m \;\wedge\; b \mid m) \;\rightarrow\; l \mid m \big).$$

4.3 Factorization into Primes

4.3.1 Primes and the Fundamental Theorem of Arithmetic

In this section we prove the well-known fact that prime factorization of integers is unique. This statement is true more generally for certain types of rings (see Chapter 5), for example for the ring of polynomials over a field. Even though rings were not introduced so far, we give hints as to how a formulation can be generalized from the integers to more general rings.

Definition 4.6. A positive integer $p > 1$ is called *prime* if the only positive divisors of p are 1 and p. An integer greater than 1 that is not a prime is called *composite*[9].[10]

[9] German: zusammengesetzt

[10] Note that 1 is neither prime nor composite.

This notion of having only trivial divisors extends to other rings, for example the ring of polynomials over $\mathbb{R}$. In such a general context, the property is called *irreducible* rather than *prime*. The term *prime* is in general used for the property that if p divides a product of elements, then it divides at least one of them (see Lemma 4.7 below). For the integers, these two concepts are equivalent. The next lemma states one direction of this equivalence.

The following theorem is called the fundamental theorem of arithmetic.

Theorem 4.6. *Every positive integer can be written uniquely (up to the order in which factors are listed) as the product of primes.*[11]

4.3.2 Proof of the Fundamental Theorem of Arithmetic *

Lemma 4.7. *If p is a prime which divides the product $x_1 x_2 \cdots x_n$ of some integers $x_1, \ldots, x_n$, then p divides one of them, i.e., $p \mid x_i$ for some $i \in \{1, \ldots, n\}$.*

Proof. The proof is by induction on n. The claim is trivially true for $n = 1$ (induction basis). Suppose it is true for some general n (induction hypothesis). To prove the claim for $n + 1$ (induction step), suppose that $p \mid x_1 \cdots x_{n+1}$. We let $y := x_1 \cdots x_n$ (and hence $p \mid y x_{n+1}$) and look at the two cases $p \mid y$ and $p \nmid y$ separately. If $p \mid y$, then $p \mid x_i$ for some $1 \leq i \leq n$, due to the induction hypothesis, and we are done. If $p \nmid y$, then, since p has no positive divisor except 1 and p, we have $\gcd(p, y) = 1$. By Corollary 4.5 there are integers u and v such that $up + vy = 1$. Hence we have

$$x_{n+1} = (up + vy)x_{n+1} = (u x_{n+1})p + v(y x_{n+1}).$$

Because p divides both terms $(u x_{n+1})p$ and $v(y x_{n+1})$ in the sum on the right side, it follows that it also divides the sum, i.e., $p \mid x_{n+1}$, which concludes the proof. □

We now prove Theorem 4.6:

Proof. We first need to prove that a factorization into primes exists and then that it is unique.

The existence is proved by contradiction. Every prime can obviously be factored into primes. Let n be the smallest positive integer which has no prime factorization. Since it can not be a prime, we have $n = km$ with $1 < k, m < n$. Since both k and m can be factored into primes, so can $km = n$, a contradiction. Hence there is no smallest n that cannot be factored into primes, and therefore every $n \geq 1$ can be factored into primes.

To prove the uniqueness of the prime factorization, suppose towards a contradiction that an integer n can be factored in two (possibly different) ways as a product of primes,

$$n = p_1^{a_1} p_2^{a_2} \cdots p_r^{a_r} = q_1^{b_1} q_2^{b_2} \cdots q_s^{b_s},$$

where the primes $p_1, \ldots, p_r$ and also the primes $q_1, \ldots, q_s$ are put in an increasing order and where we have written products of identical primes as powers (here $a_i > 0$ and

[11] Note that 1 has zero prime factors, which is allowed.

$b_i > 0$). Then for every i, $p_i \mid n$ and thus $p_i \mid q_1^{b_1} q_2^{b_2} \cdots q_s^{b_s}$. Hence, by Lemma 4.7, $p_i \mid q_j$ for some j and, because q_j is a prime and $p_i > 1$, we have $p_i = q_j$. Similarly for every j, $q_j = p_i$ for some i. Thus the set of primes are identical, i.e., $r = s$ and $p_i = q_i$ for $1 \leq i \leq r$. To show that the corresponding exponents a_i and b_i are also identical, suppose that $a_i < b_i$ for some i. We can divide both expressions by $p_i^{a_i}$, which results in two numbers that are equal, yet one is divisible by p_i while the other is not. This is impossible since if two numbers are equal, then they have the same divisors. □

4.3.3 Expressing gcd and lcm

The fundamental theorem of arithmetic assures that integers a and b can be written as

$$a = \prod_i p_i^{e_i} \qquad \text{and} \qquad b = \prod_i p_i^{f_i}.$$

This product can be understood in two different ways. Either it is over all primes, where all but finitely many of the e_i are 0, or it is over a fixed agreed set of primes. Either view is correct. Now we have

$$\gcd(a,b) = \prod_i p_i^{\min(e_i,f_i)}$$

and

$$\operatorname{lcm}(a,b) = \prod_i p_i^{\max(e_i,f_i)}.$$

It is easy to see that

$$\gcd(a,b) \cdot \operatorname{lcm}(a,b) \;=\; ab$$

because for all i we have

$$\min(e_i,f_i) + \max(e_i,f_i) \;=\; e_i + f_i.$$

4.3.4 Non-triviality of Unique Factorization *

It is worth-while pointing out that this theorem is not self-evident, as it may appear to the reader completely familiar with it. There are in fact examples of rings in which the unique factorization into irreducible elements does not hold. We give two examples, one with unique factorization into irreducible elements, and one without.

Example 4.5. Let $i = \sqrt{-1}$ denote the complex imaginary unit. The Gaussian integers

$$\mathbb{Z}[i] = \mathbb{Z}[\sqrt{-1}] = \{a + bi \mid a, b \in \mathbb{Z}\}$$

are the complex numbers whose real and imaginary parts are both integers. Since the norm (as complex numbers) is multiplied when two elements of $\mathbb{Z}[i]$ are multiplied, the units (actually four) are the elements with norm 1, namely $1, i, -1$, and $-i$. Units are elements that divide every other element. An irreducible element p in $\mathbb{Z}[i]$ is an element whose only divisors are units and associates of p (i.e., elements up where u is a unit). By a generalization of the arguments given for $\mathbb{Z}$ one can show that factorization into irreducible elements is unique in $\mathbb{Z}[i]$. (This is not obvious.)

Example 4.6. Consider now a slightly twisted version of the Gaussian integers:

$$\mathbb{Z}[\sqrt{-5}] = \{a + b\sqrt{5}i \mid a, b \in \mathbb{Z}\}.$$

Like the Gaussian integers, this set is closed with respect to addition and multiplication (of complex numbers). For example,

$$(a + b\sqrt{5}i)(c + d\sqrt{5}i) = ac - 5bd + (bc + ad)\sqrt{5}i.$$

The only units in $\mathbb{Z}[\sqrt{-5}]$ are 1 and -1. One can check easily, by ruling out all possible divisors with smaller norm, that the elements $2, 3, 1+\sqrt{5}i$, and $1-\sqrt{5}i$ are irreducible. The element 6 can be factored in two different ways into irreducible elements:

$$6 = 2 \cdot 3 = (1 + \sqrt{5}i)(1 - \sqrt{5}i).$$

4.3.5 Irrationality of Roots *

As a consequence of the unique prime factorization we can prove:

Theorem 4.8. *$\sqrt{n}$ is irrational unless n is a square ($n = c^2$ for some $c \in \mathbb{Z}$).*

Proof. Suppose $\sqrt{n} = a/b$ for two integers a and b. Then $a^2 = nb^2$. If n is not a square, it contains at least one prime factor p with an odd power. Since the number of prime factors p in any square is even, we have a contradiction: a^2 contains an even number of factors p while nb^2 contains an odd number of factors p. This is impossible according to Theorem 4.6. □

Note that this proof is simpler and more general than the proof given in Example 2.28 because there we have not made use of the unique prime factorization of the integers.

4.3.6 A Digression to Music Theory *

An octave in music corresponds to the doubling of the frequency of a tone. Similarly, a fifth[12] corresponds to a ratio 3 : 2, a musical fourth[13] corresponds to a ratio $4:3$, and a major and minor third[14] correspond to the ratios $5:4$ and $6:5$, respectively.

No multiple of fifths (or fourths) yields a multiple of octaves, since otherwise we would have $(\frac{3}{2})^n = 2^m$ for some n and m, which is equivalent to $3^n = 2^{m+n}$. This implies that one cannot tune a piano so that all intervals are correct since on the piano (considered to be extended to many octaves), one hits a higher octave after a certain number (namely 12) of fifths. It can be viewed as a number-theoretic miracle that tuning a piano is nevertheless possible with only very small inaccuracies. If one divides the octave into 12 equal (half-tone) intervals, a half-tone corresponds to a frequency ratio of $\sqrt[12]{2} \approx 1.05946$. Three, four, five, and seven half-tones yield frequency ratios of

$$2^{1/4} = 1.1892 \approx 6/5,$$

[12]German: Quinte
[13]German: Quarte
[14]German: Terz

$2^{1/3} = 1.2599 \approx 5/4$,
$2^{5/12} = 1.33482 \approx 4/3$, and
$2^{7/12} = 1.49828 \approx 3/2$,

approximating the minor third, major third, fourth, and fifth astonishingly well.

One can view these relations also as integer approximations. For example, we have $531'441 = 3^{12} \approx 2^{19} = 524'288$, which implies that $(\frac{3}{2})^{12} \approx 2^7$, i.e., 12 fifths are approximately seven octaves.

A piano for which every half-tone has the same frequency ratio, namely $\sqrt[12]{2}$, is called a *well-tempered*[15] piano. The reason why music is a pleasure, despite its theoretically unavoidable inaccuracy, is that our ear is trained to "round tones up or down" as needed.

4.4 Some Basic Facts About Primes *

4.4.1 The Density of Primes

The following fact was known already to Euclid.

Theorem 4.9. *There are infinitely many primes.*

Proof. To arrive at a contradiction, suppose that the set of primes is finite, say $P = \{p_1, \ldots, p_m\}$. Then the number $n = \prod_{i=1}^{m} p_i + 1$ is not divisible by any of the primes $p_1, \ldots, p_m$ and hence n is either itself a prime, or divisible by a prime not in $\{p_1, \ldots, p_m\}$. In either case, this contradicts the assumption that $p_1, \ldots, p_m$ are the only primes. □

This is a non-constructive existence proof. We now give a constructive existence proof for another number-theoretic fact.

Theorem 4.10. *Gaps between primes can be arbitrarily large, i.e., for every $k \in \mathbb{N}$ there exists $n \in \mathbb{N}$ such that the set $\{n, n+1, \cdots, n+k-1\}$ contains no prime.*

Proof. Let $n = (k+1)! + 2$. Then for any l with $2 \le l \le k+1$, l divides $(k+1)! = n-2$ and hence l also divides $(k+1)! + l = n - 2 + l$, ruling out $n, n+1, \ldots, n+k-1$ as possible primes. □

Example 4.7. The largest gap between two primes below 100 is 8. Which are these primes?

There exists a huge body of literature on the density and distribution of primes. We only state the most important one of them.

Definition 4.7. The *prime counting function* $\pi : \mathbb{R} \to \mathbb{N}$ is defined as follows: For any real x, $\pi(x)$ is the number of primes $\le x$.

[15] German: wohltemperiert

The following theorem proved by Hadamard and de la Vallée Poussin in the 19th century states that the density of primes $\le x$ is approximately $1/\ln(x)$. This shows that if one tries to find a large (say 1024 bit) prime, for instance for cryptographic purposes, then a randomly selected odd integer has reasonable chances of being prime. Much more precise estimates for $\pi(x)$ are known today.

Theorem 4.11. $\lim_{x\to\infty} \frac{\pi(x)\ln(x)}{x} = 1.$

Two of the main open conjectures on prime numbers are the following:

Conjecture 4.1. There exist infinitely many twin primes, i.e., primes p for which also $p+2$ is prime.

Conjecture 4.2 (Goldbach)**.** Every even number greater than 2 is the sum of two primes.

4.4.2 Remarks on Primality Testing

How can we test whether a given integer n is a prime? We can test whether any smaller prime is a divisor of n. The following lemma provides a well-known short-cut. In a practical implementation, one might not have a list of primes up to $\sqrt{n}$ and can instead simply try all odd numbers as possible divisors.

Lemma 4.12. *Every composite integer n has a prime divisor $\le \sqrt{n}$.*

Proof. If n is composite, it has a divisor a with $1 < a < n$. Hence $n = ab$ for $b > 1$. Either $a \le \sqrt{n}$ or $b \le \sqrt{n}$ since otherwise $ab > \sqrt{n} \cdot \sqrt{n} = n$. Hence n has a divisor c with $1 < c \le \sqrt{n}$. Either c is prime or, by Theorem 4.6, has a prime divisor $d < c \le \sqrt{n}$. □

For large integers, trial-division up to the square root is hopelessly inefficient. Let us briefly discuss the algorithmic problem of testing primality.

Primes are of great importance in cryptography. In many applications, one needs to generate very large primes, with 1024 or even 2048 bits. In many cases, the primes must remain secret and it must be infeasible to guess them. They should therefore be selected uniformly at random from the set of primes in a certain interval, possibly satisfying some further restrictions for security or operational reasons.

Such primes can be generated in three different ways. The first approach is to select a random (odd) integer from the given interval (e.g. $[10^{1023}, 10^{1024}-1]$) and to apply a general primality test. Primality testing is a very active research area. The record in general primality testing is around 8.000 decimal digits and required sophisticated techniques and very massive computing power. As a celebrated theoretical breakthrough (with probably little practical relevance), it was proved in 2002 that primality testing is in **P**, i.e., there is a worst-case polynomial-time algorithm for deciding primality of an integer.[16]

The second approach is like the first, but instead of a primality test one performs a probabilistic compositeness test. Such a test has two outcomes, "composite" and "possibly prime". In the first case, one is certain that the number is composite, while in the

[16] M. Agrawal, N. Kayal, and N. Saxena, PRIMES is in P, *Annals of Mathematics* vol. 160, pp. 781–793.

other case one has good chances that the number is a prime, without being certain. More precisely, one can fix a (very small) probability ϵ (e.g. $\epsilon = 10^{-100}$) and then perform a test such that for any composite integer, the probability that the test does not output "composite" is bounded by ϵ.

A third approach is to construct a prime together with a proof of primality. As we might see later, the primality of an integer n can be proved if one knows part of the factorization of $n-1$.

4.5 Congruences and Modular Arithmetic

4.5.1 Modular Congruences

We consider the following motivating example:

Example 4.8. Fermat's famous "last theorem", proved recently by Wiles, states that the equation $x^n + y^n = z^n$ has no solution in positive integers x, y, z for $n \geq 3$. Here we consider a similar question. Does $x^3 + x^2 = y^4 + y + 1$ have a solution in integers x and y?
The answer is "no", and the proof is surprisingly simple: Obviously, x must be either even or odd. In both cases, $x^3 + x^2$ is even. On the other hand, $y^4 + y + 1$ is odd no matter whether y is even or odd. But an even number cannot be equal to an odd number.

Here is another example whose solution requires a generalization of the above trick.

Example 4.9. Prove that $x^3 - x = y^2 + 1$ has no integer solutions.

Definition 4.8. For $a, b, m \in \mathbb{Z}$ with $m \geq 1$, we say that *a is congruent to b modulo m* if m divides $a - b$. We write $a \equiv b \pmod{m}$ or simply $a \equiv_m b$, i.e.,

$$a \equiv_m b \quad \stackrel{\text{def}}{\iff} \quad m \mid (a-b).$$

Example 4.10. We have $23 \equiv_7 44$ and $54321 \equiv_{10} 1$. Note that $a \equiv_2 b$ means that a and b are either both even or both odd.

Example 4.11. If $a \equiv_2 b$ and $a \equiv_3 b$, then $a \equiv_6 b$. The general principle underlying this example will be discussed later.

The above examples 4.8 and 4.9 make use of the fact that if an equality holds over the integers, then it must also hold modulo 2 or, more generally, modulo any modulus m. In other words, for any a and b,

$$a = b \implies a \equiv_m b \tag{4.1}$$

for all m, i.e., the relation $\equiv_m$ is reflexive ($a \equiv_m a$ for all a). It is easy to verify that this relation is also symmetric and transitive, which was already stated in Chapter 3:

Lemma 4.13. *For any $m \geq 1$, $\equiv_m$ is an equivalence relation on $\mathbb{Z}$.*

The implication (4.1) can be turned around and can be used to prove the inequality of two numbers a and b:

$$a \not\equiv_m b \implies a \neq b.$$

The following lemma shows that modular congruences are compatible with the arithmetic operations on $\mathbb{Z}$.

Lemma 4.14. *If $a \equiv_m b$ and $c \equiv_m d$, then*

$$a + c \equiv_m b + d \qquad \text{and} \qquad ac \equiv_m bd.$$

Proof. We only prove the first statement and leave the other proof as an exercise. We have $m \mid (a-b)$ and $m \mid (c-d)$. Hence m also divides $(a-b)+(c-d) = (a+c)-(b+d)$, which is the definition of $a + c \equiv_m b + d$. □

Corollary 4.15. *Let $f(x_1, \ldots, x_k)$ be a multi-variate polynomial in k variables with integer coefficients, and let $m \geq 1$. If $a_i \equiv_m b_i$ for $1 \leq i \leq k$, then*

$$f(a_1, \ldots, a_k) \equiv_m f(b_1, \ldots, b_k).$$

Proof. Evaluating a polynomial can be achieved by a sequence of additions and multiplications. In each such step the congruence modulo m is maintained, according to Lemma 4.14. □

4.5.2 Modular Arithmetic

There are m equivalence classes of the equivalence relation $\equiv_m$, namely $[0], [1], \ldots, [m-1]$. Each equivalence class $[a]$ has a natural representative $R_m(a) \in [a]$ in the set

$$\mathbb{Z}_m := \{0, \ldots, m-1\}$$

of remainders modulo m.[17]

In the following we are often interested only in the remainder of an integer (e.g. the result of a computation) modulo some modulus m. Addition and multiplication modulo m can be considered as operations on the set $\mathbb{Z}_m$. We will be interested in this structure in Chapter 5 where we will see that it is an important example of a so-called ring.

[17] Recall that $R_m(a)$ denotes the remainder when a is divided by m.

Example 4.12. Is $n = 84877 \cdot 79683 - 28674 \cdot 43879$ even or odd? The answer is trivial and does not require the computation of n. The product of two odd numbers is odd, the product of an even and an odd numbers is even, and the difference of an odd and an even number is odd. Thus n is odd.

The following lemma establishes the simple connection between congruence modulo m and remainders modulo m. The proof is easy and left as an exercise.

Lemma 4.16. *For any $a, b, m \in \mathbb{Z}$ with $m \geq 1$,*

(i) $a \equiv_m R_m(a)$.

(ii) $a \equiv_m b \iff R_m(a) = R_m(b)$.

The above lemma together with Lemma 4.14 implies that if in a computation involving addition and multiplication one is interested only in the remainder of the result modulo m, then one can compute remainders modulo m at any intermediate step (thus keeping the numbers small), without changing the result. This is referred to as *modular arithmetic*.

Corollary 4.17. *Let $f(x_1, \ldots, x_k)$ be a multi-variate polynomial in k variables with integer coefficients, and let $m \geq 1$. Then*

$$R_m\big(f(a_1, \ldots, a_k)\big) = R_m\big(f(R_m(a_1), \ldots, R_m(a_k))\big).$$

Proof. By Lemma 4.16 (i) we have $a_i \equiv_m R_m(a_i)$ for all i. Therefore, using Corollary 4.15 we have $f(a_1, \ldots, a_k) \equiv_m f(R_m(a_1), \ldots, R_m(a_k))$. Thus, using Lemma 4.16 (ii) we obtain the statement to be proved. □

Example 4.13. Compute 7^{100} modulo 24. We make use of the fact that $7^2 = 49 \equiv_{24} 1$. Thus $R_{24}(7^{100}) = R_{24}((7^2)^{50}) = R_{24}(R_{24}(7^2)^{50}) = R_{24}(1^{50}) = R_{24}(1) = 1$.

Example 4.14. Remainders can be used to check the correctness of calculations (which were, for instance, carried out by hand). If an error occurred during the computation, it is likely that this error also occurs when the computation is considered modulo some m. To check the result n of a computation one can compare $R_m(n)$ with the remainder modulo m obtained by continuously reducing intermediate results of the computation. The modulus $m = 9$ is especially suited because $R_9(n)$ can be easily computed by adding the decimal digits of n (prove this!), and computing the remainder modulo 9 of this sum. For instance, to check whether $247 \cdot 3158 = 780026$ is correct one can compute $R_9(247) = R_9(2+4+7) = 4$ and $R_9(3158) = R_9(3+1+5+8) = 8$ to obtain $R_9(247 \cdot 3158) = R_9(4 \cdot 8) = 5$. On the other hand we have $R_9(780026) = R_9(7+8+2+6) = 5$. Hence the result can be correct.

Example 4.15. A similar test can be performed for $m = 11$. $R_{11}(n)$ can be computed by adding the decimal digits of n with alternating sign modulo 11. This test, unlike that for $m = 9$, detects the swapping of digits.

Example 4.16. The larger m, the more likely it is that a calculation error is detected. How could one implement a similar test for $m = 99$, how for $m = 101$?

4.5.3 Multiplicative Inverses

Consider the problem of finding the solutions x for the congruence equation

$$ax \equiv_m b.$$

Obviously, if x is a solution, then so is $x + km$ for any $k \in \mathbb{Z}$. Hence we can restrict the consideration to solutions in $\mathbb{Z}_m$. Of special interest is the case where $\gcd(a, m) = 1$ and $b = 1$.

Lemma 4.18. *The congruence equation*

$$ax \equiv_m 1$$

has a solution $x \in \mathbb{Z}_m$ if and only if $\gcd(a, m) = 1$. The solution is unique.

Proof. ($\Longrightarrow$) If x satisfies $ax \equiv_m 1$, then $ax = km + 1$ for some k. Note that $\gcd(a, m)$ divides both a and m, hence also $ax - km$, which is 1. Thus $\gcd(a, m) = 1$. Therefore, if $\gcd(a, m) > 1$, then no solution x exists.

($\Longleftarrow$) Assume now that $\gcd(a, m) = 1$. According to Corollary 4.5 there exist integers u and v such that $ua + vm = \gcd(a, m) = 1$. Since $vm \equiv_m 0$ we have $ua \equiv_m 1$. Hence $x = u$ is a solution in $\mathbb{Z}$, and thus $x = R_m(u)$ is a solution in $\mathbb{Z}_m$.

To prove uniqueness of x in $\mathbb{Z}_m$, suppose there is another solution $x' \in \mathbb{Z}_m$. Then $ax - ax' \equiv_m 0$, thus $a(x - x') \equiv_m 0$ and hence m divides $a(x - x')$. Since $\gcd(a, m) = 1$, m must divide $(x - x')$.[18] Therefore $R_m(x) = R_m(x')$ and hence $R_m(x)$ is the unique solution in $\mathbb{Z}_m$. □

Definition 4.9. If $\gcd(a, m) = 1$, the unique solution $x \in \mathbb{Z}_m$ to the congruence equation $ax \equiv_m 1$ is called the *multiplicative inverse of a modulo m*. One also uses the notation $x \equiv_m a^{-1}$ or $x \equiv_m 1/a$.

Example 4.17. The multiplicative inverse of 5 modulo 13 is 8 since $5 \cdot 8 = 40 \equiv_{13} 1$.

[18] If k divides mn and $\gcd(k, n) = 1$, then k divides m. (Prove this!)

The multiplicative inverse of a modulo m can efficiently be computed using the so-called extended Euclidean algorithm. Note that if $\gcd(a, m) \neq 1$, then a has no multiplicative inverse modulo m.

4.5.4 The Chinese Remainder Theorem

We now consider a system of congruences for an integer x.

Example 4.18. Find an integer x for which $x \equiv_3 1$, $x \equiv_4 2$, and $x \equiv_5 4$. A solution is $x = 34$ as one can easily verify. This is the only solution in $\mathbb{Z}_{60}$, but by adding multiples of 60 to x one obtains further solutions.

The following theorem, known as the *Chinese Remainder Theorem (CRT)*, states this for general systems of congruences. The proof of the theorem is constructive: it shows how a solution x can be constructed efficiently.

Theorem 4.19. *Let $m_1, m_2, \ldots, m_r$ be pairwise relatively prime integers and let $M = \prod_{i=1}^r m_i$. For every list $a_1, \ldots, a_r$ with $0 \leq a_i < m_i$ for $1 \leq i \leq r$, the system of congruence equations*

$$
\begin{aligned}
x &\equiv_{m_1} a_1 \\
x &\equiv_{m_2} a_2 \\
&\cdots \\
x &\equiv_{m_r} a_r
\end{aligned}
$$

for x has a unique solution x satisfying $0 \leq x < M$.

Proof. Let $M_i = M/m_i$. Hence $\gcd(M_i, m_i) = 1$ because every factor m_k (where $k \neq i$) of M_i is relatively prime to m_i, and thus so is M_i. Thus there exists an N_i satisfying

$$M_i N_i \equiv_{m_i} 1.$$

Note that for all $k \neq i$ we have $M_i \equiv_{m_k} 0$ and thus

$$M_i N_i \equiv_{m_k} 0.$$

Therefore

$$\sum_{i=1}^{r} a_i M_i N_i \ \equiv_{m_k} \ a_k$$

for all k. Hence the integer x defined by

$$x = R_M\left(\sum_{i=1}^{r} a_i M_i N_i\right)$$

satisfies all the congruences. In order to prove uniqueness, observe that for two solutions x' and x'', $x' - x'' \equiv_{m_i} 0$ for all i, i.e., $x' - x''$ is a multiple of all the m_i and hence of $\text{lcm}(m_1, \ldots, m_r) = M$. Thus $x' \equiv_M x''$. □

The Chinese Remainder Theorem has several applications. When one is interested in a computation modulo M, then the moduli m_i can be viewed as a coordinate system. One can project the numbers of interest modulo the m_i, and perform the computation in the r projections (which may be more efficient than computing directly modulo M). If needed at the end, one can reconstruct the result from the projections.

Example 4.19. Compute $R_{35}(2^{1000})$. We can do this computation modulo 5 and modulo 7 separately. Since $2^4 \equiv_5 1$ we have $2^{1000} \equiv_5 1$. Since $2^3 \equiv_7 1$ we have $2^{1000} \equiv_7 2$. This yields $2^{1000} \equiv_{35} 16$ since 16 is the (unique) integer $x \in [0, 34]$ with $x \equiv_5 1$ and $x \equiv_7 2$.

4.6 Application: Diffie-Hellman Key-Agreement

Until the 1970's, number theory was considered one of the purest of all mathematical disciplines in the sense of being furthest away from any useful applications. However, this has changed dramatically in the 1970's when crucial applications of number theory in cryptography were discovered.

