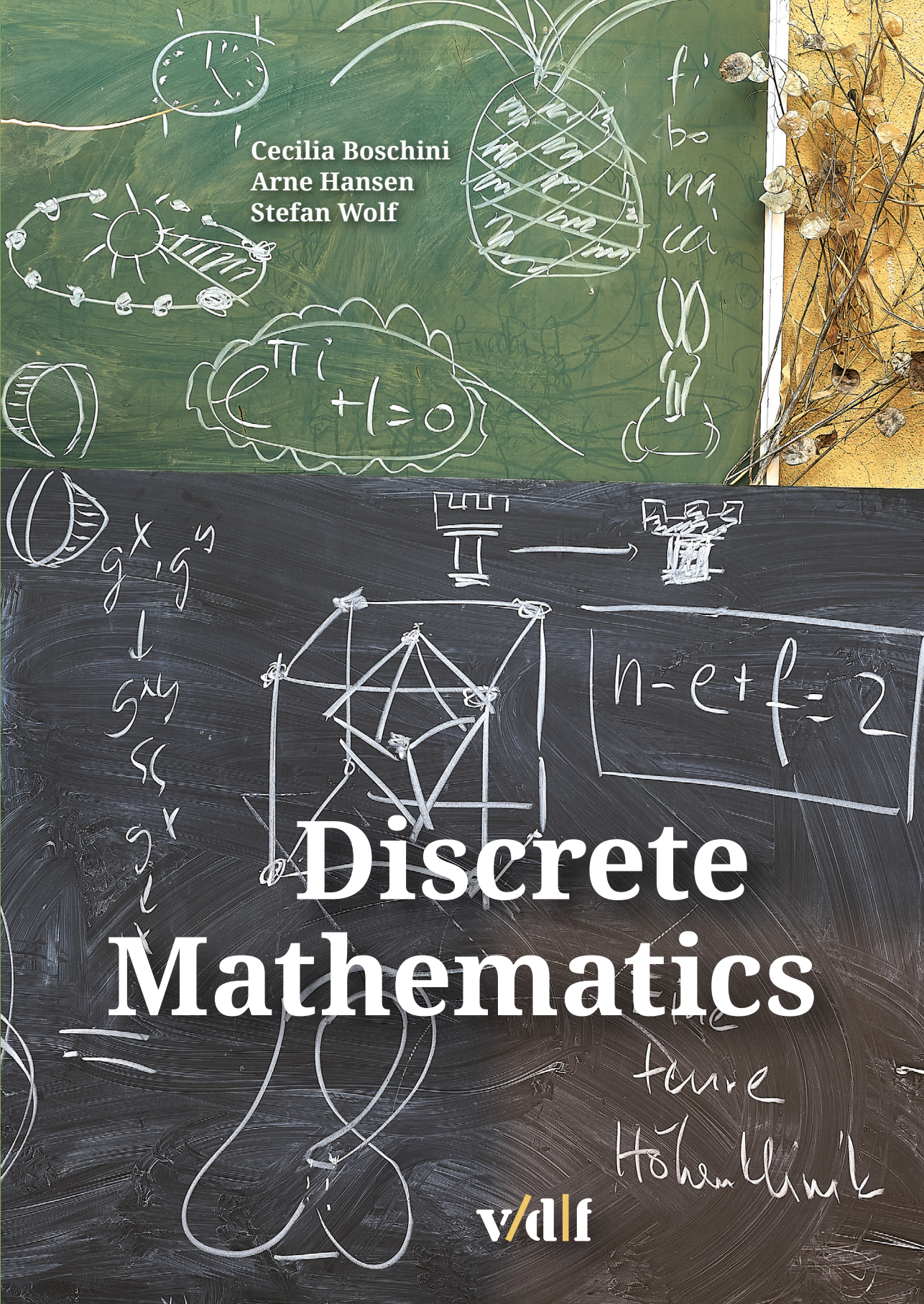


Discrete Mathematics

vdlf

Cecilia Boschini · Arne Hansen · Stefan Wolf

Discrete Mathematics

v/d|f

Bibliographic Information published by Die Deutsche Nationalbibliothek
Die Deutsche Nationalbibliothek lists this publication in the Internet at
http://dnb.dnb.de.

ISBN 978-3-7281-4109-5 (Printversion)

Download open access:
ISBN 978-3-7281-4110-1 / DOI 10.3218/4110-1

www.vdf.ethz.ch
verlag@vdf.ethz.ch

Contents

Chapter 0

What is Discrete Mathematics?

0.1 What is *"Mathematics"*?

Of the two terms in the title of this book, "mathematics" is the more common one. It is, however, not the easier one to define. *Mathematics* is a tradition, a style of thinking and a way of dealing with problems in technology and natural science, but also a mean to simply satisfy artistoïd aesthetic aspirations: Most mathematicians describe their work as akin to a quest for beauty. Much has been written about the ineffable elegance of some theorems or formulas (the most famous specimen being Euler's formula, $e^{i\pi} + 1 = 0$).

Not only are there various perspectives onto mathematics and how mathematicians regard their work. There are profound philosophical questions about the *nature* of mathematics that are still debated today. While it is held within *Platonism* that mathematics is *discovered*, *constructivists* regard mathematics as a rather artificial construct.

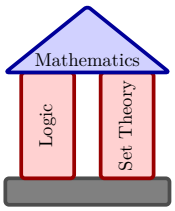


Figure 1: The pillars of Mathematics.

Irrespective of the stand on the philosophical nature of mathematics, one can single out two principal pillars of today's mathematics: *logic* and *set theory*.[1] Logic determines how to reason within Mathematics, i.e., what is considered a valid *proof*. Set theory describes the *objects* we deal with. In Chapter 2, we introduce propositional logic, in Chapter 3 set theory.

Example (Set theory). One of the definitions of the natural numbers relies merely on nesting sets of the empty set.

$$\begin{aligned} 0 &:= \emptyset \\ 1 &:= \{\emptyset\} \\ 2 &:= \{\{\emptyset\}, \emptyset\} \\ &\vdots \end{aligned}$$

This exemplifies how set theory serves to define basic mathematical entities.

0.2 What is *"discrete"*?

With *discrete* we usually refer to *finite* or *countably infinite sets*. While it is intuitively clear what is meant by *finite* (a set of elements we can label with natural numbers $1, 2, 3, \ldots, N$), *countably infinite* needs slightly more explanation. A set S is called *countably infinite* if one can match each element in S with a unique natural number. Formally speaking, such a matching is a function. This leads us to the following definition.

Definition 0.1 (Countable set)**.** A set S is called *countable* if there exists a one-to-one or injective[2] function $f : S \to \mathbb{N}$. If f is also surjective[3], then S is called countably *infinite*.

To better understand the concept of countability, we consider two common examples.

Example (The rational numbers are countable). Even though there seem to be many more rational numbers than natural numbers, we can construct a matching showing that the set is countably infinite. The rational numbers

[1] The picture of the two pillars provides a useful intuition. Albeit it being a simplified and in some regard questionable perspective. For one, set theory comes itself with an logical foundation that cannot be transferred to the logic pillar. Another doubt stems from observation that we ourselves, including our senses and brains, appear in the object world which we associated with the set-theory pillar. This bars us from a neutral perspective that is taken in the picture leading us to a more involved perspective as implied by Gonseth's position "Logic is first of all a natural science."

[2] This is to say, each element gets assigned a unique natural number, or in other words, no natural number is matched to two different elements in S.

[3] That is, every natural number is matched to an element in S.

can be associated with dots in a two-dimensional space, with the enumerator on the x-axis and the denominator on the y-axis, or *vice versa*. We can then number the dots in a spiral, as shown in Figure 2.

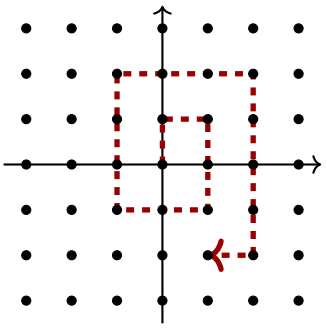


Figure 2: Matching the rational and the natural numbers

Example (The real numbers are *not* countable). One can prove by contradiction that there is no such matching for the real numbers. Indeed, let us assume that we can assign a unique natural number to every real number in the interval $[0, 1]$. If we represent the real numbers between 0 and 1 through the binary representation, we obtain a list similar to the following:

$$\begin{array}{ll} 1 & 0.0101011100110\ldots \\ 2 & 0.1001100101001\ldots \\ 3 & 0.0001011001010\ldots \\ \vdots & \vdots \end{array}$$

Surprisingly, we can construct a real number N that is not contained in the list: The i^{th} digit of the binary representation of N is just the flipped value of the i^{th} digit of the i^{th} real number:

$$\begin{array}{ll} 1 & 0.\mathbf{0}101011100110\ldots \\ 2 & 0.1\mathbf{0}01100101001\ldots \\ 3 & 0.00\mathbf{0}1011001010\ldots \\ \vdots & \vdots \\ \text{constructed real number } N = & 0.111\ldots \end{array}$$

The number we constructed differs from every element in the list and is, therefore, not contained in the list. This yields a contradiction with the initial assumption that we could assign a unique natural number to *every* real number. Thus, the size of the set of real numbers is strictly larger than the size of the

set of natural numbers (in the sense that there is no bijective map in-between the two):

$$|\mathbb{R}| > |\mathbb{Q}| = |\mathbb{N}| \ .$$

0.3 Why bother?

What is the relevance of Discrete Mathematics? On the one hand, it is the Mathematics that arises in daily life (including logic, the art of sound reasoning). On the other hand, as computers have only finite states, informatics is in a sense *applied Discrete Mathematics*. In fact, at the end of this book we will see that *cryptography*, the science of secure transmission of information between computers, strongly relies on mathematical concepts. As such, one could even argue that the importance of Mathematics is not only scientific, but *political*.

Chapter 1

Motivation

In this first chapter, we look into some introductory examples that give a flavor of some of the topics, themes, and methods appearing in these notes.

1.1 Swapping knights

Consider a reduced chess board with two knights of each color as shown in Figure 1.1. Is it possible to transform the configuration on the left into the configuration on the right using regular knight moves (never having more than one piece on a square, and no "taking")?

The knights never reach the middle field. Furthermore, we can simplify the situation by interpreting the "neighborship" of squares through knight moves, and represent that in a graph (Figure 1.2, left side). In fact, such a graph can be drawn in an even simpler way, as shown in the right side of Figure 1.2. Now, we have a very adequate, because simple, model: Each possible field is then a *node* in the graph, which is simply a *cycle*: The knights can move around the

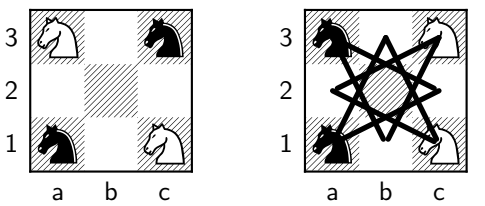


Figure 1.1: Reduced chessboard with knights

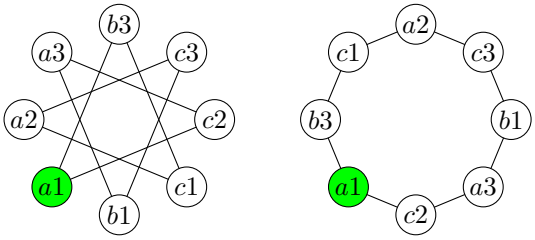


Figure 1.2: Graphs representing the knights swapping problem.

loop, but not change their order. Thus the answer is that the transformation is *impossible*.

This example contains already some basic mathematical principles and tools of problem solving:

Modelling: Finding the right model for a problem is half the solution.

Graphs are often a good structure to model discrete problems.

Abstraction: Focus on the relevant, and get rid of the irrelevant (like in art).

1.2 Combinatorics

Imagine you want to create a necklace with p pearls, with p being a prime number. You are given pearls in a different colors, and you have at least p pearls for each color (so that you can make monochrome necklaces). How many different necklaces can you make?

At first glance, the answer seems to be a^p: One can choose the first pearl in a different colors, i.e., in a different ways, and analogously for the others. According to this reasoning, the number of necklaces would be the product of the possible choices for each pearl, thus a^p.

We are not quite satisfied yet: Since we can *rotate* some of the patterns into others, these a^p necklaces are in fact not distinct! Roughly speaking, we have counted every pattern p times (once per possible rotation), so we should divide the total number by p. But that is not entirely true: The one-colored necklaces appear just once. It is important to note, however, that these are the only exceptions (from being counted p times) since p is a prime number: Periodic configurations like in Figure 1.3, where a certain pattern of 4-necklaces is counted 2 times, cannot occur. The reason is that the number of times must be a divisor of the length, which again we chose to be a prime number, having as only divisors 1 and p.

Hence, to count the possible necklaces, we first consider all those having pearls of at least two colors, which are $a^p - a$, and divide them by the number

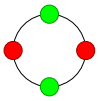


Figure 1.3: Necklace with symmetries, p not being prime.

p of possible rotations. Then we add the monochrome necklaces:

$$N = \frac{a^p - a}{p} + a \ .$$

A surprising consequence of this problem is that for a prime number p, $a^p - a$ is divisible by p. This is called *Fermat's little theorem*, and it plays an important role in cryptography, namely, the public-key protocol RSA we introduce later in the course.

1.3 Connections without crossings

Imagine a settlement with three houses and three plants (electricity, water, gas), as shown in Figure 1.4. *Is it possible to connect the three houses to each of the plants without crossings in the connections?*

This problem can again be turned into a graph, namely $K_{3,3}$, as shown in Figure 1.5. The question then becomes: *Can the associated graph be drawn such that edges (which need not be straight lines) merely meet in nodes?* If yes, the graph is called *planar*. An example of a planar graph is the "complete graph with 4 nodes," the K_4 shown in Figure 1.6.

We ultimately want to show that the $K_{3,3}$ is not planar. To do so, we first try to better understand the properties of planar graphs.

Graphs and Polyhedra. There is an interesting analogy between graphs and polyhedra. A polyhedron can be associated with a graph: Imagine the

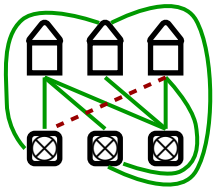


Figure 1.4: Houses with connections. The third house cannot be connected anymore without intersecting other connections.

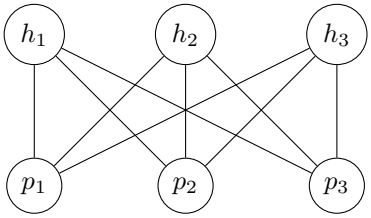


Figure 1.5: Graph representing the houses connection problem.

polyhedron was made out of elastic rubber. One could now remove one face and press the remaining part into a plane: This yields a planar graph. Edges of the polyhedron become edges of the graph, and corners turn into nodes of the graph. Note that the face that was initially removed corresponds now to the infinite region outside the graph (which as always considered as a proper region). Figure 1.7 shows the graph corresponding to a cube.

Looking at different polyhedra (or at planar graphs), it becomes apparent that the number of vertices (nodes) n, the number of edges e and the number of faces f (regions) are closely related. Indeed, *Euler's polyhedron formula* states, for any polyhedron (and equivalently for planar graphs), we have

$$n - e + f = 2 \ .$$

Before looking at a sketch of the proof, we can apply the formula to the graph in Figure 1.5 to answer our initial question. Indeed,

$$n - e + f = 6 - 9 + 13 = 10 \neq 2 \ .$$

Sketch of proof of Euler's polyhedron formula. To conclude the example, we show that Euler's formula holds true for all planar graphs (thus for the one in Figure 1.5 too!). The proof consists of two parts: First, we show the proposition for trees, which form a simpler class of graphs. Then we generalize this to arbitrary planar graphs.

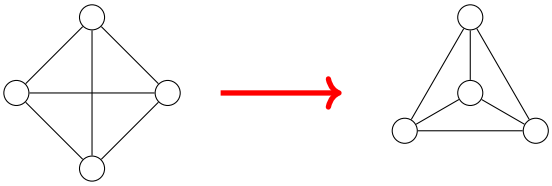


Figure 1.6: The graph K_4 is an example of a planar graph.

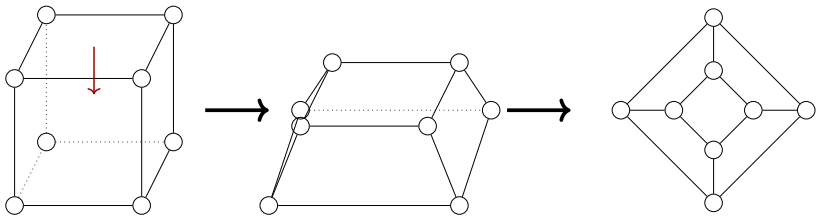


Figure 1.7: A cube can be transformed into a graph by removing the bottom (turns into outer region) and pressing the top down while spreading out the lower nodes.

Euler's formula for trees. A *tree* is a connected graph without cycles; an example is shown in Figure 1.8. It is obviously planar, and it contains only one region, the infinite region around it.

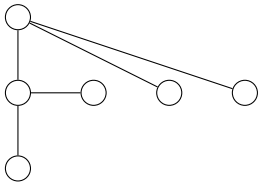


Figure 1.8: A tree.

We show *by induction* (cf. Appendix **??**) that Euler's formula is always satisfied for trees. If there is one node, then the formula is satisfied as $n = 1$, $e = 0$, and $f = 1$. For the *induction step*, we assume that the formula holds for a tree with n nodes and conclude that it holds for a tree with $n + 1$ nodes. Adding a node requires to add an edge as well[1]. The resulting tree has $e + 1$ edges and $n + 1$ nodes, and therefore,

$$\underbrace{\tilde{n}}_{=n+1} - \underbrace{\tilde{e}}_{=e+1} + \underbrace{\tilde{f}}_{=1} = n - e + f = 2 \ ,$$

where the second equality holds due to the induction assumption.

General planar graphs. A general planar graph can be characterized by a *spanning tree* and its *dual graph*: A spanning tree is a subgraph connecting all nodes that is at the same time a tree. The dual graph to a given spanning tree is again a tree, constructed by placing a node in each *region* and connecting

[1]Note that adding one node to a tree requires adding *exactly* one edge. Indeed, as there is always already a path that connects two nodes of a tree, connecting the new node to the tree through two (or more) edges creates a cycle.

them without crossing the spanning tree. An example of construction of the spanning tree and of the dual graph is shown in Figure 1.9.

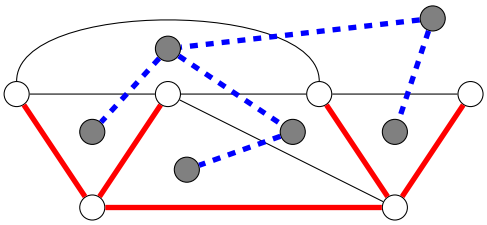


Figure 1.9: Example of a graph (composed by the white nodes and by the black and red edges) with its spanning tree (composed by the white nodes and only the red edges), and the dual graph to this spanning tree (composed by the gray nodes and by the blue, dotted edges).

The spanning tree has the same number of nodes as the original graph $n_s = n$, and one less edge than the nodes: $e_s = n_s - 1$. The number of nodes in the dual tree are the same as the number of regions in the graph, $n_d = f$. Furthermore, each edge of the graph is either part of the spanning tree or crossed by one of the edges of the dual tree. Putting all this together, yields

$$e = e_s + e_d = (n_s - 1) + (n_d - 1) = n - 1 + f - 1 = n + f - 2 \ ,$$

which is again Euler's formula.

1.4 Coloring maps

An example of an application of Euler's formula is to derive upper bounds on the number of colors required to color maps of regions: Imagine a map showing countries and their borders[2]. All two countries sharing a border are considered neighbors. How many colors are needed to color the map in a way that no neighbors have the same color? Three colors are not always enough, as Figure 1.10 shows.

Again, the problem can be translated to planar graphs: Each country is replaced by a node, and neighboring countries are connected by edges. So can the nodes be colored with 4 colors requiring that neighbors are always colored differently? The answer is *yes*, as a computer-aided proof showed some decades ago. As that proof can never be verified directly by humans, there has been

[2]For simplicity's sake, we assume countries to be in one piece, i.e., there are no exclaves like *Campione d'Italia*.

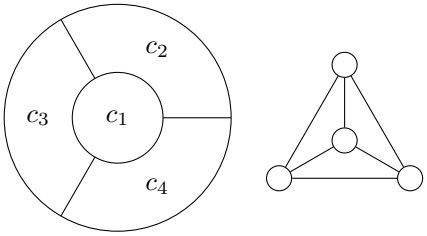


Figure 1.10: The figure on the left shows that it is possible to arrange four countries such that they are all neighbors to one another. Thus, we need at least four colors. The corresponding graph, the K_4 again, is shown on the right.

some argument on the validity of the proof. A shorter proof shows that 5 colors are sufficient. In the following, we sketch a very simple proof for 6 colors.

Proposition 1.1 (6-Coloring). Six colors suffice to color a map (or equivalently the nodes of a planar, non-multi graph).

Proof. We prove the proposition by induction.

Base case. For 6 or less nodes the proposition trivially holds.

Induction hypothesis. We assume that the proposition holds for planar graphs with at most k nodes.

Induction step. We show that, given the induction hypothesis, the proposition holds also for graphs with $k + 1$ nodes. The plan is as follows:

- Eliminate a node v.
- Use the induction hypothesis to conclude that there exists a coloring for the reduced graph.
- Find a suitable coloring for the eliminated node v.

While the first two points are clear and unproblematic, we ask whether there is always a suitable coloring of v. One way to guarantee that is to choose v such that it has only 5 neighbors (or less); then, there is always one color possible (out of the 6). And indeed, we have the following fact, following from the planarity of the graph.

Lemma 1.1. *In every planar graph, there exists at least one node v with at most 5 neighbors.*

Proof of Lemma. By *contradiction*: Assume that the statement is false, i.e., that each node has at least 6 neighbors. We derive a contradiction to Euler's formula.

As every node has at least 6 neighbors, and each edge connects exactly two nodes, the edges and the nodes are related as follows:

$$2e \geq 6n \ .$$

The number of regions and edges are related similarly: Every region is bounded by at least 3 edges; every edge limits at most two regions, so:

$$2e \geq 3f \ .$$

Putting this together, we obtain

$$n + f \leq \frac{e}{3} + \frac{2}{3}e = e \quad \Rightarrow \quad n + f - e \leq 0 \ .$$

This is in contradiction with Euler's formula and completes the proof. □

Since we can always find a node v with 5 or less neighbors to reduce a graph with $k + 1$ nodes to a graph with k nodes, this completes the induction step, and the proof of the proposition.