In a seminal 1976 paper[19], Diffie and Hellman proposed the revolutionary concept of *public-key cryptography*. Most security protocols, and essentially all those used on the Internet, are based on public-key cryptography. Without this amazing and paradoxical invention, security on the Internet would be unthinkable.

Consider the *key distribution* problem. In order to encrypt the communication between two parties, say Alice and Bob, they need a secret key known only to them. How can they obtain such a key in a setting, like the Internet, where they initially share no secret information and where they are connected only by an insecure communication channel to which a potential adversary has access? We describe the famous Diffie-Hellman protocol which allows to solve this seemingly paradoxical problem.

The Diffie-Hellman protocol (see Figure 4.2), as originally proposed[20], makes use of exponentiation modulo a large prime p, for instance with 2048 bits. While $y = R_p(g^x)$ can be computed efficiently (how?), even if p, g and x are numbers of several hundred or thousands of digits, computing x when given p, g and y is generally (believed to be) computationally infeasible. This problem is known

[19] W. Diffie and M.E. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654, 1976.

[20] Since then, other versions, for instance based on elliptic curves, have been proposed.

as (a version of) the *discrete logarithm problem*. The security of the Diffie-Hellman protocol is based on this asymmetry in computational difficulty. Such a function, like $x \mapsto R_p(g^x)$, is called a one-way function: it is easy to compute in one direction but computationally very hard to invert.[21]

The prime p and the basis g (e.g. $g = 2$) are public parameters, possibly generated once and for all for all users of the system. The protocol is symmetric, i.e., Alice and Bob perform the same operations. The exchange of the so-called *public keys* y_A and y_B must be authenticated, but not secret.[22] It is easy to see that Alice and Bob end up with the same value $k_{AB} = k_{BA}$ which they can use as a secret key for encrypting subsequent communication.[23] In order to compute k_{AB} from y_A and y_B, an adversary would have to compute either x_A or x_B, which is believed to be infeasible.

Alice	**insecure channel**	**Bob**
select x_A at random from $\{0,\ldots,p-2\}$		select x_B at random from $\{0,\ldots,p-2\}$
$y_A := R_p(g^{x_A})$		$y_B := R_p(g^{x_B})$
	$\xrightarrow{y_A}$	
	$\xleftarrow{y_B}$	
$k_{AB} := R_p(y_B^{x_A})$		$k_{BA} := R_p(y_A^{x_B})$

$$k_{AB} \equiv_p y_B^{x_A} \equiv_p (g^{x_B})^{x_A} \equiv_p g^{x_A x_B} \equiv_p k_{BA}$$

Figure 4.1: The Diffie-Hellman key agreement protocol.

A mechanical analogue of a one-way function is a padlock without a key.[24] The mechanical analog of the Diffie-Hellman protocol is shown in Figure 4.2. Alice and Bob can exchange their locks (closed) and keep a copy in the open state. Then they can both generate the same configuration, namely the two locks

[21] It is not known whether one-way functions actually exist, but it is conjectured that exponentiation modulo a prime p is a one-way function for most p.

[22] This can be achieved by recognizing the other party's voice in a phone call or, indirectly, by the use of public-key certificates.

[23] More precisely, they can derive a common secret key, for example by applying a hash function to K_{AB}.

[24] A padlock with a key corresponds to a so-called *trapdoor one-way function* which is not considered here.

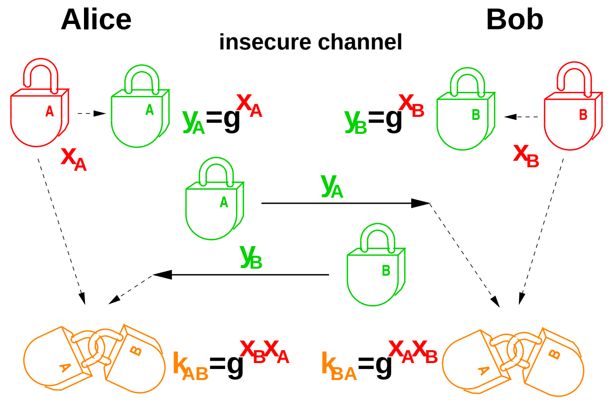

Figure 4.2: Mechanical analog of the Diffie-Hellman protocol.

interlocked. For the adversary, this is impossible without breaking open one of the locks.

Another famous (and more widely used) public-key cryptosystem, the so-called RSA-system invented in 1977 and named after Rivest, Shamir and Adleman[25], will be discussed later. Its security is based on the (conjectured) computational difficulty of factoring large integers.

[25] They received the Turing award in 2003.

Chapter 5

Algebra

5.1 Introduction

5.1.1 What Algebra is About

In a nutshell, algebra is the mathematical study of structures consisting of a set and certain operations on the set. Examples are the integers $\mathbb{Z}$, the rational numbers $\mathbb{Q}$, and the set of polynomials with coefficients from some domain, with the respective addition and multiplication operations. A main goal in algebra is to understand the properties of such algebraic systems at the highest level of generality and abstraction. For us, an equally important goal is to understand the algebraic systems that have applications in Computer Science.

For instance, one is interested in investigating which properties of the integers are responsible for the unique factorization theorem. What do the integers, the polynomials with rational or real coefficients, and several other structures have in common so that the unique factorization theorem holds? Why does it not hold for certain other structures?

The benefit of identifying the highest level of generality and abstraction is that things often become simpler when unnecessary details are eliminated from consideration, and that a proof must be carried out only once and applies to all structures captured at the given level of generality.

5.1.2 Algebraic Structures

Definition 5.1. An *operation* on a set S is a function[1] $S^n \to S$, where $n \geq 0$ is called the *"arity"*[2] of the operation.

[1]In some cases, the function is only partial.
[2]German: Stelligkeit

Operations with arity 1 and 2 are called unary and binary operations, respectively. An operation with arity 0 is called a constant; it is a fixed element from the set S, for instance the special element 1 in $\mathbb{Z}$. In many cases, only binary operations are actually listed explicitly,

Definition 5.2. An *algebra* (or *algebraic structure* or Ω*-algebra*) is a pair $\langle S; \Omega \rangle$ where S is a set (the *carrier*[3] of the algebra) and $\Omega = (\omega_1, \ldots, \omega_n)$ is a list of operations on S.[4]

5.1.3 Some Examples of Algebras

We give a few examples of algebras, some of which we will discuss in more detail later.

Example 5.1. $\langle \mathbb{Z}; +, -, 0, \cdot, 1 \rangle$ denotes the integers with the two binary operations $+$ and $\cdot$, the unary operation $-$ (taking the negative) and the two constants 0 and 1, the neutral elements of addition and multiplication.

We sometimes omit some details and write simply $\langle \mathbb{Z}; +, \cdot \rangle$ instead of $\langle \mathbb{Z}; +, -, 0, \cdot, 1 \rangle$ when the negation operation and the special elements 0 and 1 are understood. More generally, we sometimes drop unary and nullary operations. This notation is actually more common in the literature (but less precise). This is a purely notational issue.

Example 5.2. $\langle \mathbb{Z}_m; \oplus \rangle$ (and $\langle \mathbb{Z}_m; \odot \rangle$) denote the integers modulo m with addition modulo m (and multiplication modulo m) as the only binary operation.

Example 5.3. $\langle \mathcal{P}(A); \cup, \cap, \overline{} \rangle$ is the power set of a set A with union, intersection, and complement operations.

5.2 Monoids and Groups

In this section we look at algebras $\langle S; * \rangle$ with one binary operation and possibly one unary and one nullary operation. The binary operation can be denoted arbitrarily, for instance by $*$. It is often denoted $+$, in which case it is called *addition*, or $\cdot$, in which case it is called *multiplication*. But it is important to note that the name of the operation is not of mathematical relevance.

We discuss three special properties that $\langle S; * \rangle$ can have, (1) neutral elements, (2) associativity, and (3) inverse elements, as well as combinations of these.

[3]German: Trägermenge
[4]This definition, though very general, does not capture all algebraic systems one might be interested in. A more general type of algebraic system, called *heterogeneous* algebraic systems, can have several carrier sets.

5.2.1 Neutral Elements

Definition 5.3. A *left [right] neutral element* (or *identity element*) of an algebra $\langle S; *\rangle$ is an element $e \in S$ such that $e * a = a$ $[a * e = a]$ for all $a \in S$. If $e * a = a * e = a$ for all $a \in S$, then e is simply called *neutral element*.

If the operation is called addition, then e is usually denoted as 0, and if it is called multiplication, then e is usually denoted as 1.

Lemma 5.1. *If $\langle S; *\rangle$ has both a left and a right neutral element, then they are equal. In particular $\langle S; *\rangle$ can have at most one neutral element.*

Proof. Suppose that e and e' are left and right neutral elements, respectively. Then, by definition, $e * e' = e'$ (considering e as a left neutral element), but also $e * e' = e$ (considering e' as a right neutral element). Thus $e' = e$. □

Example 5.4. The empty sequence ϵ is the neutral element of $\langle \Sigma^*; |\rangle$, where Σ^* is the set of sequences over the alphabet Σ and $|$ denotes concatenation of sequences.

5.2.2 Associativity and Monoids

The operation in the previous example, sequence concatenation, has a very useful property: When all sequences are written one after the other, with spaces between sequences, it does not matter in which order one performs the concatenation operations. In short, sequence concatenation is associative.

Definition 5.4. A binary operation $*$ on a set S is *associative* if $a*(b*c) = (a*b)*c$ for all $a, b, c \in S$.

Not all operations are associative:

Example 5.5. An example of a non-associative operation on the integers is exponentiation: $(a^b)^c \neq a^{(b^c)}$ in general.

Associativity is a very special property of an operation, but it is of crucial importance in algebra. Associativity of $*$ means that the element $a_1 * a_2 * \cdots * a_n$ (for any $a_1, \ldots, a_n \in S$) is uniquely defined and independent of the order in which elements are combined. For example,

$$(((a * b) * c) * d) = ((a * b) * (c * d)) = (a * ((b * c) * d)).$$

This justifies the use of the notation $\sum_{i=1}^n a_i$ if the operation $*$ is called addition, and $\prod_{i=1}^n a_i$ if the operation $*$ is called multiplication.

Note that up to now, and also in the next section, we do not yet pay attention to the fact that some operations are commutative. In a sense, commutativity is less important than associativity.

Some standard examples of associate operations are addition and multiplication in various structures: $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{Z}_m$.

Definition 5.5. A *monoid* is an algebra $\langle M; *, e\rangle$ where $*$ is associative and e is the neutral element.

Some standard examples of monoids are $\langle \mathbb{Z}; +, 0\rangle$, $\langle \mathbb{Z}; \cdot, 1\rangle$, $\langle \mathbb{Q}; +, 0\rangle$, $\langle \mathbb{Q}; \cdot, 1\rangle$, $\langle \mathbb{R}; +, 0\rangle$, $\langle \mathbb{R}; \cdot, 1\rangle$, $\langle \mathbb{Z}_m; \oplus, 0\rangle$, and $\langle \mathbb{Z}_m; \odot, 1\rangle$.

Example 5.6. $\langle \Sigma^*; |, \epsilon\rangle$ is a monoid since, as mentioned above, concatenation of sequences is associative.

Example 5.7. For a set A, the set A^A of functions $A \to A$ form a monoid with respect to function composition. The identity function id (defined by $\mathsf{id}(a) = a$ for all $a \in A$) is the neutral element. According to Lemma 3.7, relation composition, and therefore also function composition, is associative. The algebra $\langle A^A; \circ, \mathsf{id}\rangle$ is thus a monoid.

5.2.3 Inverses and Groups

Definition 5.6. A *left [right] inverse element*[5] of an element a in an algebra $\langle S; *, e\rangle$ with neutral element e is an element $b \in S$ such that $b * a = e$ $[a * b = e]$. If $b * a = a * b = e$, then b is simply called an *inverse* of a.

To prove the uniqueness of the inverse (if it exists), we need $*$ to be associative:

Lemma 5.2. *In a monoid $\langle M; *, e\rangle$, if $a \in M$ has a left and a right inverse, then they are equal. In particular, a has at most one inverse.*

Proof. Let b and c be left and right inverses of a, respectively, i.e., we have $b * a = e$ and $a * c = e$, Then

$$b = b * e = b * (a * c) = (b * a) * c = e * c = c,$$

where we have omitted the justifications for the steps. □

Example 5.8. Consider again $\langle A^A; \circ, \mathsf{id}\rangle$. A function $f \in A^A$ has a left inverse only if it is injective, and it has a right inverse only if it is surjective. Hence f has an inverse f^{-1} if and only if f is bijective. In this case, $f \circ f^{-1} = f^{-1} \circ f = \mathsf{id}$.

[5] or simply *left [right] inverse*.

Now follows one of the most fundamental definitions of algebra.

Definition 5.7. A *group* is an algebra $\langle G; *, \widehat{\ }, e\rangle$ satisfying the following axioms:

G1 $*$ is associative.

G2 e is a neutral element: $a * e = e * a = a$ for all $a \in G$.

G3 Every $a \in G$ has an inverse element $\widehat{a}$, i.e., $a * \widehat{a} = \widehat{a} * a = e$.

We can write $\langle G; *\rangle$ (or simply G if $*$ is understood) instead of $\langle G; *, \widehat{\ }, e\rangle$. If the operation $*$ is called addition $(+)$ [multiplication $(\cdot)$], then the inverse of a is denoted $-a$ [a^{-1} or $1/a$] and the neutral element is denoted 0 [1].

Some standard examples of groups are $\langle \mathbb{Z}; +, -, 0\rangle$, $\langle \mathbb{Q}; +, -, 0\rangle$, $\langle \mathbb{Q} \setminus \{0\}; \cdot, ^{-1}, 1\rangle$, $\langle \mathbb{R}; +, -, 0\rangle$, $\langle \mathbb{R} \setminus \{0\}; \cdot, ^{-1}, 1\rangle$, and $\langle \mathbb{Z}_m; \oplus, \ominus, 0\rangle$.

Definition 5.8. A group $\langle G; *\rangle$ (or monoid) is called *commutative* or *abelian*[6] if $a * b = b * a$ for all $a, b \in G$.

We summarize a few facts we encountered already earlier for the special case of the integers $\mathbb{Z}$. The group is the right level of abstraction for describing these facts. The proofs are left as exercises.

Lemma 5.3. *For a group $\langle G; *, \widehat{\ }, e\rangle$, we have for all $a, b, c \in G$:*

(i) $\widehat{(\widehat{a})} = a$.

(ii) $\widehat{a * b} = \widehat{b} * \widehat{a}$.

(iii) *Left cancellation law:* $\quad a * b = a * c \implies b = c$.

(iv) *Right cancellation law:* $\quad b * a = c * a \implies b = c$.

(v) *The equation* $a * x = b$ *has a unique solution x for any a and b.*
So does the equation $x * a = b$.

5.2.4 (Non-)minimality of the Group Axioms

In mathematics, one generally wants the axioms of a theory to be minimal. One can show that the group axioms as stated are not minimal. One can simplify axiom **G2** by only requesting that $a * e = a$; call this new axiom **G2'**. The equation $e * a = a$ is then implied (by all axioms). The proof of this fact is left as an exercise. Similarly, one can simplify **G3** by only requesting that $a * \widehat{a} = e$; call this new axiom **G3'**. The equation $\widehat{a} * a = e$ is then implied. The proof for this is as follows:

$$
\begin{array}{rcll}
\widehat{a} * a & = & (\widehat{a} * a) * e & \textbf{(G2')} \\
 & = & (\widehat{a} * a) * (\widehat{a} * \widehat{\widehat{a}}) & \textbf{(G3')}\text{, i.e., def. of right inverse of } \widehat{a}) \\
 & = & \widehat{a} * (a * (\widehat{a} * \widehat{\widehat{a}})) & \textbf{(G1)} \\
 & = & \widehat{a} * ((a * \widehat{a}) * \widehat{\widehat{a}}) & \textbf{(G1)} \\
 & = & \widehat{a} * (e * \widehat{\widehat{a}}) & \textbf{(G3)} \\
 & = & (\widehat{a} * e) * \widehat{\widehat{a}} & \textbf{(G1)} \\
 & = & \widehat{a} * \widehat{\widehat{a}} & \textbf{(G2')} \\
 & = & e & \textbf{(G3')}\text{, i.e., def. of right inverse of } \widehat{a})
\end{array}
$$

[6] Named after the Norwegian mathematician Niels Henrik Abel.

5.2.5 Some Examples of Groups

Example 5.9. The set of invertible (non-singular) $n \times n$ matrices over the real numbers with matrix multiplication form a group, with the identity matrix as the neutral element. This group is not commutative for $n \geq 2$.

Example 5.10. Recall that the sequences with concatenation and the empty sequence as the neutral element form a non-commutative monoid. This is not a group because inverses cannot be defined (except for the empty sequence).

Example 5.11. For a given structure R supporting addition and multiplication (to be called a *ring* later), let $R[x]$ denote the set of polynomials with coefficients in R. $\langle \mathbb{Z}[x]; +\rangle$, $\langle \mathbb{Q}[x]; +\rangle$, and $\langle \mathbb{R}[x]; +\rangle$ are abelian groups, where $+$ denotes polynomial addition. $\langle \mathbb{Z}[x]; \cdot\rangle$, $\langle \mathbb{Q}[x]; \cdot\rangle$, and $\langle \mathbb{R}[x]; \cdot\rangle$ are commutative monoids, where $\cdot$ denotes polynomial multiplication. The neutral element is the polynomial 1. Like $\langle \mathbb{Z}; \cdot\rangle$, $\langle R[x]; \cdot\rangle$ is not a group, for any R.

Example 5.12. Let S_n be the set of $n!$ permutations of n elements, i.e., the set of bijections $\{1, \ldots, n\} \to \{1, \ldots, n\}$. A bijection f has an inverse f^{-1}. S_n is a subset of the set of functions $\{1, \ldots, n\} \to \{1, \ldots, n\}$. It follows from the associativity of function composition that the composition of permutations is also associative. The group $\langle S_n; \circ, ^{-1}, id\rangle$ is called the *symmetric group* on n elements. S_n is non-abelian for $n \geq 3$.

Another important source of (usually non-abelian) groups are symmetries and rotations of a geometric figure, mapping the figure to itself (but permuting the vertices). The neutral element is the identity mapping and the inverse element is, for an axial or point symmetry, the element itself. For a rotation, the inverse element is the inverse rotation. To form a group, the closure under composition must be considered. For example, the composition of two axial symmetries corresponds to a rotation by twice the angle between the axes. Such a group is a subset (a subgroup) of the set of permutations of the vertices.

Example 5.13. Consider a square in the plane, with nodes labeled A, B, C, D. Now consider operations which map the square to itself, but with the vertices permuted. Consider the four reflections with respect to one of the two middle parallels or one of the two diagonals. The closure under composition of these four elements also includes the four rotations by $0°$ (the neutral element), by $90°$, $180°$, and by $270°$. These 8 elements (reflections and rotations) form a group, which we denote by $S_{\square}$. It is called the symmetry group of the square. If the vertices of the square are labeled A, B, C, D, then these eight geometric operations each corresponds to a permutation of the set $\{A, B, C, D\}$. For example, the rotation by $90°$ corresponds to the permutation $(A, B, C, D) \to (B, C, D, A)$. Note that the set of four rotations also form a group, actually a subgroup of the above described group and also a subgroup of the group of permutations on $\{A, B, C, D\}$.[7]

Example 5.14. It is left as an exercise to figure out the symmetry group of the three-dimensional cube.

5.3 The Structure of Groups

5.3.1 Direct Products of Groups

Definition 5.9. The *direct product* of n groups $\langle G_1; *_1\rangle, \ldots, \langle G_n; *_n\rangle$ is the algebra

$$\langle G_1 \times \cdots \times G_n; \star\rangle,$$

where the operation $\star$ is component-wise:

$$(a_1, \ldots, a_n) \star (b_1, \ldots, b_n) = (a_1 *_1 b_1, \ldots, a_n *_n b_n).$$

Lemma 5.4. *$\langle G_1 \times \cdots \times G_n; \star\rangle$ is a group, where the neutral element and the inversion operation are component-wise in the respective groups.*

Proof. Left as an exercise. □

Example 5.15. Consider the group $\langle \mathbb{Z}_5; \oplus\rangle \times \langle \mathbb{Z}_7; \oplus\rangle$. The carrier of the group is $\mathbb{Z}_5 \times \mathbb{Z}_7$. The neutral element is $(0, 0)$. If we denote the group operation by $\star$, then we have $(2, 6) \star (4, 3) = (1, 2)$. Also, $\widehat{(2, 6)} = (3, 1)$. It follows from the Chinese remainder theorem that $\langle \mathbb{Z}_5; \oplus\rangle \times \langle \mathbb{Z}_7; \oplus\rangle$ is isomorphic to $\langle \mathbb{Z}_{35}; \oplus\rangle$, a concept introduced in the following subsection.

[7] We point out that one can consider the described operations also as bijections of the real plane, i.e., as functions $\mathbb{R}^2 \to \mathbb{R}^2$.

5.3.2 Group Homomorphisms

Homomorphisms are a central concept in mathematics and also in Computer Science. A homomorphism is a structure-preserving function from an algebraic structure into another algebraic structure. Here we only introduce homomorphisms of groups.

Definition 5.10. For two groups $\langle G; *, \widehat{\ }, e\rangle$ and $\langle H; \star, \widetilde{\ }, e'\rangle$, a function $\psi : G \to H$ is called a *group homomorphism* if, for all a and b,

$$\psi(a * b) \;=\; \psi(a) \star \psi(b).$$

If ψ is a bijection from G to H, then it is called an *isomorphism*, and we say that G and H are isomorphic and write $G \simeq H$.

We use the symbol $\widetilde{\ }$ for the inverse operation in the group H. The proof of the following lemma is left as an exercise:

Lemma 5.5. *A group homomorphism ψ from $\langle G; *, \widehat{\ }, e\rangle$ to $\langle H; \star, \widetilde{\ }, e'\rangle$ satisfies*

(i) $\psi(e) = e'$,

(ii) $\psi(\widehat{a}) = \widetilde{\psi(a)}$ *for all* a.

The concept of an isomorphism is more general than for algebraic systems in the strict sense, and it applies to more general algebraic structures, for instance also to relations and hence to graphs.

Example 5.16. The group $\langle \mathbb{Z}_6; \oplus\rangle \times \langle \mathbb{Z}_{10}; \oplus\rangle$ is isomorphic to $\langle \mathbb{Z}_2; \oplus\rangle \times \langle \mathbb{Z}_{30}; \oplus\rangle$. The isomorphism $\psi : \mathbb{Z}_6 \times \mathbb{Z}_{10} \to \mathbb{Z}_2 \times \mathbb{Z}_{30}$ is easily checked to be given by $\psi((a, b)) = (a', b')$ where $a' \equiv_2 a$ (i.e., $a' = R_2(a)$) and b' is given by $b' \equiv_3 a$ and $b' \equiv_{10} b$ (where the Chinese remainder theorem can be applied).

Example 5.17. The logarithm function is a group homomorphism from $\langle \mathbb{R}^{>0}, \cdot\rangle$ to $\langle \mathbb{R}, +\rangle$ since $\log(a \cdot b) = \log a + \log b$.

We give two familiar examples of relevant homomorphisms that are not isomorphisms.

Example 5.18. If one considers the three-dimensional space $\mathbb{R}^3$ with vector addition, then any projection on a plane through the origin or a line through the origin are homomorphic images of $\mathbb{R}^3$. A special case is the projection onto an axis of the coordinate system, which abstracts away all but one coordinate.

Example 5.19. Consider the set of real-valued $n \times n$ matrices. The determinant is a homomorphism (with respect to multiplication) from the set of matrices to $\mathbb{R}$. We have $\det(AB) = \det(A)\det(B)$.

5.3.3 Subgroups

For a given algebra, for example a group or a ring (see Section 5.5), a subalgebra is a subset that is by itself an algebra of the same type, i.e., a subalgebra is a subset of an algebra closed under all operations. For groups we have specifically:

Definition 5.11. A subset $H \subseteq G$ of a group $\langle G; *, \widehat{\ }, e\rangle$ is called a *subgroup* of G if $\langle H; *, \widehat{\ }, e\rangle$ is a group, i.e., if H is closed with respect to all operations:

(1) $a * b \in H$ for all $a, b \in H$,

(2) $e \in H$, and

(3) $\widehat{a} \in H$ for all $a \in H$.

Example 5.20. For any group $\langle G; *, \widehat{\ }, e\rangle$, there exist two trivial subgroups: the subset $\{e\}$ and G itself.

Example 5.21. Consider the group $\mathbb{Z}_{12}$ (more precisely $\langle \mathbb{Z}_{12}; \oplus, \ominus, 0\rangle$). The following subsets are all the subgroups: $\{0\}$, $\{0, 6\}$, $\{0, 4, 8\}$, $\{0, 3, 6, 9\}$, $\{0, 2, 4, 6, 8, 10\}$, and $\mathbb{Z}_{12}$.

Example 5.22. The set of symmetries and rotations discussed in example 5.13, denoted $S_\square$, constitutes a subgroup (with 8 elements) of the set of 24 permutations on 4 elements.

5.3.4 The Order of Group Elements and of a Group

In the remainder of Section 5.3 we will use a *multiplicative notation* for groups, i.e., we denote the group operation as "$\cdot$" (which can also be omitted) and use the corresponding multiplicative notation. But it is important to point out that this is only a notational convention that entails no loss of generality of the kind of group operation. In many cases (but not always) we denote the neutral element of a multiplicatively written group as 1. The inverse of a is denoted a^{-1} or $1/a$, and a/b stands for ab^{-1}. Furthermore, we use the common notation for powers of elements: For $n \in \mathbb{Z}$, a^n is defined recursively:

- $a^0 = e$,
- $a^n = a \cdot a^{n-1}$ for $n \geq 1$, and
- $a^n = (a^{-1})^{|n|}$ for $n \leq -1$.

It is easy to see that for all $m, n \in \mathbb{Z}$

$$a^m \cdot a^n = a^{m+n} \qquad \text{and} \qquad (a^m)^n = a^{mn}.$$

Definition 5.12. Let G be a group and let a be an element of G. The *order*[8] of a, denoted $\mathrm{ord}(a)$, is the least $m \geq 1$ such that $a^m = e$, if such an m exists, and $\mathrm{ord}(a)$ is said to be infinite otherwise, written $\mathrm{ord}(a) = \infty$.

By definition, $\mathrm{ord}(e) = 1$. If $\mathrm{ord}(a) = 2$ for some a, then $a^{-1} = a$; such an a is called self-inverse.

Example 5.23. The order of 6 in $\langle \mathbb{Z}_{20}; \oplus, \ominus, 0\rangle$ is 10. This can be seen easily since $60 = 10 \cdot 6$ is the least common multiple of 6 and 20. The order of 10 is 2, and we note that 10 is self-inverse.

Example 5.24. The order of any axial symmetry in the group $S_\square$ (see example 5.13) is 2, while the order of the $90°$-rotation (and also of the $270°$-rotation) is 4.

Example 5.25. The order of any integer $a \neq 0$ in $\langle \mathbb{Z}; +\rangle$ is ∞.

Example 5.26. Consider the group S_5 of permutations on the set $\{1, 2, 3, 4, 5\}$. What is the order of the permutations described by $(1, 2, 3, 4, 5) \to (3, 1, 2, 4, 5)$, by $(1, 2, 3, 4, 5) \to (1, 2, 3, 5, 4)$, and by $(1, 2, 3, 4, 5) \to (2, 3, 1, 5, 4)$?

The following lemma implies that the sequence of powers of an element of a finite group is periodic. It does not hold in every monoid (why?).

Lemma 5.6. *In a finite group G, every element has a finite order.*

Proof. Since G is finite, we must have $a^r = a^s = b$ for some r and s with $r < s$ (and some b). Then $a^{s-r} = a^s \cdot a^{-r} = b \cdot b^{-1} = e$. □

Definition 5.13. For a finite group G, $|G|$ is called the *order* of G.[9]

5.3.5 Cyclic Groups

If G is a group and $a \in G$ has finite order, then for any $m \in \mathbb{Z}$ we have

$$a^m = a^{R_{\mathrm{ord}(a)}(m)}.$$

Definition 5.14. For a group G and $a \in G$, the *group generated by a*, denoted $\langle a\rangle$, is defined as

$$\langle a\rangle \stackrel{\text{def}}{=} \{a^n \mid n \in \mathbb{Z}\}.$$

[8] German: Ordnung

[9] Note that the term "order" has two different (but related) meanings.

It is easy to see that $\langle a\rangle$ is a group, actually the smallest subgroup of a group G containing the element $a \in G$. For finite groups we have

$$\langle a\rangle \stackrel{\text{def}}{=} \{e, a, a^2, \ldots, a^{\text{ord}(a)-1}\}.$$

Definition 5.15. A group $G = \langle g\rangle$ generated by an element $g \in G$ is called *cyclic*, and g is called a *generator* of G.

Being cyclic is a special property of a group. Not all groups are cyclic! A cyclic group can have many generators. In particular, if g is a generator, then so is g^{-1}.

Example 5.27. The group $\langle\mathbb{Z}_n; \oplus\rangle$ is cyclic for every n, where 1 is a generator. The generators of $\langle\mathbb{Z}_n; \oplus\rangle$ are all $g \in \mathbb{Z}_n$ for which $\gcd(g, n) = 1$, as the reader can prove as an exercise.

Example 5.28. The additive group of the integers, $\langle\mathbb{Z}; +, -, 0\rangle$, is an infinite cyclic group generated by 1. The only other generator is -1.

Theorem 5.7. *A cyclic group of order n is isomorphic to $\langle\mathbb{Z}_n; \oplus\rangle$ (and hence abelian).*

In fact, we use $\langle\mathbb{Z}_n; \oplus\rangle$ as our standard notation of a cyclic group of order n.

Proof. Let $G = \langle g\rangle$ be a cyclic group of order n (with neutral element e). The bijection $\mathbb{Z}_n \to G : i \mapsto g^i$ is a group isomorphism since $i \oplus j \mapsto g^{i+j} = g^i * g^j$. □

5.3.6 Application: Diffie-Hellman for General Groups

The Diffie-Hellman protocol was described in Section 4.6 for the group $\mathbb{Z}_p^*$ (this notation is defined below), but the concept of a group was not yet introduced there. As an application of general cyclic groups we mention that the Diffie-Hellman protocol works just as well in any cyclic group $G = \langle g\rangle$ for which computing x from g^x (i.e., the *discrete logarithm problem*) is computationally infeasible. Of course, one needs to apply a suitable mapping from G to a reasonable key space.

Elliptic curves (not discussed here) are an important class of cyclic groups used in cryptography.

5.3.7 The Order of Subgroups

The following theorem is one of the fundamental results in group theory. We state it without proof.

Theorem 5.8 (Lagrange). *Let G be a finite group and let H be a subgroup of G. Then the order of H divides the order of G, i.e., $|H|$ divides $|G|$.*

The following corollaries are direct applications of Lagrange's theorem.

Corollary 5.9. *For a finite group G, the order of every elements divides the group order, i.e.,* $\text{ord}(a)$ *divides $|G|$ for every $a \in G$.*

Proof. $\langle a\rangle$ is a subgroup of G of order $\text{ord}(a)$, which according to Theorem 5.8 must divide $|G|$. □

Corollary 5.10. *Let G be a finite group. Then $a^{|G|} = e$ for every $a \in G$.*

Proof. We have $|G| = k \cdot \text{ord}(a)$ for some k. Hence

$$a^{|G|} = a^{k\cdot\text{ord}(a)} = \left(a^{\text{ord}(a)}\right)^k = e^k = e.$$ □

Corollary 5.11. *Every group of prime order*[10] *is cyclic, and in such a group every element except the neutral element is a generator.*

Proof. Let $|G| = p$ with p prime. For any a, the order of the subgroup $\langle a\rangle$ divides p. Thus either $\text{ord}(a) = 1$ or $\text{ord}(a) = p$. In the first case, $a = e$ and in the latter case $G = \langle a\rangle$. □

Groups of prime order play a very important role in cryptography.

5.3.8 The Group $\mathbb{Z}_m^*$ and Euler's Function

We noted earlier that the set $\mathbb{Z}_m = \{0, \ldots, m-1\}$ is a group with respect to addition modulo m, denoted $\oplus$. We also noted that multiplication modulo m, denoted $\odot$ (where the modulus m is usually clear from the context), is of interest as well. However, $\mathbb{Z}_m$ is not a group with respect to multiplication modulo m. For example, in $\mathbb{Z}_{12}$, 8 has no inverse. We remember (see Section 4.5.3) that $a \in \mathbb{Z}_m$ has a multiplicative inverse if and only if $\gcd(a, m) = 1$. In order to obtain a group, we must exclude those a from $\mathbb{Z}_m$ for which $\gcd(a, m) \neq 1$. Thus we define

Definition 5.16. $\mathbb{Z}_m^* \stackrel{\text{def}}{=} \{a \in \mathbb{Z}_m \mid \gcd(a, m) = 1\}$.

[10] i.e., $|G| = p$ for some prime p.

Definition 5.17. The *Euler function* $\varphi : \mathbb{Z}^+ \to \mathbb{Z}^+$ is defined as the cardinality of $\mathbb{Z}_m^*$:

$$\varphi(m) = |\mathbb{Z}_m^*|.$$

Example 5.29. $\mathbb{Z}_{18}^* = \{1, 5, 7, 11, 13, 17\}$. Hence $\varphi(18) = 6$.

If p is a prime, then $\mathbb{Z}_p^* = \{1, \ldots, p-1\} = \mathbb{Z}_p \setminus \{0\}$, and $\varphi(p) = p - 1$.

Lemma 5.12. *If the prime factorization of m is $m = \prod_{i=1}^r p_i^{e_i}$, then*[11]

$$\varphi(m) = \prod_{i=1}^r (p_i - 1)p_i^{e_i - 1}.$$

Proof. For a prime p and $e \geq 1$ we have

$$\varphi(p^e) = p^{e-1}(p-1)$$

since exactly every pth integer in $\mathbb{Z}_{p^e}$ contains a factor p and hence $\varphi(p^e) = p^{e-1}(p-1)$ elements contain no factor p. For $a \in \mathbb{Z}_m$ we have $\gcd(a, m) = 1$ if and only if $\gcd(a, p_i^{e_i}) = 1$ for $i = 1, \ldots, r$. Since the numbers $p_i^{e_i}$ are pairwise relatively prime, the Chinese remainder theorem implies that there is a one-to-one correspondence between elements of $\mathbb{Z}_m$ and lists $(a_1, \ldots, a_r)$ with $a_i \in \mathbb{Z}_{p_i^{e_i}}$. Hence, using the above, there is also a one-to-one correspondence between elements of $\mathbb{Z}_m^*$ and lists $(a_1, \ldots, a_r)$ with $a_i \in \mathbb{Z}_{p_i^{e_i}}^*$. There are $\prod_{i=1}^r (p_i - 1)p_i^{e_i - 1}$ such lists. □

Theorem 5.13. $\langle \mathbb{Z}_m^*; \odot, ^{-1}, 1 \rangle$ *is a group.*

Proof. $\mathbb{Z}_m^*$ is closed under $\odot$ because if $\gcd(a, m) = 1$ and $\gcd(b, m) = 1$, then $\gcd(ab, m) = 1$. This is true since if ab and m have a common divisor > 1, then they also have a common prime divisor > 1, which would be a divisor of either a or b, and hence a common divisor of a and m or of b and m, contradicting that $\gcd(a, m) = 1$ and $\gcd(b, m) = 1$.

The associativity of $\odot$ is inherited from the associativity of multiplication in $\mathbb{Z}$. Moreover, 1 is a neutral element and inverses exist (see Section 4.5.3). Thus $\langle \mathbb{Z}_m^*; \odot, ^{-1}, 1 \rangle$ is a group. □

Example 5.30. In $\mathbb{Z}_{18}^* = \{1, 5, 7, 11, 13, 17\}$ we have $5 \odot 13 = 11$ and $11^{-1} = 5$ since $11 \odot 5 = 1$ (i.e., $R_{18}(11 \cdot 5) = 1$).

[11] Alternatively, $\varphi(m)$ could be defined as $\varphi(m) = m \cdot \prod_{\substack{p|m \\ p \text{ prime}}} \left(1 - \frac{1}{p}\right)$.

Example 5.31. In $\mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ we have $4 \odot 6 = 2$ and $7^{-1} = 8$ since $7 \odot 8 = 1$.

Now we obtain the following simple but powerful corollary to Theorem 5.8.

Corollary 5.14 (Fermat, Euler)**.** *For all $m \geq 2$ and all a with $\gcd(a, m) = 1$,*

$$a^{\varphi(m)} \equiv_m 1.$$

In particular, for every prime p and every a not divisible by p,

$$a^{p-1} \equiv_p 1.$$

Proof. This follows from Corollary 5.10 for the group $\mathbb{Z}_m^*$ of order $\varphi(m)$. □

The special case for primes was known already to Fermat.[12] The general case was proved by Euler, actually before the concept of a group was explicitly introduced in mathematics.

We state the following theorem about the structure of $\mathbb{Z}_m^*$ without proof. Of particular importance and interest is the fact that $\mathbb{Z}_p^*$ is cyclic for every prime p.

Theorem 5.15. *The group $\mathbb{Z}_m^*$ is cyclic if and only if $m = 2$, $m = 4$, $m = p^e$, or $m = 2p^e$, where p is an odd prime and $e \geq 1$.*

Example 5.32. The group $\mathbb{Z}_{19}^*$ is cyclic, and 2 is a generator. The powers of 2 are $2, 4, 8, 16, 13, 7, 14, 9, 18, 17, 15, 11, 3, 6, 12, 5, 10, 1$. The other generators are $3, 10, 13, 14$, and 15.

5.4 Application: RSA Public-Key Encryption

The RSA public-key cryptosystem, invented in 1977 by Rivest, Shamir, and Adleman[13], is used in many security protocols on the Internet, for instance in TLS/SSL. Like the Diffie-Hellman protocol it allows two parties to communicate securely, even if the communication channel is insecure, provided only that

[12] This result can be used as a primality test. Actually, the term "compositeness test" is more appropriate. To test whether a number n is prime one chooses a base a and checks whether $a^{n-1} \equiv_n 1$. If the condition is violated, then n is clearly composite, otherwise n could be a prime. Unfortunately, it is not guaranteed to be a prime. In fact, there are composite integers n for which $a^{n-1} \equiv_n 1$ for all a with $\gcd(a, n) = 1$. (Can you find such an n?) For more sophisticated versions of such a probabilistic test one can prove that for every composite n, the fraction of test values for which the corresponding condition is satisfied is at most $1/4$. Thus, by repeating the test sufficiently many times, the confidence that n is prime can be increased beyond any doubt. This is useful in practice, but it does not provide a proof that n is prime.

[13] R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, Vol. 21, No. 2, pp. 120–126, 1978.

they can authenticate each other's public keys (respectively the Diffie-Hellman values). Moreover, the RSA system can be used as a digital signature scheme (see below). RSA was the first cryptographic system offering this important functionality.

5.4.1 e-th Roots in a Group

To understand the RSA system, all we need is the following simple theorem which is a consequence of Lagrange's theorem (Theorem 5.8).

Theorem 5.16. *Let G be some finite group (multiplicatively written), and let $e \in \mathbb{Z}$ be relatively prime to $|G|$ (i.e. $\gcd(e, |G|) = 1$). The function $x \mapsto x^e$ is a bijection and the (unique) e-th root of $y \in G$, namely $x \in G$ satisfying $x^e = y$, is*

$$x = y^d,$$

where d is the multiplicative inverse of e modulo $|G|$, i.e.,

$$ed \equiv_{|G|} 1.$$

Proof. We have $ed = k \cdot |G| + 1$ for some k. Thus, for any $x \in G$ we have

$$(x^e)^d = x^{ed} = x^{k \cdot |G|+1} = \underbrace{(x^{|G|})^k}_{=1} \cdot x = x,$$

which means that the function $y \mapsto y^d$ is the inverse function of the function $x \mapsto x^e$ (which is hence a bijection). The under-braced term is equal to 1 because of Corollary 5.10. □

When $|G|$ is known, then d can be computed from $ed \equiv_{|G|} 1$ by using the extended Euclidean algorithm No general method is known for computing e-th roots in a group G without knowing its order. This can be exploited to define a public-key cryptosystem.

5.4.2 Description of RSA

Motivated by the Diffie-Hellman protocol also based on modular exponentiation, Rivest, Shamir and Adleman suggested as a possible class of groups the groups $\mathbb{Z}_n^*$, where $n = pq$ is the product of two sufficiently large secret primes, p and q. The order of $\mathbb{Z}_n^*$,

$$|\mathbb{Z}_n^*| = \varphi(n) = (p-1)(q-1),$$

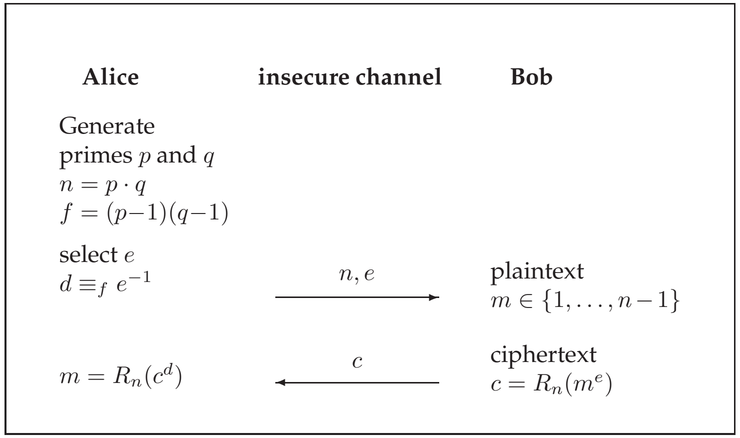

Figure 5.1: The naïve RSA public-key cryptosystem. Alice's public key is the pair (n, e) and her secret key is d. The public key must be sent to Bob via an authenticated channel. Bob can encrypt a message, represented as a number in $\mathbb{Z}_n$, by raising it to the eth power modulo n. Alice decrypts a ciphertext by raising it to the dth power modulo n.

can be computed only if the prime factors p and q of n are known.[14] The (public) encryption transformation is defined by

$$m \mapsto c = R_n(m^e),$$

and the (secret) decryption transformation is defined by

$$c \mapsto m = R_n(c^d),$$

where d can be computed according to

$$ed \equiv_{(p-1)(q-1)} 1.$$

The naïve[15] RSA public-key cryptosystem[16] is summarized in Figure 5.1.

The RSA system is usually (for instance in the TLS/SSL protocol) used only for *key management*, not for encrypting actual application data. The message

[14]One can show that one can efficiently compute p and q when given $(p-1)(q-1)$. (How?)

[15]The described version of using RSA is not secure, for several reasons. One reason is that it is deterministic and therefore an attacker can check potential messages by encrypting them himself and comparing the result with the ciphertext.

[16]The RSA encryption was defined above as a permutation on $\mathbb{Z}_n^*$. But one can show that encryption and decryption work for all $m \in \mathbb{Z}_n$. Thus the condition $\gcd(m, n) = 1$ need not be checked.

m is an encryption key (typically a short-term session key) for a conventional cryptosystem which is used to encrypt the actual application data (e.g. of a TLS session).

5.4.3 On the Security of RSA *

Let us have a brief look at the security of the RSA public-key system.[17] It is not known whether computing e-th roots modulo n (when $\gcd(e, \varphi(n)) = 1$) is easier than factoring n, but it is widely believed that the two problems are computationally equivalent.[18] Factoring large integers is believed to be computationally infeasible. If no significant breakthrough in factoring is achieved and if processor speeds keep improving at the same rate as they are now (using the so-called Moore's law), a modulus with 2048 bits appears to be secure for the next 15 years, and larger moduli (e.g. 8192 bits) are secure for a very long time.

Obviously, the system is insecure unless Bob can make sure he obtains the correct public key from Alice rather than a public key generated by an adversary and posted in the name of Alice. In other words, the public key must be sent from Alice to Bob via an authenticated channel. This is usually achieved (indirectly) using a so-called public-key certificate signed by a trusted certification authority. One also uses the term *public-key infrastructure (PKI)*. Explaining these concepts is beyond the scope of this course.

It is important to point out that for a public-key system to be secure, the message must be randomized in an appropriate manner. Otherwise, when given an encrypted message, an adversary can check plaintext messages by encrypting them and comparing them with the given encrypted message. If the message space is small (e.g. a bit), then this would allow to efficiently break the system.

5.4.4 Digital Signatures *

The RSA system can also be used for generating digital signatures. A digital signature can only be generated by the entity knowing the secret key, but it can be verified by anyone, e.g. by a judge, knowing the public key. Alice's signature s for a message m is

$$s = R_n(z^d) \quad \text{for} \quad z = m||h(m),$$

where $||$ denotes concatenation and h is a suitable function introducing redundancy into the message and the string z is naturally understood as an element of $\mathbb{Z}_n$.[19] A signature can be verified by raising it to the e-th power modulo n and checking that it is of the correct form $m||h(m)$. The message is recovered from the signature.

[17] But note that a cryptographic security analysis is much more involved.

[18] In fact, for a generic model of computation, this equivalence was proved in: D. Aggarwal and U. Maurer, *Breaking RSA generically is equivalent to factoring*, IEEE Transactions on Information Theory, vol. 62, pp. 6251–6259, 2016.

[19] This can be a so-called cryptographic hash function. Without such additional redundancy, every $s \in \mathbf{Z}_n^*$ would be a legitimate signature, and hence forging a signature would be trivial.

5.5 Rings and Fields

We now consider algebraic systems with two binary operations, usually called addition and multiplication.

5.5.1 Definition of a Ring

Definition 5.18. A *ring* $\langle R; +, -, 0, \cdot, 1\rangle$ is an algebra for which
- (i) $\langle R; +, -, 0\rangle$ is a commutative group.
- (ii) $\langle R; \cdot, 1\rangle$ is a monoid.
- (iii) $a(b+c) = (ab) + (ac)$ and $(b+c)a = (ba) + (ca)$ for all $a, b, c \in R$ (left and right distributive laws).

A ring is called *commutative* if multiplication is commutative ($ab = ba$).[20]

Example 5.33. $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$, and $\mathbb{C}$ are (commutative) rings.

Example 5.34. $\langle \mathbb{Z}_m; \oplus, \ominus, 0, \odot, 1\rangle$ is a commutative ring. Since $\langle \mathbb{Z}_m; \oplus, \ominus, 0\rangle$ is an abelian group and $\langle \mathbb{Z}_m; \odot, 1\rangle$ is a monoid, it only remains to check the distributive law, which is inherited from the distributive law for the integers.

We list some simple facts about rings.

Lemma 5.17. *For any ring $\langle R; +, -, 0, \cdot, 1\rangle$, and for all $a, b \in R$,*

- **(i)** $0a = a0 = 0$.
- **(ii)** $(-a)b = -(ab)$.
- **(iii)** $(-a)(-b) = ab$.
- **(iv)** *If R is non-trivial (i.e., if it has more than one element), then $1 \neq 0$.*

Proof. Proof of (i): We have

$$\begin{aligned}
0 &= -(a0) + a0 && (0 = -b + b \text{ for all } b \in R, \text{ e.g. for } b = a0)\\
&= -(a0) + a(0+0) && (0 + 0 = 0)\\
&= -(a0) + (a0 + a0) && \text{(distributive law)}\\
&= (-(a0) + a0) + a0 && \text{(associativity of } +)\\
&= 0 + a0 && (-b + b = 0 \text{ for all } b \in R)\\
&= a0 && (0 + b = b \text{ for all } b \in R)
\end{aligned}$$

[20] One can show (as an exercise) that ring addition must be commutative, i.e., commutativity of addition follows from the remaining ring axioms. The stated ring axioms are hence not minimal. The word "commutative" in (i) could be dropped.

The dual equation $0a = 0$ is proved similarly.[21]

The proofs of (ii), (iii), and (iv) are left as exercises. □

This lemma makes explicit that in a non-trivial ring, 0 has no multiplicative inverse since, according to (i) and (iv), $0a = 1$ is not possible. Thus requesting $\langle R; \cdot, 1\rangle$ to be a group rather than a monoid would make no sense.

Definition 5.19. The *characteristic* of a ring is the order of 1 in the additive group if it is finite, and otherwise the characteristic is defined to be 0 (not infinite).

Example 5.35. The characteristic of $\langle \mathbb{Z}_m; \oplus, \ominus, 0, \odot, 1\rangle$ is m. The characteristic of $\mathbb{Z}$ is 0.

5.5.2 Units and the Multiplicative Group of a Ring

Definition 5.20. An element u of a ring R is called a *unit*[22] if u is invertible, i.e., $uv = vu = 1$ for some $v \in R$. (We write $v = u^{-1}$.[23]) The set of units of R is denoted by R^*.

Example 5.36. The units of $\mathbb{Z}$ are -1 and 1: $\mathbb{Z}^* = \{-1, 1\}$.

Example 5.37. The units of $\mathbb{R}$ are all non-zero elements of $\mathbb{R}$: $\mathbb{R}^* = \mathbb{R} \setminus \{0\}$.

Example 5.38. The ring of Gaussian integers (see Example 4.5) contains four units: $1, i, -1$, and $-i$. For example, the inverse of i is $-i$.

Example 5.39. The set of units of $\mathbb{Z}_m$ is $\mathbb{Z}_m^*$ (Definition 5.16).[24]

Lemma 5.18. *For a ring R, R^* is a multiplicative group (the group of units of R).*

Proof. We need to show that R^* is closed under multiplication, i.e., that for $u \in R^*$ and $v \in R^*$, we also have $uv \in R^*$, which means that uv has an inverse. The inverse of uv is $v^{-1}u^{-1}$ since $(uv)(v^{-1}u^{-1}) = uvv^{-1}u^{-1} = uu^{-1} = 1$. R^* also contains the neutral element 1 (since 1 has an inverse). Moreover, the associativity of multiplication in R^* is inherited from the associativity of multiplication in R (since elements of R^* are also elements of R and the multiplication operation is the same). □

[21] This also follows from commutativity of multiplication, but the proof shows that the statement holds even for non-commutative rings.
[22] German: Einheit
[23] The inverse, if it exists, is unique.
[24] In fact, we now see the justification for the notation $\mathbb{Z}_m^*$ already introduced in Definition 5.16.

5.5.3 Divisors

In the following R denotes a commutative ring.

Definition 5.21. [25] For $a, b \in R$ we say that a *divides* b, denoted $a \mid b$, if there exists $c \in R$ such that $b = ac$. In this case, a is called a *divisor*[26] of b and b is called a *multiple*[27] of a.

Note that every non-zero element is a divisor of 0. Moreover, 1 and -1 are divisors of every element.

Lemma 5.19. *In any commutative ring,*

(i) *If $a \mid b$ and $b \mid c$, then $a \mid c$, i.e., the relation $\mid$ is transitive.*

(ii) *If $a \mid b$, then $a \mid bc$ for all c.*

(iii) *If $a \mid b$ and $a \mid c$, then $a \mid (b + c)$.*

Proof. Proof of (i). $a \mid b \Longrightarrow \exists d\ (b = ad)$. Also, $b \mid c \Longrightarrow \exists e\ (c = be)$. Thus $c = be = (ad)e = a(de)$, i.e., $a \mid c$.
The proofs of (ii) and (iii) are left as an exercise. □

As mentioned in Section 4.2.3, the concept of a greatest common divisor not only applies to integers, but to any commutative ring:

Definition 5.22. For ring elements a and b (not both 0), a ring element d is called a *greatest common divisor* of a and b if d divides both a and b and if every common divisor of a and b divides d, i.e., if

$$d \mid a \ \wedge\ d \mid b \ \wedge\ \forall c\, \big((c \mid a \wedge c \mid b) \ \to\ c \mid d\big).$$

5.5.4 Zerodivisors and Integral Domains

Definition 5.23. An element $a \neq 0$ of a commutative ring R is called a *zerodivisor*[28] if $ab = 0$ for some $b \neq 0$ in R.

Definition 5.24. An *integral domain*[29] is a (nontrivial[30]) commutative ring without zerodivisors: $\forall a \forall b\ \big(ab = 0 \ \to\ a = 0 \vee b = 0\big)$.

[25] Recall that this definition was already stated as Definition 4.1 for the special case of integers.
[26] German: Teiler
[27] German: Vielfaches
[28] German: Nullteiler
[29] German: Integritätsbereich
[30] i.e., $1 \neq 0$

Example 5.40. $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$, and $\mathbb{C}$ are integral domains.

Example 5.41. $\mathbb{Z}_m$ is not an integral domain if m is not a prime. Any element of $\mathbb{Z}_m$ not relatively prime to m is a zerodivisor.

Lemma 5.20. *In an integral domain, if $a \mid b$, then c with $b = ac$ is unique (and is denoted by $c = \frac{b}{a}$ or $c = b/a$ and called quotient).*[31]

Proof. Suppose that $b = ac = ac'$ for some c and c'. Then

$$0 = ac + (-(ac')) = a(c + (-c'))$$

and thus, because $a \neq 0$ and there are no zero-divisors, we must have $c+(-c') = 0$ and hence $c = c'$. □

5.5.5 Polynomial Rings

Definition 5.25. A *polynomial* $a(x)$ over a commutative ring R in the indeterminate x is a formal expression of the form

$$a(x) = a_d x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0 \quad = \sum_{i=0}^{d} a_i x^i.$$

for some non-negative integer d, with $a_i \in R$. The *degree* of $a(x)$, denoted $\deg(a(x))$, is the greatest i for which $a_i \neq 0$. The special polynomial 0 (i.e., all the a_i are 0) is defined to have degree "minus infinity".[32] Let $R[x]$ denote the set of polynomials (in x) over R.