□

1.5 Exercises

In the following, the reader can find some exercises to apply the basic concepts shown in this chapter, including the proof techniques of induction and contradiction (cf. Appendix **??**).

Exercise 1.1 (Chess and Domino). A chessboard is composed by 8×8 small squares, each of which exactly corresponds to half a domino stone. Clearly, the board can now be covered by 32 domino stones. We now take away from the chessboard two diagonally opposite corner squares (i.e., two squares that share one vertex but no edge). Can the rest be covered by exactly 31 domino stones? Give such a covering or proof that it cannot exist.

Exercise 1.2 (Rubik's Cube in Walnut). You would like to saw a wooden cube into $3 \times 3 \times 3 = 27$ small cubes. The obvious solution requires *six* cuts. Now, you are allowed to rearrange, after each cut, the obtained parts. More precisely, you can staple them on top of each other, and then perform a straight cut through the entire tower of pieces. Can the number of required cuts be reduced in this way? In other words, can you find a procedure that allows for cutting the cube with five cuts, or can you find an argument that such a thing is impossible?

Exercise 1.3 (Shake Hands). Show that in any group of six (or more) people, at least one of the following two statements is always true:

- There are three people who have never shaken hands with each other.
- There are three people who all already have shaken hands with each other.

Try to solve this problem by modeling the situation as a graph.

Exercise 1.4 (Proof Techniques). This exercise aims to familiarize the reader with two basic techniques for mathematical proofs: proofs by inductions and proofs by contradiction (cf. Appendix **??**).

1. Prove by induction that for all $n \in \mathbb{N}$, $\sum_{i=1}^{n} (2i - 1) = n^2$.
2. What is wrong with the following induction proof?
   Statement: All people have the same hair color. We prove that the following statement holds for all n: In a group of n people, all have the same hair color.
   Base case: For $n = 1$, this is obviously true. In a "group" of a single person, clearly there is only one hair color.
   Induction hypothesis: We assume that the statement is true for n.
   Induction step: Consider a group of $n + 1$ people. Let A and B be two different subsets of this group, each of size n. By the induction hypothesis, both A and B must consist of people having the same hair

color. Now, because of the people in both groups, the two hair colors in the two groups must be identical, too. Consequently, all $n+1$ must have the same hair color. QED.

3. Prove by contradiction that $\sqrt[3]{2}$ is not a rational number.

4. Prove that there exist irrational numbers s and t such that s^t is rational.

   Hint. Start by considering the number $\sqrt{2}^{\sqrt{2}}$ and distinguish different cases.

Exercise 1.5 (Sylvester – hard). Consider n points in the plane with the following property: Whenever we draw a line connecting two of the points, then at least one additional point must lie on the line. Prove that in this case, *all* points must lie on *one line*.

Hint. Try to prove it by contradiction: Assume that the statement is false (with the goal of deriving a contradiction from it). If the assumption is false, then there must be points that do not lie on the line connecting some other points. Consider now the one point that lies *the closest* to such a line *without* actually lying *on* it. Do not forget our assumption: On every connection, there is a third point!

Chapter 2

Propositional Logic

In this chapter, we discuss *propositional logic* (PL), a branch of logic concerned with propositions and, in particular, *how composed propositions can be obtained from atomic ones, and how their truth value is determined by the one of their parts.*

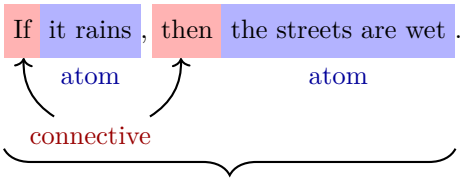
The chapter follows closely the book by *Uwe Schoening*: "Logic for Computer Scientists."

2.1 Definitions

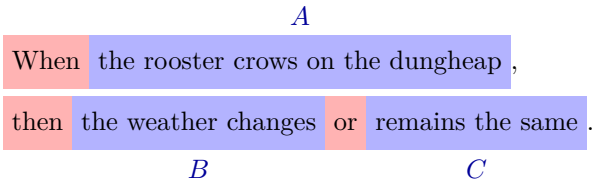
Logic has been described as the *realm of true and false* in the same sense that ethics is the one of *good and bad* or aesthetics of *beautiful and ugly.*

Definition 2.1 (Proposition, Atom, Connective)**.** A *proposition* is a sentence, expression, or formula which is either *true* or *false*, *i.e.*, is *truth-definite.* An *atom* or *atomic proposition* is a basic proposition that is not composed of other propositions. A *connective* links (generally) two propositions to a new proposition.

Example 2.1. The examples show how *atoms* (or atomic proposition) can be connected by connectives to form a composed proposition:



The following proposition connects three atoms A, B, C:



The conditioning atom is without effect as the second proposition is always true. The atom C is just the negation of the atom B: $C = \neg B$. Therefore $C \vee B = (\neg B) \vee B = \texttt{true}$. Such a proposition that is always true due to its form is called a *tautology*.

Propositions can be confusing — and lead to logical *paradoxa* or *antinomies* — when they refer to themselves. The following is a proposition, and it is *false*:

This is not a proposition .

This one sounds like, but does not seem to be, a proper proposition as one can, in principle, not decide whether it is *true* or *false*:

This proposition is false .

The latter example makes think of the "liar antinomy": *Epimenides* from Crete says: "All Cretans are liars." So are they?

2.1.1 Connectives as truth functions

A *connective* is a function applied to propositions (or their truth values), yielding another. They can be fully characterized by a *truth table*. As truth can take one of *two* values (*true* and *false*), just like *bits* in a computer, there is a close relation between connectives and *logical gates* (the basic building blocks of a processor).

Here are some examples of connectives.

Conjunction. The AND gate is the connective that returns true if and only if both arguments are true:

A	B	$A \wedge B$
`true`	`true`	`true`
`true`	`false`	`false`
`false`	`true`	`false`
`false`	`false`	`false`

Negation. The NOT gate returns the opposite of the input truth value:

A	$\neg A$
`true`	`false`
`false`	`true`

Together, the AND and the NOT gates allow for realizing any other gate: This set of gates is *universal*. Interestingly, just one single gate — the NAND — is already universal all by itself.

NAND (NOT AND). The two gates above can be combined into a new gate, the negated AND gate, short NAND gate:

$$A \mid B := \neg(A \wedge B) \ .$$

The truth table is the inverted AND table:

A	B	$A \mid B$
`true`	`true`	`false`
`true`	`false`	`true`
`false`	`true`	`true`
`false`	`false`	`true`

To get an idea of how the NAND gate can serve to realize all other gates, one obtains the NOT gate by using A for both inputs: $\neg A$ ' = ' $A \mid A$. One can now use this NOT gate and the NAND to obtain an AND gate, and so on.

Example 2.2. NAND is used when considering two mutually exclusive options, that might be false at the same time:

- $A =$ "This polygon has exactly 4 edges."
- $B =$ "This polygon has exactly 5 edges."
- $A|B =$ "The number of edges of this polygon is NOT equal to 4 AND 5 *at the same time*."

Inclusive disjunction. The OR gate returns *true* whenever *at least one* of the two inputs is true:

A	B	$A \vee B$
`true`	`true`	`true`
`true`	`false`	`true`
`false`	`true`	`true`
`false`	`false`	`false`

Exclusive disjunction. The exclusive OR or XOR gate returns *true* if and only if *exactly one* of the two arguments is *true*:

A	B	$A \oplus B$
`true`	`true`	`false`
`true`	`false`	`true`
`false`	`true`	`true`
`false`	`false`	`false`

$\Leftrightarrow$

A	B	$A \oplus B$
1	1	0
1	0	1
0	1	1
0	0	0

The XOR symbol resembles a *plus sign* for a reason: If we replace `false` by zero and `true` by 1 — a notation we adopt from now on — then the XOR is just addition modulo 2.

Example 2.3 (Disjunction). In everyday language both inclusive and exclusive disjunction are generally expressed by *or*.

- $A =$ "Applicants for this job should have a PhD."
- $B =$ "Applicants for this job should have teaching experience."
- $A \vee B =$ "Applicants for this job should have a PhD OR teaching experience." (a person having both would obviously not been discarded)

It is possible to clarify that one means an exclusive disjunction by using "either. . . or."

- $A =$ "Billy is a cat."
- $B =$ "Billy is a dog."
- $A \vee B =$ "Billy is EITHER a cat OR a dog."

Logical equivalence. This is the "inverse" of the XOR, returning *true* (1) if and only if the arguments have the same truth value:

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

We have

$$A \leftrightarrow B \text{ ‘}=\text{’ } \neg(A \oplus B) \ ;$$

we leave it open for the moment in which exact sense they are "equal" (as strings, they are not).

Implication. The *implication* from A to B is *true* whenever the fact that A is *true* unavoidably means that also B is *true*. Here, A is called the *premise* and B the *conclusion*.

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

(2.1)

Conversely, $A \to B$ does not mean that if B is *true* then also A is:

$$A \to B \text{ ‘} \neq \text{’ } B \to A \ :$$

Implication is asymmetric. Also, it is not true that negations $\neg$ can simply be added:

$$A \to B \text{ ‘} \neq \text{’ } (\neg A) \to (\neg B) \ .$$

However, the *contraposition law* always holds:

$$A \to B \text{ ‘} = \text{’ } (\neg B) \to (\neg A) \ .$$

This asymmetry requires that one be particularly careful how to prove a statement: The line of reasoning cannot simply be inverted. (If an obvious truth, for instance “$0 = 0$,” is shown to imply a statement S, then S is proven. If, on the other hand, S is shown to imply “$0 = 0$,” this is actually no valid argument for the truth of S.)

However, if both implications hold, then we obtain *equivalence*:

$$(A \to B) \wedge (B \to A) \text{ ‘} = \text{’ } A \leftrightarrow B \ .$$

It is possible to prove statements of equivalence by separately showing both implications.

What is the meaning of ‘=’? The expressions on the right and on the left-hand sides of the “equalities” were not the same on the level of *strings*: They are *syntactically different*. However, their truth value is always the same, for every truth assignment of the involved atoms: they are *semantically equivalent*. We write $\equiv$ instead of $=$ from now on.

Let us look at *syntax* and *semantics* of propositional logic separately.

2.2 Syntax of propositional logic

In *syntax* (of logic as well as of a programming language), we specify what a correct string (formula, program) is, independently of its respective “meaning” or “function.” The following recursive definition specifies what is a correct formula: The *atomic formulas* are, and the ways are given in which formulas can be composed.

Definition 2.2 (Syntax)**.** Syntactically correct formulas $\mathcal{E}_\mathcal{D}$ with a set of atoms $\mathcal{D} := \{A, B, C, \ldots\}$ are

- atomic formulas in $\mathcal{D}$;
- if f and g are syntactically correct formulas, then also $(\neg f)$, $(f \wedge g)$ and $(f \vee g)$ are syntactically correct.

These are all syntactically correct formulas.

In short, the only permitted formulas for now are atoms, as well as compositions using $\neg, \wedge$, and $\vee$. Note that the other connectives listed above are not explicitly part of the syntax, but are in fact implicitly included: We already mentioned that the "allowed ones" are *universal*.

We visualize Definition 2.2 in a *syntax diagram* (see Figure 2.1).

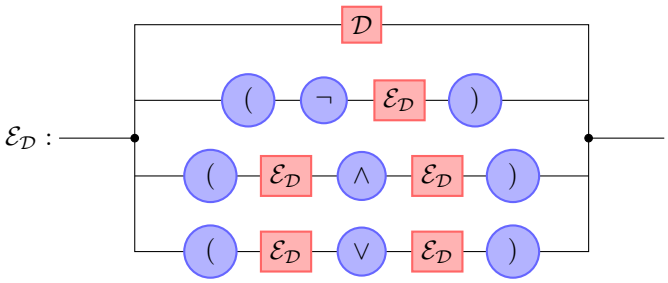


Figure 2.1: The *syntax diagram* for propositional logic.

Example 2.4. The following are syntactically correct formulas. (Note that no brackets can be left away so far, according to the definition.)

- A
- $(\neg A)$
- $(A \wedge B)$
- $(A \vee B)$
- $(\neg(A \wedge B))$
- $F := (((A \wedge B) \vee (\neg C)) \wedge (A \vee B))$

The (structure of the) latter formula, F, can be visualized in a graph called a *syntax tree* (see Figure 2.2). Here, subtrees correspond to *partial formulas* (substrings which are formulas in their own right). The partial formulas of F are

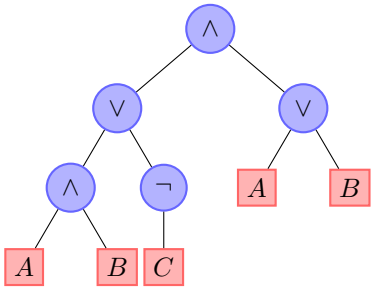


Figure 2.2: The *syntax tree* of the formula F.

$$\{A, B, C, (A \wedge B), (\neg C), ((A \wedge B) \vee (\neg C)), (A \vee B), F\}$$

According to our definition, the following formulas are *not* correct (as brackets are missing):

- $A \wedge B \wedge C$
- $A \wedge B \vee C$

Whether it makes sense to allow for omitting these brackets or not depends on whether the truth value depends on how the brackets are placed. Thus, only "truth calculus" *semantics* can decide such questions.

2.3 Semantics of propositional logic

Definition 2.3 (Assignment)**.** A *truth assignment* $\mathcal{A} : \mathcal{D} \to \{0, 1\}$ is a function that assigns a truth value to every atom. This *truth function* is extended to all syntactically correct formulas by simply evaluating the formula, based the truth values for its atoms and the connectives used.

Example 2.5 (Truth Assignment and Function)*.* Let us consider a set of three atoms A, B, C. A *truth assignment* is, for instance,

$$\begin{aligned} \mathcal{A} : A &\mapsto 0 \\ B &\mapsto 1 \\ C &\mapsto 0. \end{aligned}$$

The function is extended to formulas: For $F = (((A \wedge B) \vee (\neg C)) \wedge (A \vee B))$ (as above), the assignment yields $\mathcal{A}(F) = 1$, as can be easily seen using the given syntax tree, evaluating bottom-up.

Definition 2.4 (Semantic Behavior / Truth Vector)**.** The *semantic behavior* or *truth vector* of a proposition is the list of truth values for all possible assignments.

This behavior or vector can be expressed completely in a *truth table*.

Example 2.6. For the formula F in the example above, we can evaluate the tree bottom-up. If we wish a full list of the results for all possible inputs, it is more efficient to evaluate the subformulas and write the result below the connective:

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

(2.2)

We say that two formulas are *semantically equivalent* if they yield the same output on the same input. Equivalent formulas can be exchange against each other in any logical context; they are "logically the same."

Definition 2.5 (Semantic Equivalence)**.** Two formulas are *semantically equivalent* if they have the same truth value for all assignments of their atomic formulas. We write $F \equiv G$ or $F \Leftrightarrow G$.

Example 2.7 (Equivalent Formulas). The following two formulas are equivalent and yield the same red columns:

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

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

Thus,

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

We introduce the "truth values" 0 and 1 as syntactically correct formulas, abbreviating (the shortest) unsatisfiable formula and tautology, respectively.

Definition 2.6. We define the formulas

$$0 :\equiv (A \wedge (\neg A)) \ ,$$
$$1 :\equiv (A \vee (\neg A)) \ .$$

Definition 2.7 (Tautology, Unsatisfiable Formula)**.** If a formula F is semantically equivalent to 0, $F \equiv 0$, then F is called *unsatisfiable*.

If a formula F is semantically equivalent to 1, $F \equiv 1$, then F is called a *tautology*.

Similarly, we introduce additional connectives as abbreviations: For instance, the XOR:

$$A \oplus B :\equiv ((A \wedge (\neg B)) \vee ((\neg A) \wedge B)) \ .$$

This is the equivalence connective:

$$A \leftrightarrow B \ :\equiv \ (A \wedge B) \vee ((\neg A) \wedge (\neg B)) \ .$$

Semantic equivalence is an *equivalence relation*: It partitions the set of all formulas into disjoint subsets, groups of semantically equivalent formulas, the *equivalence classes*. It structures the set of all syntactically correct formulas as visualized in Figure 2.3. For instance, the formulas $(B \vee (\neg B)), (\neg(\neg A) \vee (\neg(\neg(\neg A))))$ are all equivalent (to 1, actually) and, therefore, in the same equivalence class: The class of all tautologies.

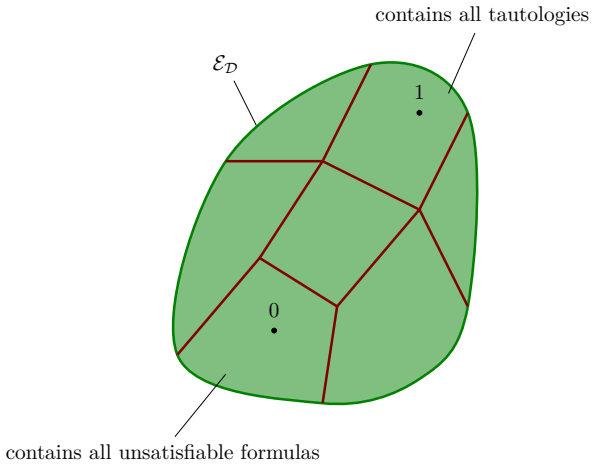


Figure 2.3: Equivalence classes in the set of all syntactically correct formulas $\mathcal{E}_\mathcal{D}$.

Number of equivalence classes. The set of all syntactically correct formulas $\mathcal{E}_\mathcal{D}$ is infinite. For instance, it contains the sequence of formulas A, $(\neg A)$, $((\neg(\neg A))$, etc. The number of equivalence classes, however, is *finite* (if the number of atomic formulas is). If we assume, for instance, that there are 26 atomic formulas $\mathcal{D} = \{A, B, C, \ldots, Z\}$, then there are 2^{26} different input configurations (or lines if we wrote it as in 2.2). As each line specifies an entry (0 or 1) of the truth vector, there are $2^{(2^{26})}$ different semantic behaviors, i.e., equivalence classes: a finite but *very* large number.

We express the semantic equivalence of *two* formulas through a property of *one single* formula.

Theorem 2.1. *Two formulas F and G are semantically equivalent, i.e., $F \Leftrightarrow G$, if and only if the formula $F \leftrightarrow G$ is a tautology.*

Note that $\leftrightarrow$ connects F and G syntactically, whereas $\Leftrightarrow$ connects the two semantically.

Proof. The formula $F \leftrightarrow G \equiv (F \wedge G) \vee ((\neg F) \wedge (\neg G))$ is a tautology if and only if F and G have the same truth values for all assignments. Thus, it is a tautology if and only if F and G are semantically equivalent. □

Example 2.8 (Logical Laws). The following list contains important examples of semantic equivalences:

- Idempotence
$$(F \wedge F) \equiv F \qquad (F \vee F) \equiv F$$
- Symmetry
$$(F \wedge G) \equiv (G \wedge F) \qquad (F \vee G) \equiv (G \vee F)$$
- Associativity
$$(A \wedge (B \wedge C)) \equiv ((A \wedge B) \wedge C) \qquad (A \vee (B \vee C)) \equiv ((A \vee B) \vee C$$
- Absorption
$$((F \wedge G) \vee F) \equiv F \qquad ((F \vee G) \wedge F) \equiv F$$
- Distributivity
$$(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H)) \qquad (F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H)$$
- de Morgan
$$(\neg(F \wedge G)) \equiv ((\neg F) \vee (\neg G)) \qquad (\neg(F \vee G)) \equiv ((\neg F) \wedge (\neg G))$$

- Double negation
$$(\neg(\neg F)) \equiv F$$

The de Morgan laws give rise to a duality in the set of laws: Take any law, swap the AND and OR, and you get another valid law.

Note that there is also distributivity for AND and XOR:

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

This does, however, not hold if the two connectives are swapped (just as in arithmetic, there is a distributive law: $a(b+c) = ab + ac$, but *not* $a + bc = (a+b)(a+c)$ in general). In fact, XOR and AND can be sees as addition and multiplication of logic.

Simplification of notation. So far, we have been sticking closely to the syntax permitted by Definition 2.2. We introduce some simplifications, motivated by the equivalences above:

- We allow all connectives introduced in the first part ($\oplus$, $\rightarrow$, $\leftrightarrow$), as they are equivalent to formulas with the basic connectives.
- If brackets do not change the truth behavior, they can be dropped. For instance, because of associativity, we can allow $A \wedge B \wedge C$. In general, we write
$$\bigwedge_{i=1}^{n} A_i \equiv A_1 \wedge A_2 \wedge \ldots \wedge A_n \ , \qquad \bigvee_{i=1}^{n} A_i \equiv A_1 \vee A_2 \vee \ldots \vee A_n \ .$$
- We introduce the following *priority rules* (operator precedence): $\neg, (\wedge, \vee), (\oplus, \leftarrow, \rightarrow, \leftrightarrow)$. Again brackets can be dropped and the formula is read in accordance with these priority rules. For instance,
$$A \wedge \neg B \rightarrow C$$
stands for
$$(A \wedge (\neg B)) \rightarrow C \ .$$

2.4 Normal forms

Given a syntactically correct formula, one obtains its semantic behavior by writing down the truth table. Is it conversely possible to construct a syntactically correct formula that reproduces *any given truth vector*? The following example shows how this can be done.

Example 2.9. Let us consider the truth table with the atoms A, B, and C. We want to find a composed formula F that produces the truth vector in the last column:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

One way to construct a formula F with the desired semantics is to consider all the rows that have to be *true*: The formula can express that "we are" in row 3, row 4, row 5, row 6, *or* row 8. "To be in" a certain row means that the atoms (or their negation in case of $\neg$) are *all* true:

$$\begin{aligned} F &\equiv (\text{row } 3) \vee (\text{row } 4) \vee (\text{row } 5) \vee (\text{row } 6) \vee (\text{row } 8) \\ &\equiv (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \\ &\qquad \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge C) \end{aligned}$$

We call this disjunction of conjunctions (OR or ANDs) a *Disjunctive Normal Form (DNF)*. Applying the logical rules above, we can reduce the formula to a simpler, semantically equivalent one: Employing associativity we obtain

$$(\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \equiv (\neg A \wedge B) \vee \underbrace{(\neg C \wedge C)}_{\equiv 0} \equiv \neg A \wedge B \ .$$

We obtain

$$\begin{aligned} &\underbrace{(\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge C)}_{\equiv \neg A \wedge B} \vee \underbrace{(A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C)}_{\equiv A \wedge \neg B} \vee (A \wedge B \wedge C) \\ &\quad \equiv \underbrace{(\neg A \wedge B) \vee (A \wedge \neg B)}_{\equiv A \oplus B} \vee (A \wedge B \wedge C) \equiv A \oplus B \vee A \cdot B \cdot C \end{aligned}$$

using multiplication as an abbreviation of the AND.

A second approach to finding a formula F is to consider the *false* rows: We are not in row 1 *and* not in row 2 *and* not in row 7. Each of these rows is false if *any* of the subformulas is false. Thus, we connect the atoms (or their negations) by ORs:

$$\begin{aligned} F &\equiv \neg(\text{row } 1) \wedge \neg(\text{row } 2) \wedge \neg(\text{row } 7) \\ &\equiv (A \vee B \vee C) \wedge (A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee C) \ . \end{aligned}$$

This conjunction of disjunctions (AND of ORs) is called a *Conjunctive Normal Form (CNF)*. Using associativity, we simplify again, to

$$F \equiv (A \vee B) \wedge (\neg A \vee \neg B \vee C) \ ,$$

where we used the distributive property twice to obtain the second and third equivalences. The DNF and CNF obtained for F are syntactically different but, due to construction — we started out from the same truth vector — semantically equivalent.

DNF

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

CNF

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

Figure 2.4: Truth tables of the DNF and CNF from Example 2.9.

We define the normal forms CNF and DNF.

Definition 2.8 (Literal)**.** If $A \in \mathcal{D}$ is an atom then A and $\neg A$ are called literals: A literal is an atom or a negated atom.

Definition 2.9 (Conjunctive Normal Form)**.** A formula F is in *Conjunctive Normal Form (CNF)* if there exist literals $L_{i,j}$ such that

$$\begin{aligned} F &= \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} L_{i,j} \right) \\ &= (L_{1,1} \vee L_{1,2} \vee \ldots \vee L_{1,m_1}) \\ &\quad \wedge (L_{2,1} \vee \ldots \vee L_{2,m_2}) \\ &\quad \wedge \ldots \\ &\quad \wedge (L_{n,1} \vee \ldots \vee L_{n,m_n}) \ . \end{aligned}$$

Definition 2.10 (Disjunctive Normal Form)**.** A formula F is in *Disjunctive Normal Form (DNF)* if there exist literals $L_{i,j}$ such that

$$F = \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} L_{i,j} \right) .$$

Note that, in these definitions, we put equalities ($=$), not merely equivalences ($\equiv$): The normal forms are also *syntactic* notions.

2.5 Models and semantic conclusion

Physics asks: In our world, what formulas are true? In logic, we often take the opposed stand: For a given formula, what is the set of worlds in which it is true. Such a world is called a *model* for the formula; it makes it *true*. In the case of propositional logic: For a given formula, what truth assignments make it true?

Definition 2.11 (Model)**.** Let F be a formula and $\mathcal{A}$ an assignment which renders F true. Then $\mathcal{A}$ is a *model* of F. We write

$$\mathcal{A} \vDash F .$$

Example 2.10. Let us consider the formula F and its truth table:

$F = (A$	$\wedge$	$B)$	$\rightarrow$	C
0	0	0	1	0
0	0	0	1	1
0	0	1	1	0
0	0	1	1	1
1	0	0	1	0
1	0	0	1	1
1	1	1	0	0
1	1	1	1	1

Only the assignment in the seventh row is *not* a model of F.

Some of the properties of *models* are the following:

- Two formulas F and G are semantically equivalent, i.e., $F \equiv G$, if and only if they have the same models:

$$\mathcal{A} \vDash F \quad \text{iff} \quad \mathcal{A} \vDash G$$

- A formula F is a tautology if and only if *every* assignment $\mathcal{A}$ is a model for F (F is true in any "thinkable" world).

- A formula F is unsatisfiable if and only if there it has *no* model.

Based on the notion of models, we define the one-sided variant of semantic equivalence.

Definition 2.12 (Semantic conclusion)**.** G is a *semantic conclusion* of F_1, ..., F_n if every model $\mathcal{A}$ of all the F_i is also a model of G. We say that $F_1, \ldots, F_n$ semantically implies G. We write $\{F_a, \ldots, F_n\} \models G$ or $\{F_a, \ldots, F_n\} \Rightarrow G$.

Semantic conclusion is the one-sided variant of semantic equivalence (they relate just like $\rightarrow$ and $\leftrightarrow$ on the syntactical level). In particular, two formulas F and G are semantically equivalent if and only if F is the semantic conclusion of G, and *vice versa*.

Note that the symbol $\models$ has two meanings: It indicates semantic conclusion as well as models.

Example 2.11. Let us take a closer look at the following set of formulas:

$$\{A, A \rightarrow B, B \rightarrow C\} \ .$$

Do they semantically imply C? The only model of the first formula is the assignment $\mathcal{A}_0(A) = 1$. The models of the second are (corresponding to the rows 1,2 and 4 in 2.1)

$$\begin{aligned} \mathcal{A}_1 &: A \mapsto 0, B \mapsto 0 \\ \mathcal{A}_2 &: A \mapsto 0, B \mapsto 1 \\ \mathcal{A}_3 &: A \mapsto 1, B \mapsto 1 \ , \end{aligned}$$

and similarly for the last formula:

$$\begin{aligned} \mathcal{A}_4 &: B \mapsto 0, C \mapsto 0 \\ \mathcal{A}_5 &: B \mapsto 0, C \mapsto 1 \\ \mathcal{A}_6 &: B \mapsto 1, C \mapsto 1 \ . \end{aligned}$$

So only the assignment

$$\tilde{\mathcal{A}} : A \mapsto 1, B \mapsto 1, C \mapsto 1$$

is a model that renders all formulas true, i.e., a common model for all formulas. Since this is also a model for the atomic formula C we obtain

$$\{A, A \rightarrow B, B \rightarrow C\} \models C \ .$$

The semantic conclusion is related to the syntactical implication just as semantic to syntactic equivalence (Theorem 2.1):

Theorem 2.2. *A set of formulas $\{F_1, \ldots, F_2\}$ semantically implies a formula G — $\{F_1, \ldots, F_n\} \vDash G$ — if and only if the formula $\bigwedge_{i=1}^{n} F_i \to G$ is a tautology.*

We carefully distinguish between the syntactical and the semantic levels: The expression $\{F_1, \ldots, F_n\} \vDash G$ relates the set of formulas $\{F_1, \ldots, F_n\}$ *semantically* with the formula G. The expression $\bigwedge_{i=1}^{n} F_i \to G$ connects the two *syntactically* and yields another syntactically correct formula. Only by demanding this formula to be a tautology does a semantic criterion emerge.

Proof. The implication $A \to B$ is equivalent to $\neg A \vee B$. Therefore,

$$\begin{aligned} \bigwedge_{i=1}^{n} F_i \to G &\equiv \neg(F_1 \wedge F_2 \wedge \ldots \wedge F_n) \vee G \\ &\equiv \neg F_1 \vee \neg F_2 \vee \ldots \vee \neg F_n \vee G \ . \end{aligned}$$

This formula being a tautology means: If an assignment renders all F_i true — $\mathcal{A}(F_i) = 1 \ \forall i$ — then the assignment must also make G true. But that is exactly the definition of a semantic conclusion. □

Remark 1. A formula F is a tautology if and only if F is a conclusion of 1, i.e., $1 \vDash F$: F is a tautology if and only if the implication $1 \to F$ is a tautology.

A formula F is unsatisfiable if and only if 0 is a conclusion of F, i.e., $F \vDash 0$: F is unsatisfiable if the implication $F \to 0$ is a tautology.

2.6 Proof theory of propositional logic

The goal of a proof theory is to decide semantic questions such as

- Is F a tautology?
- Is F unsatisfiable?
- Does $\{F_1, \ldots, F_n\} \vDash G$ hold?

Let us first of all observe that this is, in the end, always the same type of question, which can be made to boil down to the question whether a certain formula is a *tautology* or not. Let us thus look at different flavors of that particular question.

Example 2.12 (Tautology Problem of CNF). Given a formula F in CNF, how can we decide whether it is a tautology? Since F is in CNF, it is a *conjunction* (AND) of subformulas F_i, which are themselves *disjunctions* (OR) of literals:

$$F = \underbrace{(L_{1,1} \vee \ldots L_{1,m_1})}_{=:F_1} \wedge \ldots \wedge \underbrace{(L_{n,1} \vee \ldots L_{n,m_n})}_{=:F_n} = F_1 \wedge \ldots \wedge F_n \ .$$

Observe first that a conjunction of F_i's is a tautology if and only of all the F_i are. Second, the F_i's are tautologies if at least one atom appears twice therein, once unnegated and once negated. (Otherwise, a violating assignment can always be found.)

Example 2.13 (Tautology Problem of DNF). Can we derive a similarly simple criterion for a DNF-formula F:

$$F = \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} L_{i,j} \right) = F_1 \vee \ldots \vee F_n \ ?$$

Is F is a tautology or not? Unfortunately, the problem does not *"localize"* (reduce to similar questions independently for the subformulas) in the same sense: The *relation* between subformulas is important here. Clearly, what can always be done is a *truth table*. However, the size of that grows *exponentially* with the number of different atoms in the formula.

Relating DNF and CNF. Is it easy to change from *one* normal form to the other, for a formula F? No, but what we *can* do is to obtain the *negation* of F in the *other* normal form in which F itself is given, using *de Morgan*'s law:

$$\begin{aligned} \neg F = \neg \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} L_{i,j} \right) &\equiv \bigwedge_{i=1}^{n} \left(\neg \bigwedge_{j=1}^{m_i} L_{i,j} \right) \\ &\equiv \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} \neg L_{i,j} \right) \ . \end{aligned}$$

As F being a tautology is equivalent to $\neg F$ being unsatisfiable, the *satisfiability* problem for a DNF can be solved analogously to the *tautology* problem of a CNF: They are both simple. The other problems (satisfiability for CNF, tautology for DNF, change from CNF to DNF and *vice versa* for some F) seem hard. Let us look at this is some more detail.

Comparison of computational hardness. Let us compare the hardness of the tautology problems for CNF and DNF, respectively: If F is given in CNF, one merely has to check the double occurrence of an atom (once positive, once negated) in each of the subformulas. This can be done in essentially *linear time*, i.e., the number of steps being upper-bounded by a linear function in the length of the formula), simply going through the formula. On the other hand, in the case of F being a DNF, one has to write down the entire truth table, containing 2^l rows. The number of computational steps is thus *exponential* in the length l, growing much faster. In a sense. the two problems are opposite extremal cases of hardness. Let us look at that in some more depth.

2.6.1 Short excursion into complexity theory

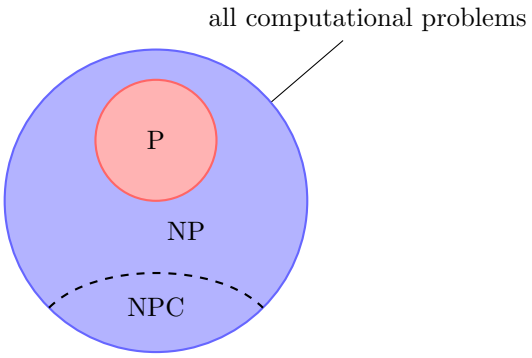


Figure 2.5: The set of computational problems is divided into classes of different complexity.

Complexity theory is about classifying computational problems by their computational difficulty. The set P contains all problems that can be solved in polynomial time, i.e., the number of computational steps being at most a polynomial function (such as x, x^2, $20x^{100}$, ...) of the input size. P is a subset of NP, containing all problems for which one can *verify* a given solution in polynomial time, whereas the solution may not be found in polynomial time.

A subset of NP are the NP-complete problems (NPC): Problems in NPC are as hard as *any* other problem in NP. That means that *any* problem in NP can be *reduced* to any problem in NPC in polynomial time. Consequently, any problem in NPC can be reduced to another problem in NPC in polynomial time. This means that NPC problems can be used as *indicators* of simplicity for *all* NP problems: If you can solve *one* NPC problem in polynomial time, then you can solve *all* of them. (In fact, you have then shown P=NP=NPC, which would be an overly surprising result that makes you very famous; most people believe that the three classes P, NP, and NPC are all different from each other — no one has proven that so far, however.)

Example 2.14 (Elements of P). P contains

- the tautology problem for CNF;
- the (un)satisfiability problem for DNF.

Even more, these decision problems can be solved not only in polynomial but even in *linear* time; they are among the simplest even among P problems.

Example 2.15 (Elements in NPC). NPC contains

- the tautology problem for DNF;

- the satisfiability problem for CNF
- the problem to find a semantically equivalent DNF for a given CNF, and *vice versa*.

The last follows from the first two: If we could find a G in CNF for any F in DNF with $F \equiv G$ in polynomial time, we could solve the tautology problem for a DNF in polynomial time by first turning it into a CNF and then applying the criteria above.

Real-life problems? Are problems in real life hard? Yes, they are, as one can see in the following example.

Example 2.16 (Sudoku). *Sudoku* is an NPC problem, as it boils down to the satisfiability of a CNF: As one has to find a solution satisfying *all* conditions on the rows, columns, and subboxes, it is a problem of the form

$$(\text{condition 1}) \wedge (\text{condition 2}) \wedge \dots \tag{2.3}$$

Each of these conditions can be satisfied in different ways:

$$\text{condition 1} = \text{way 1} \vee \text{way 2} \vee \text{way 3} \vee \dots$$

This is a CNF; to find a solution is the same as proving *satisfiability*. The satisfiability problem for a CNF is in NPC; so is *Sudoku*.

Actually, this is a common structure of real-life problems, such as flight plans or schedules. Usually there is a number of necessary conditions yielding a formula of the form 2.3. Each condition can be met in various ways, so the subformulas are disjunctions. Thus, one is usually looking for an assignment satisfying a CNF.

2.7 The resolution calculus

Although there is no hope to solve satisfiability (SAT) for CNF formulas efficiently *always*, we discuss here a calculus that can do it *often*: *Resolution*.

Example 2.17. The task is to prove the following semantic conclusion

$$\{A, A \to B, B \to C, C \to D, D \to E\} \vDash E \ .$$

Employing Theorem 2.2, the task turns into showing that the formula

$$\underbrace{(A \wedge (A \to B) \wedge (B \to C) \wedge (C \to D) \wedge (D \to E))}_{=:F} \to E$$

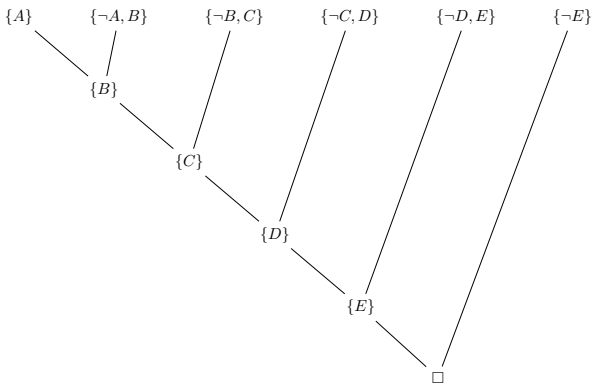


Figure 2.6: Deducing a contradiction from the clauses.

is a tautology. Remember $F \to E = \neg F \vee E$. We turn that tautology problem into the *unsatisfiability* problem for

$$\neg(\neg F \vee E) \equiv F \wedge \neg E \ .$$

Explicating F yields a CNF:

$$A \wedge (\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee D) \wedge (\neg D \vee E) \wedge \neg E \ .$$

The subformulas of a CNF are often called *clauses* and written as sets, containing the literals of the clause:

$$\{\{A\}, \{\neg A, B\}, \{\neg B, C\}, \{\neg C, D\}, \{\neg D, E\}, \{\neg E\}\} \ .$$

We now show by contradiction that there does not exist an assignment $\mathcal{A}$ that renders all clauses true: Let us assume that such an assignment $\mathcal{A}$ does exist. We deduce further clauses from the existing ones which are also satisfied by that same assignment: If the assignment $\mathcal{A}$ renders all clauses true, it must hold that $\mathcal{A}(A) = 1$. Thus $\neg A$ is false, and the second clause can be true only if $\mathcal{A}(B) = 1$. This argument is repeated, all together, and they are represented in the tree of Figure 2.6. In the last step, we obtain that the assignment must make both E and $\neg E$ true — a contradiction: Our initial assumption was wrong, and the assignment in question cannot exist in the first place. We have shown the formula to be unsatisfiable.

Example 2.18. Prove the semantic conclusion

$$\{A \vee B, A \to C, A \to C, B \to C\} \vDash C \ .$$

This means showing that the formula

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

is unsatisfiable. Figure 2.7 shows the clauses and their resolution.

We explain the rationale of a resolution step by the following example:

Example 2.19. Every assignment that renders both $\{A, B, \neg C\}$ and $\{\neg A, \neg E\}$ true must also render $\{B, \neg C, \neg E\}$ true. To prove this we have to distinguish the cases

1. $\mathcal{A}(A) = 0$: Then $\{B, \neg C\}$ must be true and, therefore, also $\{B, \neg C, \neg E\}$.
2. $\mathcal{A}(A) = 1$: Then $\{\neg E\}$ must be true and, therefore, also $\{B, \neg C, \neg E\}$.

Whatever truth value A is assigned, the clause $\{B, \neg C, \neg E\}$ is always true.

Definition 2.13 (Resolution Step)**.** Let C_1, C_2, and R be clauses. R is the *resolvent* of C_1 and C_2 if there exists a literal L such that L is in C_1 and $\neg L$ is in C_2 and R contains all literals in C_1 and C_2 except of L and $\neg L$.

In set notation:

$$F = C_1 \setminus \{L\} \cup C_2 \setminus \{\neg L\} \ .$$

The rationale is that C_1 and C_2 semantically imply R

$$\{C_1, C_2\} \vDash R \ .$$

The following examples show that resolution does not merely show unsatisfiability but also yields an assignment if the formula is satisfiable.

Example 2.20. Is the following formula satisfiable?

$$F = (A \vee B \vee \neg C) \wedge (\neg A \vee \neg E) \wedge (\neg C \vee D \vee E) \wedge C \wedge (\neg D \vee \neg C) \ .$$

From the resolution in Figure 2.8, one obtains a model

$$\begin{aligned} \mathcal{A} : A &\mapsto 0 \\ B &\mapsto 1 \\ C &\mapsto 1 \\ D &\mapsto 0 \\ E &\mapsto 1 \end{aligned}$$

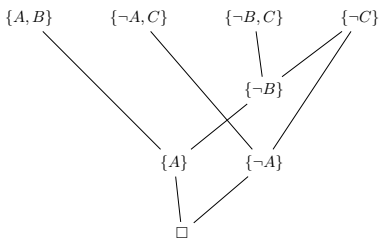


Figure 2.7: Clauses and resolution from Example 2.18.

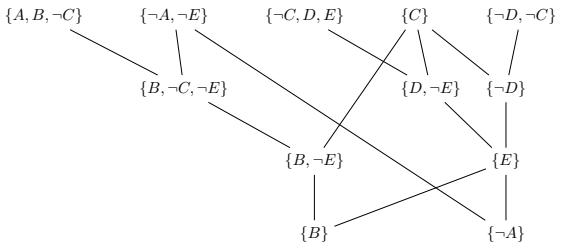


Figure 2.8: Resolution from Example 2.20.

That is, if a formula is *not* unsatisfiable, then the resolution calculus leads us to a satisfying assignment.

A functional programming language directly based on resolution calculus is PROLOG.

2.8 Exercises

Exercise 2.1 (Pigs). Every old pig eats a lot, and every healthy pig eats a lot. On a farm, there are pigs that eat a lot and there are pigs that do not. Which of the following statements follow from these premises?

1. There are both old and young pigs on the farm.
2. There are young pigs on the farm.
3. All pigs that do not eat a lot are young.
4. Some young pigs on the farm are sick.
5. Some old pigs on the farm are sick.
6. All young pigs on the farm are sick.

Exercise 2.2 (Syntax of Propositional Logic). In this exercise, the notion of a *correct formula* is to be understood according to the restrictive definition from class, represented by the syntax diagram: No brackets are to be omitted, and the only allowed connectives are $\wedge$, $\vee$, and $\neg$.

Decide for the following strings whether they are correct formulas. For those that are, draw the syntax tree and evaluate the formulas for the assignment $\mathcal{A}(A) = \mathcal{A}(B) = \mathcal{A}(C) = \texttt{true}$.

1. $(((A \wedge B) \vee C) \wedge ((A \wedge B) \vee (\neg C)))$
2. $((A \wedge A \wedge B))$
3. $((A \vee (B \vee C)) \wedge ((A \vee B) \vee (\neg C)))$
4. $(\neg\neg\neg A)$
5. $(\neg(\neg A)) = A$

Exercise 2.3 (Semantics of Propositional Logic). In this exercise, we use the "liberal" syntax: Parantheses can be omitted, and we allow other connectives besides $\wedge$, $\vee$, and $\neg$, such as: $\rightarrow$, $\leftrightarrow$, or $\oplus = \text{XOR}$.