Actually, it is mathematically better (but less common) to think of a polynomial simply as a finite list $(a_0, a_1, \ldots, a_{d-1}, a_d)$ of elements of R. There is no need to think of a variable x which suggests that it can be understood as a function $R \to R$. A polynomial can, but need not, be considered as such a function (see Section 5.7).

Addition and multiplication in $R[x]$ are defined as usual. Consider polynomials $a(x) = \sum_{i=0}^{d} a_i x^i$ of degree d and $b(x) = \sum_{i=0}^{d'} b_i x^i$ of degree d'. The sum of $a(x)$ and $b(x)$ is a polynomial of degree at most $\max(d, d')$ and is defined as

$$a(x) + b(x) \;=\; \sum_{i=0}^{\max(d,d')} (a_i + b_i)\, x^i,$$

[31]Note that the terms $\frac{b}{a}$ (or b/a) are defined only if $a \mid b$.

[32]The interpretation of "minus infinity" is that it is a quantity which remains unchanged when an arbitrary integer is added to it.

where here and in the following coefficients with index greater than the degree are understood to be 0. The product of $a(x)$ and $b(x)$ is defined as[33]

$$\begin{aligned} a(x)b(x) &= \sum_{i=0}^{d+d'} \left(\sum_{k=0}^{i} a_k b_{i-k} \right) x^i \;=\; \sum_{i=0}^{d+d'} \left(\sum_{u+v=i} a_u b_v \right) x^i \\ &= a_d b_{d'} x^{d+d'} + \cdots + (a_0 b_2 + a_1 b_1 + a_2 b_0)x^2 + (a_0 b_1 + a_1 b_0)x + a_0 b_0. \end{aligned}$$

The i-th coefficient of $a(x)b(x)$ is $\sum_{k=0}^{i} a_k b_{i-k} = \sum_{u+v=i} a_u b_v$, where the sum is over all pairs (u, v) for which $u + v = i$ as well as $u \geq 0$ and $v \geq 0$.

The degree of the product of polynomials over a ring R is, by definition, at most the sum of the degrees. It is equal to the sum if R is an integral domain, which implies that the highest coefficient is non-zero: $a_d b_{d'} \neq 0$ if $a_d \neq 0$ and $b_{d'} \neq 0$.

Example 5.42. Consider the ring $\mathbb{Z}_7$ and let $a(x) = 2x^2+3x+1$ and $b(x) = 5x+6$. Then

$$a(x) + b(x) \;=\; 2x^2 + (3+5)x + (1+6) \;=\; 2x^2 + x$$

and

$$a(x)b(x) \;=\; (2\cdot 5)x^3 + (3\cdot 5 + 2\cdot 6)x^2 + (1\cdot 5 + 3\cdot 6)x + 1\cdot 6 \;=\; 3x^3 + 6x^2 + 2x + 6.$$

Theorem 5.21. *For any commutative ring R, $R[x]$ is a commutative ring.*

Proof. We need to prove that the conditions of Definition 5.18 (i.e., the ring axioms) are satisfied for $R[x]$, assuming they are satisfied for R. We first observe that since multiplication in R is commutative, so is multiplication in $R[x]$.

Condition (i) requires that $R[x]$ is an abelian group with respect to (polynomial) addition. This is obvious from the definition of addition. Associativity and commutativity of (polynomial) addition are inherited from associativity and commutativity of addition in R (because R is a ring). The neutral element is the polynomial 0, and the inverse in the group (i.e., the negative) of $a(x) = \sum_{i=0}^{d} a_i x^i$ is $-a(x) = \sum_{i=0}^{d} (-a_i)x^i$.

Condition (ii) requires that $R[x]$ is a monoid with respect to (polynomial) multiplication. The polynomial 1 is the neutral element, which is easy to see. That multiplication is associative can be seen as follows. Let $a(x)$ and $b(x)$ as above, and $c(x) = \sum_{i=0}^{d''} c_i x^i$. Using the above definition of $a(x)b(x)$, we have

$$\big(a(x)b(x)\big)c(x) \;=\; \sum_{i=0}^{d+d'+d''} \left(\sum_{j=0}^{i} \left(\sum_{u+v=j} a_u b_v \right) c_{i-j} \right) x^i$$

[33]Note that, for example, the highest coefficient $\sum_{k=0}^{d+d'} a_k b_{i-k}$ is in the formula defined as a sum of $d + d' + 1$ terms, but all but one of them (namely for $k = d$) are zero.

$$= \sum_{i=0}^{d+d'+d''} \left(\sum_{u+v+w=i} (a_u b_v) c_w \right) x^i.$$

If one computes $a(x)\big(b(x)c(x)\big)$, one arrives at the same expression, by making use of associativity of multiplication in R, i.e., the fact that $(a_u b_v)c_w = a_u(b_v c_w) = a_u b_v c_w$.

Condition (iii), the distributive law, can also be shown to follow from the distributive law holding for R. □

Lemma 5.22.

(i) *If D is an integral domain, then so is $D[x]$.*

(ii) *The units of $D[x]$ are the constant polynomials that are units of D: $D[x]^* = D^*$.*

Proof. Left as an exercise. □

Example 5.43. Lemma 5.22 implies that for an integral domain D, the set $D[x][y]$ of polynomials in y with coefficients in $D[x]$, is also an integral domain. One can also view the elements of $D[x][y]$ as polynomials in two indeterminates, denoted $D[x, y]$.

5.5.6 Fields

Definition 5.26. A *field*[34] is a nontrivial commutative ring F in which every nonzero element is a unit, i.e., $F^* = F \setminus \{0\}$.

In other words, a ring F is a field if and only if $\langle F \setminus \{0\}; \cdot, ^{-1}, 1\rangle$ is an abelian group.

Example 5.44. $\mathbb{Q}, \mathbb{R}$, and $\mathbb{C}$ are fields, but $\mathbb{Z}$ and $R[x]$ (for any ring R) are not fields.

Theorem 5.23. *$\mathbb{Z}_p$ is a field if and only if p is prime.*

Proof. This follows from our earlier analysis of $\mathbb{Z}_p^*$, namely that $\mathbb{Z}_p \setminus \{0\}$ is a multiplicative group if and only if p is prime. □

In the following we denote the field with p elements by $\mathrm{GF}(p)$ rather than $\mathbb{Z}_p$. As explained later, "GF" stands for Galois field. Galois discovered finite fields around 1830.

[34] German: Körper

Fields are of crucial importance because in a field one can not only add, subtract, and multiply, but one can also divide by any nonzero element. This is the abstraction underlying many algorithms like those for solving systems of linear equations (e.g. by Gaussian elimination) or for polynomial interpolation. Also, a vector space, a crucial concept in mathematics, is defined over a field, the so-called base field. Vector spaces over $\mathbb{R}$ are just a special case.

Example 5.45. Solve the following system of linear equations over $\mathbb{Z}_{11}$:

$$\begin{aligned} 5x \oplus 2y &= 4 \\ 2x \oplus 7y &= 9 \end{aligned}$$

Solution: Eliminate x by adding 2 times the first and $\ominus 5 = 6$ times the second equation, resulting in

$$\underbrace{(2 \odot 5 \oplus 6 \odot 2)}_{=0} x + \underbrace{(2 \odot 2 \oplus 6 \odot 7)}_{=2} y = \underbrace{2 \odot 4 \oplus 6 \odot 9}_{=7},$$

which is equivalent to $2y = 7$. Thus $y = 2^{-1} \odot 7 = 6 \odot 7 = 9$. This yields

$$x = 2^{-1} \odot (9 \ominus 7 \odot y) = 6 \odot (9 \oplus 4 \odot 9) = 6 \odot 1 = 6.$$

Later we will describe a few applications of finite fields. Here we give another example of a finite field.

Example 5.46. We describe a field with 4 elements, $F = \{0, 1, A, B\}$, by giving the function tables of addition and multiplication:

$+$	0	1	A	B
0	0	1	A	B
1	1	0	B	A
A	A	B	0	1
B	B	A	1	0

$\cdot$	0	1	A	B
0	0	0	0	0
1	0	1	A	B
A	0	A	B	1
B	0	B	1	A

This field is not isomorphic to the ring $\mathbb{Z}_4$, which is not a field. We explain the construction of this 4-element field in Section 5.8.2.

Theorem 5.24. *A field is an integral domain.*

Proof. In a field, every non-zero element is a unit, and in an integral domain, every non-zero element must *not* be a zero-divisor. It hence suffices to show that in any commutative ring, a unit $u \in R$ is not a zerodivisor. To arrive at a contradiction, assume that $uv = 0$ for some v. Then we must have $v = 0$ since

$$v = 1v = u^{-1}uv = u^{-1}0 = 0,$$

and hence u is not a zerodivisor. □

5.6 Polynomials over a Field

Recall Definition 5.25 of a polynomial over a ring R. As mentioned several times, the set $R[x]$ of polynomials over R is a ring with respect to the standard polynomial addition and multiplication. The neutral elements of addition and multiplication are the polynomials 0 and 1, respectively.

Polynomials over a field F are of special interest, for reasons to become clear. Namely, they have properties in common with the integers, $\mathbb{Z}$.

5.6.1 Factorization and Irreducible Polynomials

For $a, b \in \mathbb{Z}$, if b divides a, then also $-b$ divides a. The analogy for polynomials is as follows. If $b(x)$ divides $a(x)$, then so does $v \cdot b(x)$ for any nonzero $v \in F$ because if $a(x) = b(x) \cdot c(x)$, then $a(x) = vb(x) \cdot \big(v^{-1}c(x)\big)$. Among the polynomials $vb(x)$ (for $v \in F$), which are in a certain sense associated to each other, there is a distinguished one, namely that with leading coefficient 1. This is analogous to b and $-b$ being associated in $\mathbb{Z}$ (see Section 5.6.3) and the positive one being distinguished.

Definition 5.27. A polynomial $a(x) \in F[x]$ is called *monic*[35] if the leading coefficient is 1.

Example 5.47. In GF(5)$[x]$, $x + 2$ divides $x^2 + 1$ since $x^2 + 1 = (x + 2)(x + 3)$. Also, $2x + 4$ divides $x^2 + 1$ since $x^2 + 1 = (2x + 4)(3x + 4)$. More generally, $v \cdot (x+2)$ divides x^2+1 for any $v \in \text{GF}(5)$ because $x^2+1 = v(x+2) \cdot v^{-1}(x+3)$.

One can factor polynomials, similarly to the factorization of integers.

Example 5.48. In GF(7)$[x]$ we have

$$x^3 + 2x^2 + 5x + 2 \;=\; (x + 5)(x^2 + 4x + 6).$$

In GF(2)$[x]$ we have

$$x^6 + x^5 + x^4 + x^3 + 1 \;=\; (x^2 + x + 1)(x^4 + x + 1).$$

Definition 5.28. A polynomial $a(x) \in F[x]$ with degree at least 1 is called *irreducible* if it is divisible only by constant polynomials and by constant multiples of $a(x)$.

The notion of irreducibility in $F[x]$ corresponds to the notion of primality in $\mathbb{Z}$, in a sense to be made more precise in Section 5.6.3.

[35] German: monisch, normiert

It follows immediately from the definition (and from the fact that the degrees are added when polynomials are multiplied) that every polynomial of degree 1 is irreducible. Moreover, a polynomial of degree 2 is either irreducible or the product of two polynomials of degree 1. A polynomial of degree 3 is either irreducible or it has at least one factor of degree 1. Similarly, a polynomial of degree 4 is either irreducible, has a factor of degree 1, or has an irreducible factor of degree 2.

Irreducibility of a polynomial of degree d can be checked by testing all irreducible polynomials of degree $\leq d/2$ as possible divisors. Actually, it suffices to test only the monic polynomials because one could always multiply a divisor by a constant, for example the inverse of the highest coefficient. This irreducibility test is very similar to the primality test which checks all divisors up to the square root of the number to be tested.

Example 5.49. In GF(5)$[x]$, $x^2 + 2$ is irreducible since (as one can check) $x + \alpha$ does not divide $x^2 + 2$, for all $\alpha \in \text{GF}(5)$.

Example 5.50. In GF(7)$[x]$, $x^2 + 4x + 6$ is irreducible since $x + \alpha$ does not divide $x^2 + 4x + 6$, for all $\alpha \in \text{GF}(7)$.

Example 5.51. In GF(2)$[x]$, $x^2 + x + 1$ is irreducible because neither x nor $x + 1$ divides $x^2 + x + 1$. It is easy to see that this is the only irreducible polynomial of degree 2 over GF(2). There are two irreducible polynomials of degree 3 over GF(2), namely $x^3 + x + 1$ and $x^3 + x^2 + 1$. Moreover, there are three irreducible polynomials of degree 4 over GF(2), which the reader can find as an exercise.

Not only the concepts of divisors and division with remainders (see below) carries over from $\mathbb{Z}$ to $F[x]$, also the concept of *the* greatest common divisor can be carried over. Recall that $F[x]$ is a ring and hence the notion of a greatest common divisor is defined. For the special type of ring $F[x]$, as for $\mathbb{Z}$, one can single out one of them.

Definition 5.29. The monic polynomial $g(x)$ of largest degree such that $g(x) \mid a(x)$ and $g(x) \mid b(x)$ is called *the* greatest common divisor of $a(x)$ and $b(x)$, denoted $\gcd(a(x), b(x))$.

Example 5.52. Consider GF(7)$[x]$. Let $a(x) = x^3 + 4x^2 + 5x + 2$ and $b(x) = x^3 + 6x^2 + 4x + 6$. One can easily check that $a(x) = (x + 1)(x^2 + 3x + 2)$ and $b(x) = (x + 3)(x^2 + 3x + 2)$. Thus $\gcd(a(x), b(x)) = x^2 + 3x + 2$.

Example 5.53. Consider GF(2)$[x]$. Let $a(x) = x^3+x^2+x+1$ and $b(x) = x^2+x+1$. Then $\gcd(a(x), b(x)) = 1$.

5.6.2 The Division Property in $F[x]$

Let F be a field. The ring $F[x]$ has strong similarities with the integers $\mathbb{Z}$ (see Section 5.6.3). Both these integral domains have the special property that one can divide one element a by another element $b \neq 0$, resulting in a quotient q and a remainder r which are unique when r is required to be "smaller" than the divisor. In case of the integers, the "size" of $b \in \mathbb{Z}$ is given by the absolute value $|b|$, and the "size" of a polynomial $b(x) \in F[x]$ can be defined as its degree $\deg(b(x))$.

Theorem 5.25. *Let F be a field. For any $a(x)$ and $b(x) \neq 0$ in $F[x]$ there exist a unique $q(x)$ (the quotient) and a unique $r(x)$ (the remainder) such that*

$$a(x) = b(x) \cdot q(x) + r(x) \quad \textit{and} \quad \deg(r(x)) < \deg(b(x)).$$

Proof sketch. We first prove the existence of $q(x)$ and $r(x)$ and then the uniqueness. If $\deg(b(x)) > \deg(a(x))$, then $q(x) = 0$ and $r(x) = a(x)$. We thus assume that $\deg(b(x)) \leq \deg(a(x))$. Let $a(x) = a_m x^m + \cdots$ and $b(x) = b_n x^n + \cdots$ with $n \leq m$, where "$\cdots$" stands for lower order terms. The first step of polynomial division consists of subtracting $a_m b_n^{-1} b(x) x^{m-n}$ from $a(x)$, resulting in a polynomial of degree at most $m-1$.[36] Continuing polynomial division finally yields $q(x)$ and $r(x)$, where $\deg(r(x)) < \deg(b(x))$ since otherwise one could still subtract a multiple of $b(x)$.

To prove the uniqueness, suppose that

$$a(x) = b(x)q(x) + r(x) = b(x)q'(x) + r'(x),$$

where $\deg(r(x)) < \deg(b(x))$ and $\deg(r'(x)) < \deg(b(x))$. Then

$$b(x)[q(x) - q'(x)] = r'(x) - r(x).$$

Since $\deg(r'(x) - r(x)) < \deg(b(x))$, this is possible only if $q(x) - q'(x) = 0$, i.e., $q(x) = q'(x)$, which also implies $r'(x) = r(x)$.[37] □

In analogy to the notation $R_m(a)$, we will denote the remainder $r(x)$ of the above theorem by $R_{b(x)}(a(x))$.

Example 5.54. Let F be the field GF(7) and let $a(x) = x^3 + 2x^2 + 5x + 4$ and $b(x) = 2x^2 + x + 1$. Then $q(x) = 4x + 6$ and $r(x) = 2x + 5$ since

$$(x^3 + 2x^2 + 5x + 4) = (2x^2 + x + 1) \cdot (4x + 6) + (2x + 5),$$

[36] Note that here it is important that F is a field since otherwise the existence of b_n^{-1} is not guaranteed.

[37] Note that here we have made use of the fact that $\deg(u(x)v(x)) = \deg(u(x)) + \deg(v(x))$. We point out that this only holds in an integral domain. (Why?) Recall that a field is an integral domain (Theorem 5.24).

Example 5.55. In GF(2)$[x]$ we have

$$(x^4 + x^3 + x^2 + 1) : (x^2 + 1) = x^2 + x \quad \text{with remainder } x + 1.$$

Note that in GF(2), $-1 = 1$. For example, $-x = x$ and $-x^2 = x^2$

5.6.3 Analogies Between $\mathbb{Z}$ and $F[x]$, Euclidean Domains *

In this section we describe the abstraction underlying both $\mathbb{Z}$ and $F[x]$.

Definition 5.30. In an integral domain, a and b are called *associates*, denoted $a \sim b$, if $a = ub$ for some unit u.

Definition 5.31. In an integral domain, a non-unit $p \in D \setminus \{0\}$ is *irreducible* if, whenever $p = ab$, then either a or b is a unit.[38]

The units in $\mathbb{Z}$ are 1 and -1 and the units in $F[x]$ are the non-zero constant polynomials (of degree 0). In $\mathbb{Z}$, a and $-a$ are associates.

Example 5.56. In $\mathbb{Z}$, 6 and -6 are associates. In GF(5)$[x]$, $x^2 + 2x + 3$, $2x^2 + 4x + 1$, $3x^2 + x + 4$, and $4x^2 + 3x + 2$ are associates.

For $a \in D$ one can define one associate to be distinguished. For $\mathbb{Z}$ the distinguished associate of a is $|a|$, and for $a(x) \in F[x]$ the distinguished associate of $a(x)$ is the monic polynomial associated with $a(x)$. If we consider only the distinguished associates of irreducible elements, then for $\mathbb{Z}$ we arrive at the usual notion of prime numbers.[39]

We point out that the association relation is closely related to divisibility. The proof of the following lemma is left as an exercise.

Lemma 5.26. $a \sim b \iff a \mid b \,\wedge\, b \mid a$.

There is one more crucial property shared by both integral domains $\mathbb{Z}$ and $F[x]$ (for any field F), described in the following abstract definition.

Definition 5.32. A *Euclidean domain* is an integral domain D together with a so-called *degree function* $d : D \setminus \{0\} \to \mathbb{N}$ such that

(i) For every a and $b \neq 0$ in D there exist q and r such that $a = bq + r$ and $d(r) < d(b)$ or $r = 0$.

(ii) For all nonzero a and b in D, $d(a) \leq d(ab)$.

Example 5.57. The Gaussian integers $\mathbb{Z}[\sqrt{-1}]$ discussed earlier are a Euclidean domain where the degree of $a + bi$ is $\sqrt{a^2 + b^2}$, i.e., the absolute value (of complex numbers).

One can prove that in a Euclidean domain, the greatest (according to the degree function) common divisor is well-defined, up to taking associates, i.e., up to multiplication

[38] In other words, p is divisible only by units and associates of p.

[39] There is a notion of a *prime* element of a ring, which is different from the notion of an irreducible element, but for the integers $\mathbb{Z}$ the two concepts coincide.

by a unit. The condition $d(r) < d(b)$ guarantees that the gcd can be computed in the well-known manner by continuous division. This procedure terminates because $d(r)$ decreases monotonically in each division step.

The following theorem can be proved in a manner analogous to the proof of the unique factorization theorem for $\mathbb{Z}$. One step is to show that a Euclidean domain is a principle ideal domain.

Theorem 5.27. *In a Euclidean domain every element can be factored uniquely (up to taking associates) into irreducible elements.*

5.7 Polynomials as Functions

5.7.1 Polynomial Evaluation

For a ring R, a polynomial $a(x) \in R[x]$ can be interpreted as a function $R \to R$ by defining *evaluation* of $a(x)$ at $\alpha \in R$ in the usual manner. This defines a function $R \to R : \alpha \mapsto a(\alpha)$.

Example 5.58. Consider the field $GF(5)$ and the polynomial $a(x) = 2x^3+3x+1$. Then $a(0) = 1$, $a(1) = 1$, $a(2) = 3$, $a(3) = 4$, and $a(4) = 1$.

The following lemma is easy to prove:

Lemma 5.28. *Polynomial evaluation is compatible with the ring operations:*

- *If $c(x) = a(x) + b(x)$, then $c(\alpha) = a(\alpha) + b(\alpha)$ for any α.*
- *If $c(x) = a(x) \cdot b(x)$, then $c(\alpha) = a(\alpha) \cdot b(\alpha)$ for any α.*

5.7.2 Roots

Definition 5.33. Let $a(x) \in R[x]$. An element $\alpha \in R$ for which $a(\alpha) = 0$ is called a *root*[40] of $a(x)$.

Example 5.59. The polynomial $x^3 - 7x + 6$ in $\mathbb{R}[x]$ has 3 roots: $-3, 1$, and 2. The polynomial $x^2 + 1$ in $\mathbb{R}[x]$ has no root. The polynomial $(x^3 + 2x^2 + 5x + 2)$ in GF(7)$[x]$ has 2 as its only root. The polynomial $(x^4 + x^3 + x + 1)$ in GF(2)$[x]$ has the root 1.

Lemma 5.29. *For a field F, $\alpha \in F$ is a root of $a(x)$ if and only if $x - \alpha$ divides $a(x)$.*

[40]German: Nullstelle oder Wurzel

Proof. ($\Longrightarrow$) Assume that α is a root, i.e., $a(\alpha) = 0$. Then, according to Theorem 5.25, we can write $a(x)$ as

$$a(x) = (x - \alpha)q(x) + r(x),$$

where $\deg(r(x)) < \deg(x - \alpha) = 1$, i.e., $r(x)$ is a constant r, where

$$r \;=\; a(x) - (x - \alpha)q(x).$$

Setting $x = \alpha$ in the above equation gives

$$r \;=\; a(\alpha) - (\alpha - \alpha)q(\alpha) \;=\; 0 - 0 \cdot q(\alpha) \;=\; 0.$$

Hence $x - \alpha$ divides $a(x)$.
($\Longleftarrow$) To prove the other direction, assume that $x - \alpha$ divides $a(x)$, i.e., $a(x) = (x - \alpha)q(x)$ for some $q(x)$. Then $a(\alpha) = (\alpha - \alpha)q(\alpha) = 0$, i.e., α is a root of $a(x)$. □

Lemma 5.29 implies that an irreducible polynomial of degree ≥ 2 has no roots.

Corollary 5.30. *A polynomial $a(x)$ of degree 2 or 3 over a field F is irreducible if and only if it has no root.*[41]

Proof. A reducible polynomial of degree 2 or 3 has a factor of degree 1 and hence a root. An irreducible polynomial has no root because according to Lemma 5.29, such a root would correspond to a (linear) factor. □

Theorem 5.31. *For a field F, a nonzero*[42] *polynomial $a(x) \in F[x]$ of degree d has at most d roots.*

Proof. To arrive at a contradiction, suppose that $a(x)$ has degree d but $e > d$ roots, say $\alpha_1, \ldots, \alpha_e$. Then the polynomial $\prod_{i=1}^{e}(x - \alpha_i)$ divides $a(x)$. Since this is a polynomial of degree e, $a(x)$ has degree at least e, and hence more than d, which is a contradiction. □

5.7.3 Polynomial Interpolation

It is well-known that a polynomial of degree d over $\mathbb{R}$ can be interpolated from any $d + 1$ values. Since the proof requires only the properties of a field (rather than the special properties of $\mathbb{R}$), this interpolation property holds for polynomials over any field F. This fact is of crucial importance in many applications.

[41]Note that this statement is not true for polynomials of degree ≥ 4.
[42]Note that every $\alpha \in F$ is a root of the polynomial 0.

Lemma 5.32. *A polynomial $a(x) \in F[x]$ of degree at most d is uniquely determined by any $d+1$ values of $a(x)$, i.e., by $a(\alpha_1), \ldots, a(\alpha_{d+1})$ for any distinct $\alpha_1, \ldots, \alpha_{d+1} \in F$.*

Proof. Let $\beta_i = a(\alpha_i)$ for $i = 1, \ldots, d+1$. Then $a(x)$ is given by Lagrange's interpolation formula:

$$a(x) = \sum_{i=1}^{d+1} \beta_i u_i(x),$$

where the polynomial $u_i(x)$ is given by

$$u_i(x) = \frac{(x-\alpha_1)\cdots(x-\alpha_{i-1})(x-\alpha_{i+1})\cdots(x-\alpha_{d+1})}{(\alpha_i-\alpha_1)\cdots(\alpha_i-\alpha_{i-1})(\alpha_i-\alpha_{i+1})\cdots(\alpha_i-\alpha_{d+1})}.$$

Note that for $u_i(x)$ to be well-defined, all constant terms $\alpha_i - \alpha_j$ in the denominator must be invertible. This is guaranteed if F is a field since $\alpha_i - \alpha_j \neq 0$ for $i \neq j$. Note also that the denominator is simply a constant and hence $u_i(x)$ is indeed a polynomial of degree d. It is easy to verify that $u_i(\alpha_i) = 1$ and $u_i(\alpha_j) = 0$ for $j \neq i$. Thus the polynomials $a(x)$ and $\sum_{i=1}^{d+1} \beta_i u_i(x)$ agree when evaluated at any α_i, for all i. We note that $a(x)$ has degree at most d.[43]

It remains to prove the uniqueness. Suppose there is another polynomial $a'(x)$ of degree at most d such that $\beta_i = a'(\alpha_i)$ for $i = 1, \ldots, d+1$. Then $a(x) - a'(x)$ is also a polynomial of degree at most d, which (according to Theorem 5.31) can have at most d roots, unless it is 0. But $a(x) - a'(x)$ has indeed the $d+1$ roots $\alpha_1, \ldots, \alpha_{d+1}$. Thus it must be the 0-polynomial, which implies $a(x) = a'(x)$. □

5.8 Finite Fields

So far we have seen the finite field $GF(p)$, where p is prime. In this section we discuss all other finite fields.

5.8.1 The Ring $F[x]_{m(x)}$

We continue to explore the analogies between the rings $\mathbb{Z}$ and $F[x]$. In the same way as we can compute in the integers $\mathbb{Z}$ modulo an integer m, yielding the ring $\langle \mathbb{Z}_m; \oplus, \ominus, 0, \odot, 1\rangle$, we can also compute in $F[x]$ modulo a polynomial $m(x)$. Let $R_{m(x)}(a(x))$ denote the (unique) remainder when $a(x)$ is divided by $m(x)$. The concept of congruence modulo $m(x)$ is defined like congruence modulo m. For $a(x), b(x) \in F[x]$,

$$a(x) \equiv_{m(x)} b(x) \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad m(x) \mid \big(a(x) - b(x)\big).$$

[43] The degree can be smaller than d.