Decide with truth tables which of the following formulas (which are all syntactically correct) are tautologies, and which are unsatisfiable:

1. $A \rightarrow (B \rightarrow (A \leftrightarrow B))$
2. $A \rightarrow (\neg B \rightarrow (A \oplus B))$
3. $(A \oplus B) \leftrightarrow (A \leftrightarrow B)$
4. $((A \oplus B) \oplus \neg A) \leftrightarrow B$

5. $(A \to B) \wedge (C \to A) \to \neg(B \oplus C)$

Exercise 2.4 (Normal Forms). Find, for each of the formulas below, a semantically equivalent formula in CNF and in DNF:

1. $A \to A \vee B$
2. $(\neg A \to B \wedge C) \leftrightarrow \neg C$
3. $A \oplus B \oplus C$

Exercise 2.5 (Conclusions). Show that these are all correct semantic conclusions:

1. $\neg(F \oplus G)\,,\ F \vee G \models F \wedge G$
2. $F_1 \to F_2\,,\ F_2 \to F_3\,,\ F_3 \to F_1 \models (F_1 \leftrightarrow F_2) \wedge (F_2 \leftrightarrow F_3)$
3. $(L \vee F) \wedge (\neg L \vee G) \models F \vee G$

Exercise 2.6 (The Sheep and the 40 Lions). A sheep is among 40 hungry lions. In this exercise, we want to answer the question whether it should be worried. Let the following rules be valid:

- The sheep cannot be subdivided: If it is being eaten, then by exactly *one* lion.
- This lion, however, then falls into a deep digestive sleep, and he can be eaten by one of his colleagues. In other words, a lion becomes a sheep after having eaten.
- If a lion has to fear being eaten after having eaten, he would not eat. If he has nothing to fear, he eats.
- The lions are not only hungry, but also intelligent and rational, and they rely on the others being so, too. (For the poor sheep, this assumption is of no help.)

1. Let, for $n \geq 0$, B_n be the proposition: "If a sheep is among n lions, then it will be eaten by one of them."

   Determine the truth values of B_0, B_1, B_2, and B_3.

2. For $n \geq 1$, represent the truth value of B_n *recursively* as a function of B_{n-1}.

3. Using this, determine B_{40} and, generally, B_n as a function of n.

Exercise 2.7 (Resolution). Decide for the following formulas in CNF whether they are tautologies, unsatisfiable, or neither of the two. In the first two cases, give a proof. In the third, give both an assignment that makes the formula TRUE and FALSE.

1. $(A \vee B) \wedge (\neg B \vee D \vee E) \wedge (A \vee \neg D \vee E) \wedge \neg A$.
2. $(A \vee B) \wedge \neg E \wedge (\neg B \vee D) \wedge (\neg D \vee E) \wedge (\neg A \vee B)$.
3. $(A \vee H \vee H \vee \neg A) \wedge (D \vee \neg B \vee E \vee B) \wedge (F \vee C \vee E \vee \neg C \vee G)$.

Exercise 2.8 (Consequences). Decide through resolution whether the following semantic consequences are correct or not:

1. $A \vee B,\ \neg B \vee D,\ \neg D \vee E,\ \neg A \vee B \ \models\ E$.
2. $\{A, B, C\},\ \{\neg A, \neg B, C\} \ \models\ \{C\}$.

Exercise 2.9 (Sushi). I have a lot of potatoes, oil, and rice at home, but I would like to eat sushi. What to do? My Japanese neighbour will surely give me wasabi if I bring her both beef and fries. With potatoes and oil, I can make fries as well as potato salad (as much as required, of both). For sushi, I need fish and rice and wasabi. My other neighbor (who just turned veggie) will certainly give me fish and beef if I get him some rice and potato salad.

So in the end, will I be able to have my sushi? Please answer this question using resolution.

Chapter 3

Set Theory

Let us turn to the second pillar of mathematics, namely, *set theory*. As mentioned in the Introduction, *all* objects in mathematics are sets. For instance, the natural numbers can be defined inductively, starting from the empty set, through never-ending formation of new sets:

$$\begin{aligned} 0 &:= \emptyset \\ 1 &:= \{0\} = \{\emptyset\} \\ 2 &:= \{0,1\} = \{\emptyset, \{\emptyset\}\} \\ &\vdots \\ n &:= \{0,1,\ldots,n-1\} \ . \end{aligned}$$

3.1 Basic notions

A set is a collection of objects — where all these objects are *sets* themselves (remember: *all* objects are). Thus, set theory is about a relation, called the *element relation*, among sets.

Definition 3.1 (Element Relation)**.** The relation "is an element of" relates an object (set) x [1] with a set A. One says "x is an element of A" or "x is in A" or "A contains x" and writes

$$x \in A \ .$$

On the other hand, if x is not in A, one writes

$$x \notin A \quad \text{or} \quad \neg(x \in A) \ .$$

[1] It is common to label sets with capital letters and their elements with small letters — although, again, also the lowercase-letter objects are sets.

3.1.1 "Cantor's Paradise"

Georg Cantor is the founder or set theory, where his definition of what a set is, and what objects it can contain, was very *liberal* in the sense that a set was a collection of *any* kind of objects with the only condition that these objects be distinguishable from each other.

Definition 3.2 (Cantor's naive approach)**.** Any collection of well-distinguished objects is a *set*.

In this "liberal" definition, it is, in particular, not excluded that a set contains itself as a member. Unfortunately, this possibility leads to a serious problem: If a set can contain itself (or not, of course), then exactly this property can be used to form a new set: *The set M of all elements that do not contain themself as an element*:

$$M := \{B | B \notin B\} \ .^2$$

Then, what about this question: Does M contain itself? If it does, it does not, according to the definition — a classical antinomy: Naïve set theory crashes. This antinomy, which is due to the mathematician and philosopher *Bertrand Russell*, was, by Russell himself, put as follows: A barber is a man who shaves every man who does not shave himself. Does the barber shave himself? Again, he does exactly when he does not.

That was the end of Cantor's Paradise. The way out is a set of stricter rules about what can be a set. It will still be the case that sets contain sets (there is nothing else, after all), but with the new rules, there is no more such a thing as "the set of all sets." And in particular, it never occurs that a set contains itself. The most famous set of such rules goes back to *Ernst Zermelo* and *Abraham Fränkel* (ZF), extended by the somewhat mysterious "axiom of choice" (*ZFC*).

3.1.2 Zermelo/Fraenkel/Choice (ZFC) set theory

In a nutshell, ZFC set theory is a set of axioms describing how sets can be formed from other sets.

We introduce symbols from predicate logic (which we have not studied here) which we use in the text simply as linguistic abbreviations.

Definition 3.3 (Quantifiers)**.** In order to express that a statement holds for *all* cases of a kind, one uses the *universal quantifier* $\forall$. If a statement holds for at least one instance of a kind, the *existential quantifier* $\exists$ is used.

[2] In this context the symbol "|" means "*such that*" (and not NAND as in Chapter 2).

Example 3.1. In terms of quantifiers, the statement "all natural numbers are non-negative" is

$$\forall x \in \mathbb{N}\,(x \geq 0)$$

(read: "for all x in $\mathbb{N}$, x is greater or equal to zero"). Similarly, the statement "there exists at least one natural number greater than 100" can be written as

$$\exists x \in \mathbb{N}\,(x > 100)$$

The first axiom of ZFC, called *extensionality*, states that a set is completely determined by its elements. It implies, in particular, that the order in which the elements are listed is irrelevant, and that an object cannot be in a set multiple times.

Axiom 1 (Extensionality Axiom)**.** *Two sets are equal if they contain the same elements:*

$$\forall A\ \forall B\ (\forall x\ (x \in A \Leftrightarrow x \in B) \Rightarrow A = B)\ .$$

Note that the axiom implies that the sets are equal *if and only if* they contain the same elements. The sufficiency of that condition is stated by the axiom, whereas its necessity is purely *logical*: The logic of equality requires that if two objects are equal, then they have the same properties.

Example 3.2. As the following sets contain the same elements, they are equal:

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

Neither the order nor multiple occurrences matter. The following sets are not equal:

$$\{a, b, c\} \neq \{\{a\}, \{b\}, \{c\}\}$$

In particular, a and the set containing a, i.e., $\{a\}$, are *not* equal.

Predicates, i.e., properties, yield subsets of a given set: It is possible to single out in a set all the elements of the original set having that certain property.

Definition 3.4 (Predicate)**.** For a given set A, a predicate is a function $P : A \to \{\texttt{false}, \texttt{true}\}$. P can also be regarded as a property that all $x \in A$ have for which $P(x) = \texttt{true}$.

Axiom 2 (Subsets from Predicates)**.** *Given a set A and a predicate P on A, the collection of all elements that have the property P (i.e. for which P is true)*

$$B := \{x \in A \mid P(x) = \texttt{true}\} = \{x \in A \mid P(x)\}$$

is another set.

Example 3.3. One can define a predicate on the natural numbers to be true if and only if the argument is less or equal to 10. This yields the set

$$A := \{x \in \mathbb{N} \mid x \leq 10\} \ .$$

With a second predicate on A returning true if and only if the argument is prime, we obtain

$$B := \{x \in A \mid \text{IsPrime}(x)\} = \{2, 3, 5, 7\} \ .$$

Definition 3.5 (Subset)**.** A set A is a subset of another set B if all elements of A are also elements of B:

$$A \subseteq B \quad :\Leftrightarrow \quad \forall x(x \in A \Rightarrow x \in B) \ .$$

Using the subset relation we can formulate a first theorem.

Theorem 3.1. *If a set A is a subset of B and B a subset of A, then the two sets are equal:*

$$A \subseteq B \wedge B \subseteq A \quad \Rightarrow \quad A = B \ .$$

Proof. The theorem is a direct consequence of the axiom of extensionality. As A is a subset of B, any element in A is an element of B and *vice versa* any element in B is an element of A. Thus, an element x is an element of A if and only if it is an element of B, i.e., A and B contain the same elements. □

Of course, set theory makes sense only if sets exist. Normally, the existence of the empty set is postulated. It is surprising enough that, from this alone, the whole rich zoo of mathematical objects blossoms out of this semen. In fact, it is only necessary to postulate the existence of *at least one set* A, then the empty set can be gotten out of it with the predicate rule, and the predicate of *inequality*:

$$\emptyset := \{x \in A \mid x \neq x\} = \{\} \ .$$

Thus, we obtain the empty set as the subset of A not containing any element from A. By extensionality, the empty set is *unique*. Furthermore, it is a subset of any set.

Let us now consider different ways of forming new sets from given ones.

Definition 3.6 (Intersection)**.** Given two sets A and B, the *intersection* $A \cap B$ consists of all elements contained in *both*:

$$x \in A \cap B \quad :\Leftrightarrow \quad x \in A \wedge x \in B \ .$$

Note that this is *not* a new axiom. Any intersection can just be written in terms of predicates:

$$A \cap B = \{x \in A \mid x \in B\} \ .$$

The union of two sets cannot be written in terms of predicates. We thus require another axiom.

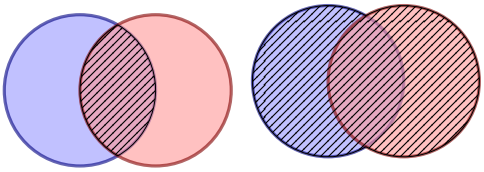


Figure 3.1: Intersection and union of two sets.

Axiom 3 (Union)**.** *Given two sets A and B, their* union $A \cup B$ *containing all elements of A as well as all elements of B is a set:*

$$x \in A \cup B \quad :\Leftrightarrow \quad x \in A \vee x \in B \ .$$

To compare two sets, we need to define the (asymmetric) difference and the symmetric difference of two sets.

Definition 3.7 (Difference)**.** The *difference* of two sets A and B — denoted by $A \setminus B$ — is defined as the set of all elements contained in A but not in B, i.e.,

$$x \in A \setminus B \quad :\Leftrightarrow \quad x \in A \wedge x \notin B \ .$$

Definition 3.8 (Symmetric Difference)**.** The *symmetric difference* is defined using the *XOR*:

$$x \in A \triangle B \quad :\Leftrightarrow \quad x \in A \oplus x \in B \ .$$

The symmetric difference — as the name suggests — is the symmetric version of the difference. It is equal to the union of $A \setminus B$ and $B \setminus A$ as well as to the union of A and B minus their intersection:

$$A \triangle B = (A \setminus B) \cup (B \setminus A) = (A \cup B) \setminus (A \cap B) \ .$$

The definitions above show the close relation between set theory and logic: Using the logical connectives, we can define corresponding set operations.

The construction of the *power set* goes beyond that, and it is the most "powerful" of the ZFC set-forming axioms, allowing for constructing in particular very large sets from smaller ones. The power set of a set A is the set of all subsets of A.

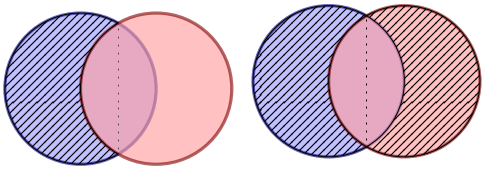


Figure 3.2: Difference and symmetric difference of two sets.

Definition 3.9 (Power Set)**.** The *power set* of a set A is the set of all subsets of A, i.e.,

$$x \in \mathbb{P}(A) :\Leftrightarrow x \subseteq A \ .$$

In particular, the power set contains the empty set and A itself.

The power set is also denoted 2^A as the cardinality, i.e. the number of elements, of the power set is just

$$|\mathbb{P}(A)| = 2^{|A|}$$

if A is finite. This is since for each of the $|A|$ elements of A, we can decide whether we choose it or not for the subset: We have $|A|$ binary choices, multiplying up to $2^{|A|}$ total choices, i.e., different subsets.

Definition 3.10 (Complement)**.** If a set A is defined as the subset of some larger set $\mathcal{U}$, usually called the *universe*, then the *complement* of A is

$$\bar{A} := \mathcal{U} \setminus A \ .$$

Families. Any union or intersection of sets, indexed by a set, yields another set. For a family of sets A_i with an index $i \in I$, we define

$$\begin{aligned} x \in \bigcup_{i \in I} A_i \quad &:\Leftrightarrow \quad \exists i \in I (x \in A_i) \\ x \in \bigcap_{i \in I} A_i \quad &:\Leftrightarrow \quad \forall i \in I (x \in A_i) \ . \end{aligned}$$

(These are well-defined due to associativity.)

Let us finally look at this mysterious "axiom of choice." Intuitively, it asks for a quite unsurprising fact: If you have a family of all non-empty sets, then it is possible to choose exactly one element out of each of the (non-empty) sets in the family. (Another way of putting it is: The Cartesian product of a family of non-empty sets is non-empty.) What is so mysterious about that? Although it is intuitive — why would the claimed *not* be possible? — it has counter-intuitive consequences, such as the *Banach/Tarski paradox*: A unit ball can be cut into five peaces (subsets) that can be rearranged using only rotations and translations, for obtaining *two unit balls of the same size*. This suggests that our intuition is somehow inconsistent. In this sense, Banach/Tarski is "only" a paradox, and not an antinomy: It appears weird to us, but it is not an intrinsic logical problem of the theory. Which does not mean that it does not have consequences: For instance, it is not possible to define a universal volume function. The reason why the axiom of choice appears so innocent to us is that we apply it to *finite* families, whereas the strange consequences come when you apply it beyond this, to infinite families. Major parts of mathematics depend on that axiom, such as most of functional analysis, which is the basis for quantum mechanics.

The axiom of choice. Given a family of sets, each containing at least one element, it is possible to make a selection of exactly one object from each set.

3.1.3 Laws derived from logic

The close relation between logic and set theory mentioned above allows us to infer set-theoretic laws from logical laws. One obtains the correspondent of a logical law in set theory by replacing $\wedge$ by $\cap$, $\vee$ by $\cup$, $\neg$ by the complement and the semantic equivalence by equality of sets.

Idempotence in logic states that the formula $A \wedge A$ is semantically equivalent to A, i.e., $A \wedge A \equiv A$. In terms of sets, this becomes

$$A \cap A = A \ .$$

Similarly we obtain from $A \vee A = A$,

$$A \cup A = A \ .$$

Absorption states that a formula $A \wedge (A \vee B)$ is semantically equivalent to A. For sets, this turns into

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

Distributivity for logical conjunction is $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$ and turns into

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

Finally also *de Morgan's laws* have set-theoretic correspondents. If we have a reference set $\mathcal{U}$, and can thus apply the complement operation, we obtain

$$\overline{(A \cup B)} = \overline{A} \cap \overline{B}$$

as the set analogue of $\neg(A \vee B) = \neg A \wedge \neg B$.

We prove one of the identities as an example. Note that one way to show that two sets are equal is to show that each is a subset of the other. For showing that a set A is a subset of B, we show that any element $x \in A$ also belongs to B.

Example 3.4. For any two sets A, B, prove that:

1. $A \cap (A \cup B) = A$ **(Absorption Law)**.
2. $(A \cap B) \cup (A \cap \overline{B}) = A$.

Proof. We assume that U is the universe set for part 2 and hence $A, B \subseteq U$.

1. We show that these two sets are equal by showing that each is a subset of the other. First we show that $A \cap (A \cup B) \subseteq A$. Suppose $x \in A \cap (A \cup B)$ be any element of this set. Then, $x \in A$ and $x \in A \cup B$ by the definition of intersection. Since $x \in A$, we have proved that the left-hand side is a subset of the right-hand side.

   Conversely, let $x \in A$ be an arbitrary element of A. Then by the definition of union, $x \in A \cup B$ as well. Since both of these are true, $x \in A \cap (A \cup B)$ by the definition of intersection, and we have shown that the right-hand side is a subset of the left-hand side.

   The two directions together imply the equality.

2. Again, we show that these two sets are equal by showing that each is a subset of the other. Suppose $x \in (A \cap B) \cup (A \cap \overline{B})$. Then we know that either $x \in (A \cap B)$ or $x \in (A \cap \overline{B})$ (or both). In either case, by the definition of intersection, this forces $x \in A$. Thus, we have shown that the left-hand side is a subset of the right-hand side.

   For the opposite direction, suppose $x \in A$. Then $x \in U = B \cup \overline{B}$. By definition of union, there are two cases: $x \in B$ or $x \in \overline{B}$. In the former case, x is an element of $(A \cap B)$ and therefore also an element of $(A \cap B) \cup (A \cap \overline{B})$. In the latter cases, $x \in \overline{B}$ and therefore x is an element of $(A \cap \overline{B})$ and therefore also an element of $(A \cap B) \cup (A \cap \overline{B})$. Hence the equality follows.

□

3.1.4 The Cartesian product

In the 17$^{\text{th}}$ century, the early modern philosopher René Descartes introduced the *Cartesian product* to describe the location of points by their coordinates. Only later Cartesian products were formalized in set theory employing ordered pairs. Consider, for example, points in the two-dimensional plane, as shown in Figure 3.3. The points (x, y) and (y, x) are generally not the same. The order of the coordinates does matter: Differently from unordered pairs, two ordered pairs (a, b) and (c, d) are equal if and only if $a = c$ and $b = d$. How can we define such an ordered pair in set theory?

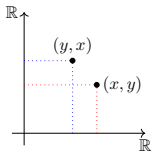


Figure 3.3: The order of the coordinates matters.

Definition 3.11 (Ordered Pair)**.** The *ordered pair* (x, y) is defined as

$$(x, y) := \{\{x\}, \{x, y\}\} \ .$$

Let us verify whether this definition leads to the equality property for two points (a, b) and (c, d), mentioned above. The ordered pairs correspond by definition to the sets

$$\Big\{\{\ a\ \}, \Big\{a,\ b\ \Big\}\Big\} \qquad \Big\{\{\ c\ \}, \Big\{c,\ d\ \Big\}\Big\} \ .$$

If the two sets are equal, then a and c necessarily have to be equal. Therefore, as also the elements $\{a, b\}$ and $\{c, d\}$ have to be equal, b and d have to be equal.

A special case of an ordered pair is the one containing the same element twice. Then the set is equal to the set containing the set containing the element:

$$(x, x) = \{\{x\}, \{x, x\}\} = \{\{x\}, \{x\}\} = \{\{x\}\} \ .$$

Having established the notion of an ordered pair, we can now define the Cartesian product of two sets.

Definition 3.12 (Cartesian Product)**.** Given two sets A and B, their Cartesian product is defined as the set containing all ordered pairs:

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

If one of the two is the empty set, the Cartesian product is empty:

$$A \times \emptyset = B \times \emptyset = \emptyset \ .$$

Generally, if neither A nor B are empty, and they are not equal, $A \neq B$, then the Cartesian product is not symmetric, i.e.,

$$A \times B \neq B \times A \ .$$

Example 3.5. Considering the sets $A = \{1\}$ and $B = \{2, 3\}$ we obtain the Cartesian products:

$$A \times B = \{(1, 2), (1, 3)\}$$
$$B \times A = \{(2, 1), (3, 1)\}$$

The definitions of ordered pairs extends naturally to ordered lists of more than two numbers, so called *tuples*. Given a finite index set, $I = \{1, \ldots, k\}$ we define the Cartesian product of k sets as

$$\bigtimes_{i \in I} A_i := \{(a_1, \ldots, a_k) \mid \forall i \in I(a_i \in A_i)\} \ .$$

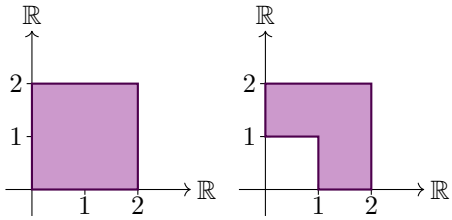


Figure 3.4: The shaded area to the left is the Cartesian product $[0,2]^2$, the one to the right the difference $[0,2]^2 \setminus [0,1]^2$.

Example 3.6. Given an interval

$$[0,2] := \{x \in \mathbb{R} \mid x \geq 0 \wedge x \leq 2\} \ ,$$

the Cartesian product $[0,2]^2 = [0,2] \times [0,2]$ is a square with length 2 as shown in Figure 3.4. Another subset of $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$, $[0,2]^2 \setminus [0,1]^2$, is shown in the figure as well.

Example 3.7 (Order Relation). Using the order relation $\leq$ we can define the following subset

$$R := \left\{(x,y) \in \mathbb{R}^2 \mid x \leq y\right\}$$

pictured in Figure 3.5. So far, order relations were not formally defined. We make up for that in the next section, turning things around: We *define* the relation $\leq$ *to be* the *equal to* the set R.

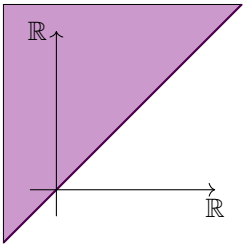


Figure 3.5: The area above the diagonal corresponds to the order relation $\geq$.

3.2 Relations

Definition 3.13 (Binary relation)**.** A (binary) relation R from A to B is a subset of the Cartesian product of A and B:

$$R \subseteq A \times B \ .$$

We speak of *a relation from A to B* (we have in mind here functions from A to B). In the special case where $A = B$, a relation R from A to A is usually called *relation on A* (except, again, in the case of a function from A to A). For a pair $(a, b) \in R$, we write aRb.

We apply the set calculus from above on relations since they *are* simply sets.

Example 3.8. The intersection of the relations $\leq$ and $\geq$ is equal to the equality relation. Set operations are indicated by blue, relations by red:

$$\leq \ \cap \ \geq \ = \ =$$

Similarly, the symmetric difference of the two yields the inequality relation:

$$\leq \ \triangle \ \geq \ = \ \neq$$

The relation $<$ is contained in $\leq$, and $>$ in $\geq$:

$$< \ \subseteq \ \leq \qquad > \ \subseteq \ \geq$$

The complement of $\leq$ is $>$.

Example 3.9. We introduce relations on the integers that are of particular interest in number theory and cryptography:

$$\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\} \ .$$

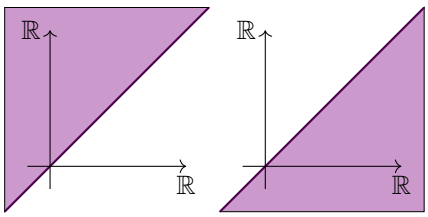


Figure 3.6: Merely the diagonal line with $x = y$ is contained in both relations.

An integer b is said to be divisible by another integer a (or a is a divisor of b) if there exists $c \in \mathbb{Z}$ such that $a \cdot c = b$. Divisibility is usually denoted by "|," a symbol we have already seen used to mean "such that." This is one of the many cases in which using mathematical notation creates confusion instead of clarity. To avoid misunderstandings, the convention is to use either " |" or ":" to mean "such that," depending on the context. Accordingly, the formal definition of divisibility is

$$a \mid b \quad :\Leftrightarrow \quad \exists c \in \mathbb{Z} : \; a \cdot c = b \; .$$

For instance, 4 is divisible by 2, and 9 by -3, but 5 not by 2:

$$2 \mid 4 \qquad -3 \mid 9 \qquad 2 \nmid 5 \; .$$

Divisibility is a binary relation on $\mathbb{Z}$:

$$\mid \; \subseteq \; \mathbb{Z} \times \mathbb{Z} = \mathbb{Z}^2 \; .$$

All relations we have seen before on $\mathbb{R}$ can be reduced to relations on $\mathbb{Z}$ by intersection:

$$\leq_{\mathbb{Z}} \; = \; \leq_{\mathbb{R}} \; \cap \; \mathbb{Z}^2$$

Finally, *congruence modulo* m is defined as

$$a \equiv b \pmod{m} \quad :\Leftrightarrow \quad m \mid (a - b) \; ,$$

i.e., two integer numbers a and b are congruent modulo m if the difference of the two is divisible by m. This is the same as saying that the integer division of each a and b by m yields the same remainder. For instance, 3 and 5 are congruent modulo 2, while 3 and 4 are not:

$$3 \equiv 5 \pmod{2} \qquad 3 \not\equiv 4 \pmod{2} \; .$$

Each natural number m defines such a congruence relation $\equiv_m$ with the modulo m. The intersection of two such congruence relations yields another one with the least common multiple[3] (lcm) being the modulus:

$$\equiv_m \; \cap \; \equiv_n \; = \; \equiv_{\mathrm{lcm}(m.n)}$$

[3]The least common multiple of two integers a and b is the smallest number $c \in \mathbb{Z}$ such that there exist integers $m, n \in \mathbb{Z}$ with $a \cdot m = c = b \cdot n$. For relatively prime numbers a and b, the least common multiple is simply their product, $c = a \cdot b$.

So one obtains, for example,

$$\equiv_2 \cap \equiv_2 = \equiv_2 \qquad \equiv_2 \cap \equiv_3 = \equiv_6$$

Symmetry of relations. Two important types of relations are *order relations* and *equivalence relations*. Whereas they overlap in certain properties, mainly *transitivity*, they essentially differ in one: Orders are *anti*-symmetric, whereas equivalences are *symmetric*. Symmetry hereby means that an ordered pair (a, b) is in R if and only if the swapped pair (b, a) is also in R. Anti-symmetry, on the other hand, means that at most one of the two can hold for any two $a \neq b$.

Inequality relations such as $\leq$, $\geq$, $>$ and $<$ are anti-symmetric relations. They impose on the set, on which they are defined, a *hierarchy*.

Equality, congruence, and semantic equivalence are *equivalence relations*. These relations yield a partition of the set they are defined on. That means they divide the set into subsets, each containing all elements equivalent to one another. Equivalence relations are often denoted by $\sim$.

Example 3.10 (Order Relations). Let us return to the example of divisibility on $\mathbb{Z}$. It is an order relation. For now we restrict the relation (by intersection) to the subset $A := \{1, 2, 3, 4, 5, 6\} \subseteq \mathbb{Z}$. The order relation on this finite set can be visualized in a graph, as in Figure 3.7. Following the arrows, also for multiple steps, yields the ordered pairs in $|$. Similarly, we obtain the graph 3.8 for divisibility on $B = \{1, 2, 3, 5, 6, 10, 15, 30\}$. The corresponding *Hasse diagram* is the two-dimensional projection of a cube. The order relation $\subseteq$ on the power set

$$\mathbb{P}(\{1, 2, 3\}) = \{\emptyset, \{1\}, \{2\}, \ldots, \{1, 2, 3\}\}$$

represents the same cube. One can now extend the graph to a projection of a 4-dimensional hypercube. This corresponds either to the divisibility relation on the set $\{1, 2, 3, 4, 5, 7, \ldots, 210\}$ or the subset relation on the power set $\mathbb{P}(\{1, 2, 3, 4\})$. Considering the latter, we extend the cube by another linked cube containing the set with 4 as Figure 3.9.

Example 3.11 (Equivalence relation). Let us consider an equivalence relation on $\{1, 2, 3, 4, 5, 6\}^2 = \{(i, j) \mid 1 \leq i, j \leq 6\}$ defined as

$$(a, b) \sim (c, d) \quad :\Leftrightarrow \quad a \cdot d = b \cdot c \ .$$

Intuitively, the *ratio* of the two numbers in two equivalent pairs is equal. The relation yields the following equivalence classes:

$$\{1/1, 2/2, 3/3, 4/4, 5/5, 6/6\},$$
$$\{1/2, 2/4, 3/6\}, \{2/1, 4/2, 6/3\},$$
$$\{1/3, 3/6\}, \ldots$$
$$\{1/4\}, \ldots$$

This is how the rational numbers are defined in terms of the integers: as the set of such equivalence classes: We write

$$\mathbb{Q} := \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})/\sim .$$

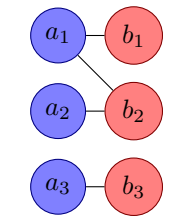
3.2.1 Representation of relations

Relations on finite sets A and B can be represented by either *binary matrices* or *bipartite graphs*.

In matrix representation, each row corresponds to an element in A and each column to an element in B. The entry is then 1 if and only if the corresponding pair (a, b) is in R:

$$\begin{array}{c} \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{array} \begin{array}{c} \begin{array}{cccc} b_1 & b_2 & \cdots & b_n \end{array} \\ \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 0 & 1 & \cdots & 0 \\ & & & \\ 0 & 0 & \cdots & 1 \end{pmatrix} \end{array}$$

In the bipartite graph, the nodes on the left correspond to elements in A, the ones on the right to elements in B. The nodes are linked if and only if $(a, b) \in R$.



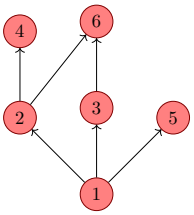


Figure 3.7: The graph shows divisibility on the set A. The number 1 divides all other numbers and is thus the root of the directed path. Its direct neighbours are the prime numbers. Finally, the integers 4 and 6 are connected to their prime factors.

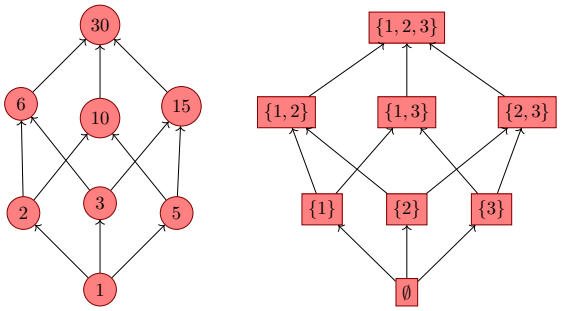


Figure 3.8: The graph on the left shows the divisibility of the set B. 1 divides all other numbers and is thus the root of the directed path. The direct neighbors are prime numbers.

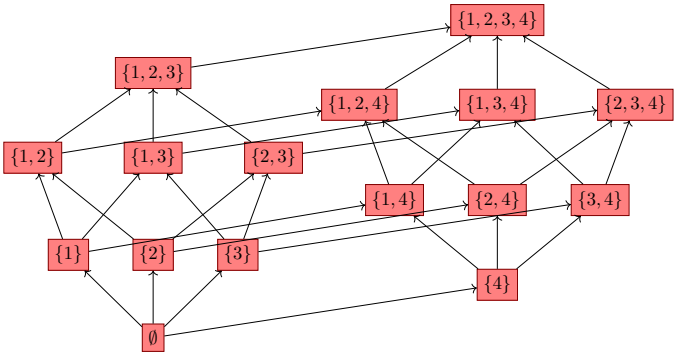


Figure 3.9: The graph of the subset relation on the power set $\mathbb{P}(\{1, 2, 3, 4\})$ corresponds to the 4-dimensional hypercube.

In the special case of *relations on a set* A, i.e., where $A = B$ holds, representations can be made so that every element a of A is drawn only once, and an arrow connects a_1 and a_2 if and only if the pair (a_1, a_2) is in the relation. What results is a *graph*. (It is, however, not a *simple* graph since it can have loops, i.e., a node connected by *itself* via an arrow.) In some way, a graph is in fact nothing but (the representation of) a relation on its vertex set.

3.2.2 Properties of relations

We consider different properties of relations. We restrict ourselves to relations and a set A, i.e., the special case where $A = B$.

Reflexivity. A relation R on A is *reflexive* if for any element $a \in A$ the pair (a, a) is in R:

$$\forall a \in R : \ (a, a) \in R \ .$$

In matrix representation, the corresponding matrix has only ones on the diagonal. The representing graph has a loop for each of its nodes.

Example (Reflexive relations).

$$\equiv_m \qquad = \qquad \leq \qquad \geq$$

Anti-reflexivity. A relation R on A is *anti-reflexive* if

$$\forall a \in A : (a, a) \notin R \ .$$

The diagonal of the corresponding matrix is 0, and there are no loops in the associated graph.

Example (Anti-reflexive relations).

$$> \qquad < \qquad \neq$$

Symmetry. A relation R on A is *symmetric* if

$$\forall a, b \in A : \quad (a, b) \in R \ \leftrightarrow \ (b, a) \in R \ .$$

For finite sets A the corresponding binary matrix is symmetric: That is $M_R^T = M_R$. In the graph representation, symmetry means that whenever there is an arrow in one direction, there is also one in the inverse direction. Thus, symmetric relations are often represented as *undirected* graphs, with undirected edges instead of arrows.

Example (Symmetric relations).

$$\equiv_m \qquad =$$

Anti-symmetry. A relation R on A is *anti-symmetric* if

$$\forall a, b \in A : \quad (a, b) \in R \wedge (b, a) \in R \ \Rightarrow \ a = b$$

Example (Anti-symmetric relations).

$$\leq \qquad < \qquad \geq \qquad >$$

In the matrix representation: Whenever a non-diagonal position $(i.j)$ holds a 1, then the mirrored position (j, i) must hold a 0. (A 0 is both positions is possible.)

Note that also the strict inequalities such as $<$ are anti-symmetric as the condition is merely a *conclusion* in the definition, *not* an equivalence: The relation to hold in both directions can only happen for equal entries, but it does not have to. For $<$, the condition $((a, b) \in R \wedge (b, a) \in R)$ is simply never met — which makes the conclusion always true.

Transitivity. A relation R on A is transitive if

$$\forall a, b, c \in A: \quad (a, b) \in R \wedge (b, c) \in R \;\Rightarrow\; (a, c) \in R \; .$$

Example. This is the central property shared by both equivalence and order relations. Transitive are

$$= \qquad \equiv_m \qquad \leq \qquad < \qquad \subseteq \; .$$

The negations of some of the above are not transitive:

$$\neq \; .$$

The only "positive" one that comes to mind which is intransitive is the element relation

$$\in \; .$$

We look at three types or relations: Equivalences, orders, and functions.

3.2.3 Equivalence relations

Definition 3.14 (Equivalence relation)**.** A relation R on a set A is called an equivalence relation if it is reflexive, symmetric, transitive.

Example 3.12 (Age group). Let us consider the set of all humans, $A := \{\text{humans}\}$. Their age group is an equivalence relation[4] formally defined as

$$a \sim b \quad :\Leftrightarrow \quad a \text{ and } b \text{ are born in the same year} \; .$$

This equivalence relation introduces a partition on the set of all humans. They are being grouped by their year of birth.

Equivalence relations are dividing the ground set A into "classes," they *partition* the set. We formally define what a partition of a set is, and then we prove the claimed equality of equivalence relations and partitions.

[4] To show that the relation "the same age group" is an equivalence relation, one needs to show that it satisfies the properties in the definition. Rather obviously, it does so since it is based on an equality.

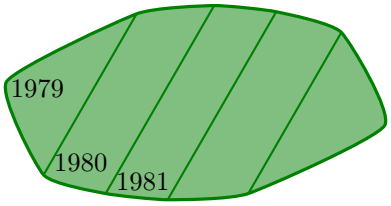


Figure 3.10

Definition 3.15 (Partition)**.** Let A be a set. A *partition of* A is a family of sets $(A_i)_{i\in I}$ with the following two properties:

- Their union yields A: $\bigcup_{i\in I} A_i = A$,
- they are pairwisely disjoint: $A_i \cap A_j = \emptyset \ \forall i, j \in I, i \neq j$.

Theorem 3.2. *Any equivalence relation yields a partition, and vice versa. More precisely, if* $\sim$ *is an equivalence relation on* A*, then the equivalence classes*

$$[a] := \{x \in A \mid x \sim a\} \subseteq A$$

are a partition of A*.*
Conversely, if $(A_i)_{i\in I}$ *is a partition of* A*, then the relation*

$$x \sim y \quad :\Leftrightarrow \quad \exists i \in I : x \in A_i \wedge y \in A_i$$

is an equivalence relation.

Proof. We show that, for a given equivalence relation $\sim$, the equivalence classes yield a partition. As any equivalence class $[a]$ contains at least a by reflexivity, it follows

$$A \supseteq \bigcup_{a\in A} [a] \supseteq \bigcup_{a\in A} \{a\} = A \quad \Rightarrow \bigcup_{a\in A} [a] = A \ .$$

It remains to show that the equivalence classes are *disjoint*. In fact, if two equivalence classes are not disjoint, then they are equal. This follows from transitivity, as follows: Consider two equivalence classes, $[x]$ and $[y]$, with a non-empty intersection: That implies there exists an element $z \in A$, such that $z \in [x] \cap [y]$. Therefore, $z \sim x$ and $z \sim y$, and thus by transitivity $x \sim y$. Employing again transitivity, we obtain that any other element $x' \in [x]$ is equivalent to y and thus in $[y]$, we obtain $[x] \subseteq [y]$. The same argument can be made to obtain $[y] \subseteq [x]$, and thus $[x] = [y]$.

The second part of the theorem is trivial.

□

Example 3.13 (Semantic Equivalence). Semantic equivalence is an equivalence relation on the set of all syntactically correct formulas. It gives rise to a partition with $2^{(2^n)}$ classes if there are n atoms.

Example 3.14 (Congruence Modulo n). The relation "congruence modulo m," $\equiv_m$, is an equivalence relation on $\mathbb{Z}$ any $m \in \mathbb{Z}$. It yields m equivalence classes:

$$\begin{aligned}
[0] &= \{\ldots,\ (-2)\cdot m,\ (-1)\cdot m,\ 0,\ m,\ 2\cdot m,\ 3\cdot m,\ \ldots\} \\
[1] &= \{\ldots,\ (-2)\cdot m+1,\ (-1)\cdot m+1,\ 1,\ m+1,\ 2\cdot m+1,\ 3\cdot m+1,\ \ldots\} \\
[2] &= \{\ldots,\ (-2)\cdot m+2,\ (-1)\cdot m+2,\ 2,\ m+2,\ 2\cdot m+2,\ 3\cdot m+2,\ \ldots\} \\
[3] &= \{\ldots,\ (-2)\cdot m+3,\ (-1)\cdot m+3,\ 3,\ m+3,\ 2\cdot m+3,\ 3\cdot m+3,\ \ldots\} \\
&\vdots \\
[m-1] &= \{\ldots,\ (-2)\cdot m-1,\ (-1)\cdot m-1,\ -1,\ m-1,\ 2\cdot m-1,\ 3\cdot m-1,\ \ldots\}
\end{aligned}$$

We can now transfer the algebraic structure of $\mathbb{Z}$ to the set of equivalence classes: Based on the arithmetic operations on integers, we define operations on the set of equivalence classes as

$$\mathbb{Z}_m \ := \ \mathbb{Z}/\equiv_m \ = \ \{[0], [1], [2], \ldots, [m-1]\} \ .$$

Addition is defined as

$$[a] + [b] := [a+b] \ .$$

We have chosen two particular representatives of the equivalence classes, namely, $a \in [a]$ and $b \in [b]$, and used these to define a new equivalence class, $[a+b]$, as the result of the addition. This operation must be independent of the choice of the representative for the definition to be meaningful — this fact is called *well-definedness*. That is, for *any* pair of representatives, $a' \in [a]$ and $b' \in [b]$, the sum should be in the corresponding equivalence class, $a'+b' \in [a+b]$. To show that this is actually the case, we note that any $a' \in [a]$ differs from a only by an integer multiple of m. The same holds for $b' \in [b]$:

$$a' = a + k\cdot m \qquad b' = b + l\cdot m \qquad k, l \in \mathbb{Z} \ .$$

Therefore, the sum of a' and b' turns out to be

$$a' + b' = a + b + (k+l)\cdot m \ ,$$

and it differs by integer multiple of m from $a+b$. Thus, it is in the same equivalence class.

Multiplication. Similar to addition, we define multiplication in $\mathbb{Z}_m$ as

$$[a] \cdot [b] = [a \cdot b] \ .$$

Again, we must verify that the addition is *well-defined*, i.e., independent of the choice of the representatives. In the same manner as above, one multiplies $a' = a + k \cdot m$ and $b' = b + l \cdot m$ to obtain

$$a' \cdot b' = a \cdot b + m \cdot \underbrace{(a \cdot l + b \cdot k + k \cdot l \cdot m)}_{\in \mathbb{Z}} \ .$$

Thus, $a \cdot b$ and $a' \cdot b'$ differ by an integer multiple of m and are thus congruent modulo m. With the two operations $\cdot$ and $+$ on $\mathbb{Z}_m$ we obtain an algebraic structure, called a *ring*.

3.2.4 Order relations

Definition 3.16 (Partial Order)**.** A relation $\leq$ on a set A is a partial order if it satisfies the following properties:

- reflexive,
- anti-symmetric: $x \leq y \wedge y \leq x \Rightarrow x = y$,
- transitive.

Example 3.15 (Partial orders). The following are partial orders:

- For the sets $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$, the relations $\leq$ and $\geq$ are partial orders. The relations $<$ and $>$ are anti-symmetric but not reflexive and, hence, not partial orders according to our definition.
- The divisibility relation $|$ on $\mathbb{N}$ is a partial order. This is not true for $\mathbb{Z}$, as $a \neq -a$; divisibility here is not anti-symmetric. a is a divisor of $-a$ and *vice versa*, but they are not equal (for all $a \neq 0$).
- The subset relation $\subseteq$ on a powerset $\mathbb{P}(B)$ is a partial order.

Remark 2. Note that we do not ask for every pair to be *comparable*. This condition reads

$$\forall x, y \in A : x \leq y \vee y \leq x$$

and defines *total order*, also called *linear order* or *chain*.

Whether an order is partial or total can easily be seen from the directed graph associated with the order. In Figure 3.7, corresponding to the subset order, not all nodes are connected, and thus the order is not total. The order $\leq$ on $\mathbb{R}$ has a linear graph (or a chain) and is a total order.

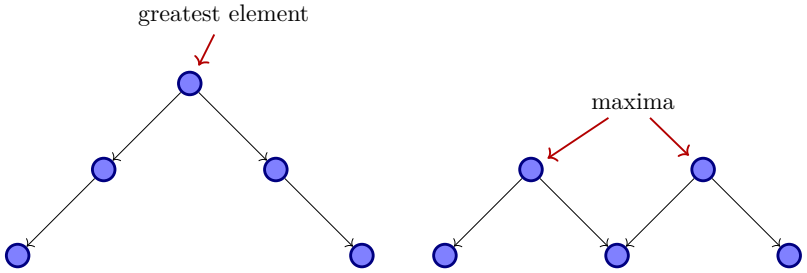


Figure 3.11: The left shows an order with just one maximum, the greatest element. The right shows an order with two maxima, and no unique greatest element.

Notions.

- The *maximal elements* in a set A with partial order $\leq$ satisfy

$$\nexists y \in A : y \neq x \wedge y \geq x \ .$$

- An element $x \in A$ is the *greatest element* if

$$\forall y \in A : \quad x \geq y \ .$$

Note that there are possibly multiple maxima, whereas the greatest element is unique, see Figure 3.11. If there is a greatest element, it is also maximal, and there are no other maxima besides it.

Example 3.16. Let us consider the set $A = \{1, 2, 3\}^2$ first with the order

$$a = (a_1, a_2) \leq a' = (a'_1, a'_2) \quad :\Leftrightarrow \quad a_1 \leq a'_1 \wedge a_2 \leq a'_2 \ .$$

This order is not total, as the pairs $(1, 2)$ and $(2, 1)$ are not comparable. This can be easily seen by looking at its Hasse diagram (cf. Figure 3.12). Another order on the same set is

$$a = (a_1, a_2) \leq a' = (a'_1, a'_2) \quad :\Leftrightarrow \quad a_1 < a'_1 \vee (a_1 = a'_1 \wedge a_2 \leq a'_2) \ .$$

This second order corresponds to an alphabetical order. The first component where the two elements differ decides about the order: It is the *lexicographic order*, and it is total (see its Hasse diagram in Figure 3.13).