The proof of the following lemma is analogous to the proof that congruence modulo m is an equivalence relation on $\mathbb{Z}$.

Lemma 5.33. *Congruence modulo $m(x)$ is an equivalence relation on $F[x]$, and each equivalence class has a unique representative of degree less than $\deg(m(x))$.*

Example 5.60. Consider $\mathbb{R}[x]$ or $\mathbb{Q}[x]$. We have, for example,

$$5x^3 - 2x + 1 \equiv_{3x^2+2} 8x^3 + 1 \equiv_{3x^2+2} -\tfrac{16}{3}x + 1$$

as one can easily check. Actually, the remainder when $5x^3 - 2x + 1$ is divided by $3x^2 + 2$ is $-\frac{16}{3}x + 1$.

Example 5.61. Consider $\mathrm{GF}(2)[x]$. Example 5.55 can be rephrased as $R_{x^2+1}(x^4 + x^3 + x^2 + 1) = x + 1$.

Definition 5.34. Let $m(x)$ be a polynomial of degree d over F. Then

$$F[x]_{m(x)} \stackrel{\text{def}}{=} \big\{a(x) \in F[x] \;\big|\; \deg(a(x)) < d\big\}.$$

We state a simple fact about the cardinality of $F[x]_{m(x)}$ when F is finite.

Lemma 5.34. *Let F be a finite field with q elements and let $m(x)$ be a polynomial of degree d over F. Then $|F[x]_{m(x)}| = q^d$.*

Proof. We have

$$F[x]_{m(x)} = \big\{a_{d-1}x^{d-1} + \cdots + a_1 x + a_0 \;\big|\; a_0, \ldots, a_{d-1} \in F\big\}.$$

There are q^d choices for $a_0, \ldots, a_{d-1}$. □

$F[x]_{m(x)}$ is derived from $F[x]$ in close analogy to how the ring $\mathbb{Z}_m$ is derived from the ring $\mathbb{Z}$.

Lemma 5.35. *$F[x]_{m(x)}$ is a ring with respect to addition and multiplication modulo $m(x)$.*[44]

Proof. $F[x]_{m(x)}$ is a group with respect to polynomial addition.[45] The neutral element is the polynomial 0 and the negative of $a(x) \in F[x]_{m(x)}$ is $-a(x)$. Associativity is inherited from $F[x]$.

[44] It is important to point out that we are considering three algebraic systems, namely F, $F[x]$, and $F[x]_{m(x)}$. Each system has an addition and a multiplication operation, and we use the same symbols "+" and "·" in each case, letting the context decide which one we mean. This should cause no confusion. The alternative would have been to always use different symbols, but this would be too cumbersome. Note that, as mentioned above, addition (but not multiplication) in $F[x]$ and $F[x]_{m(x)}$ are identical.

[45] Note that the sum of two polynomials is never reduced modulo $m(x)$ because the degree of the sum is at most the maximum of the two degrees. In other words, $a(x)+b(x)$ in $F[x]$ and $a(x)+b(x)$ in $F[x]_{m(x)}$ are the same operation when restricted to polynomials of degree less than $\deg(m(x))$.

$F[x]_{m(x)}$ is a monoid with respect to polynomial multiplication. The neutral element is the polynomial 1. Associativity of multiplication is inherited from $F[x]$, as is the distributive law. □

The following lemma can be proved in analogy to Lemma 4.18.

Lemma 5.36. *The congruence equation*

$$a(x)b(x) \equiv_{m(x)} 1$$

(for a given $a(x)$) has a solution $b(x) \in F[x]_{m(x)}$ if and only if $\gcd(a(x), m(x)) = 1$. *The solution is unique.*[46] *In other words,*

$$F[x]^*_{m(x)} = \{a(x) \in F[x]_{m(x)} \mid \gcd(a(x), m(x)) = 1\}.$$

Inverses in $F[x]^*_{m(x)}$ can be computed efficiently by a generalized version of Euclid's gcd-algorithm, which we do not discuss here.

5.8.2 Constructing Extension Fields

The following theorem is analogous to Theorem 5.23 stating that $\mathbb{Z}_m$ is a field if and only if m is prime.

Theorem 5.37. *The ring $F[x]_{m(x)}$ is a field if and only if $m(x)$ is irreducible.*[47]

Proof. For an irreducible polynomial $m(x)$, we have $\gcd(a(x), m(x)) = 1$ for all $a(x) \neq 0$ with $\deg(a(x)) < \deg(m(x))$ and therefore, according to Lemma 5.36, $a(x)$ is invertible in $F[x]_{m(x)}$. In other words, $F[x]^*_{m(x)} = F[x]_{m(x)} \setminus \{0\}$. If $m(x)$ is not irreducible, then $F[x]_{m(x)}$ is not a field because nontrivial factors of $m(x)$ have no multiplicative inverse. □

In Computer Science, the fields of most interest are finite fields, i.e., $F[x]_{m(x)}$ where F itself is a finite field. But before we discuss finite fields, we illustrate this new type of field based on polynomial arithmetic using a well-known example of an infinite field.

Example 5.62. The polynomial x^2+1 is irreducible in $\mathbb{R}[x]$ because x^2+1 has no root in $\mathbb{R}$. Hence, according to Theorem 5.37, $\mathbb{R}[x]_{x^2+1}$ is a field. The elements of $\mathbb{R}[x]_{x^2+1}$ are the polynomials of degree at most 1, i.e., of the form $ax+b$. Addition and multiplication are defined by

$$(ax+b)+(cx+d) = (a+c)x+(b+d)$$

[46] This $b(x)$ (if it exists) is called the inverse of $a(x)$ modulo $m(x)$.
[47] $F[x]_{m(x)}$ is called an *extension field* of F.

and

$$\begin{aligned}(ax+b)\cdot(cx+d) &= R_{x^2+1}((ax+b)\cdot(cx+d)) \\ &= R_{x^2+1}(acx^2+(bc+ad)x+bd) \\ &= (bc+ad)x+(bd-ac).\end{aligned}$$

The last step follows from the fact that $R_{x^2+1}(x^2) = -1$. The reader may have noticed already that these addition and multiplication laws correspond to those of the complex numbers $\mathbb{C}$ when $ax+b$ is interpreted as the complex number $b+ai$. Indeed, $\mathbb{R}[x]_{x^2+1}$ is simply $\mathbb{C}$ or, more precisely, $\mathbb{R}[x]_{x^2+1}$ is isomorphic to $\mathbb{C}$. In fact, this appears to be the most natural way of defining $\mathbb{C}$.

This example raises a natural question: Can we define other extension fields of $\mathbb{R}$, or, what is special about $\mathbb{C}$? There are many other irreducible polynomials of degree 2, namely all those corresponding to a parabola not intersecting with the x-axis. What is, for example, the field $\mathbb{R}[x]_{2x^2+x+1}$? One can show that $\mathbb{R}[x]_{m(x)}$ is isomorphic to $\mathbb{C}$ for every irreducible polynomial of degree 2 over $\mathbb{R}$. Are there irreducible polynomials of higher degree over $\mathbb{R}$? The answer, as we know, is negative. Every polynomial in $\mathbb{R}[x]$ can be factored into a product of polynomials of degree 1 (corresponding to real roots) and polynomials of degree 2 (corresponding to pairs of conjugate complex roots). The field $\mathbb{C}$ has the special property that a polynomial of degree d has *exactly* d roots in $\mathbb{C}$. For the field $\mathbb{R}$, this is not true. There are no irreducible polynomials of degree > 1 over $\mathbb{C}$.

Example 5.63. The polynomial x^2+x+1 is irreducible in GF(2)$[x]$ because it has no roots. Hence, according to Theorem 5.37, GF(2)$[x]_{x^2+x+1}$ is a field. The elements of GF(2)$[x]_{x^2+x+1}$ are the polynomials of degree at most 1, i.e., of the form $ax+b$. Addition is defined by

$$(ax+b)+(cx+d) = (a+c)x+(b+d).$$

Note that the "$+$" in $ax+b$ is in GF(2) (i.e., in $\mathbb{Z}_2$), and the middle "$+$" in $(ax+b)+(cx+d)$ is to be understood in GF(2)$[x]_{x^2+x+1}$, i.e., as polynomial addition. Multiplication is defined by

$$\begin{aligned}(ax+b)\cdot(cx+d) &= R_{x^2+x+1}((ax+b)\cdot(cx+d)) \\ &= R_{x^2+x+1}(acx^2+(bc+ad)x+bd) \\ &= (bc+ad+ac)x+(bd+ac).\end{aligned}$$

The last step follows from the fact that $R_{x^2+x+1}(x^2) = -x-1 = x+1$ (since $-1 = 1$ in GF(2)). It now becomes clear that this field with 4 elements is that of Example 5.46. The reader can check that $A = x$ and $B = x+1$ works just as well as $A = x+1$ and $B = x$.

$+$	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
0	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
1		0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
x			0	1	x^2+x	x^2+x+1	x^2	x^2+1
$x+1$				0	x^2+x+1	x^2+x	x^2+1	x^2
x^2					0	1	x	$x+1$
x^2+1						0	$x+1$	x
x^2+x							0	1
x^2+x+1								0

Figure 5.2: The addition table for GF(8) constructed with the irreducible polynomial x^3+x+1.

$\cdot$	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
0	0	0	0	0	0	0	0	0
1		1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
x			x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
$x+1$				x^2+1	x^2+x+1	x^2	1	x
x^2					x^2+x	x	x^2+1	1
x^2+1						x^2+x+1	$x+1$	x^2+x
x^2+x							x	x^2
x^2+x+1								$x+1$

Figure 5.3: The multiplication table for GF(8) constructed with the irreducible polynomial x^3+x+1.

Example 5.64. The polynomial x^3+x+1 over GF(2) is irreducible since it has no roots (it evaluates to 1 for both 0 and 1). The field GF(8) (also written GF(2^3)) consists of the 8 polynomials of degree ≤ 2 over GF(2). The tables for addition and multiplication are shown in Figures 5.2 and 5.3. In this field we have, for example,

$$(x+1)/(x^2+1) = (x+1)(x^2+1)^{-1} = (x+1)x = x^2+x.$$

5.8.3 Some Facts About Finite Fields *

Theorem 5.37 gives us a method for constructing a new field from an existing field F, provided we can find an irreducible polynomial $m(x)$ in $F[x]$. When F is a finite field, then so is $F[x]_{m(x)}$. The proofs of most facts stated in this section are beyond the scope of this course.

The theory of finite fields was founded by the French mathematician Evariste Galois

(1811–1832).[48] In his honor, finite fields are called *Galois fields*. A field with q elements is usually denoted by GF(q) (independently of how it is constructed).

Theorem 5.38. *For every prime p and every $d \geq 1$ there exists an irreducible polynomial of degree d in* GF$(p)[x]$. *In particular, there exists a finite field with p^d elements.*

The following theorem states that the finite fields we have seen so far, $\mathbb{Z}_p$ for prime p and GF$(p)[x]_{m(x)}$ for an irreducible $m(x)$, are all finite fields. There are no other finite fields. Moreover, one obtains no new finite fields by taking an irreducible polynomial, say of degree d', over some extension field GF(p^d), resulting in the field GF$(p^{dd'})$. Such a field can always be constructed directly using an irreducible polynomial of degree dd' over GF(p).

Theorem 5.39. *There exists a finite field with q elements if and only if q is a power of a prime. Moreover, any two finite fields of the same size q are isomorphic.*

The last claim justifies the use of the notation GF(q) without making explicit how the field is constructed. Different constructions yield different representations (naming) of the field elements, but not different fields. However, it is possible that some representations are better suited than others for the efficient hardware or software implementation of the field arithmetic.

Theorem 5.40. *The multiplicative group of every finite field* GF(q) *is cyclic.*

Note that the multiplicative group of GF(q) has order $q-1$ and has $\varphi(q-1)$ generators.

Example 5.65. One can check that the fields GF(2^2) and GF(2^3) have multiplicative groups of orders 3 and 7, which are both prime. Therefore all elements except 1 (and 0 of course) are generators of the multiplicative group.

5.9 Application: Error-Correcting Codes

5.9.1 Definition of Error-Correcting Codes

Finite fields are of great importance in Computer Science and have many applications, one of which is discussed in this section.

Error-correcting codes are used in many communication protocols and other applications. For example, the digital information on a CD is stored in such a manner that even if some of the information is lost (e.g. because of a scratch or dirt on the disc), the entire information can still be reconstructed without quality degradation, as long as sufficiently much of the data is still available.

[48] His interest was in proving that polynomial equations over $\mathbb{R}$ of fifth or higher degree have in general no closed form solution in radicals (while equations of up to fourth degree do). His major contributions to mathematics were recognized by the leading mathematicians only many years after his death. He died in a duel at the age of 21. The story goes that he wrote down major parts of his theory during the last night before the duel.

There are two types of problems that can occur in data transmission or when reading data from a storage medium. First, data can be erased, meaning that when reading (or receiving) it one realizes that it is missing. Second, data can contain errors. The second type of problem is more severe because it is not even known where in a data stream the errors occurred. A good error-correcting scheme can handle both problems.

Definition 5.35. A (n, k)-*encoding function* E for some alphabet $\mathcal{A}$ is an injective function that maps a list $(a_0, \ldots, a_{k-1}) \in \mathcal{A}^k$ of k (information) symbols to a list $(c_0, \ldots, c_{n-1}) \in \mathcal{A}^n$ of $n > k$ (encoded) symbols in $\mathcal{A}$, called *codeword*:

$$E: \mathcal{A}^k \to \mathcal{A}^n : (a_0, \ldots, a_{k-1}) \mapsto E((a_0, \ldots, a_{k-1})) = (c_0, \ldots, c_{n-1}).$$

For an encoding function E one often consider the set

$$\mathcal{C} = \mathsf{Im}(E) = \{E((a_0, \ldots, a_{k-1})) \mid a_0, \ldots, a_{k-1} \in \mathcal{A}\}$$

of codewords, which is called an error-correcting code.

Definition 5.36. An (n, k)-*error-correcting code* over the alphabet $\mathcal{A}$ with $|\mathcal{A}| = q$ is a subset of $\mathcal{A}^n$ of cardinality q^k.

It is natural to use as the alphabet $\mathcal{A} = \{0, 1\}$, i.e., to take bits as the basic unit of information. However, for several reasons (one being the efficiency of encoding and in particular decoding), one often considers larger units of information, for example bytes (i.e., $\mathcal{A} = \{0, 1\}^8$).

Definition 5.37. The *Hamming distance* between two strings of equal length over a finite alphabet $\mathcal{A}$ is the number of positions at which the two strings differ.

Definition 5.38. The *minimum distance* of an error-correcting code $\mathcal{C}$, denoted $d_{\min}(\mathcal{C})$, is the minimum of the *Hamming distance* between any two codewords.

Example 5.66. The following code is a $(5, 2)$-code over the alphabet $\{0, 1\}$:

$$\{(0, 0, 0, 0, 0),\ (1, 1, 1, 0, 0),\ (0, 0, 1, 1, 1),\ (1, 1, 0, 1, 1)\}.$$

The minimum distance is 3.

5.9.2 Decoding

Definition 5.39. A *decoding function* D for an (n, k)-encoding function is a function $D: \mathcal{A}^n \to \mathcal{A}^k$.

The idea is that a good decoding function takes an arbitrary list $(r_0, \ldots, r_{n-1}) \in \mathcal{A}^n$ of symbols[49] and decodes it to the most plausible (in some sense) information vector $(a_0, \ldots, a_{k-1})$. Moreover, a good decoding function should be efficiently computable.

The error-correcting capability of a code $\mathcal{C}$ can be characterized in terms of the number t of errors that can be corrected. More precisely:

Definition 5.40. A decoding function D is *t-error correcting* for encoding function E if for any $(a_0, \ldots, a_{k-1})$

$$D\big((r_0, \ldots, r_{n-1})\big) = (a_0, \ldots, a_{k-1})$$

for any $(r_0, \ldots, r_{n-1})$ with Hamming distance at most t from $E\big((a_0, \ldots, a_{k-1})\big)$. A code $\mathcal{C}$ is *t-error correcting* if there exists E and D with $\mathcal{C} = \mathsf{Im}(E)$ where D is t-error correcting.

Theorem 5.41. *A code $\mathcal{C}$ with minimum distance d is t-error correcting if and only if $d \geq 2t + 1$.*

Proof. ($\Longleftarrow$) If any two codewords have Hamming distance at least $2t + 1$ (i.e., differ in at least $2t+1$ positions), then it is impossible that a word $(r_0, \ldots, r_{n-1}) \in \mathcal{A}^n$ could result from two different codewords by changing t positions. Thus if $(r_0, \ldots, r_{n-1})$ has distance at most t from a codeword $(c_0, \ldots, c_{n-1})$, then this codeword is uniquely determined. The decoding function D can be defined to decode to (one of) the nearest codeword(s) (more precisely, to the information resulting (by E) in that codeword).
($\Longrightarrow$) If there are two codewords that differ in at most $2t$ positions, then there exists a word $(r_0, \ldots, r_{n-1})$ which differs from both codewords in at most t positions; hence it is possible that t errors can not be corrected. $\square$

Example 5.67. A code with minimum distance $d = 5$ can correct $t = 2$ errors. The code in Example 5.66 can correct a single error ($t = 1$).

5.9.3 Codes based on Polynomial Evaluation

A very powerful class of codes is obtained by polynomial interpolation if $\mathcal{A}$ has a field structure, i.e., $\mathcal{A} = \mathrm{GF}(q)$ for some q:

[49] For example the list of symbols received after transmission of a codeword over a noisy channel or read from a storage medium like a CD.

Theorem 5.42. *Let $\mathcal{A} = \mathrm{GF}(q)$ and let $\alpha_0, \ldots, \alpha_{n-1}$ be arbitrary distinct elements of* $\mathrm{GF}(q)$. *Consider the encoding function*

$$E((a_0, \ldots, a_{k-1})) = \big(a(\alpha_0), \ldots, a(\alpha_{n-1})\big),$$

where $a(x)$ is the polynomial

$$a(x) = a_{k-1}x^{k-1} + \cdots + a_1 x + a_0.$$

This code has minimum distance $n - k + 1$.

Proof. The polynomial $a(x)$ of degree $k-1$ can be interpolated from any k values, i.e., from any k codeword symbols. If two polynomials agree for k arguments (or, equivalently, if two codewords agree at k positions), then they are equal. This means that two *different* codewords cannot agree at k positions. Hence any two codewords disagree in at least $n - k + 1$ positions. □

An (n, k)-code over the field $\mathrm{GF}(2^d)$ can be interpreted as a binary (dn, dk)-code (over $\mathrm{GF}(2)$). The minimum distance of this binary code is at least that of the original code because two different $\mathrm{GF}(2^d)$-symbols must differ in at least one bit (but can of course differ in more than one bit).

Example 5.68. Polynomial codes as described are used for storing information on Compact Discs. In fact, the coding scheme of CD's makes use of two different such codes, but explaining the complete scheme is beyond the scope of this course on discrete mathematics. The field is $\mathrm{GF}(2^8)$ defined by an irreducible polynomial of degree 8 over $\mathrm{GF}(2)$ and the two codes are a $(32, 28)$-code over $\mathrm{GF}(2^8)$ and a $(28, 24)$-code over $\mathrm{GF}(2^8)$, both with minimum distance 5.

Chapter 6

Logic

6.1 Introduction

In Chapter 2 we have introduced some basic concepts of logic, but the treatment was quite informal. In this chapter we discuss the foundations of logic in a mathematically rigorous manner. In particular, we clearly distinguish between the syntax and the semantics of a logic and between syntactic derivations of formulas and logical consequences they imply. We also introduce the concept of a logical calculus and define soundness and completeness of a calculus. Moreover, we discuss in detail a concrete calculus for propositional logic, the so-called resolution calculus.

At a very general level, the goal of logic is to provide a framework for expressing mathematical statements and for expressing and verifying proofs for such statements. A more ambitious, secondary goal can be to provide tools for *automatically* or semi-automatically generating a proof for a given statement.

A treatment of logic usually begins with a chapter on propositional logic[1] (see Section 6.5), followed by a chapter on predicate (or first-order) logic[2] (see Section 6.6), which can be seen as an extension of propositional logic. There are several other logics which are useful in Computer Science and in mathematics, including temporal logic, modal logic, intuitionistic logic, and logics for reasoning about knowledge and about uncertainty. Most if not all relevant logics contain the logical operators from propositional logic, i.e., $\wedge, \vee, \neg$ (and the derived operators $\rightarrow$ and $\leftrightarrow$), as well as the quantifiers ($\forall$ and $\exists$) from predicate logic.

Our goal is to present the general concepts that apply to all types of logics in a unified manner, and then to discuss the specific instantiations of these

[1] German: Aussagenlogik
[2] German: Prädikatenlogik

concepts for each logic individually. Therefore we begin with such a general treatment (see Sections 6.2, 6.3, and 6.4) before discussing propositional and predicate logic. From a didactic viewpoint, however, it will be useful to switch back and forth between the generic concepts of Sections 6.2, 6.3, and 6.4 and the concrete instantiations of Sections 6.5 and 6.6.

We give a general warning: Different treatments of logic often use slightly or sometimes substantially different notation.[3] Even at the conceptual level there are significant differences. One needs to be prepared to adopt a particular notation used in a particular application context. However, the general principles explained here are essentially standard.

We also refer to the book by Kreuzer and Kühling and that by Schöning mentioned in the preface of these lecture notes.

6.2 Proof Systems

6.2.1 Definition

In a formal treatment of mathematics, all objects of study must be described in a well-defined syntax. Typically, syntactic objects are finite strings over some alphabet Σ, for example the symbols allowed by the syntax of a logic or simply the alphabet $\{0,1\}$, in which case syntactic objects are bit-strings. Recall that Σ^* denotes the set of finite strings of symbols from Σ.

In this section, the two types of mathematical objects we study are

- *mathematical statements* of a certain type and
- *proofs* for this type of statements.

By a statement type we mean for example the class of statements of the form that a given number n is prime, or the class of statements of the form that a given graph G has a Hamiltonian cycle (see below), or the class of statements of the form that a given formula F in propositional logic is satisfiable.

Consider a fixed type of statements. Let $\mathcal{S} \subseteq \Sigma^*$ be the set of (syntactic representations of) mathematical statements of this type, and let $\mathcal{P} \subseteq \Sigma^*$ be the set of (syntactic representations of) proof strings.[4]

Every statement $s \in \mathcal{S}$ is either true or false. The *truth function*

$$\tau : \mathcal{S} \to \{0,1\}$$

[3] For example, in some treatments the symbol $\Rightarrow$ is used for $\rightarrow$, which can be confusing.

[4] Membership in $\mathcal{S}$ and also in $\mathcal{P}$ is assumed to be efficiently checkable (for some notion of efficiency).

assigns to each $s \in \mathcal{S}$ its truth value $\tau(s)$. This function τ defines the meaning, called the *semantics*, of objects in $\mathcal{S}$.[5]

An element $p \in \mathcal{P}$ is either a (valid) proof for a statement $s \in \mathcal{S}$, or it is not. This can be defined via a *verification function*

$$\phi : \mathcal{S} \times \mathcal{P} \to \{0,1\},$$

where $\phi(s,p) = 1$ means that p is a valid proof for statement s.

Without strong loss of generality we can in this section consider

$$\mathcal{S} = \mathcal{P} = \{0,1\}^*,$$

with the understanding that any string in $\{0,1\}^*$ can be interpreted as a statement by defining syntactically wrong statements as being false statements.

Definition 6.1. A *proof system*[6] is a quadruple $\Pi = (\mathcal{S}, \mathcal{P}, \tau, \phi)$, as above.

We now discuss the two fundamental requirements for proof systems.

Definition 6.2. A proof system $\Pi = (\mathcal{S}, \mathcal{P}, \tau, \phi)$ is *sound*[7] if no false statement has a proof, i.e., if for all $s \in \mathcal{S}$ for which there exists $p \in \mathcal{P}$ with $\phi(s,p) = 1$, we have $\tau(s) = 1$.

Definition 6.3. A proof system $\Pi = (\mathcal{S}, \mathcal{P}, \tau, \phi)$ is *complete*[8] if every true statement has a proof, i.e., if for all $s \in \mathcal{S}$ with $\tau(s) = 1$, there exists $p \in \mathcal{P}$ with $\phi(s,p) = 1$.

In addition to soundness and completeness, one requires that the function ϕ be *efficiently computable* (for some notion of efficiency).[9] We will not make this formal, but it is obvious that a proof system is useless if proof verification is computationally infeasible. Since the verification has to generally examine the entire proof, the length of the proof cannot be infeasibly long.[10]

[5] In the context of logic discussed from the next section onwards, the term semantics is used in a specific restricted manner that is compatible with its use here.

[6] The term proof system is also used in different ways in the mathematical literature.

[7] German: korrekt

[8] German: vollständig

[9] The usual efficiency notion in Computer Science is so-called *polynomial-time computable* which we do not discuss further.

[10] An interesting notion introduced in 1998 by Arora et al. is that of a *probabilistically checkable proof (PCP)*. The idea is that the proof can be very long (i.e., exponentially long), but that the verification only examines a very small random selection of the bits of the proof and nevertheless can decide correctness, except with very small error probability.

6.2.2 Examples

Example 6.1. An undirected *graph* consists of a set V of nodes and a set E of edges between nodes. Suppose that $V = \{0, \ldots, n-1\}$. A graph can then be described by the so-called *adjacency matrix*, an $n \times n$-matrix M with $\{0,1\}$-entries, where $M_{i,j} = 1$ if and only if there is an edge between nodes i and j. A graph with n nodes can hence be represented by a bit-string of length n^2, by reading out the entries of the matrix row by row.

We are now interested in proving that a given graph has a so-called *Hamiltonian cycle*, i.e., that there is a closed path from node 1 back to node 1, following edges between nodes, and visiting every node exactly once. We are also interested in the problem of proving the negation of this statement, i.e., that a given graph has *no* Hamiltonian cycle. Deciding whether or not a given graph has a Hamiltonian cycle is considered a computationally very hard decision problem (for large graphs).[11]

To prove that a graph has a Hamiltonian cycle, one can simply provide the sequence of nodes visited by the cycle. A value in $V = \{0, \ldots, n-1\}$ can be represented by a bit-string of length $\lceil \log_2 n \rceil$, and a sequence of n such numbers can hence be represented by a bit-string of length $n \lceil \log_2 n \rceil$. We can hence define $\mathcal{S} = \mathcal{P} = \{0,1\}^*$.