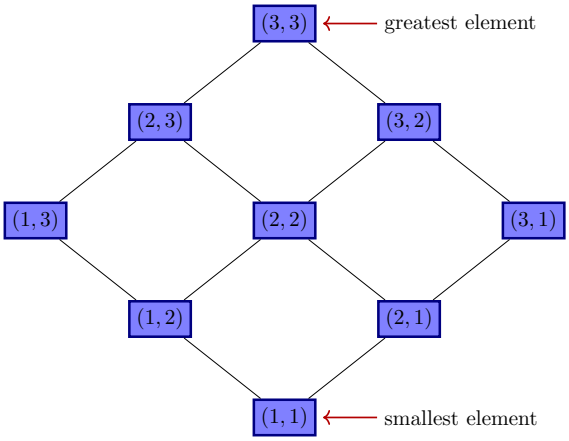


Figure 3.12: Hasse diagram for the first order relation.

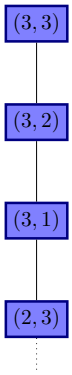
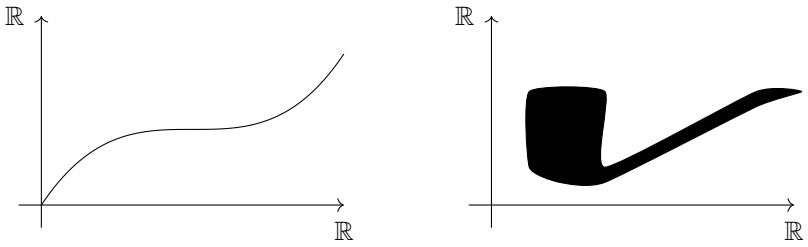


Figure 3.13: The lexicographic order has a linear Hasse diagram as it is a total order.

3.3 Functions

For two sets A and B, a *function from A to B* is a relation that "assigns" to every element $a \in A$ exactly one element $b \in B$.



Ceci est une fonction. Ceci n'est pas une fonction.

Figure 3.14: Example for a function and a relation that is not a function.

Definition 3.17 (Function)**.** A relation $f \subseteq A \times B$ is *a functional relation from A to B* or, usually, just a *function from A to B*, written $f : A \to B$, if it satisfies the following properties:

- $\forall a \in A\ \exists b \in B :\ (a, b) \in f$ (Existence of functional value);
- $(a, b) \in f \wedge (a, b') \in f \ \Rightarrow\ b = b'$ (Uniqueness).

Sometimes existence and uniqueness are written expressed together as

$$\forall a \in A\ \exists! b :\ (a, b) \in f\ ,$$

where the exclamation mark stands for "only one."

The definition identifies a function with the corresponding set of pairs (a, b), i.e., the *graph* of the function. The following notations are commonly used for $(a, b) \in f$:

$$f(a) = b \qquad f : a \mapsto b\ .$$

Definition 3.18.

- A function $f : A \to B$ is called *injective* or *one-to-one* if

$$\forall a, a' \in A : \qquad a \neq a' \quad \Rightarrow \quad f(a) \neq f(a')\ .$$

- A function $f : A \to B$ is called *surjective* or *onto* if

$$\forall b \in B\ \exists a \in A :\ f(a) = b\ .$$

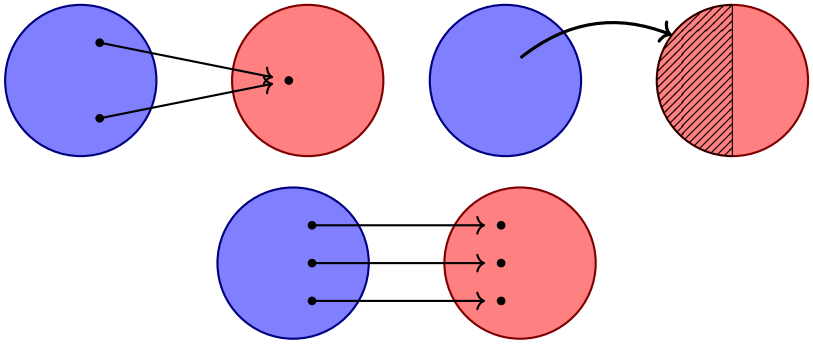


Figure 3.15: The first function has a collision and is thus not injective. The second has an image that is not equal to B and thus not surjective. The third shows a bijective function.

- A function $f : A \to B$ is called *bijective* if it is both, injective and surjective.

A function is *injective* if each image corresponds to a *unique* pre-image. There are no collisions as shown in Figure 3.15. An injective function has an *inverse*, which is defined on its *image*, i.e., from the set $f(A)$ to A.

Surjectivity, on the other hand, means that all points in the set B are reached, i.e., the image $f(A)$ of f equals B. Finally, *bijectivity* means both properties together and yields a one-to-one correspondence between the two sets. Also, there is an inverse function $f^{-1} : B \to A$.

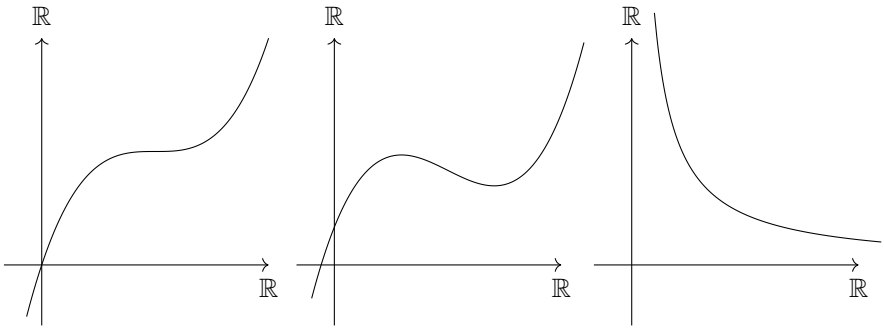


Figure 3.16: These three functions $f_1, f_2, f_3 : \mathbb{R} \to \mathbb{R}$ illustrate the properties of functions. The function on the left is bijective, the one in the middle is surjective but not injective, and the one on the right is injective but not surjective.

The existence of functions with the above properties can be used to introduce relation of sets with respect to their size (cardinality), the hope being that this is an order relation. (Is it? If yes, is it total?)

Definition 3.19 (Cardinality)**.** We say that a set A is at most as big as another B, or that the *cardinality* of a set A is less or equal than the cardinality of another set B, written $A \preccurlyeq B$, if there exists an *injective* function from A to B:

$$A \preccurlyeq B \quad :\Leftrightarrow \quad \exists f : A \to B \text{ injective} .$$

The sets are *equal in size* (they have the *same cardinality*) if there exists a bijective function between them:

$$A \approx B \quad :\Leftrightarrow \quad \exists f : A \to B \text{ bijective} .$$

The relation $\preccurlyeq$ is an order relation as shown below, and yields a hierarchy of sets with respect to size: The relation $\preccurlyeq$ is reflexive, as the identity function

$$\mathrm{id} : A \to A \qquad f(a) = a$$

is injective (actually bijective). Therefore, $A \preccurlyeq A$.

In order to prove the *transitivity* of the relation, we introduce the *composition* of functions: Given a function $f : A \to B$ and a function $g : B \to C$, we define a function $h := g \circ f : A \to C$ by $h(a) = g(f(a))$. If both f and g are injective, then $g \circ f$ is also injective.

A proof of this statement can be given by contradiction: If $g \circ f$ was not injective, there existed elements a and a' such that $g(f(a)) = g(f(a'))$. Employing the injectivity of f, we know that $f(a) =: b \neq f(a') =: b'$. Thus, there exist two arguments b and b', $b \neq b'$, such that $g(b) = g(b')$, yielding a contradiction with g being injective: If $A \preccurlyeq B$ and $B \preccurlyeq C$, then there exist injective functions $f : A \to B$ and $g : B \to C$. The existence of the injective function $g \circ f$ proves $A \preccurlyeq C$, and transitivity.

To complete the proof for $\preccurlyeq$ being a partial order, it remains to show (a statement resembling) anti-symmetry:

$$A \preccurlyeq B \wedge B \preccurlyeq A \quad \Rightarrow \quad A \approx B . \tag{3.1}$$

It is not intuitively clear why this should hold, and it is the statement of a fine, subtle theorem by Cantor, Schröder, and Bernstein.

Theorem 3.3 (Cantor/Schröder/Bernstein)**.** *For any two sets A and B, we have*

$$A \preccurlyeq B \wedge B \preccurlyeq A \quad \Rightarrow \quad A \approx B .$$

Proof. Imagine a park (associated with a set A) with a house (associated with a set B) as shown in Figure 3.18. In the house, there is a map of the park.

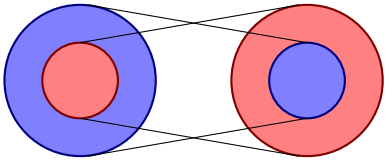


Figure 3.17: Injective maps map the first set into the second and *vice versa*.

If we look closely enough, we see the house on the map again. This house on the map in turn contains a map of the park containing a house containing the park, and so on, and so on, *ad infinituum*.

As the house is inside the park, i.e. $B \subseteq A$, there exists an injective function $f : A \to B$. On the other hand, the map of the park (so the park itself) is inside the house, and there is an injective map $g : B \to A$.

Bijection. We define a bijection from the park (circle) to the house (square): Points in sets of the green type "park\house" (i.e., $A \setminus g(B)$ or "circle\square" in Figure 3.19) are mapped to their correspondent "an iteration level below." On the other hand, red points in sets "house\park" (i.e., $B \setminus f(A)$ or "square\circle") are mapped to themselves (on the same level). The mechanism is depicted in Figure 3.19. □

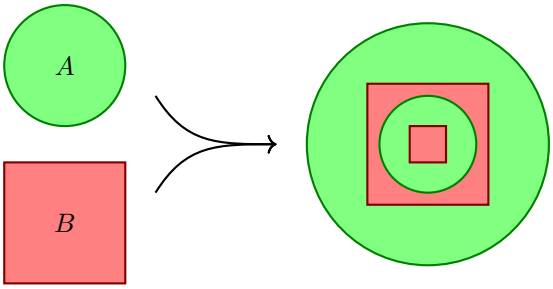


Figure 3.18: The park, represented by the green circle, with the house depicted by the red square. The smaller green circle in the house corresponds to the map of the park.

We conclude that $\preccurlyeq$ satisfies all requirements of a partial order. Indeed it is even a *total* order (we do not prove this here). Thus, it is natural to ask whether there is a *greatest* element (or maximal element, which is the same in a total order). Clearly, there is a unique minimum, a smallest element: The empty set — from which everything starts. In contrast to that, there are no

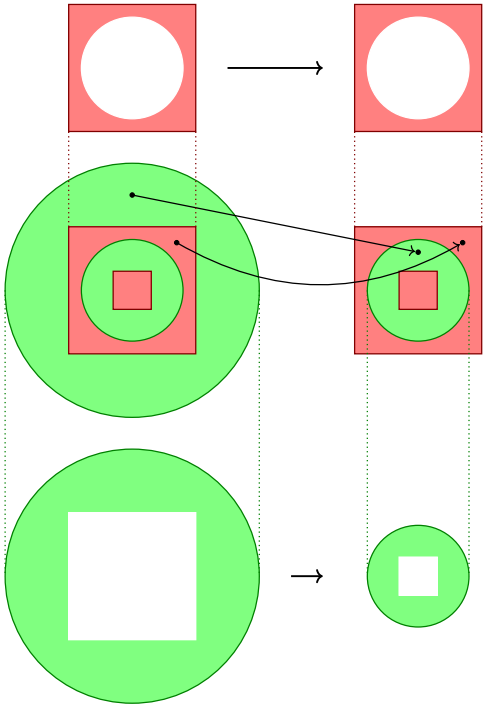


Figure 3.19: The bijective map from A to B.

greatest elements: To any set, as large it may be, there is always a strictly larger one: its power set: $A \preccurlyeq \mathbb{P}(A)$. Thus, there is no greatest set. The proof of this fact goes back to Cantor, and it is a generalization of the argument give right at the beginning of the text that the real numbers and strictly greater than the natural numbers. (In fact, the real numbers are exactly as large as the power set of the natural numbers.)

Theorem 3.4 (Cantor)**.** *The cardinality of any set A strictly smaller than the cardinality of its power set:*

$$A \not\approx \mathbb{P}(A) \ ,$$

thus, $A \prec \mathbb{P}(A)$.

Proof. We show that no function

$$f : A \to \mathbb{P}(A)$$

is surjective. We note that any element $a \in A$ is either an element of its image $f(a)$ or not:

$$a \in f(a) \vee a \notin f(a) \ .$$

Define the set

$$B := \{a \in A \mid a \notin f(a)\} \subseteq A \; .$$

We now show by contradiction that there does not exist an element $b \in A$ such that $f(b) = B$. Assume such a b did exist: If $b \in f(b)$, then $b \notin B = f(b)$. This is a contradiction. So there is no surjective function from A to its power set and no bijection.

(There is a simple *in*jective map from A to its power set $\mathbb{P}(A)$, mapping all elements in A to the corresponding one-element sets:

$$f : A \to \mathbb{P}(A) \qquad f : a \mapsto \{a\} \, .$$

Thus, $A \preccurlyeq \mathbb{P}(A)$ and therefore $A \prec \mathbb{P}(A)$.) □

This statement opens the door for larger and larger sets, larger and larger types of infinities: The real numbers are not the largest set there is; take their power set, and then the power set of that power set, and so on, to infinity. Then, go on and take the union of all the sets you have — and the power set again, etc., this *transfinite induction never ends, not even at infinity.* There are infinitely many different infinities. And so on.

As a side remark, it is quite ironic that this famous "diagonal argument" by Cantor is very similar in its underlying idea to Russell's "barber" argument that made Cantor's paradise, his "naïve" set theory, collapse. How could Cantor not see it? (Perhaps one is sometimes blind for arguments endangering one's own ideas and thoughts, even when the former resemble the latter, i.e., even if one's thinking style questions questions and limits itself. One of *G.W.F. Hegel*'s thoughts is that this is always the case: Everything is born with its own crack, its own end.)

Let us now return to the ground, to the topic of this course: *Finite* sets, and to counting their size: *Combinatorics, yeah*!

3.4 Exercises

Exercise 3.1 (Set Calculus). Prove the following rules for sets A, B and C.

1. Absorption:
$$A \cap (A \cup B) = A \qquad A \cup (A \cap B) = A \tag{3.2}$$
2. Distributivity:
$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \qquad A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \tag{3.3}$$
3. De Morgan:
$$\overline{A \cup B} = \overline{A} \cap \overline{B} \qquad \overline{A \cap B} = \overline{A} \cup \overline{B} \tag{3.4}$$
4. Discuss the obvious duality, *i.e.*, the possibility to swap $\cap$ and $\cup$ in a particular rule and to obtain another valid rule this way.

Exercise 3.2 (Hasse Diagrams). 1. Draw the Hasse diagram for the inclusion relation on the set $\mathbf{P}(\{1, 2, 3, 4, 5\})$.

2. Draw the Hasse diagram for the divisibility relation on the set of divisors of the number 2310.
3. Compare the diagrams. What do they represent? Explain your observations.

Exercise 3.3. Family lunch. Consider the sets

$$A = \{10; 20; 30; 40; 60; 400\}, \quad A' = \{10; 20; 30; 40; 60\}$$

$$B = \{1; 2; 3; 4; 6\} \quad C = \{1; 2; 3; 5; 6\}$$

and the relation R:= "x and y start with the same number." Which of the following options yield a function?

1. $x \in A$ and $y \in B$
2. $x \in A$ and $y \in C$
3. $x \in A'$ and $y \in B$

Chapter 4

Combinatorics

Combinatorics is a collection of methods, principles, tools, techniques, and facts to count the size of finite sets with some structure. We start by introducing some basic notions by means of an example.

4.1 Basic notions

Example 4.1 (Manhattan). Imagine walking from a point A to another point B on a rectangular grid such as the streets of Manhattan, as shown in Figure 4.1. The question is: *How many shortest paths are there from A to B?* We will see two different ways to answer the question yielding the same answer (hopefully!): The first approach is recursive (in algorithms, this is called *divide et impera*; the second one is "one-shot" (not recursive) and based on counting permutations of the steps in a path from A to B.

Divide and conquer. The key observation for the recursive argument is this: The number of shortest paths from A to B equals *the sum of the numbers*

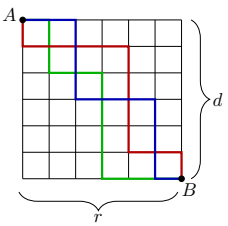


Figure 4.1: Some of the shortest paths are highlighted in colors.

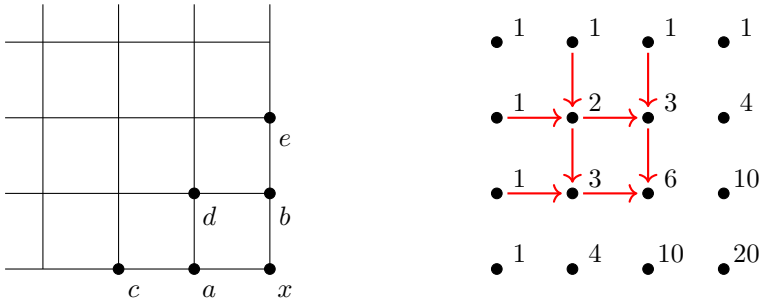


Figure 4.2: In the figure to the left the variables a, b, c, d, e, and x refer to the number of paths from A to the corresponding points. Note that they do not refer to the coordinates. The figure on the right shows the number of shortest paths from A to its neighboring points.

of shortest paths from A to the two neighbors of B pointing towards A (i.e., the one above B and the one to the left of B). Why? Well, every shortest paths from A to B must pass through *exactly one* of these two points — all others unavoidably make some detour. Let x be the number of shortest paths from A to B, then $x = a + b$ if a and b are the corresponding numbers for the described points (see Figure 4.2).

The nature of such a recursive approach — reducing the problem in question to the same problem, but "smaller" — is to *continue* the reduction until one gets to the trivial base case, which in is our case: counting the number of shortest paths between the only vertically or only horizontally connected points:

$$\begin{aligned} x &= a + b \\ a &= c + d \\ b &= d + e. \end{aligned} \tag{4.1}$$

In this way, we reduce the computation of x until we get to points to which the number of shortest paths from A is one (Figure in 4.2). The pattern of numbers that emerges in the grid is the *Pascal triangle*, shown in Figure 4.3, rotated by 45 °. The recursive law underlying the Pascal triangle is the same: An internal number is the sum of the two neighbors "upstairs," and the marginal numbers equal one. The positive-diagonal levels are labeled by $k = 0, 1, 2, \ldots$, the horizontal levels are labeled by $n = 0, 1, 2, \ldots$. The entries of the triangle are denoted by

$$\binom{n}{k}.$$

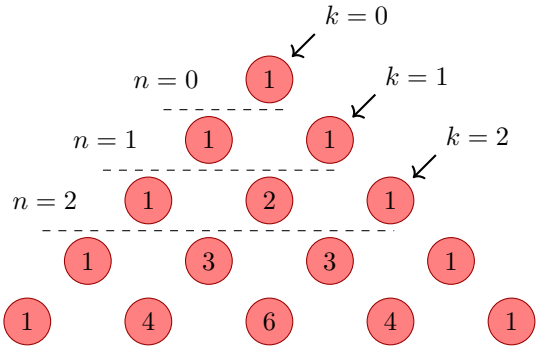


Figure 4.3: The Pascal triangle. The positive diagonals are labeled by k, the horizontal levels by n.

This entity is usually called the *binomial coefficient* "n choose k." Our observation that the number of paths can be calculated from the number of paths ending in previous nodes (i.e., equation 4.1), in terms of n and k yields the following recursive formula

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \tag{4.2}$$

With the base-case defined as

$$\binom{n}{0} := \binom{n}{n} := 1 \ .$$

The binomial coefficients are completely determined, as they can be computed for any n and k by building up the triangle employing the recursion relation in Equation (4.2).

(Note that, *algorithmically* speaking, simply applying the recursion would lead to a very inefficient computation: After all, computing the coefficient would be reduced to, ultimately, adding up 1's. The reason for the inefficiency is that the (smaller) subproblems to which the initial problem is reduced are solved over and over again, even if they have been already. The remedy when such simple recursion fails in this way is *dynamic programming*: Filling up a table with the intermediate results (in our case: the Pascal triangle) to avoid the repetitions.)

We can now write the number of shortest paths by setting $n = d + r$ and $k = r$ to obtain

$$x = \binom{d+r}{r} \ .$$

Counting Step Sequences. A second way to approach the problem is to take into account the different step sequences: We denote a step *down* with a capital D and a step to the *right* with R. Every sequence of steps forming one of the shortest paths from A to B as, for instance,

$$R_1 D_1 R_2 R_3 \cdots D_{d-1} D_d R_r$$

contains r steps to the right and d steps down. A first idea is that the number of shortest paths from A to B is the number of *permutations* of these $d + r$ steps, which equals $(d + r)!$:[1].

$$(d + r) \cdot (d + r - 1) \cdot (d + r - 2) \cdot 2 \cdot 1 = (d + r)!$$

This, however, counts each paths multiply, since permuting the different steps down among each other has no effect on the path. Specifically, the two sequences

$$R_1 R_2 D_1 \qquad R_2 R_1 D_1$$

lead to the same path. More precisely, we have counted each path how many times? The number of permutations among the D's is $d!$, and of the R's $r!$ correspondingly, so $d!r!$ times altogether. Thus, the number x of paths in question is

$$x = \frac{(r + d)!}{r! \cdot d!} \ .$$

The fact that the two approaches yield the same result means for the binomial coefficients, replacing $d + r = n$, $r = k$ and $d = n - k$, that

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{k(k-1)(k-2)\cdots 2 \cdot 1} \ . \tag{4.3}$$

Let us now close the circle and show the recursive formula directly for the explicit representation of the binomial coefficients. More specifically, we show that the right-hand side of 4.3 satisfies the base conditions and the recursion relation. Let us first check the base case: By convention $0! = 1$. Indeed,

$$\frac{n!}{0!n!} = 1 \ .$$

We verify the recursion relation:

$$\begin{aligned} \frac{n!}{k!(n-k)!} + \frac{n!}{(k+1)!(n-k+1)!} &= \frac{n!}{(k-1)!(n-k)!k} + \frac{n!}{(k-1)!(n-k)!(n-k+1)} \\ &= \frac{n!(n-k+1+k)}{k!(n-k+1)} = \frac{(n+1)!}{k!(n+1-k)!} \end{aligned}$$

[1] The number of permutations (i.e., different total order relations on a set) of n elements is $n!$. To see this, imagine we put the n elements one by one into an order. For the first element, we can choose among n elements, for second among $n - 1$, and so on. We obtain, therefore, $n \cdot (n-1) \cdots 2 \cdot 1 = n!$ different orders (or permutations).