Now we can let τ be the function defined by $\tau(s) = 1$ if and only if $|s| = n^2$ for some n and the n^2 bits of s encode the adjacency matrix of a graph containing a Hamiltonian cycle. If $|s|$ is not a square or if s encodes a graph without a Hamiltonian cycle, then $\tau(s) = 0$.[12] Moreover, we can let ϕ be the function defined by $\phi(s,p) = 1$ if and only if, when s is interpreted as an $n \times n$-matrix M and when p is interpreted as a sequence of n different numbers $(a_1, \ldots, a_n)$ with $a_i \in \{0, \ldots, n-1\}$ (each encoded by a bit-string of length $\lceil \log_2 n \rceil$), then the following is true:

$$M_{a_i, a_{i+1}} = 1$$

for $i = 1, \ldots, n-1$ and

$$M_{a_n, a_1} = 1.$$

This function ϕ is efficiently computable. The proof system is sound because a graph without Hamiltonian cycle has no proof, and it is complete because every graph with a Hamiltonian cycle has a proof. Note that each s with $\tau(s) = 1$ has at least n different proofs because the starting point in the cycle is arbitrary.

Example 6.2. Let us now consider the opposite problem of proving the inexistence of a Hamiltonian cycle in a given graph. In other words, in the above example we define $\tau(s) = 1$ if and only if $|s| = n^2$ for some n and the n^2 bits

[11] The best known algorithm has running time exponential in n. The problem is actually NP-complete, a concept that will be discussed in a later course on theoretical Computer Science.

[12] Note that τ defines the meaning of the strings in $\mathcal{S}$, namely that they are meant to encode graphs and that we are interested in whether a given graph has a Hamiltonian cycle.

of s encode the adjacency matrix of a graph *not* containing Hamiltonian cycle. In this case, no sound and complete proof system (with reasonably short and efficiently verifiable proofs) is known. It is believed that no such proof system exists.

Example 6.3. Let again $\mathcal{S} = \mathcal{P} = \{0,1\}^*$, and for $s \in \{0,1\}^*$ let $n(s)$ denote the natural number whose (standard) binary representation is s, with the convention that leading 0's are ignored. (For example, $n(101011) = 43$ and $n(00101) = 5$.) Now, let τ be the function defined as follows: $\tau(s) = 1$ if and only if $n(s)$ is *not* a prime number. Moreover, let ϕ be the function defined by $\phi(s,p) = 1$ if and only if $n(s) = 0$, or if $n(s) = 1$, or if $n(p)$ divides $n(s)$ and $1 < n(p) < n(s)$. This function ϕ is efficiently computable. This is a proof system for the non-primality (i.e., compositeness) of natural numbers. It is sound because every s corresponding to a prime number $n(s)$ has no proof since $n(s) \neq 0$ and $n(s) \neq 1$ and $n(s)$ has no divisor d satisfying $1 < d < n(s)$. The proof system is complete because every natural number n greater than 1 is either prime or has a prime factor q satisfying $1 < q < n$ (whose binary representation can serve as a proof).

Example 6.4. Let us consider the opposite problem, i.e., proving primality of a number $n(s)$ represented by s. In other words, in the previous example we replace "not a prime" by "a prime". It is far from clear how one can define a verification function ϕ such that the proof system is sound and complete. However, such an efficiently computable function ϕ indeed exists. Very briefly, the proof that a number $n(s)$ (henceforth we simply write n) is prime consists of (adequate representations of):

1) the list $p_1, \ldots, p_k$ of distinct prime factors of $n-1$,
2) a (recursive) proof of primality for each of $p_1, \ldots, p_k$[13]
3) a generator g of the group $\mathbb{Z}_n^*$.

The exact representation of these three parts of the proof would have to be made precise, but we omit this here as it is obvious how this could be done.

The verification of a proof (i.e., the computation of the function ϕ) works as follows.

- If $n = 2$ or $n = 3$, then the verification stops and returns 1.[14]
- It is tested whether $p_1, \ldots, p_k$ all divide $n-1$ and whether $n-1$ can be written as a product of powers of $p_1, \ldots, p_k$ (i.e., whether $n-1$ contains no other prime factor).

[13] recursive means that the same principle is applied to prove the primality of every p_i, and again for every prime factor of $p_i - 1$, etc.

[14] One could also consider a longer list of small primes for which no recursive primality proof is required.

- It is verified that
$$g^{n-1} \equiv_n 1$$
and, for all $i \in \{1, \ldots, k\}$, that
$$g^{(n-1)/p_i} \not\equiv_n 1.$$
(This means that g has order $n-1$ in $\mathbb{Z}_n^*$).
- For every p_i, an analogous proof of its primality is verified (recursively).

This proof system for primality is sound because if n is not a prime, then there is no element of $\mathbb{Z}_n^*$ of order $n-1$ since the order of any element is at most $\varphi(n)$, which is smaller than $n-1$. The proof system is complete because if n is prime, then $GF(n)$ is a finite field and the multiplicative group of any finite field, i.e., $\mathbb{Z}_n^*$, is cyclic and has a generator g. (We did not prove this statement in this course.)[15]

6.2.3 Discussion

The examples demonstrate the following important points:

- While proof verification must be efficient (in some sense not defined here), *proof generation* is generally not (or at least not known to be) efficient. For example, finding a proof for the Hamiltonian cycle example requires to find such a cycle, a problem that, as mentioned, is believed to be very hard. Similarly, finding a primality proof as discussed would require the factorization of $n-1$, and the factoring problem is believed to be hard. In general, finding a proof (if it exists) is a process requiring insight and ingenuity, and it cannot be efficiently automated.
- A proof system is always restricted to a certain type of mathematical statement. For example, the proof system of Example 6.1 is very limited in the sense that it only allows to prove statements of the form "graph G has a Hamiltonian cycle".
- Proof verification can in principle proceed in very different ways. The proof verification method of logic, based on checking a sequence of rule applications, is (only) a special case.
- Asymmetry of statements and their negation: Even if a proof system exists for a certain type of statements, it is quite possible that for the negation of the statements, no proof system (with efficient verification) exists.

[15]Actually, a quite efficient deterministic primality test was recently discovered by Agrawal et al., and this means that primality can be checked without a proof. In other words, there exists a trivial proof system for primality with empty proofs. However, this fact is mathematically considerably more involved than the arguments presented here for the soundness and completeness of the proof system for primality.

6.2.4 Proof Systems in Theoretical Computer Science *

The concept of a proof system appears in a more concrete form in theoretical computer science (TCS), as follows. Statements and proofs are bit-strings, i.e., $\mathcal{S} = \mathcal{P} = \{0,1\}^*$. The predicate τ defines the set $L \subseteq \{0,1\}^*$ of strings that correspond to true statements:
$$L = \{s \mid \tau(s) = 1\}.$$
Conversely, every subset $L \subseteq \{0,1\}^*$ defines a predicate τ. In TCS, such a set L of strings is called a *formal language*, and one considers the problem of proving that a given string s is in the language, i.e., $s \in L$. A proof for $s \in L$ is called a *witness* of s, often denoted as w, and the verification function $\phi(s, w)$ defines whether a string w is a witness for $s \in L$.

One then considers the special case where the length of w is bounded by a polynomial of the length of s and where the function ϕ must be computable in polynomial time, i.e., by a program with worst-case running time polynomial in the length of s. Then, the important class NP of languages is the set of languages for which such a polynomial-time computable verification function exists.

As mentioned in a footnote, a type of proof system of special interest are so-called *probabilistically checkable proofs (PCP)*.

An important extension of the concept of proof systems are so-called *interactive proofs*.[16] In such a system, the proof is not a bit-string, but it consists of an interaction (a protocol) between the prover and the verifier, where one tolerates an immensely small (e.g. exponentially small) probability that a verifier accepts a "proof" for a false statement. The reason for considering such interactive proofs are:

- Such interactive proofs can exist for statements for which a classical (non-interactive) proof does not exist. For example, there exists an interactive proof system for the *non*-Hamiltonicity of graphs.
- Such interactive proofs can have a special property, called *zero-knowledge*, which means that the verifier learns absolutely nothing (in a well-defined sense) during the protocol, except that the statement is true. In particular, the verifier cannot prove the statement to somebody else.
- Zero-knowledge proofs (especially non-interactive versions, so-called NIZK's) are of crucial importance in a large number of applications, for example in sophisticated block-chain systems.

6.3 Elementary General Concepts in Logic

The purpose of this section is to introduce the most basic concepts in logic in a general manner, not specific to a particular logic. However, this section is best appreciated by considering concrete examples of logics, in particular propositional logic and predicate logic. Without discussing such examples in parallel to introducing the concepts, this section will be hard to appreciate. We will discuss the general concepts and the concrete examples in parallel, going back and forth between Section 6.3 and Sections 6.5 and 6.6.

[16]This topic is discussed in detail in the Master-level course *Cryptographic Protocols* taught by Martin Hirt and Ueli Maurer.

6.3.1 The General Goal of Logic

A goal of logic is to provide a specific proof system $\Pi = (\mathcal{S}, \mathcal{P}, \tau, \phi)$ for which a very large class of mathematical statements can be expressed as an element of $\mathcal{S}$.

However, such a proof system $\Pi = (\mathcal{S}, \mathcal{P}, \tau, \phi)$ can never capture *all* possible mathematical statements. For example, it usually does not allow to capture (self-referential) statements about Π, such as "Π is complete", as an element of $\mathcal{S}$. The use of common language is therefore unavoidable in mathematics and logic (see also Section 6.7).

In logic, an element $s \in \mathcal{S}$ consists of one or more formulas (e.g. a formula, or a set of formulas, or a set of formulas and a formula), and a proof consists of applying a certain sequence of syntactic steps, called a *derivation* or a *deduction*. Each step consists of applying one of a set of allowed syntactic rules, and the set of allowed rules is called a *calculus*. A rule generally has some place-holders that must be instantiated by concrete values.

In standard treatments of logic, the syntax of $\mathcal{S}$ and the semantics (the function τ) are carefully defined. In contrast, the function ϕ, which consists of verifying the correctness of each rule application step, is not completely explicitly defined. One only defines rules, but for example one generally does not define a syntax for expressing how the place-holders of the rules are instantiated.[17]

6.3.2 Syntax, Semantics, Interpretation, Model

6.3.3 Syntax

A *logic* is defined by the *syntax* and the *semantics*. The basic concept in any logic is that of a *formula*[18].

Definition 6.4. The *syntax* of a logic defines an alphabet Λ (of allowed symbols) and specifies which strings in Λ^* are formulas (i.e., are syntactically correct).

The semantics (see below) defines under which "conditions" a formula is *true* (denoted as 1) or *false* (denoted as 0).[19] What we mean by "conditions" needs to be made more precise and requires a few definitions.

Some of the symbols in Λ (e.g. the symbols A and B in propositional logic or the symbols P and Q in predicate logic) are understood as variables, each of which can take on a value in a certain domain associated to the symbol.

[17] In a fully computerized system, this must of course be (and indeed is) defined.
[18] German: Formel
[19] There are logics (not considered here) with more than two truth values, for example a logic with confidence or belief values indicating the degree of confidence in the truth of a statement.

6.3.4 Semantics

Definition 6.5. The *semantics* of a logic defines (among other things, see below) a function *free* which assigns to each formula $F = (f_1, f_2, \ldots, f_k) \in \Lambda^*$ a subset $\mathit{free}(F) \subseteq \{1, \ldots, k\}$ of the indices. If $i \in \mathit{free}(F)$, then the symbol f_i is said to occur *free* in F.[20]

The same symbol $\beta \in \Lambda$ can occur free in one place of F (say $f_3 = \beta$ where $3 \in \mathit{free}(F)$) and not free in another place (say $f_5 = \beta$ where $5 \notin \mathit{free}(F)$).

The free symbols of a formula denote kind of variables which need to be assigned fixed values in their respective associated domains before the formula has a truth value. This assignment of values is called an interpretation:

Definition 6.6. An *interpretation* consists of a set $\mathcal{Z} \subseteq \Lambda$ of symbols of Λ, a domain (a set of possible values) for each symbol in $\mathcal{Z}$, and a function that assigns to each symbol in $\mathcal{Z}$ a value in its associated domain.[21]

Often (but not in propositional logic), the domains are defined in terms of a so-called *universe* U, and the domain for a symbol in Λ can for example be U, or a function $U^k \to U$ (for some k), or a function $U^k \to \{0, 1\}$ (for some k).

Definition 6.7. An interpretation is *suitable*[22] for a formula F if it assigns a value to all symbols $\beta \in \Lambda$ occurring free in F.[23]

Definition 6.8. The *semantics* of a logic also defines a function[24] σ assigning to each formula F, and each interpretation $\mathcal{A}$ suitable for F, a truth value $\sigma(F, \mathcal{A})$ in $\{0, 1\}$.[25] In treatments of logic one often writes $\mathcal{A}(F)$ instead of $\sigma(F, \mathcal{A})$ and calls $\mathcal{A}(F)$ *the truth value of* F *under interpretation* $\mathcal{A}$.[26]

[20] The term "free" is not standard in the literature which instead uses special terms for each specific logic, but as we see later it coincides for the notion of free variables in predicate logic.
[21] There may be restrictions for what is an allowed interpretation.
[22] German: passend
[23] A suitable interpretation can also assign values to symbols $\beta \in \Lambda$ not occurring free in F.
[24] We assume that the set of formulas and the set of interpretations are well-defined.
[25] Note that different free occurrences of a symbol $\beta \in \Lambda$ in F are assigned the same value, namely that determined by the interpretation.
[26] This notation in the literature is unfortunately a bit ambiguous since $\mathcal{A}$ is used for two different things, namely for an interpretation as well as for the function induced by the interpretation which assigns to every formula the truth value (under that interpretation). We nevertheless use the notation $\mathcal{A}(F)$ instead of $\sigma(F, \mathcal{A})$ in order to be compatible with most of the literature.

Definition 6.9. A (suitable) interpretation $\mathcal{A}$ for which a formula F is true, (i.e., $\mathcal{A}(F) = 1$) is called a *model* for F, and one also writes

$$\mathcal{A} \models F.$$

More generally, for a set M of formulas, a (suitable) interpretation for which all formulas in M are true is called a model for M, denoted as

$$\mathcal{A} \models M.$$

If $\mathcal{A}$ is not a model for M one writes $\mathcal{A} \not\models M$.

6.3.5 Connection to Proof Systems *

We now explain how the semantics of a logic (the function σ in Definition 6.8) is connected to the semantics of a proof systems (the function τ in Definition 6.1).

First we should remark that one can treat logic in a similarly informal manner as one treats other fields of mathematics. There can be variations on how much is formalized in the sense of proof systems. Concretely, there are the following two options for formalizing a logic:

- In addition to formulas, also interpretations are considered to be formal objects, i.e., there is a syntax for writing (at least certain types of) interpretations. In this case, statements can correspond to pairs $(F, \mathcal{A})$, and the function σ corresponds to the function τ (in the sense of proof systems).
- Only formulas are formal objects and interpretations are treated informally, e.g. in words or some other informal notation. This is the typical approach in treatments of logic (also in this course). This makes perfect sense if the formal statements one wants to prove only refer to formulas, and not to specific interpretations. Indeed, many statements about formulas are of this form, for example the statement that a formula F is a tautology, the statement that F is satisfiable (or unsatisfiable), or the statement that a formula G is a logical consequence of a formula F, i.e., $F \models G$. Note that to prove such statements it is not necessary to formalize interpretations.

6.3.6 Satisfiability, Tautology, Consequence, Equivalence

Definition 6.10. A formula F (or set M of formulas) is called *satisfiable*[27] if there exists a model for F (or M),[28] and *unsatisfiable* otherwise. The symbol $\perp$ is used for an unsatisfiable formula.[29]

[27]German: erfüllbar

[28]Note that the statement that M is satisfiable is *not* equivalent to the statement that every formula in M is satisfiable.

[29]The symbol $\perp$ is not a formula itself, i.e., it is not part of the syntax of a logic, but if used in expressions like $F \equiv \perp$ it is to be understood as standing for an arbitrary unsatisfiable formula. For example, $F \equiv \perp$ means that F is unsatisfiable.

Definition 6.11. A formula F is called a *tautology*[30] or *valid*[31] if it is true for every suitable interpretation. The symbol $\top$ is used for a tautology.

The symbol $\perp$ is sometimes called *falsum*, and $\top$ is sometimes called *verum*.

Definition 6.12. A formula G is a *logical consequence*[32] of a formula F (or a set M of formulas), denoted

$$F \models G \qquad (\text{or} \quad M \models G\ ),$$

if every interpretation suitable for both F (or M) and G, which is a model for F (for M), is also a model for G.[33]

Definition 6.13. Two formulas F and G are *equivalent*, denoted $F \equiv G$, if every interpretation suitable for both F and G yields the same truth value for F and G, i.e., if each one is a logical consequence of the other:

$$F \equiv G \quad \overset{\text{def}}{\iff} \quad F \models G \ \text{ and } \ G \models F.$$

A set M of formulas can be interpreted as the conjunction (AND) of all formulas in M since an interpretation is a model for M if and only if it is a model for *all* formulas in M.[34] If M is the empty set, then, by definition, every interpretation is a model for M, i.e., the empty set of formulas corresponds to a tautology.

Definition 6.14. If F is a tautology, one also writes $\models F$.

That F is unsatisfiable can be written as $F \models \perp$.

6.3.7 The Logical Operators $\wedge$, $\vee$, and $\neg$

Essentially all logics contain the following recursive definitions as part of the syntax definition.

Definition 6.15. If F and G are formulas, then also $\neg F$, $(F \wedge G)$, and $(F \vee G)$ are formulas.

[30]German: Tautologie

[31]German: gültig, allgemeingültig

[32]German: (logische) Folgerung, logische Konsequenz

[33]The symbol $\models$ is used in two slightly different ways: with a formula (or set of formulas), and also with an interpretation on the left side. This makes sense because one can consider a set M of formulas as defining a set of interpretations, namely the set of models for M.

[34]More formally, let G be any formula (one of the many equivalent ones) that corresponds to the conjunction of all formulas in M. Then $M \models F$ if and only if $G \models F$.

A formula of the form $(F \wedge G)$ is called a conjunction, and a formula of the form $(F \vee G)$ is called a disjunction.

We introduce some notational conventions for the use of parentheses. The outermost parentheses of a formula can be dropped, i.e., we can write $F \wedge G$ instead of $(F \wedge G)$. Moreover, parentheses not needed because of associativity of $\wedge$ or $\vee$ (which is actually a consequence of the semantics defined below) can also be dropped.

The implication introduced in Section 2.3 can be understood simply as a notational convention: $F \to G$ stands for $\neg F \vee G$.[35] Similarly, the symbol $F \leftrightarrow G$ stands for $(F \wedge G) \vee (\neg F \wedge \neg G)$.

The semantics of the logical operators $\wedge$, $\vee$, and $\neg$ is defined as follows (in any logic where these operators exist):

Definition 6.16.

$\mathcal{A}((F \wedge G)) = 1$ if and only if $\mathcal{A}(F) = 1$ and $\mathcal{A}(G) = 1$.

$\mathcal{A}((F \vee G)) = 1$ if and only if $\mathcal{A}(F) = 1$ or $\mathcal{A}(G) = 1$.

$\mathcal{A}(\neg F) = 1$ if and only if $\mathcal{A}(F) = 0$.

Some basic equivalences were already discussed in Section 2.3.2 and are now stated for any logic that includes the logical operators $\wedge$, $\vee$, and $\neg$:

Lemma 6.1. *For any formulas F, G, and H we have*

1) $F \wedge F \equiv F$ *and* $F \vee F \equiv F$ *(idempotence);*
2) $F \wedge G \equiv G \wedge F$ *and* $F \vee G \equiv G \vee F$ *(commutativity);*
3) $(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$ *and* $(F \vee G) \vee H \equiv F \vee (G \vee H)$ *(associativity);*
4) $F \wedge (F \vee G) \equiv F$ *and* $F \vee (F \wedge G) \equiv F$ *(absorption);*
5) $F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$ *(distributive law);*
6) $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$ *(distributive law);*
7) $\neg\neg F \equiv F$ *(double negation);*
8) $\neg(F \wedge G) \equiv \neg F \vee \neg G$ *and* $\neg(F \vee G) \equiv \neg F \wedge \neg G$ *(de Morgan's rules);*
9) $F \vee \top \equiv \top$ *and* $F \wedge \top \equiv F$ *(tautology rules);*
10) $F \vee \bot \equiv F$ *and* $F \wedge \bot \equiv \bot$ *(unsatisfiability rules).*
11) $F \vee \neg F \equiv \top$ *and* $F \wedge \neg F \equiv \bot$.

Proof. The proofs follow directly from Definition 6.16. For example, the claim

[35] Alternatively, one could also define $\to$ to be a symbol of the syntax, in which case one would also need to extend the semantics to provide an interpretation for $\to$. This subtle distinction between notational convention or syntax extension is not really relevant for us. We can simply use the symbol $\to$.

$\neg(F \wedge G) \equiv \neg F \vee \neg G$ follows from the fact that for any suitable interpretation, we have $\mathcal{A}(\neg(F \wedge G)) = 1$ if and only if $\mathcal{A}(F \wedge G) = 0$, and hence if and only if either $\mathcal{A}(F) = 0$ or $\mathcal{A}(G) = 0$, i.e., if and only if either $\mathcal{A}(\neg F) = 1$ or $\mathcal{A}(\neg G) = 1$, and hence if and only if $\mathcal{A}(\neg F \vee \neg G) = 1$. □

6.3.8 Logical Consequence vs. Unsatisfiability

We state the following facts without proofs, which are rather obvious. These lemmas are needed for example to make use of the resolution calculus (see Section 6.5.5), which allows to prove the unsatisfiability of a set of formulas, to also be able to prove that a formula F is a tautology, or to prove that a formula G is logical consequence of a given set $\{F_1, F_2, \ldots, F_k\}$ of formulas.

Lemma 6.2. *A formula F is a tautology if and only if $\neg F$ is unsatisfiable.*

Lemma 6.3. *The following three statements are equivalent:*

1. $\{F_1, F_2, \ldots, F_k\} \models G$,
2. $(F_1 \wedge F_2 \wedge \cdots \wedge F_k) \to G$ *is a tautology,*
3. $\{F_1, F_2, \ldots, F_k, \neg G\}$ *is unsatisfiable.*

6.3.9 Theorems and Theories

We can consider at least four types of statements one may want to prove in the context of using a logic:

1. Theorems in an axiomatically defined theory (see below),
2. Statements about a formula or a set of formulas, for example that F is satisfiable or that a set M of formulas is unsatisfiable.
3. The statement $\mathcal{A} \models F$ for a given interpretation $\mathcal{A}$ and a formula F.
4. Statements *about* the logic, for example that a certain calculus for the logic is sound.

To describe the first type of statements, consider a fixed logic, for instance predicate logic discussed in Section 6.6, and consider a set T of formulas, where the formulas in T are called the *axioms* of the theory. Any formula F for which

$$T \models F$$

is called a *theorem* in theory T. For example, the axioms of group theory are three formulas in predicate logic, and any theorem in group theory (e.g. Lagrange's theorem) is a logical consequence of the axioms.

Consider two theories T and T', where T' contains all the axioms of T plus one or more additional axioms. Then every theorem in T is also a theorem in T' (but not vice versa). In the special case where $T = \emptyset$, a theorem in $T = \emptyset$ is a tautology in the logic. Tautologies are useful because they are theorems in any theory, i.e., for any set of axioms.

Example 6.5. The formula $\neg\exists x\forall y\ \big(P(y,x) \leftrightarrow \neg P(y,y)\big)$ is a tautology in predicate logic, as proved in Section 6.6.9.

6.4 Logical Calculi

6.4.1 Introduction

As mentioned in Section 6.3.1, the goal of logic is to provide a framework for expressing and verifying proofs of mathematical statements. A proof of a theorem should be a purely syntactic derivation consisting of simple and easily verifiable steps. In each step, a new syntactic object (typically a formula, but it can also be a more general object involving formulas) is derived by application of a derivation rule or inference rule, and at the end of the derivation, the desired theorem appears. The syntactic verification of a proof does not require any intelligence or "reasoning between the lines", and it can in particular be performed by a computer.

Checking a proof hence simply means to execute a program. Like in computer programming, where the execution of a program is a dumb process while the design of a program is generally an intelligent, sometimes ingenious process, the verification of a proof should be a dumb process while devising a proof is an intelligent, creative, and sometimes ingenious process.

A well-defined set of rules for manipulating formulas (the syntactic objects) is called a *calculus*. Many such calculi have been proposed, and they differ in various ways, for example in the syntax, the semantics, the expressiveness, how easy or involved it is to write a proof, and how long a proof will be.

When defining a calculus, there is a trade-off between simplicity (e.g. a small number of rules) and versatility. For a small set of rules, proving even simple logical steps (like the substitution of a sub-formula by an equivalent sub-formula) can take a very large number of steps in the calculus.

It is beyond the scope of this course to provide an extensive treatment of various logical calculi.

6.4.2 Hilbert-Style Calculi

As mentioned, there are different types of logical calculi. For the perhaps most intuitive type of calculus, the syntactic objects that are manipulated are formulas. This is sometimes called a Hilbert-style calculus. There is also another type of calculi, often called *sequent calculi* (which we will not discuss in this course), where the syntactic objects are more complex objects than just formulas. The following refers to Hilbert-style calculi.

Definition 6.17. A *derivation rule* or *inference rule*[36] is a rule for deriving a formula from a set of formulas (called the precondition or premises). We write

$$\{F_1,\ldots,F_k\}\ \vdash_R\ G$$

if G can be derived from the set $\{F_1,\ldots,F_k\}$ by rule R.[37]

The derivation rule $\{F_1,\ldots,F_k\} \vdash_R G$ is often also written as

$$\frac{F_1\ F_2\ \cdots\ F_k}{G}\quad (R),$$

where spaces separate the formulas above the bar.

Derivation is a purely syntactic concept. Derivation rules apply to syntactically correct (sets of) formulas. Some derivation rules (e.g. resolution, see Section 6.5.5) require the formulas to be in a specific format.

Typically such a derivation rule is defined as a rule that involves *place-holders* for formulas (such as F and G), which can be instantiated with any concrete formulas. In order to apply such a rule one must instantiate each place-holder with a concrete formula.

Definition 6.18. (Informal.) The *application of a derivation rule R to a set M of formulas* means

1. Select a subset N of M.
2. For the place-holders in R: specify formulas that appear in N such that $N \vdash_R G$ for a formula G.
3. Add G to the set M (i.e., replace M by $M \cup \{G\}$).

Example 6.6. Two derivation rules for propositional and predicate logic are

$$\{F\wedge G\}\ \vdash\ F \qquad \text{and} \qquad \{F,G\}\ \vdash\ F\wedge G$$

[36] German: Schlussregel

[37] Formally, a derivation rule is a relation from the power set of the set of formulas to the set of formulas.

The left rule states that if one has already derived a formula of the form $F \wedge G$, where F and G are arbitrary formulas, then one can derive F. The second rule states that for any two formulas F and G that have been derived, one can also derive the formula $F \wedge G$. For example, an application of the right rule yields

$$\{A \vee B,\ C \vee D\} \ \vdash\ (A \vee B) \wedge (C \vee D),$$

where F is instantiated as $A \vee B$ and G is instantiated as $C \vee D$. More rules are discussed in Section 6.4.4.

Definition 6.19. A (logical) *calculus*[38] K is a finite set of derivation rules: $K = \{R_1, \ldots, R_m\}$.

Definition 6.20. A *derivation*[39] of a formula G from a set M of formulas in a calculus K is a finite sequence (of some length n) of applications of rules in K (see Def. 6.18), leading to G. More precisely, we have

- $M_0 := M$,
- $M_i := M_{i-1} \cup \{G_i\}$ for $1 \leq i \leq n$, where $N \ \vdash_{R_j}\ G_i$ for some $N \subseteq M_{i-1}$ and for some $R_j \in K$, and where
- $G_n = G$.

We write

$$M \ \vdash_K\ G$$

if there is a derivation of G from M in the calculus K.

The above treatment of syntactic derivation is not completely general. In some contexts (e.g. in so-called Natural Deduction for predicate logic, which is a so-called sequent calculus), one needs to keep track not only of the derived formulas, but also of the history of the derivation, i.e., the derivation steps that have led to a given formula.

6.4.3 Soundness and Completeness of a Calculus

A main goal of logic is to formalize reasoning and proofs. One wants to perform purely syntactic manipulations on given formulas, defined by a calculus, to arrive at a new formula which is a logical consequence. In other words, if we use a calculus, the syntactic concept of derivation (using the calculus) should be related to the semantic concept of logical consequence.

[38]German: Kalkül
[39]German: Herleitung

Definition 6.21. A derivation rule R is *correct* if for every set M of formulas and every formula F, $M \ \vdash_R F$ implies $M \models F$:

$$M \ \vdash_R F \ \Longrightarrow\ M \models F.$$

Example 6.7. The two rules of Example 6.6 are correct, but the rule

$$\{F \to G,\ G \to F\} \ \vdash\ F \wedge G$$

is not correct. To see this, note that if F and G are both false, then $F \to G$ and $G \to F$ are true while $F \wedge G$ is false.

Definition 6.22. A calculus K is *sound*[40] or *correct* if for every set M of formulas and every formula F, if F can be derived from M then F is also a logical consequence of M:

$$M \ \vdash_K F \ \Longrightarrow\ M \models F,$$

and K is *complete*[41] if for every M and F, if F is a logical consequence of M, then F can also be derived from M:

$$M \models F \ \Longrightarrow\ M \ \vdash_K F.$$

A calculus is hence sound and complete if

$$M \ \vdash_K F \ \Longleftrightarrow\ M \models F,$$

i.e., if logical consequence and derivability are identical. Clearly, a calculus is sound if and only if every derivation rule is correct. One writes $\vdash_K F$ if F can be derived in K from the empty set of formulas. Note that if $\vdash_K F$ for a sound calculus, then $\models F$, i.e., F is a tautology.

6.4.4 Some Derivation Rules

In this section we discuss a few derivation rules for propositional logic and any logic which contains propositional logic. We do not provide a complete and compactly defined calculus, just a few rules. For singleton sets of formulas we omit the brackets "{" and "}".

All equivalences, including the basic equivalences of Lemma 6.1, can be used as derivation rules. For example, the following derivation rules are correct:

$$\neg\neg F \ \vdash\ F \qquad F \wedge G \ \vdash\ G \wedge F \qquad \neg(F \vee G) \ \vdash\ \neg F \wedge \neg G$$

Other natural and correct rules, which capture logical consequences, not equivalences, are:

$$F \wedge G \vdash F \qquad F \wedge G \vdash G \qquad \{F, G\} \vdash F \wedge G$$

[40]German: widerspruchsfrei
[41]German: vollständig

$$F \vdash F \vee G \qquad\qquad F \vdash G \vee F$$
$$\{F,\ F \to G\} \vdash G \qquad\qquad \{F \vee G,\ F \to H,\ G \to H\} \vdash H.$$

Such rules are not necessarily independent. For example, the rule $F \wedge G \vdash G \wedge F$ could be derived from the above three rules as follows: F can be derived from $F \wedge G$ and G can also be derived from $F \wedge G$, resulting in the set $\{F \wedge G,\ F,\ G\}$. $\{G, F\}$ is a subset of $\{F \wedge G,\ F,\ G\}$ and hence one of the above rules yields $\{G, F\} \vdash G \wedge F$.

The last rule discussed above captures case distinction (two cases). It states that if one knows that F or G is true and that each of them implies H, then we can conclude H. Such a proof step is in a sense non-constructive because it may not be known which of F or G is true.

To begin a derivation from the empty set of formulas, one can use any rule of the form $\vdash F$, where F is a tautology. The best-known such rule is

$$\vdash\ F \vee \neg F$$

called "tertium non datur (TND)" (in English: "there is no third [alternative]"), which captures the fact that a formula F can only be true or false (in which case $\neg F$ is true); there is no option in between.[42] Another rule for deriving a tautology is

$$\vdash\ \neg(F \leftrightarrow \neg F).$$

Example 6.8. The following rule can be understood as capturing the principle of proof by contradiction. (Why?)

$$\{F \vee G,\ \neg G\}\ \vdash\ F.$$

The reader can prove the correctness as an exercise.

Which set of rules constitutes an adequate calculus is generally not clear, but some calculi have received special attention. One could argue both for a small set of rules (which are considered the fundamental ones from which everything else is derived) or for a large library of rules (so there is a large degree of freedom in finding a short derivation).

6.4.5 Derivations from Assumptions

If in a sound calculus K one can derive G under the assumption F, i.e., one can prove $F \vdash_K G$, then one has proved that $F \to G$ is a tautology, i.e., we have

$$F \vdash_K G \quad \Longrightarrow \quad \models (F \to G).$$

[42] However, in so-called constructive or intuitionistic logic, this rule is not considered correct because its application does not require explicit knowledge of whether F or $\neg F$ is true.

One could therefore also extend the calculus by the new rule

$$\vdash (F \to G),$$

which is sound. Here F and G can be expressions involving place-holders for formulas.

Example 6.9. As a toy example, consider the rules $\neg\neg F \vdash F$ and $\neg(F \vee G) \vdash \neg F$. Let H be an arbitrary formula. Using the second rule (and setting $F = \neg H$) we can obtain $\neg(\neg H \vee G) \vdash \neg\neg H$. Thus, using the first rule (and setting $F = H$) we can obtain $\neg\neg H \vdash H$. Hence we have proved $\neg(\neg H \vee G) \vdash H$. As usual, this holds for arbitrary formulas G and H and hence can be understood as a rule. When stated in the usual form (with place holders F and G, the rule would be stated as $\neg(\neg F \vee G) \vdash F$.

More generally, we can derive a formula G from several assumptions, for example

$$\{F_1, F_2\} \vdash_K G \quad \Longrightarrow \quad \models \big((F_1 \wedge F_2) \to G\big).$$

6.4.6 Connection to Proof Systems *

Let us briefly explain the connection between logical calculi and the general concept of proof systems (Definition 6.2).

In a proof system allowing to prove statements of the form $M \models G$, one can let the set $\mathcal{S}$ of statements be the set of pairs (M, G). One further needs a precise syntax for expressing derivations. Such a syntax would, for example, have to include a way to express how place-holders in rules are instantiated. This aspect of a calculus is usually not made precise and therefore a logical calculus (alone) does not completely constitute a proof system in the sense of Definition 6.2. However, in a computerized system this needs to be made precise, in a language specific to that system, and such computerized system is hence a proof system in the strict sense of Section 6.2.

6.5 Propositional Logic

We also refer to Section 2.3 where some basics of propositional logic were introduced informally and many examples were already given. This section concentrates on the formal aspects and the connection to Section 6.3.

6.5.1 Syntax

Definition 6.23. (Syntax.) An *atomic formula* is a symbol of the form A_i with $i \in \mathbb{N}$.[43] A *formula* is defined as follows, where the second point is a restatement (for convenience) of Definition 6.15:

- An atomic formula is a formula.
- If F and G are formulas, then also $\neg F$, $(F \wedge G)$, and $(F \vee G)$ are formulas.

[43] A_0 is usually not used. This definition guarantees an unbounded supply of atomic formulas,

A formula built according to this inductive definition corresponds naturally to a tree where the leaves correspond to atomic formulas and the inner nodes correspond to the logical operators.

6.5.2 Semantics

Recall Definitions 6.5 and 6.6. *In propositional logic, the free symbols of a formula are all the atomic formulas.* For example, the truth value of the formula $A \wedge B$ is determined only after we specify the truth values of A and B. In propositional logic, an interpretation is called a truth assignment (see below).

Definition 6.24. (Semantics.) For a set Z of atomic formulas, an interpretation $\mathcal{A}$, called *truth assignment*[44], is a function $\mathcal{A} : Z \to \{0,1\}$. A truth assignment $\mathcal{A}$ is suitable for a formula F if Z contains all atomic formulas appearing in F (see Definition 6.7). The semantics (i.e., the truth value $\mathcal{A}(F)$ of a formula F under interpretation $\mathcal{A}$) is defined by $\mathcal{A}(F) = \mathcal{A}(A_i)$ for any atomic formula $F = A_i$, and by Definition 6.16 (restated here for convenience):

$$\begin{array}{ll} \mathcal{A}((F \wedge G)) = 1 & \text{if and only if } \mathcal{A}(F) = 1 \text{ and } \mathcal{A}(G) = 1. \\ \mathcal{A}((F \vee G)) = 1 & \text{if and only if } \mathcal{A}(F) = 1 \text{ or } \mathcal{A}(G) = 1. \\ \mathcal{A}(\neg F) = 1 & \text{if and only if } \mathcal{A}(F) = 0. \end{array}$$

Example 6.10. Consider the formula

$$F = (A \wedge \neg B) \vee (B \wedge \neg C)$$

already discussed in Section 2.3. The truth assignment $\mathcal{A} : Z \to \{0,1\}$ for $Z = \{A, B\}$ that assigns $\mathcal{A}(A) = 0$ and $\mathcal{A}(B) = 1$ is not suitable for F because no truth value is assigned to C, and the truth assignment $\mathcal{A} : Z \to \{0,1\}$ for $Z = \{A, B, C, D\}$ that assigns $\mathcal{A}(A) = 0$, $\mathcal{A}(B) = 1$, $\mathcal{A}(C) = 0$, and $\mathcal{A}(D) = 1$ is suitable and also a model for F. F is satisfiable but not a tautology.

6.5.3 Brief Discussion of General Logic Concepts

We briefly discuss the basic concepts from Section 6.3.6 in the context of propositional logic.

Specializing Definition 6.13 to the case of propositional logic, we confirm Definition 2.6: Two formulas F and G are equivalent if, when both formulas are considered as functions $M \to \{0,1\}$, where M is the union of the atomic formulas of F and G, then the two functions are identical (i.e., have the same function table).

but as a notational convention we can also write $A, B, C, \ldots$ instead of $A_1, A_2, A_3, \ldots$.

[44] German: (Wahrheits-)Belegung

Specializing Definition 6.12 to the case of propositional logic, we see that G is a logical consequence of F, i.e., $F \models G$, if the function table of G contains a 1 for at least all argument for which the function table of F contains a 1.[45]

Example 6.11. $F = (A \wedge \neg B) \vee (B \wedge \neg C)$ is a logical consequence of A and $\neg C$, i.e., $\{A, \neg C\} \models F$. In contrast, F is not a logical consequence of A and B, i.e., $\{A, B\} \not\models F$.

The basic equivalences of Lemma 6.1 apply in particular to propositional logic.

6.5.4 Normal Forms

Definition 6.25. A *literal* is an atomic formula or the negation of an atomic formula.

Definition 6.26. A formula F is in *conjunctive normal form (CNF)* if it is a conjunction of disjunctions of literals, i.e., if it is of the form

$$F = (L_{11} \vee \cdots \vee L_{1m_1}) \wedge \cdots \wedge (L_{n1} \vee \cdots \vee L_{nm_n})$$

for some literals L_{ij}.

Example 6.12. The formula $(A \vee \neg B) \wedge (\neg A \vee B \vee \neg D) \wedge \neg C$ is in CNF.

Definition 6.27. A formula F is in *disjunctive normal form (DNF)* if it is a disjunction of conjunctions of literals, i.e., if it is of the form

$$F = (L_{11} \wedge \cdots \wedge L_{1m_1}) \vee \cdots \vee (L_{n1} \wedge \cdots \wedge L_{nm_n})$$

for some literals L_{ij}.

Example 6.13. The formula $(B \wedge C) \vee (\neg A \wedge B \wedge \neg C)$ is in DNF.

Theorem 6.4. *Every formula is equivalent to a formula in CNF and also to a formula in DNF.*

Proof. Consider a formula F with atomic formulas $A_1, \ldots, A_n$ with a truth table of size 2^n.

[45] If the truth values 0 and 1 were interpreted as numbers, then $F \models G$ means that G is greater or equal to F for all arguments. This also explains why $F \models G$ and $G \models H$ together imply $F \models H$.

Given such a formula F, one can use the truth table of F to derive an equivalent formula in DNF, as follows. For every row of the function table evaluating to 1 one takes the conjunction of the n literals defined as follows: If $A_i = 0$ in the row, one takes the literal $\neg A_i$, otherwise the literal A_i. This conjunction is a formula whose function table is 1 exactly for the row under consideration (and 0 for all other rows). Then one takes the disjunction of all these conjunctions. F is true if and only if one of the conjunctions is true, i.e., the truth table of this formula in DNF is identical to that of F.

Given such a formula F, one can also use the truth table of F to derive an equivalent formula in CNF, as follows. For every row of the function table evaluating to 0 one takes the disjunction of the n literals defined as follows: If $A_i = 0$ in the row, one takes the literal A_i, otherwise the literal $\neg A_i$. This disjunction is a formula whose function table is 0 exactly for the row under consideration (and 1 for all other rows). Then one takes the conjunction of all these (row-wise) disjunctions. F is false if and only if all the disjunctions are false, i.e., the truth table of this formula in CNF is identical to that of F. □

Example 6.14. Consider the formula $F = (A \wedge \neg B) \vee (B \wedge \neg C)$ from above. The function table is

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

We therefore obtain the following DNF

$$F \equiv (\neg A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C)$$

as the disjunction of 4 conjunctions. And we obtain the following CNF

$$F \equiv (A \vee B \vee C) \wedge (A \vee B \vee \neg C) \wedge (A \vee \neg B \vee \neg C) \wedge (\neg A \vee \neg B \vee \neg C).$$

as the conjunction of 4 disjunctions.

It is often useful to transform a given formula into an equivalent formula in a certain normal form, but the CNF and the DNF resulting from the truth table as described in the proof of Theorem 6.4 are generally exponentially long. In fact, in the above example F is already given in disjunctive normal form, and the procedure has resulted in a much longer (equivalent) formula in DNF.

A transformation to CNF or DNF can also be carried out by making use of the basic equivalences of propositional logic.

Example 6.15. For the formula $\neg\big((A\wedge\neg B)\vee(B\wedge C)\big)\vee D$ we derive an equivalent formula in CNF, using the basic equivalences of Lemma 6.1:

$$\begin{aligned}
\neg\big((A \wedge \neg B) \vee (B \wedge C)\big) \vee D &\equiv \big(\neg(A \wedge \neg B) \wedge \neg(B \wedge C)\big) \vee D \\
&\equiv \big((\neg A \vee \neg\neg B) \wedge (\neg B \vee \neg C)\big) \vee D \\
&\equiv \big((\neg A \vee B) \wedge (\neg B \vee \neg C)\big) \vee D \\
&\equiv ((\neg A \vee B) \vee D) \wedge ((\neg B \vee \neg C) \vee D) \\
&\equiv (\neg A \vee B \vee D) \wedge (\neg B \vee \neg C \vee D).
\end{aligned}$$

In the first step we have used $F \wedge G \equiv \neg(\neg F \vee \neg G)$, which is a direct consequence of rule 8) of Lemma 6.1. In the second step we have applied rule 8) twice, etc.

6.5.5 The Resolution Calculus for Propositional Logic

Resolution is an important logical calculus that is used in certain computer algorithms for automated reasoning. The calculus is very simple in that it consists of a single derivation rule. The purpose of a derivation is to prove that a given set M of formulas (or, equivalently, their conjunction) is unsatisfiable.

As mentioned earlier (see Lemma 6.2), this also allows to prove that a formula F is a tautology, which is the case if and only if $\neg F$ is unsatisfiable. It also allows to prove that a formula F is a logical consequence of a set M of formulas (i.e., $M \models F$), as this is the case if and only if the set $M \cup \{\neg F\}$ is unsatisfiable (see Lemma 6.3).

The resolution calculus assumes that all formulas of M are given in conjunctive normal form (CNF, see Definition 6.26). This is usually not the case, and therefore the formulas of M must first be transformed into equivalent formulas in CNF, as explained earlier. Moreover, instead of working with CNF-formulas (as the syntactic objects), one works with an equivalent object, namely sets of clauses.

Recall (Definition 6.25) that a literal is an atomic formula or the negation of an atomic formula. For example A and $\neg B$ are literals.

Definition 6.28. A *clause* is a set of literals.

Example 6.16. $\{A, \neg B, \neg D\}$ and $\{B, C, \neg C, \neg D, E\}$ are clauses, and the empty set $\varnothing$ is also a clause.

Definition 6.29. The set of clauses associated to a formula

$$F = (L_{11} \vee \cdots \vee L_{1m_1}) \wedge \cdots \wedge (L_{n1} \vee \cdots \vee L_{nm_n})$$

in CNF, denoted as $\mathcal{K}(F)$, is the set

$$\mathcal{K}(F) \stackrel{\text{def}}{=} \{\{L_{11},\ldots,L_{1m_1}\},\ldots,\{L_{n1},\ldots,L_{nm_n}\}\}.$$

The set of clauses associated with a set $M = \{F_1,\ldots,F_k\}$ of formulas is the union of their clause sets:

$$\mathcal{K}(M) \stackrel{\text{def}}{=} \bigcup_{i=1}^{k} \mathcal{K}(F_i).$$

The idea behind this definition is that a clause is satisfied by a truth assignment if and only if it contains *some* literal that evaluates to true. In other words, a clause stands for the disjunction (OR) of its literals. Likewise, a set $\mathcal{K}(M)$ of clauses is satisfied by a truth assignment if *every* clause in $\mathcal{K}(M)$ is satisfied by it. In other words, a set of clauses stands for the conjunction (AND) of the clauses. The set $M = \{F_1,\ldots,F_k\}$ is satisfied if and only if $\bigwedge_{i=1}^{k} F_i$ is satisfied, i.e., if and only if all clauses in $\mathcal{K}(M)$ are satisfied. Note that *the empty clause corresponds to an unsatisfiable formula and the empty set of clauses corresponds to a tautology.*

Note that for a given formula (not necessarily in CNF) there are many equivalent formulas in CNF and hence many equivalent sets of clauses. Conversely, to a given set $\mathcal{K}$ of clauses one can associate many formulas which are, however, all equivalent. Therefore, one can naturally think of a set of clauses as a (canonical) formula, and the notions of satisfiability, equivalence, and logical consequence carry over immediately from formulas to clause sets.

Definition 6.30. A clause K is a *resolvent* of clauses K_1 and K_2 if there is a literal L such that $L \in K_1$, $\neg L \in K_2$, and[46]

$$K = \big(K_1 \setminus \{L\}\big) \cup \big(K_2 \setminus \{\neg L\}\big). \qquad (6.1)$$

Example 6.17. The clauses $\{A, \neg B, \neg C\}$ and $\{\neg A, C, D, \neg E\}$ have two resolvents: If A is eliminated, we obtain the clause $\{\neg B, \neg C, C, D, \neg E\}$, and if C is eliminated, we obtain the clause $\{A, \neg B, \neg A, D, \neg E\}$. Note that clauses are sets and we can write the elements in arbitrary order. In particular, we could write the latter clause as $\{A, \neg A, \neg B, D, \neg E\}$.

It is important to point out that resolution steps must be carried out one by one; one cannot perform two steps at once. For instance, in the above example, $\{\neg B, D, \neg E\}$ is *not* a resolvent and can also not be obtained by two resolution steps, even though $\{\neg B, D, \neg E\}$ would result from $\{A, \neg B, \neg C\}$ and $\{\neg A, C, D, \neg E\}$ by eliminating A and $\neg C$ from the first clause and $\neg A$ and C from the second clause.[47]

[46] For a literal L, $\neg L$ is the negation of L, for example if $L = \neg A$, then $\neg L = A$.

[47] A simpler example illustrating this is that $\{\{A, B\}, \{\neg A, \neg B\}\}$ is satisfiable, but a "double" resolution step would falsely yield $\varnothing$, indicating that $\{\{A, B\}, \{\neg A, \neg B\}\}$ is unsatisfiable.

Given a set $\mathcal{K}$ of clauses, a resolution step takes two clauses $K_1 \in \mathcal{K}$ and $K_2 \in \mathcal{K}$, computes a resolvent K, and adds K to $\mathcal{K}$. To be consistent with Section 6.4.2, one can write the resolution rule (6.1) as follows:[48]

$$\{K_1, K_2\} \ \vdash_{\mathsf{res}} \ K,$$

where equation (6.1) must be satisfied. The resolution calculus, denoted Res, consists of a single rule:

$$\mathsf{Res} = \{\mathsf{res}\}.$$

Recall that we write $\mathcal{K} \vdash_{\mathsf{Res}} K$ if K can be derived from $\mathcal{K}$ using a finite number of resolution steps.[49]

Lemma 6.5. *The resolution calculus is sound, i.e., if* $\mathcal{K} \vdash_{\mathsf{Res}} K$ *then* $\mathcal{K} \models K$.[50]

Proof. We only need to show that the resolution rule is correct, i.e., that if K is a resolvent of clauses $K_1, K_2 \in \mathcal{K}$, then K is logical consequence of $\{K_1, K_2\}$, i.e.,

$$\{K_1, K_2\} \ \vdash_{\mathsf{res}} \ K \quad \Longrightarrow \quad \{K_1, K_2\} \models K.$$

Let $\mathcal{A}$ be an arbitrary truth assignment suitable for $\{K_1, K_2\}$ (and hence also for K). Recall that $\mathcal{A}$ is a model for $\{K_1, K_2\}$ if and only if $\mathcal{A}$ makes at least one literal in K_1 true and also makes at least one literal in K_2 true.

We refer to Definition 6.30 and distinguish two cases. If $\mathcal{A}(L) = 1$, then $\mathcal{A}$ makes at least one literal in $K_2 \setminus \{\neg L\}$ true (since $\neg L$ is false). Similarly, if $\mathcal{A}(L) = 0$, then $\mathcal{A}$ makes at least one literal in $K_1 \setminus \{L\}$ true (since L is false). Because one of the two cases occurs, $\mathcal{A}$ makes at least one literal in $K = (K_1 \setminus \{L\}) \cup (K_2 \setminus \{\neg L\})$ true, which means that $\mathcal{A}$ is a model for K. □

The goal of a derivation in the resolution calculus is to derive the empty clause $\varnothing$ by an appropriate sequence of resolution steps. The following theorem states that the resolution calculus is complete with respect to the task of proving unsatisfiability.

Theorem 6.6. *A set M of formulas is unsatisfiable if and only if* $\mathcal{K}(M) \vdash_{\mathsf{Res}} \varnothing$.

Proof. The "if" part (soundness) follows from Lemma 6.5: If $\mathcal{K}(M) \vdash_{\mathsf{Res}} \varnothing$, then $\mathcal{K}(M) \models \varnothing$, i.e., every model for $\mathcal{K}(M)$ is a model for $\varnothing$. Since $\varnothing$ has no model, $\mathcal{K}(M)$ also does not have a model. This means that $\mathcal{K}(M)$ is unsatisfiable.

[48] In the literature, one usually does not use the symbol $\vdash$ in the context of resolution.

[49] In the lecture we introduce a natural graphical notation for writing a sequence of resolution steps.

[50] For convenience, the clause K is understood to mean the singleton clause set $\{K\}$. In other words, the truth value of a clause K is understood to be the same the truth value of $\{K\}$.

It remains to prove the "only if" part (completeness with respect to unsatisfiability). We need to show that if a clause set $\mathcal{K}$ is unsatisfiable, then $\varnothing$ can be derived by some sequence of resolution steps. The proof is by induction over the number n of atomic formulas appearing in $\mathcal{K}$. The induction basis (for $n = 1$) is as follows. A clause set $\mathcal{K}$ involving only literals A_1 and $\neg A_1$ is unsatisfiable if and only if it contains the clauses $\{A_1\}$ and $\{\neg A_1\}$. One can derive $\varnothing$ exactly if this is the case.

For the induction step, suppose that for every clause set $\mathcal{K}'$ with n atomic formulas, $\mathcal{K}'$ is unsatisfiable if and only if $\mathcal{K}' \vdash_{\mathsf{Res}} \varnothing$. Given an arbitrary clause set $\mathcal{K}$ for the atomic formulas $A_1, \ldots, A_{n+1}$, define the two clause sets $\mathcal{K}_0$ and $\mathcal{K}_1$ as follows. $\mathcal{K}_0$ is the clause set for atomic formulas $A_1, \ldots, A_n$ obtained from $\mathcal{K}$ by setting $A_{n+1} = 0$, i.e.,

- by eliminating all clauses from $\mathcal{K}$ containing $\neg A_{n+1}$ (which are satisfied since $\neg A_{n+1} = 1$), and
- by eliminating from each remaining clause the literal A_{n+1} if it appears in it (since having A_{n+1} in it can not contribute to the clause being satisfied).

$\mathcal{K}$ is satisfiable under the constraint $A_{n+1} = 0$ if and only if $\mathcal{K}_0$ is satisfiable.

Analogously, $\mathcal{K}_1$ is obtained from $\mathcal{K}$ by eliminating all clauses containing A_{n+1} and by eliminating from each remaining clause the literal $\neg A_{n+1}$ if it appears in it. $\mathcal{K}$ is satisfiable under the constraint $A_{n+1} = 1$ if and only if $\mathcal{K}_1$ is satisfiable.

If $\mathcal{K}$ is unsatisfiable, it is unsatisfiable both for $A_{n+1} = 0$ and for $A_{n+1} = 1$, i.e., both $\mathcal{K}_0$ and $\mathcal{K}_1$ are unsatisfiable. Therefore, by the induction hypothesis, we have $\mathcal{K}_0 \vdash_{\mathsf{Res}} \varnothing$ and $\mathcal{K}_1 \vdash_{\mathsf{Res}} \varnothing$. Now imagine that the same resolution steps leading from $\mathcal{K}_0$ to $\varnothing$ are carried out on $\mathcal{K}$, i.e., with A_{n+1}. This derivation may or may not involve clauses (of $\mathcal{K}$) that contain A_{n+1}. In the latter case (i.e., A_{n+1} not contained), the derivation of $\varnothing$ from $\mathcal{K}_0$ is also a derivation of $\varnothing$ from $\mathcal{K}$, and in the other case it corresponds to a derivation of $\{A_{n+1}\}$ from $\mathcal{K}$.