This yields the recursion relation (note that we merely shifted the index relative to equation 4.2).

We clarify the origins of the terms "choose" as well as "binomial" connected to the coefficients.

Subsets. Why do we say "n choose k" referring to the binomial coefficient? Because this is the number of ways in which we can choose k out of n elements. More statically, $\binom{n}{k}$ is equal to the number of subsets of size k of an n set.

To show that, we prove that the number of k-element subsets of an n-element set satisfies the same recursion relation, including the base cases, as the binomial coefficient. There is one subset with zero elements — the empty set — and one subset with n elements — the set itself: The base case works.

For the recursion relation, let us consider a set containing $n+1$ elements. How many k-element subset does this set have? We argue recursively: Let us single out the $(n+1)$-th element of the set. The number of k-element subsets of the full set is the number of k-subsets of the n-set (without the singled-out element) plus the number of $(k-1)$-subsets of the n-set (for counting the k including the singled-out element). Thus we obtain

$$\#(k \text{ out of } (n+1)) = \#(k \text{ out of } n) + \#((k-1) \text{ out of } n).$$

This is again the recursion relation from equation 4.2.

Binomials. A *binomial*[2] is an expression of the form $(x+y)^n$. What is the connection to the coefficients? If we carry out the multiplication and write the binomial as a *sum of products* instead of a *power of a sum* (as given), we get

$$\begin{aligned}(x+y)^n &= (x+y)\cdot(x+y)\cdots(x+y)\\ &= \underbrace{(x\cdot x\cdots x)}_{=x^n} + \underbrace{(y\cdot x\cdots x)}_{=yx^{n-1}} + \ldots + y^n\ .\end{aligned}$$

In order to get one summand, we choose one variable from each of the brackets. In total, there are 2^n summands, containing each n factors. We can now collect factors that appear in the sum multiple times. For instance, the summand x^n appears only once, since x has to be chosen from all brackets to get this term. The summand $y\cdot x^{n-1}$ appears n times. Why? It corresponds to the number of possibilities to choose the one bracket (among n) from which we choose the y-factor. Correspondingly, the term $x^{n-k}y^k$ appears $\binom{n}{k}$ times. Together, we get

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}\ .$$

That's it!

[2]Take care, it is often confused with "binominal."

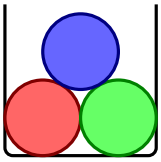


Figure 4.4: Urn with three elements.

4.2 Urn models

How many combinations of k elements can be drawn from an urn containing n elements (as shown in Figure 4.4)? The number of different combinations depends on whether the order of the combinations matters, and whether the elements are put back after each draw. To understand these cases better, we first consider the following example before turning to the general case.

Example 4.2. To keep matters simple, let us set $n = 3$ and $k = 2$. We will now go through all the four cases in detail.

Ordered, with repetition. After each draw, the element is put back into the urn. So for each draw, there are n choices. As the order matters, the two combinations (x, y) and (y, x) are not equal, similar to the Cartesian product, and we therefore obtain the following 9 combinations:

$$\begin{array}{lll} (1,1) & (1,2) & (1,3) \\ (2,3) & (2,2) & (2,3) \\ (3,3) & (3,2) & (3,3) \end{array}$$

Ordered, without repetitions. Elements once drawn from the urn are not put back again. Thus, the number of choices reduces by one with each draw and we obtain the following 6 combinations:

$$\begin{array}{lll} & (1,2) & (1,3) \\ (2,3) & & (2,3) \\ (3,3) & (3,2) & \end{array}$$

Unordered, with repetition. Again we put the elements back after each draw, but the order of the drawn combination does not matter. That is, we regard the combinations above and below the diagonal in the matrices above as equal. Therefore, we are left with 6 combinations:

$$\begin{array}{lll} (1,1) & (1,2) & (1,3) \\ & (2,2) & (2,3) \\ & & (3,3) \end{array}$$

Unordered, without repetition. If we do not put back the elements after each draw, the combinations with twice the same element, i.e., those on the diagonal, cannot occur. Thus we are left with the following 3 cases:

$$\begin{matrix}(1,2) & (1,3) \\ & (2,3)\end{matrix}$$

We would now like to generalize from the example above to arbitrary n and k. The matrix picture is somewhat misleading as it is merely helpful in the case $k = 2$. One rather multiplies the number of choices as we will see.

Ordered, with repetition. For each of the draws, there are n choices. So the number of combinations is

$$\underbrace{n \cdot n \cdots n}_{k \text{ times}} = n^k.$$

Ordered, without repetition. If the elements are not returned to the urn, the number of choices decreases by one with each draw. Thus the number of combinations is

$$n \cdot (n-1) \cdot (n-2) \cdots (n-k+1) =: n^{\underline{k}}.$$

The special case $k = n$ yields the number of permutations of n elements, i.e., $n^{\underline{n}} = n!$, as mentioned already before.

Unordered, without repetition. The number of unordered combinations of k elements without repetition corresponds to the k-element subsets of a set with n elements. Recall that two sets are equal if they contain the same elements, independent of the order. Thus, the number of combinations is given by the binomial coefficient:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Unordered, with repetition. The last case, with repetition but unordered, is slightly more complicated. A vote is an example of this case. The order of the votes does not matter, but merely how many votes each candidate got. Imagine 3 candidates and 20 voters. Therefore, $n = 3$ and $k = 20$.[3] How many different distributions of votes are there? As the order does not make

[3] As repetitions are allowed, the number of draws k might as well be bigger than the number of elements in the urn, n.

a difference, we might as well order the votes such that the votes for the first candidate come first and then the ones for the seconds, etc.:

$$\underbrace{1\ldots1\mid2\ldots2\mid3\ldots3}_{20\text{ votes}}$$

For arbitrary numbers of candidates and voters we similarly obtain

$$\underbrace{1\ldots1\mid2\ldots2\mid\ldots\mid n\cdots n}_{k\text{ votes}}$$

with $n-1$ separators $\mid$. In this notation the separators also indicate the votes. Left of the first separator are the votes for the first candidate, left of the second separator the votes for the second candidate and so on and so forth. We can therefore write

$$\underbrace{\star\star\ldots\star\mid\star\ldots\star\mid\ldots\mid\star\ldots\star}_{k\text{ stars}}$$

The combinations are characterized merely by the order of stars and separators. To see how many arrangements there are, we take $n-1+k$ positions and fill them either with stars or with separators. Then there are

$$\binom{n-1+k}{n-1}=\binom{n+k-1}{k}$$

different arrangements for k stars and $n-1$ separators as this corresponds to k-element subsets of an $(n+k-1)$-element set (or equivalently to k-element subsets by the symmetry of the binomial coefficient)[4].

The extensionally axiom of set theory establishes a connection to unordered combinations. Two sets that differ only by a permutation of the elements are equal, analog to unordered combinations. The emergence of the binomial coefficient is a result of this analogy.

4.3 Rules and strategies

When one is faced with a counting problem of some set, it is often possible to count *parts of this set* using the standard urn models. What remains is then the problem of composing the partial solutions to determine the number of elements of the set in question. We discuss some principles, some of them trivial, one — "inclusion/exclusion" — which allow for exactly this.

[4]One could also consider the number of permutations of all $n+k-1$ stars and separators. Then we have to divide by the number of permutations among the stars and among the separators as they are indistinguishable. So we obtain the binomial coefficient again.

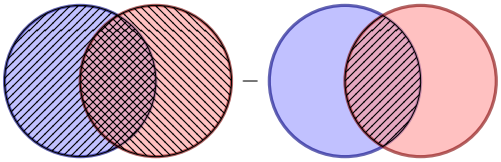


Figure 4.5: The elements in the intersection are counted twice and have to be subtracted again.

The sum rule. Consider a family $(A_i)_{i=1,\dots,n}$ of mutually disjoint sets, i.e.,

$$A_i \cap A_j = \emptyset \quad \forall i \neq j \ .$$

Then the size of the *union* of these sets is the *sum* of the sizes of the sets:

$$\left| \bigcup_{i=1}^{n} A_j \right| = \sum_{i=1}^{n} |A_i| \ .$$

The product rule. For a family of sets $(A_i)_{i=1,\dots,n}$, disjoint or not, the size of the Cartesian product of the family is equal to the *product* of the sizes of the individual sets A_i:

$$\left| \bigtimes_{i=1}^{n} A_i \right| = \prod_{i=1}^{n} |A_i| \ .$$

The equality rule. Two finite sets A, B have the same number of elements if there exists a bijective function $f : A \to B$: The function establishes a one-to-one correspondence between the elements of A and B.

The principle of inclusion/exclusion. We generalize the sum rule above to sets that are *not* mutually disjoint. Let us first consider the cases of families of two and three sets. The union of two sets, disjoint or not, is of size

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2| \ .$$

The rationale is: If we simply add the two sizes, then the "overlap," i.e., the elements that are in both sets, are counted twice; hence, we have to "remove them once" (see Figure 4.5).

Example 4.3 (The size of the union of two general sets.). How many of the numbers 1, 2, 3, ..., 100 are divisible by 2 *or* by 5? The numbers that are divisible by 2 form a set A_1 with 50 elements, the numbers divisible by 5 a set A_2 with 20 elements. Their intersection has size 10. Therefore,

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2| = 60 \ .$$

What is remarkable already in this example: Counting *one* set if reduced here to counting *three* sets — what have we won?

Let us consider the case of three sets. Again, we first sum up the sizes of the individual sets. Then, we subtract the sizes of the three pairwise intersections. We have now counted *exactly once* (as it should be) the elements that are in exactly one of the sets, and in exactly two. But what about the elements that are *in all three sets*? We have counted them thrice and then subtracted thrice, so in the end, we have not counted them at all. Thus, we need to add them at the end. We already sense now the continued change of signs that gave the name to the principle:

$$\begin{aligned} |A_1 \cup A_2 \cup A_3| &= |A_1| + |A_2| + |A_3| \\ &\quad - |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 + A_3| \\ &\quad + |A_1 \cap A_2 \cap A_3| \ . \end{aligned}$$

Instead of the thought just displayed, we could also reduce the case of three sets to repeated application of the case of two:

$$\begin{aligned} |(A_1 \cup A_2) \cup A_3| &= |A_1 \cup A_2| + |A_3| - \underbrace{|(A_1 \cup A_2) \cap A_3|}_{|(A_1 \cap A_3) \cup (A_2 \cap A_3)|} \\ &= |A_1| + |A_2| - |A_1 \cap A_2| + |A_3| - (|A_1 \cap A_3| + |A_2 \cap A_3|) \ . \end{aligned}$$

This step turns out to be crucial in our — recursive — proof of the *general* case of n sets. It states that the size of a union of sets is equal to the sum of the sizes of the individual sets, *minus* the sum of the sizes of pairwise intersections, *plus* the size of threewise intersections, *minus* the sizes of the intersections of quadruples, etc. You see now where the name comes from.(And you also see how much simpler it must often be to count intersections than unions: After all, the price is that we have to count an exponential number of them.)

Theorem 4.1 (Inclusion/Exclusion)**.** *The size of a union of sets* $(A_i)_{i=1,\dots,n}$ *is given by*

$$\left| \bigcup_{i=1}^{n} A_i \right| = \sum_{r=1}^{n} (-1)^{r-1} \sum_{1 \le i_1 < i_2 < \dots < i_r \le n} \left| \bigcap_{k=1}^{r} A_{i_k} \right| \ . \tag{4.4}$$

The sum in the statement is over all families if size r.

Proof. Induction over n. The case cases $n = 1$, $n = 2$, and $n = 3$ have been checked above. (Actually, the case $n = 2$ is not only important as a base case, but is explicitly invoked in the induction step.) Let us assume the statement is valid for all numbers up to $n \ge 2$. Consider a family of $(n + 1)$ sets — of

which we single out the $(n+1)$st set.

$$\left|\bigcup_{i=1}^{n+1} A_i\right| = \left|\left(\bigcup_{i=1}^{n} A_i\right) \cup A_{n+1}\right| = \left|\bigcup_{i=1}^{n} A_i\right| + |A_{n+1}| - \underbrace{\left|\left(\bigcup_{i=1}^{n} A_i\right) \cap A_{n+1}\right|}_{=\left|\bigcup_{i=1}^{n}(A_i \cap A_{n+1})\right|}$$

$$= \sum_{r=1}^{n}(-1)^{r-1} \sum_{1\le i_1<i_2<\ldots<i_r\le n} \left|\bigcap_{k=1}^{r} A_{i_k}\right| + |A_{n+1}| + \\ - \sum_{r=1}^{n}(-1)^{r-1} \sum_{1\le i_1<i_2<\ldots<i_r\le n} \left|\bigcap_{k=1}^{r} (A_{i_k} \cap A_{n+1})\right|$$

The last line is of the form of Equation (4.4). To see this, we expand Equation (4.4) with $(n+1)$. First, we split of the term $r=1$, then all the terms with $i_r = n+1$:

$$\sum_{r=1}^{n+1}(-1)^{r-1} \sum_{1\le i_1<i_2<\ldots<i_r\le n+1} \left|\bigcap_{k=1}^{r} A_{i_k}\right| = \sum_{k=1}^{n}|A_k| + |A_{n+1}| + \sum_{r=2}^{n+1}\ldots$$

$$= \sum_{k=1}^{n}|A_k| + |A_{n+1}| + \sum_{r=2}^{n}(-1)^{r-1} \sum_{1\le i_1<i_2<\ldots<i_r\le n} \left|\bigcap_{k=1}^{r} A_{i_k}\right| + \\ + \sum_{r=2}^{n+1}(-1)^{r-1} \sum_{1\le i_1<i_2<\ldots<i_{r-1}\le n} \left|\bigcap_{k=1}^{r-1} A_{i_k} \cap A_{n+1}\right|$$

$$= \sum_{k=1}^{n}|A_k| + |A_{n+1}| + \sum_{r=2}^{n}(-1)^{r-1} \sum_{1\le i_1<i_2<\ldots<i_r\le n} \left|\bigcap_{k=1}^{r} A_{i_k}\right| + \\ - \sum_{r=1}^{n}(-1)^{r-1} \sum_{1\le i_1<i_2<\ldots<i_r\le n} \left|\bigcap_{k=1}^{r} A_{i_k} \cap A_{n+1}\right|$$

$$= |A_{n+1}| + \sum_{r=2}^{n}(-1)^{r-1} \sum_{1\le i_1<i_2<\ldots<i_r\le n} \left|\bigcap_{k=1}^{r} A_{i_k}\right| + \\ - \sum_{r=1}^{n}(-1)^{r-1} \sum_{1\le i_1<i_2<\ldots<i_r\le n} \left|\bigcap_{k=1}^{r} A_{i_k} \cap A_{n+1}\right|$$

In the penultimate step, the index r in the last term was shifted. The emerging (-1) factor was taken in front of the sum. In the last step, the sum $\sum_{k=1}^{n}|A_k|$ was absorbed into the third term as $r=1$. □

Example 4.4 (Opera). During an opera evening with 500 guests, all the coats in the cloakroom get disordered, and every guest gets back a random coat

afterwards. Is it more likely that at least one person gets back their own coat or that nobody gets back their own coat? (Try to guess it.) Let us first consider a simpler case with three guests. The three coats in the cloakroom are permuted. At least one person gets their own coat if the permutation has a *fixed point*. For instance, the permutation

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$$

has the fixed point 3. If no guest retrieves their own coat, then the corresponding permutation is *fix-point free* (we abbreviate such a permutation by FPFP). For three coats, there are exactly two FPFPs (out of 6):

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

The probability that nobody retrieves their own coat is, hence, 1/3, whereas the probability that at least one person retrieves the own coat is, correspondingly, $1 - 1/3 = 2/3$. How do we calculate this if the number of guests (and, therefore, the number of coats) is larger? This question can be answered using the inclusion/exclusion principle 4.4: We express the set containing all permutations with at least one fix-point, denoted hereafter A, as a union of sets, and use "inclusion/exclusion" to determine the size of this union.

Let A_i be the set of permutations for which i is a fixed point. (There can be other fixed points besides it, or not.) Then,

$$\bigcup_{i=1}^{n} A_i = A \ ,$$

and the size of A can be calculated using "inclusion/exclusion." Again, at first sight it might not look like a gain to reduce counting a single set to counting an exponential number of sets. But it is: First, many of these sets have the same size, which is easy to determine, and the number of sets in such groups is easy to determine as well.

Note first that the size of each of the A_i is equal to the number of permutations of $(n-1)$ elements, i.e., $(n-1)!$. Furthermore, the size of the intersections of two different A_i is equal the number of permutations of $(n-2)$ elements, i.e., $(n-2)!$. And so on:

$$\begin{aligned} |A| = \left| \bigcup_{i=1}^{n} A_i \right| &= \sum_{r=1}^{n} (-1)^{r-1} \sum_{1 \le i_1 < i_2 < \ldots < i_r \le n} \left| \bigcap_{k=1}^{r} A_{i_k} \right| \\ &= \sum_{r=1}^{n} (-1)^{r-1} \sum_{1 \le i_1 < i_2 < \ldots < i_r \le n} (n-r)! \ . \end{aligned}$$

For each r, there are $\binom{n}{r}$ terms in the inner sum. Employing the equality $\binom{n}{r} = \frac{n!}{(n-r)!r!}$, we obtain

$$|A| = \sum_{r=1}^{n} (-1)^{r-1} \binom{n}{r} (n-r)! = n! \sum_{r=1}^{n} \frac{(-1)^{r-1}}{r!} \ .$$

The number of fixed-point-free permutations is then

$$n! - |A| = n! \left(1 - \sum_{r=1}^{n} \frac{(-1)^{r-1}}{r!} \right) = n! \sum_{r=0}^{n} \frac{(-1)^{r}}{r!} \ .$$

Using that the exponential can be written as

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} \ , \tag{4.5}$$

we obtain, with $x = -1$, that the number of fix-point-free permutations tends to $n!/e$:

$$\lim_{n \to \infty} \#\mathrm{FPFP}(n) = \frac{n!}{e} \ :$$

The probability that nobody retrieves their own coat is $1/e \approx 1/2.71828$ and thus smaller than the probability that at least one person retrieves their own coat.

What is remarkable about this is that the probability is (essentially) independent of the number of people, i.e., it converges (very fast) to a number that is neither 0 nor 1.

4.3.1 The pigeonhole principle

"If there are more pigeons inside a pigeonry than holes in the pigeonry, then at least two pigeons have to leave through the same hole." That's it!

Theorem 4.2 (Pigeonhole Principle)**.** *When n objects are distributed among k boxes, with $k < n$, then there is at least one box containing at least two objects.*

Proof. Induction over k. In the base case, with $k = 1$, there is merely one box containing all n elements, with $n \geq 2$. For the induction step, we prove that the assertion holds for $k+1$ boxes, assuming that it holds for k boxes. Let us consider the $(k+1)$-th box separately, and distinguish the following two cases:

1. The box $(k+1)$ contains *at least two* objects.
2. The box $(k+1)$ contains *at most one* object. Then, the other $n-1$ objects are distributed among the remaining k boxes. If $n > k+1$, then $n-1 > k$, and by induction hypothesis there is one box among these n which contains at least two objects.

This completes the induction step, and the proof. □

The following example illustrates a beautiful application of the principle.

Example 4.5 (Monotonic subsequences). If we are given a finite sequence of *distinct* numbers as, for instance,

$$(1, 17, 5, 3, 20, 2, 4) \ ,$$

then a subsequence is a selection of these numbers with the order given by the original sequence, such as

$$(17, 20, 4) \ .$$

A subsequence is *monotonically increasing* if the numbers are in ascending order. Correspondingly, the sequence is *monotonically decreasing* if the numbers are in descending order. The sequence

$$(1, 3, 20)$$

is monotonically increasing, whereas the sequence

$$(17, 5, 3, 2)$$

is monotonically decreasing.
The question is: For a sequence of length n, what is the *best lower bound on the length of the longest monotonic subsequence*? The answer is:

A sequence of length $m^2 + 1$ has always a monotonic subsequence of length $m + 1$, and this bound is "tight" (i.e., it cannot be improved in general).

Proof. By contradiction. Let us consider a sequence of $m^2 + 1$ elements,

$$(a_1, \ldots, a_{m^2+1}) \ .$$

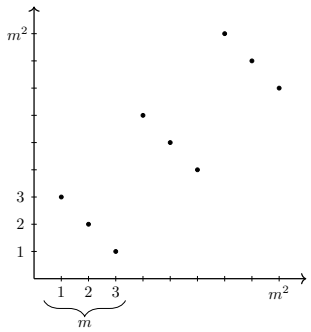


Figure 4.6: A sequence with m^2 elements and monotonic subsequences with m elements.

We assume the opposite of what we desire to show, namely, that the longest subsequence has length $l \leq m$. To each number a_i in the sequence we now associate the length c_i of the longest monotonically increasing subsequence *starting with* a_i and in particular *containing* a_i, and the length d_i of the longest monotonically decreasing subsequence starting with a_i. For each a_i, we obtain a pair (c_i, d_i). Both, c_i and d_i are less or equal to l and, therefore, less or equal to m. Thus, there are m^2 possible *different* ordered pairs. The pigeonhole principle states that there are at least two numbers with the same values (c, d):[5]

$$\begin{array}{c} (a_1, \ldots, \underset{\searrow}{a_i}, \ldots, \underset{\searrow}{a_j}, a_{m^2+1}) \\ (c,d) \qquad (c,d) \end{array}$$

We distinguish now the two cases:

$a_i < a_j$ Then we can construct a monotonically increasing subsequence of length $c + 1$ by adding a_i to the subsequence starting from a_j. This yields a contradiction, with c being the length of the longest subsequence starting from a_i.