Analogously, the derivation of $\varnothing$ from $\mathcal{K}_1$ corresponds to a derivation of $\varnothing$ from $\mathcal{K}$ or to a derivation of $\{\neg A_{n+1}\}$ from $\mathcal{K}$.

If in any of the two cases we have a derivation of $\varnothing$ from $\mathcal{K}$, we are done (since $\varnothing$ can be derived from $\mathcal{K}$, i.e., $\mathcal{K} \vdash_{\mathsf{Res}} \varnothing$). If this is not the case, then we have a derivation of $\{A_{n+1}\}$ from $\mathcal{K}$, i.e., $\mathcal{K} \vdash_{\mathsf{Res}} \{A_{n+1}\}$ as well as a derivation of $\{\neg A_{n+1}\}$ from $\mathcal{K}$, i.e., $\mathcal{K} \vdash_{\mathsf{Res}} \{\neg A_{n+1}\}$. From these two clauses one can derive $\varnothing$ by a final resolution step. This completes the proof. □

6.6 Predicate Logic (First-order Logic)

We also refer to Section 2.4 where some basics of predicate logic were introduced informally. Predicate logic is an extension of propositional logic, i.e., propositional logic is embedded in predicate logic as a special case.

6.6.1 Syntax

Definition 6.31. (Syntax of predicate logic.)

- A *variable symbol* is of the form x_i with $i \in \mathbb{N}$.[51]
- A *function symbol* is of the form $f_i^{(k)}$ with $i, k \in \mathbb{N}$, where k denotes the number of arguments of the function. Function symbols for $k = 0$ are called *constants*.
- A *predicate symbol* is of the form $P_i^{(k)}$ with $i, k \in \mathbb{N}$, where k denotes the number of arguments of the predicate.
- A *term* is defined inductively: A variable is a term, and if $t_1, \ldots, t_k$ are terms, then $f_i^{(k)}(t_1, \ldots, t_k)$ is a term. For $k = 0$ one writes no parentheses.
- A *formula* is defined inductively:
  - For any i and k, if $t_1, \ldots, t_k$ are terms, then $P_i^{(k)}(t_1, \ldots, t_k)$ is a formula, called an *atomic* formula.
  - If F and G are formulas, then $\neg F$, $(F \wedge G)$, and $(F \vee G)$ are formulas.
  - If F is a formula, then, for any i, $\forall x_i\ F$ and $\exists x_i\ F$ are formulas.

$\forall$ is called the *universal* quantifier, and $\exists$ is called the *existential* quantifier.

A formula constructed according to this inductive definition corresponds naturally to a tree where the leaves correspond to terms and the inner nodes correspond to the logical operators and the quantifiers.

To simplify notation, one usually uses function symbols f, g, h, where the number of arguments is implicit, and for constants one uses the symbols a, b, c. Similarly, one uses predicate symbols P, Q, R, where the number of arguments is implicit. Moreover, one uses variable names x, y, z instead of x_i, and sometimes also u, v, w or k, m, n. To avoid confusion one can also use $(\forall x\ F)$ and $(\exists x\ F)$ instead of $\forall x\ F$ and $\exists x\ F$.

6.6.2 Free Variables and Variable Substitution

Definition 6.32. Every occurrence of a variable in a formula is either *bound* or *free*. If a variable x occurs in a (sub-)formula of the form $\forall x\ G$ or $\exists x\ G$, then it is bound, otherwise it is free.[52] A formula is *closed*[53] if it contains no free variables.

[51] x_0 is usually not used.

[52] The occurrence of a variable x immediately following a quantifier is also bound.

Note that the same variable can occur bound and free in a formula. One can draw the construction tree (see lecture) of a formula showing how a formula is constructed according to the rules of Definition 6.31. Within the subtree corresponding to $\forall x$ or $\exists x$, all occurrences of x are bound.

Example 6.18. In the formula

$$F = Q(x) \ \lor \ \big(\forall y\ P(f(x,y)) \ \land \ \exists x\ R(x,y)\big),$$

the first two occurrences of x are free, the other occurrences are bound. The last occurrence of y is free, the other occurrences are bound.

Definition 6.33. For a formula F, a variable x and a term t, $F[x/t]$ denotes the formula obtained from F by substituting every free occurrence of x by t.

Example 6.19. For the formula F of Example 6.18 we have

$$F[x/g(a,z)] \ = \ Q(g(a,z)) \ \lor \ \big(\forall y\ P(f(g(a,z),y)) \ \land \ \exists x\ R(x,y)\big).$$

6.6.3 Semantics

Recall Definitions 6.5 and 6.6. *In predicate logic, the free symbols of a formula are all predicate symbols, all function symbols, and all occurrences of free variables.* An interpretation, called *structure* in the context of predicate logic, must hence define a universe and the meaning of all these free symbols.

Definition 6.34. An *interpretation* or *structure* is a tuple $\mathcal{A} = (U, \phi, \psi, \xi)$ where

- U is a non-empty *universe*,
- ϕ is a function assigning to each function symbol (in a certain subset of all function symbols) a function, where for a k-ary function symbol f, $\phi(f)$ is a function $U^k \to U$.
- ψ is a function assigning to each predicate symbol (in a certain subset of all predicate symbols) a function, where for a k-ary predicate symbol P, $\psi(P)$ is a function $U^k \to \{0,1\}$, and where
- ξ is a function assigning to each variable symbol (in a certain subset of all variable symbols) a value in U.

For notational convenience, for a structure $\mathcal{A} = (U, \phi, \psi, \xi)$ and a function symbol f one usually writes $f^{\mathcal{A}}$ instead of $\phi(f)$. Similarly, one writes $P^{\mathcal{A}}$ instead of $\psi(P)$ and $x^{\mathcal{A}}$ instead of $\xi(x)$. One also writes $U^{\mathcal{A}}$ rather than U to make $\mathcal{A}$ explicit.

[53] German: geschlossen

We instantiate Definition 6.7 for predicate logic:

Definition 6.35. A interpretation (structure) $\mathcal{A}$ is *suitable* for a formula F if it defines all function symbols, predicate symbols, and freely occurring variables of F.

Example 6.20. For the formula

$$F = \forall x\ \big(P(x) \lor P(f(x,a))\big),$$

a suitable structure $\mathcal{A}$ is given by $U^{\mathcal{A}} = \mathbb{N}$, by $a^{\mathcal{A}} = 3$ and $f^{\mathcal{A}}(x,y) = x+y$, and by letting $P^{\mathcal{A}}$ be the "evenness" predicate (i.e., $P^{\mathcal{A}}(x) = 1$ if and only if x is even). For obvious reasons, we will say (see below) that the formula evaluates to true for this structure.

Another suitable structure $\mathcal{A}$ for F is defined by $U^{\mathcal{A}} = \mathbb{R}$, $a^{\mathcal{A}} = 2$, $f^{\mathcal{A}}(x,y) = xy$ and by $P^{\mathcal{A}}(x) = 1$ if and only if $x \geq 0$ (i.e., $P^{\mathcal{A}}$ is the "positiveness" predicate). For this structure, F evaluates to false (since, for example, $x = -2$ makes $P(x)$ and $P(f(x,a)) = P(2x)$ false).

The semantics of a formula is now defined in the natural way as already implicitly discussed in Section 2.4.

Definition 6.36. (Semantics.) For an interpretation (structure) $\mathcal{A} = (U, \phi, \psi, \xi)$, we define the value (in U) of terms and the truth value of formulas under that structure.

- The value $\mathcal{A}(t)$ of a term t is defined recursively as follows:
  - If t is a variable, i.e., $t = x_i$, then $\mathcal{A}(t) = \xi(x_i)$.
  - If t is of the form $f(t_1, \ldots, t_k)$ for terms $t_1, \ldots, t_k$ and a k-ary function symbol f, then $\mathcal{A}(t) = \phi(f)(\mathcal{A}(t_1), \ldots, \mathcal{A}(t_k))$.
- The truth value of a formula F is defined recursively by Definition 6.16 and
  - If F is of the form $F = P(t_1, \ldots, t_k)$ for terms $t_1, \ldots, t_k$ and a k-ary predicate symbol P, then $\mathcal{A}(F) = \psi(P)(\mathcal{A}(t_1), \ldots, \mathcal{A}(t_k))$.
  - If F is of the form $\forall x\ G$ or $\exists x\ G$, then let $\mathcal{A}_{[x \to u]}$ for $u \in U$ be the same structure as $\mathcal{A}$ except that $\xi(x)$ is overwritten by u (i.e., $\xi(x) = u$):

$$\mathcal{A}(\forall x\ G) \quad = \quad \begin{cases} 1 & \text{if } \mathcal{A}_{[x \to u]}(G) = 1 \text{ for all } u \in U \\ 0 & \text{else} \end{cases}$$

$$\mathcal{A}(\exists x\ G) \quad = \quad \begin{cases} 1 & \text{if } \mathcal{A}_{[x \to u]}(G) = 1 \text{ for some } u \in U \\ 0 & \text{else.} \end{cases}$$

This definition defines the function $\sigma(F, \mathcal{A})$ of Definition 6.8. Note that the definition is recursive not only on formulas (see the second bullet of the definition), but also on structures. Namely, $\mathcal{A}(\forall x\ G)$ and $\mathcal{A}(\exists x\ G)$ are defined in terms of all structures $\mathcal{A}_{[x\to u]}(G)$ for $u \in U$. To evaluate the truth value of a formula $F = \forall x\ G$ one needs to apply Definition 6.36 recursively, for formula G and all structures $\mathcal{A}_{[x\to u]}$.

The basic concepts discussed in Section 6.3 such as satisfiable, tautology, model, logical consequence, and equivalence, are now immediately instantiated for predicate logic.

Note that the syntax of predicate logic does not require nested quantified variables in a formula to be distinct, but we will avoid such overload of variable names to avoid any confusion. For example, the formula $\forall x\ (P(x) \vee \exists y\ Q(y))$ is equivalent to $\forall x\ (P(x) \vee \exists x\ Q(x))$.

6.6.4 Predicate Logic with Equality

Reexamining the syntax of predicate logic it may surprise that the equality symbol "$=$" is not allowed. For example, $\exists x\ f(x) = g(x)$ is not a formula. However, one can extend the syntax and the semantics of predicate logic to include the equality symbol "$=$" with its usual meaning. This is left as an exercise.

6.6.5 Some Basic Equivalences Involving Quantifiers

In addition to the equivalences stated in Lemma 6.1), we have:

Lemma 6.7. *For any formulas F, G, and H, where x does not occur free in H, we have*

1) $\neg(\forall x\ F) \equiv \exists x\ \neg F$;
2) $\neg(\exists x\ F) \equiv \forall x\ \neg F$;
3) $(\forall x\ F) \wedge (\forall x\ G) \equiv \forall x\ (F \wedge G)$;
4) $(\exists x\ F) \vee (\exists x\ G) \equiv \exists x\ (F \vee G)$;
5) $\forall x\ \forall y\ F \equiv \forall y\ \forall x\ F$;
6) $\exists x\ \exists y\ F \equiv \exists y\ \exists x\ F$;
7) $(\forall x\ F) \wedge H \equiv \forall x\ (F \wedge H)$;
8) $(\forall x\ F) \vee H \equiv \forall x\ (F \vee H)$;
9) $(\exists x\ F) \wedge H \equiv \exists x\ (F \wedge H)$;
10) $(\exists x\ F) \vee H \equiv \exists x\ (F \vee H)$.

Proof. We only prove statement 7). The other proofs are analogous.

We have to show that every structure $\mathcal{A}$ that is a model for $(\forall x\ F) \wedge H$ is also a model for $\forall x\ (F \wedge H)$, and vice versa.

Recall that the definition of the semantics of a formula $\forall x\ G$ for a structure $\mathcal{A}$ is that, for all $u \in U$, $\mathcal{A}_{[x\to u]}(G) = 1$.

To prove the first direction, i.e., $(\forall x\ F) \wedge H \models \forall x\ (F \wedge H)$, suppose that $\mathcal{A}\big((\forall x\ F) \wedge H\big) = 1$ and hence[54] that (i) $\mathcal{A}(\forall x\ F) = 1$ and that (ii) $\mathcal{A}(H) = 1$. Recall that (i) means that $\mathcal{A}_{[x\to u]}(F) = 1$ for all $u \in U$, and (ii) means that $\mathcal{A}_{[x\to u]}(H) = 1$ for all $u \in U$ (since x does not occur free in H and hence $\mathcal{A}_{[x\to u]}(H) = \mathcal{A}(H)$). Therefore $\mathcal{A}_{[x\to u]}(F \wedge H) = 1$ for all $u \in U$, which means that $\mathcal{A}(\forall x\ (F \wedge H)) = 1$, which was to be proved.

To prove the other direction, i.e. $\forall x\ (F \wedge H) \models (\forall x\ F) \wedge H$, suppose that $\mathcal{A}\big(\forall x\ (F \wedge H)\big) = 1$, i.e., for all $u \in U$, $\mathcal{A}_{[x\to u]}(F \wedge H) = 1$, which means that (i) $\mathcal{A}_{[x\to u]}(F) = 1$ for all $u \in U$ and (ii) $\mathcal{A}_{[x\to u]}(H) = 1$ for all $u \in U$. By definition, (i) means that $\mathcal{A}(\forall x\ F) = 1$. Moreover, because x does not occur free in H, by (ii) we have $\mathcal{A}_{[x\to u]}(H) = \mathcal{A}(H) = 1$ for all u, which by definition means $\mathcal{A} \models H$. Hence $\mathcal{A} \models (\forall x\ F) \wedge H$. □

The following natural lemma is stated without proof.

Lemma 6.8. *If one replaces a sub-formula G of a formula F by an equivalent (to G) formula H, then the resulting formula is equivalent to F.*

Example 6.21. $\forall y\ Q(x,y)$ is a sub-formula of $\exists x\ (P(x) \vee \forall y\ Q(x,y))$. Therefore

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

because $\forall y\ Q(x,y) \equiv \neg\exists y\ \neg Q(x,y)$.

6.6.6 Substitution of Bound Variables

The following lemma states that the name of a bound variable carries no semantic meaning and can therefore be replaced by any other variable name that does not occur elsewhere. This is called *bound substitution*.

Lemma 6.9. *For a formula G in which y does not occur, we have*

- $\forall x\ G \equiv \forall y\ G[x/y]$,
- $\exists x\ G \equiv \exists y\ G[x/y]$.

Proof. For any structure $\mathcal{A} = (U, \phi, \psi, \xi)$ and $u \in U$ we have

$$\mathcal{A}_{[x\to u]}(G) \;=\; \mathcal{A}_{[y\to u]}(G[x/y]).$$

Therefore $\forall x\, G$ is true for exactly the same structures for which $\forall y\ G[x/y]$ is true. □

[54] according to the semantics of $\wedge$, see Definition 6.36

Example 6.22. The formula $\forall x\ \exists y\ (P(x, f(y)) \vee Q(g(x), a))$ is equivalent to the formula $\forall u\ \exists v\ (P(u, f(v)) \vee Q(g(u), a))$ obtained by substituting x by u and y by v.

Definition 6.37. A formula in which no variable occurs both as a bound and as a free variable and in which all variables appearing after the quantifiers are distinct is said to be in *rectified*[55] form.

By appropriately renaming quantified variables one can transform any formula into an equivalent formula in rectified form.

6.6.7 Normal Forms

It is often useful to transform a formula into an equivalent formula of a specific form, called a normal form. This is analogous to the conjunctive and disjunctive normal forms for formulas in propositional logic.

Definition 6.38. A formula of the form

$$Q_1x_1\ Q_2x_2\ \cdots Q_nx_n\ G,$$

where the Q_i are arbitrary quantifiers ($\forall$ or $\exists$) and G is a formula free of quantifiers, is said to be in *prenex form*[56].

Theorem 6.10. *For every formula there is an equivalent formula in prenex form.*

Proof. One first transforms the formula into an equivalent formula in rectified form and then applies the equivalences of Lemma 6.7 move up all quantifiers in the formula tree, resulting in a prenex form of the formula. □

Example 6.23.

$$\begin{aligned}
\neg\big(\forall x\ P(x,y)\ \wedge\ \exists y\ Q(x,y,z)\big) &\equiv \neg\big(\forall u\ P(u,y)\ \wedge\ \exists v\ Q(x,v,z)\big) \\
&\equiv \neg\forall u\ P(u,y)\ \vee\ \neg\exists v\ Q(x,v,z) \\
&\overset{(1)}{\equiv} \exists u\ \neg P(u,y)\ \vee\ \neg\exists v\ Q(x,v,z) \\
&\overset{(2)}{\equiv} \exists u\ \neg P(u,y)\ \vee\ \forall v\ \neg\ Q(x,v,z) \\
&\overset{(10)}{\equiv} \exists u\ \big(\neg P(u,y)\ \vee\ \forall v\ \neg\ Q(x,v,z)\big) \\
&\equiv \exists u\ \big(\forall v\ \neg\ Q(x,v,z)\ \vee\ \neg P(u,y)\big) \\
&\overset{(8)}{\equiv} \exists u\ \big(\forall v\ \big(\neg Q(x,v,z)\ \vee\ \neg P(u,y)\big)\big) \\
&\equiv \exists u\ \forall v\ \big(\neg Q(x,v,z)\ \vee\ \neg P(u,y)\big) \\
&\equiv \exists u\ \forall v\ \big(\neg P(u,y)\ \vee\ \neg\ Q(x,v,z)\big).
\end{aligned}$$

[55] German: bereinigt
[56] German: Pränexform

In the first step we have renamed the bound variables, in the second step we made use of the equivalence $\neg(F \wedge G) \equiv \neg F \vee \neg G$ (Lemma 6.1 8)), and then we have applied the rules of Lemma 6.7, as indicated. We have also made explicit the use of the commutative law for $\vee$ (Lemma 6.1 2)). In the second last step, the removal of parentheses is made explicit. The last step, again making use of Lemma 6.1 2), is included (only) to arrive at a form with the same order of occurrence of P and Q.

One can also transform every formula F into a formula G in prenex form that only contains universal quantifiers ($\forall$). However, such a formula is in general not equivalent to F, but only equivalent with respect to satisfiability. In other words, F is satisfiable if and only if G is satisfiable. Such a normal form is called *Skolem normal form*. This topic is beyond the scope of this course.

6.6.8 Derivation Rules

It is beyond the scope of this course to systematically discuss derivation rules for predicate logic, let alone an entire calculus. But, as an example, we discuss one such rule, called *universal instantiation* (or also *universal elimination*). It states that for any formula F and any term t, one can derive from the formula $\forall x F$ the formula $F[x/t]$, thus eliminating the quantifier $\forall$:[57]

$$\forall xF\ \ \vdash\ F[x/t]$$

This rule is justified by the following lemma (proof left as an exercise).

Lemma 6.11. *For any formula F and any term t we have*

$$\forall xF\ \ \models\ \ F[x/t].$$

6.6.9 An Example Theorem and its Interpretations

The following apparently innocent theorem is a powerful statement from which several important corollaries follow as special cases. The example illustrates that one can prove a general theorem in predicate logic and, because it is a tautology, it can then be instantiated for different structures (i.e., interpretations), for each of which it is true.

[57] Note that if x does not occur free in F, the statement still holds but in this case is trivial.

Theorem 6.12. $\neg\exists x\forall y\ \big(P(y,x) \leftrightarrow \neg P(y,y)\big)$.

Recall that the statement of the theorem means that the formula $\neg\exists x\forall y\ \big(P(y,x) \leftrightarrow \neg P(y,y)\big)$ is a tautology, i.e., true for any suitable structure, i.e., for any universe and any choice of the predicate P.

Proof. We can transform the formula by equivalence transformations:

$$\begin{aligned}\neg\exists x\ \forall y\ \big(P(y,x) \leftrightarrow \neg P(y,y)\big) &\equiv \forall x\ \neg\forall y\ \big(P(y,x) \leftrightarrow \neg P(y,y)\big)\\ &\equiv \forall x\ \exists y\ \neg\big(P(y,x) \leftrightarrow \neg P(y,y)\big)\\ &\equiv \forall x\ \exists y\ \big(P(y,x) \leftrightarrow P(y,y)\big),\end{aligned}$$

where we have made use of $\neg(F \leftrightarrow \neg G)\ \equiv\ (F \leftrightarrow G)$, which is easily checked to hold by comparing the truth tables of $\neg(A \leftrightarrow \neg B)$ and $(A \leftrightarrow B)$

To see that the latter formula (i.e., $\forall x\ \exists y\ \big(P(y,x) \leftrightarrow P(y,y)\big)$) is a tautology, let $\mathcal{A}$ be an arbitrary suitable interpretation, which defines the universe $U^{\mathcal{A}}$ and the predicate $P^{\mathcal{A}}$. Below we omit the superscripts $\mathcal{A}$ and write simply U and P. Since $\mathcal{A}$ is arbitrary, it suffices to show that

$$\mathcal{A}(\forall x\ \exists y\ \big(P(y,x) \leftrightarrow P(y,y)\big)) = 1.$$

This can be shown as follows: For every $u \in U$ we have

$$\mathcal{A}(P(u,u) \leftrightarrow P(u,u)) = 1.$$

Hence for every $u \in U$ we have

$$\mathcal{A}_{[x\to u][y\to u]}(P(y,x) \leftrightarrow P(y,y)) = 1,$$

and therefore for every fixed $u \in U$ we have

$$\mathcal{A}_{[x\to u]}(\exists y\ P(y,x) \leftrightarrow P(y,y)) = 1,$$

and therefore we have

$$\mathcal{A}(\forall x\ \exists y\ P(y,x) \leftrightarrow P(y,y)) = 1,$$

as was to be shown. □

Let us now interpret Theorem 6.12. We can instantiate it for different universes and predicates. The first interpretation is Russel's paradox:

Corollary 6.13. *There exists no set that contains all sets S that do not contain themselves, i.e., $\{S|\ S \notin S\}$ is not a set.*

Proof. We consider the universe of all sets[58] and, to be consistent with the chapter on set theory, use the variable names R instead of x and S instead of y.[59] Moreover, we consider the specific predicate P defined as $P(S,R) = 1$ if and only if $S \in R$. Then Theorem 6.12 specializes to

$$\neg\exists R\ \ \forall S\ \big(S \in R \leftrightarrow S \notin S\big).$$

This formula states that there is no set R such that for a set (say S) to be in R is equivalent to not being contained in itself ($S \notin S$). □

It is interesting to observe that Russell's paradox is a fact that holds more generally than in the universe of sets and where $P(x,y)$ is defined as $x \in y$. We state another corollary:

Example 6.24. The reader can investigate as an exercise that Theorem 6.12 also explains the so-called barber paradox (e.g. see Wikipedia) which considers a town with a single barber as well as the set of men that do not shave themselves.

The following corollary was already stated as Theorem 3.23.

Corollary 6.14. *The set $\{0,1\}^\infty$ is uncountable.*

We prove the equivalent statement: Every enumeration of elements of $\{0,1\}^\infty$ does not contain all elements of $\{0,1\}^\infty$.

Proof. We consider the universe $\mathbb{N}$ and a fixed enumeration of elements of $\{0,1\}^\infty$, and we interpret $P(y,x)$ as the yth bit of the xth sequence of the enumeration. Then Theorem 6.12, $\neg\exists x\forall y\ \big(P(y,x) \leftrightarrow \neg P(y,y)\big)$, states that there exists no index x, i.e., no sequence in the enumeration, such that for all y, the yth bit on that sequence is equal to the negation of the yth bit on the yth sequence. But the sequence given by $y \mapsto \neg P(y,y)$ is a well-defined sequence in $\{0,1\}^\infty$, and we just proved that it does not occur in the enumeration. □

Note that the proof of this corollary contains Cantor's diagonalization argument, which is hence implicite in Theorem 6.12.

We discuss a further use of the theorem. If we understand a program as describable by a finite bit-string, or, equivalently, a natural number (since there is a bijection between finite bit-strings and natural numbers), and if we consider programs that take a natural number as input and output 0 or 1, then we obtain the following theorem. (Here we ignore programs that do not halt (i.e.,

[58] The universe of all sets is not a set itself. Formally, the universe in predicate logic need not be a set (in the sense of set theory), it can be a "collection" of objects.

[59] The particular variable names (R and S) are not relevant and are chosen simply to be compatible with the chapter on set theory where sets were denoted by capital letters and Russel's proposed set was called R. Here we have deviated from the convention to use only small letters for variables.

loop forever), or, equivalently, we interpret looping as output 0.) The following corollary was already stated as Corollary 3.24.[60]

Corollary 6.15. *There are uncomputable functions* $\mathbb{N} \to \{0,1\}$.

Proof. We consider the universe $\mathbb{N}$, and a program is thought of as represented by a natural number. Let $P(y,x) = 1$ if and only if the bit that program x outputs for input y is 1. Theorem 6.12, $\neg\exists x\forall y\ \big(P(y,x) \leftrightarrow \neg P(y,y)\big)$, states that there exists no program x that (for all inputs y) computes the function $y \mapsto \neg P(y,y)$, i.e., this function is uncomputable. □

The above corollary was already discussed as Corollary 3.24, as a direct consequence of Corollary 6.14 (i.e., of Theorem 3.23). The proof given here is stronger in the sense that it provides a concrete function, namely the function $y \mapsto \neg P(y,y)$, that is not computable.[61] We state this as a corollary:

Corollary 6.16. *The function* $\mathbb{N} \to \{0,1\}$ *assigning to each* $y \in \mathbb{N}$ *the complement of what program* y *outputs on input* y*, is uncomputable.*

We point out that the corollary does not exclude the existence of a program that computes the function for an overwhelming fraction of the y, it excludes only the existence of a program that computes the function for all but finitely many arguments.

6.7 Beyond Predicate Logic *

The expressiveness of every logic is limited. For example, one can not express meta-theorems about the logic as formulas within the logic. It is therefore no surprise that the expressiveness of predicate logic is also limited.

The formula $F = \forall x\exists y\ P(x,y)$ can equivalently be stated as follows: There exists a (unary) function $f : U \to U$ such that $\forall x\ P(x, f(x))$. The function f assigns to every x one of the y for which $P(x,y)$. Such a y must exist according to F.

In other words, the pair of quantifiers $\forall x\exists y$ is equivalent to the existence of a function. However, we can not write this as a formula since function symbols are not variables and can not be used with a quantifier. The formula $\exists f\ P(x, f(x))$ is not a formula in predicate (or first-order) logic. Such formulas exist only in second-order logic, which is substantially more involved and not discussed here.

Predicate logic is actually more limited than one might think. As an example, consider the formula

$$\forall w\, \forall x\, \exists y\, \exists z\ P(w,x,y,z).$$

[60] Explaining the so-called Halting problem, namely to decide whether a given program halts for a given input, would require a more general theorem than Theorem 6.12, but it could be explained in the same spirit.

[61] This function of course depends on the concrete programming language which determines the exact meaning of a program and hence determines P.

In this formula, y and z can depend on both w and x. It is not possible to express, as a formula in predicate logic, that in the above formula, y must only depend on w and z must only depend on x. This appears to be an artificial restriction that is not desirable.