$a_i > a_j$ One obtains a monotonically decreasing subsequence of length $d+1$. This is in contradiction, with d being the length of the longest monotonically decreasing subsequence starting from a_i.

Therefore, the assumption that the length of the longest monotonic subsequence is less or equal to m leads to a contradiction. □

Is the bound tight? Yes: We can come up with a sequence of length m^2 of which the longest monotonic subsequence has length only m; see Figure 4.6.

[5]The m^2 possible pairs correspond to the holes and the $m^2 + 1$ elements of the sequence to the pigeons.

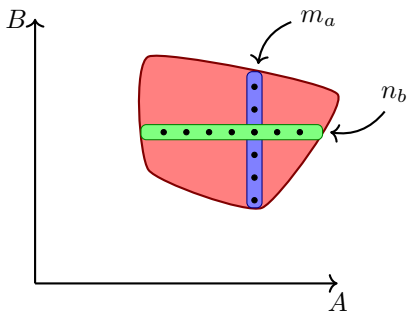


Figure 4.7: One can count the elements in S either row after row, or column after column.

4.3.2 Double counting

There are two different ways of counting the element of a relation S from A to B, i.e., of a subset of a Cartesian product $A \times B$, i.e. in a relation, $S \subseteq A \times B$: We can count the set column by column using the subsets

$$m_a := \{b \in B \mid (a, b) \in S\} \subseteq S \ ,$$

or count the elements in S row by row using the subsets

$$n_b := \{a \in A \mid (a, b) \in S\} \subseteq S \ .$$

Of course, we obtain the same result, namely, the size of S in both cases:

$$|S| = \sum_{a \in A} |m_a| = \sum_{b \in B} |n_b| \ .$$

Example 4.6. What is the "average" number of divisors of a number k? To answer this question, let us introduce the function counting the number of divisors:

$$\nu(k) := |\{l > 0 \mid l \text{ divides } k\}| \ .$$

The function takes the following values for small arguments:

$$\nu(1) = 1 \qquad \nu(2) = 2 \qquad \nu(3) = 2 \qquad \nu(4) = 3 \qquad \nu(5) = 2 \qquad \nu(6) = 4 \ .$$

Furthermore, for every prime number p, we obtain $\nu(p) = 2$. We define what we mean by "average number of divisors": For a given number n, we are interested in calculating

$$\sum_{k=1}^{n} \frac{\nu(k)}{n} \ .$$

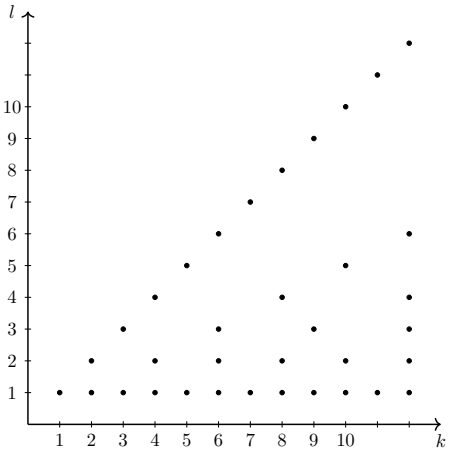


Figure 4.8: The divisors l of the numbers $k \leq n$.

Summing up the values of $\nu(k)$ is equivalent to counting the dots in Figure 4.8 column by column. The columns have a irregular pattern — whereas the rows are completely regular: Every second number is even, every third is divisible by three, etc. In particular, the number of points within each row is given by $\left\lfloor \frac{n}{k} \right\rfloor$: The fraction n/k is rounded to the next smaller integer value. (The floor function is used since 0 is not in the set: You always have to wait maximally long for the first point.) Using the principle and replacing the sum over columns by the sum over rows, we obtain

$$\frac{1}{n} \sum_{k=1}^{n} \nu(k) = \frac{1}{n} \sum_{l=1}^{n} \left\lfloor \frac{n}{k} \right\rfloor \ .$$

The rest is *calculus*: How does this number grow with n? To answer this question, we estimate the sum from above. Note, first,

$$\frac{n}{l} - 1 \leq \left\lfloor \frac{n}{l} \right\rfloor \leq \frac{n}{l} \ .$$

Therefore, we can bound the average as

$$\left(\sum_{l=1} \frac{1}{l} \right) - 1 \leq \frac{1}{n} \sum_{l=1}^{n} \left\lfloor \frac{n}{l} \right\rfloor \leq \sum_{l=1}^{n} \frac{1}{l} \ .$$

Thus, we require bounds on the sum $\sum_{l=1}^{n} 1/l$. As shown in Figure 4.9, we can approximate with functions and integrate to obtain the area below the graph.

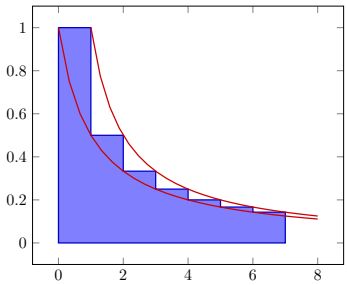


Figure 4.9: Estimation using integrals.

The upper bound is then

$$\sum_{l=1}^{n} \frac{1}{l} \leq 1 + \int_{1}^{n} \frac{1}{x} dx = 1 + \ln(n) \ ,$$

where ln is the natural logarithm (i.e., with respect to the basis e). Similarly, we obtain the lower bound

$$\sum_{l=1}^{n} \frac{1}{l} \geq \int_{0}^{n} \frac{1}{x+1} dx = \ln(n+1) \geq \ln(n) \ .$$

Putting all this together, the average of the number of divisors for numbers between 1 and n can be estimated as

$$\ln(n) - 1 \leq \frac{1}{n} \sum_{l=1}^{n} \left\lfloor \frac{n}{l} \right\rfloor \leq \ln(n) + 1 \ .$$

4.4 Binomial coefficients: Properties and approximations

As shown above, the number of k-sets that can be chosen from an n-set is given by the binomial coefficient:

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \binom{n}{n-k}$$

4.4.1 Symmetry

The binomial coefficient reflects the symmetry of the Pascal triangle as

$$\binom{n}{k} = \binom{n}{n-k} \ .$$

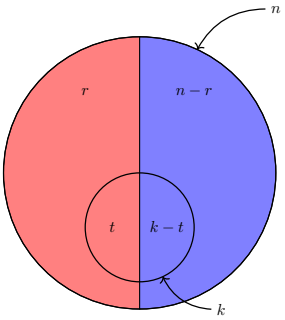


Figure 4.10

This can also be seen from the formula above.

4.4.2 Vandermonde identity

There are $\binom{n}{k}$ possibilities to choose k elements from a set containing n elements. Imagine now that there are r blue and $n-r$ red elements in the set of n elements. Then there are

$$\binom{r}{t} \cdot \binom{n-r}{k-t}$$

possibilities to choose k elements such that t of these are red, while $k-t$ are blue. If we add the numbers of k-sets from an n-set with $t=0$ elements red, $t=1$ elements red, $t=2$ elements red until we reach k red elements, it is just the same as not bothering about the colors, and we are left with $\binom{n}{k}$. Thus, we obtain the equality

$$\binom{n}{k} = \sum_{t=0}^{k} \binom{r}{t} \cdot \binom{n-r}{k-t}$$

Note that $\binom{n}{k}$ is set to zero whenever $k > n$.

4.4.3 Binomial theorem

For the sake of completeness, we mention again the binomial theorem (see also 4.3):

$$(x+y)^n = \sum_{k=1}^{n} \binom{n}{k} x^k y^{n-k}$$

If we set $x = y = 1$, we obtain the sum over one row in Pascal's triangle:

$$2^n = \sum_{k=1}^{n} \binom{n}{k}$$

This yields the number of *all* subsets of an n-set and thus the size of the power set.

Another interesting special case is $x = -1$ and $y = 1$. This yields

$$\sum_{k=1}^{n} \binom{n}{k} (-1)^k = 0^k = 0 \ . \tag{4.6}$$

That is, summing over any row with alternating sign yields zero. For rows with n being odd, the number of terms in the row is even. So 4.6 for n being odd follows directly from the symmetry of the Pascal triangle. For any row with n even it is not obvious though.

From this we can draw a conclusion for the number of strings with even and with odd parity. The parity of a string is even if the number of ones in the string is even and it is odd if the number of ones in the string is odd. So the left-hand side of the equation 4.6 is just the number of even parity strings minus the number of odd parity strings. Thus, the number of odd parity strings and the number of even parity strings are always equal. For strings with an odd number of elements, there is a bijection that shows the equal number of even and odd parity strings. One simply flips all bits. So, for instance,

$$010 \mapsto 101.$$

For strings with an even number of elements, the fact is not that obvious but follows from equation 4.6.

4.4.4 Approximation of the binomial coefficient

For large numbers n and k, the binomial coefficient is rather hard to compute. Therefore, we would like to have an estimate. First of all, we derive upper and lower bounds. To obtain a lower bound, we write the binomial coefficient with factorials and rearrange the factors as follows.

$$\begin{aligned} \binom{n}{k} &= \frac{n!}{k!(n-k)!} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{k(k-1)(k-2)\cdots 1} \\ &= \frac{n}{k} \cdot \frac{n-1}{k-1} \cdot \frac{n-2}{k-2} \cdots \frac{n-k+1}{1} \end{aligned}$$

If $n \geq k$ then the factor n/k is less or equal to $(n-1)/(k-1)$ as

$$n(k-1) = nk - n \leq (n-1)k = nk - k \qquad \Rightarrow \qquad \frac{n}{k} \leq \frac{n-1}{k-1} \ .$$

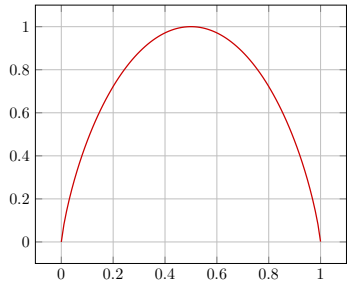


Figure 4.11: The graph of the binary entropy $h(x)$.

Repeating this argument for the subsequent factors, we obtain the lower bound

$$\binom{n}{k} \geq \left(\frac{n}{k}\right)^k$$

How much bigger can $\binom{n}{k}$ be than $(n/k)^k$? To answer this question we now derive an upper bound. Consider first the following ratio.

$$\frac{\binom{n}{k}}{\left(\frac{n}{k}\right)^k} = \underbrace{\frac{n(n-1)(n-2)\cdots(n-k+1)}{n^k}}_{\leq 1} \underbrace{\frac{k^k}{k(k-1)(k-2)\cdots 1}}_{\leq e^k} \leq \left(\frac{n}{k}\right)^k e^k \ .$$

We have employed once more the formula with factorials and arranged the factors in a convenient way. To upper bound the second part, we used the expansion of the exponential (4.5). One summand is less than the entire series.

Summarizing this we obtain the bounds:

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{n}{k}\right)^k \cdot e^k \ .$$

Stirling's formula provides a more precise estimate of the factorial

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

and can thus be used to estimate the binomial coefficient. The factors of e cancel as $e^k \cdot e^{n-k}/e^n = e^n/e^n = 1$. Further, we drop all square root factors

and obtain

$$\begin{aligned}\binom{n}{k} = \frac{n!}{k!(n-k)!} &\approx \frac{1}{\sqrt{2\pi}} \frac{\sqrt{n}}{\sqrt{k(n-k)}} \left(\frac{n}{e}\right)^n \left(\frac{e}{k}\right)^k \left(\frac{e}{n-k}\right)^{n-k} \\ &\approx \frac{n^n}{k^k (n-k)^{n-k}} = \frac{1}{\left(\frac{k}{n}\right)^k \left(\frac{n-k}{n}\right)^{n-k}} \\ &= \left(\frac{1}{\left(\frac{k}{n}\right)^{k/n} \left(\frac{n-k}{n}\right)^{(n-k)/n}}\right)^n .\end{aligned}$$

In the last two steps we merely reformulate the expression in a form that turns out to yield some insight later on.

Let us now define $x := k/n$. Thus, $(n-k)/n = 1-x$, and we can write the estimate above as

$$\binom{n}{k} \approx \left(x^x (1-x)^{1-x}\right)^{-n} = 2^{n(-x \log_2 x - (1-x)\log_2(1-x))} .$$

Further, we introduce the function

$$h(x) := -x \log_2 x - (1-x) \log_2 (1-x).$$

The function is symmetric about $x = 1/2$ and becomes 0 for both $x = 0$ and $x = 1$ and 1 for $x = 1/2$. The graph of h is shown in Figure 4.13. All this new formalism yields the estimate

$$\log_2 \binom{n}{k} \approx n h(x).$$

Why should we bother to write the estimate in a such a complicated way? The function h plays an important role in information theory, more concretely in results on data compression.

4.5 An excursion into information theory: Data compression

How much can we compress data without losing information? Let us consider a bit string generated by n coin flips. If the probability for heads and tails is $1/2$ we cannot compress this string. But if the probability for obtain zero is larger than $1/2$, i.e.,

$$x := P(0) > \frac{1}{2} ,$$

then there is redundancy, and we can compress to a smaller bit string of length l as depicted in Figure 4.12. As we are dealing with a probabilistic source of

n bit strings, we refine the question. What is the best compression, i.e., the smaller l, on average, without loss of information?

To answer this question, let us consider the set of all possible n-bit strings. What strings are likely to occur if they are produced with source generating a zero with probability $x > 1/2$ and one with a probability $1 - x$?

The single most likely string is then the zero string 0^n with a probability of x^n. If, for instance, $x = 0.9$, then $P(0^n)$ is 0.81 for $n = 2$ and 0.729 for $n = 3$. For $n \to \infty$ this probability tends toward zero.

Any string with equally many zeros and ones occurs with a probability of

$$x^{n/2} \cdot (1-x)^{n/2} \ .$$

As there are $\binom{n}{n/2}$ of these strings, the probability to draw any such string is

$$\begin{aligned}\binom{n}{n/2} x^{n/2}(1-x)^{n/2} &\approx 2^{nh(1/2)} x^{n/2}(1-x)^{n/2} \\ &= 2^n x^{n/2}(1-x)^{n/2} \\ &= \left(2 \cdot \sqrt{x(1-x)}\right)^n \ ,\end{aligned}$$

which approaches 0 when n increases. Here, we used that $\sqrt{x(1-x)} < 1/2$ for any $x > 1/2$, as can be seen from the graph shown in Figure 4.14. This is related to the geometric fact, that the rectangle with the largest area for a given circumference is a square.

Thus, for long strings, i.e., large n, the probability to draw a string with equally many zeros and ones is very small.

Which strings are instead drawn with high probability? Intuitively, we would expect the string with $n \cdot x$ zeros and $n \cdot (1-x)$ ones to be rather likely.

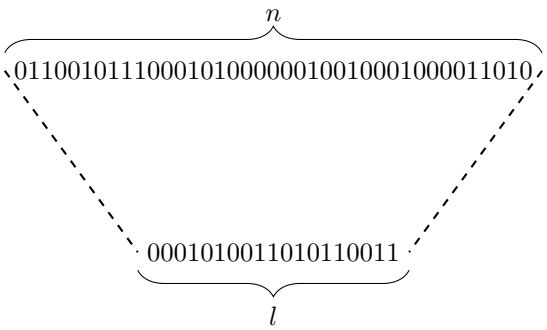


Figure 4.12: Compression of an n-bit string to an l-bit string.

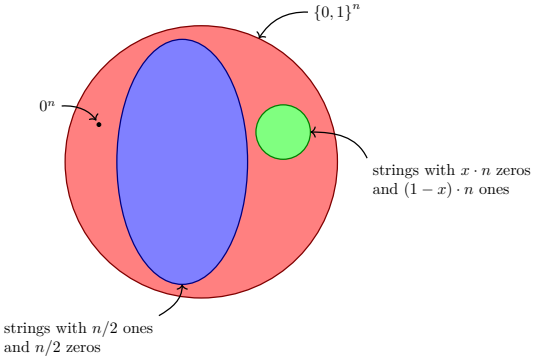


Figure 4.13: The set of all strings of length n with various subsets.

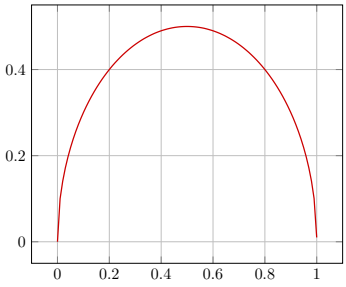


Figure 4.14: The graph of the function $f(x) = \sqrt{x(1-x)}$.

The probability to draw any such string is in fact

$$\begin{aligned} P[\#\text{ zeros} = x \cdot n] &= \binom{n}{x \cdot n} x^{x \cdot n} (1-x)^{n \cdot (1-x)} \\ &= \binom{n}{x \cdot n} 2^{n(x \log(x) + (1-x)\log(1-x))} = \binom{n}{x \cdot n} 2^{-nh(x)} \\ &\approx 2^{nh(x)} 2^{-nh(x)} = 1. \end{aligned}$$

Therefore, a lossless compression protocol merely encodes the roughly $\binom{n}{x \cdot n} \approx 2^{nh(x)}$ strings with $n \cdot x$ zeros and $(1-x) \cdot n$ ones using $l = nh(x)$ bits. So the binary entropy $h(x)$ characterizes the ultimate data compression rate or the relative "information content" of a source.

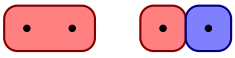
4.6 Special counting problems

4.6.1 Equivalence relations

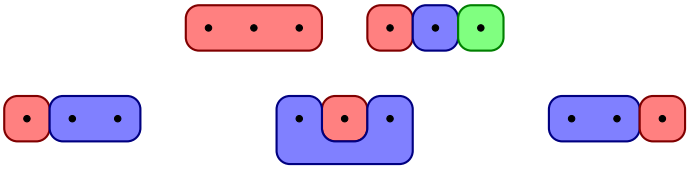
Recall from set theory that equivalence relations yield a partition of a set. The question we are now interested in this: How many equivalence relations are there for a given set with n elements? Let us first consider some simple cases with small n:

- $n = 1$: There is just one partition and therefore merely one equivalence relation.

- $n = 2$: There is are two partitions, the one containing both the elements, and the one with each element being contained in its own partition:



- $n = 3$: There are 5 partitions: one containing all elements, one with each element in its own partition, and three with two elements contained in the same partition:



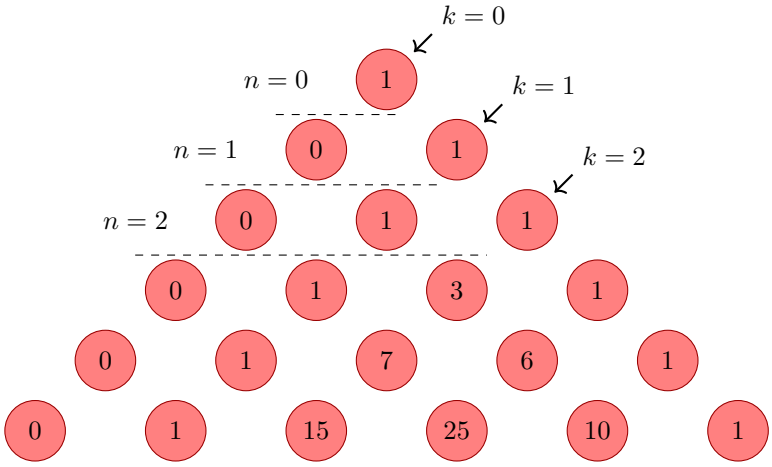


Figure 4.15: Stirling's triangle of the second kind.

Instead of trying to directly find a general answer to the question above, we first consider a slightly simpler question: How many partitions into exactly k sets are there? We denote this number by $S_{n,k}$. One can now deduce a recursive formula similar to the binomial coefficient. To do so, we separate the n-th element and distinguish the following two cases:

1. The n-th element is in its own a set. Then we have to find a $k-1$ partition of the remaining $n-1$ elements.

2. The n-th element is an element of a set containing also other elements. Imagine the other $n-1$ elements are partitioned into k sets. Then, we could add the n-th element to any of those k sets.

These considerations yield the following recursive formula:

$$S_{n,k} = S_{n-1,k-1} + k \cdot S_{n-1,k} \ .$$

This relation resembles the recursion relation for the binomial coefficient. There is merely the additional factor k. Indeed, after considering the base cases

$$\begin{aligned} S_{0,0} &= 1, \\ S_{n,0} &= 0 \qquad \forall n > 0, \\ S_{n,n} &= 1, \end{aligned}$$

we can build up a similar triangle, called Stirling's triangle of the second kind.

The number of all partitions is then given by the sum over row n in the triangle:

$$B_n := \sum_{k=0}^{n} S_{n,k}$$

There is no closed formula for this number. So, to calculate the number, one has to actually build up the triangle before calculating the sum over the row.

4.6.2 Permutations

A permutation is a bijective map:

$$\pi : \{1, \ldots, n\} \to \{1, \ldots, n\} .$$

A common notation for permutations is

$$\begin{pmatrix} 1 & 2 & \cdots & n \\ \pi(1) & \pi(2) & \cdots & \pi(n) \end{pmatrix} .$$

Example 4.7. The following is a permutation of 5 elements:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix} .$$

The permutation has a fix point, $3 \mapsto 3$, and two cycles of 2:

$$1 \mapsto 5 \mapsto 1 \qquad 2 \mapsto 4 \mapsto 2$$

Therefore, applying the permutation twice, yields the identity, $\pi^2 = \mathrm{id}$. In other words, the permutation is self-inverse, $\pi^{-1} = \pi$.

Example 4.8. Let us consider a permutation with a more complicated cycle structure:

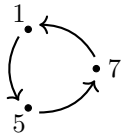
$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 5 & 9 & 8 & 10 & 7 & 6 & 1 & 3 & 4 & 2. \end{pmatrix}$$

There are the following cycles:

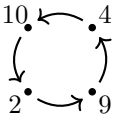
- A fix point at 6:

- There is one cycle of 2:

- There is one cycle of 3:



- There is one cycle of 4:



The counting problem we consider is this: How many permutations of n with exactly k cycles are there? As before, we aim for a recursion relation. Abusing slightly the notation, we denote the number permutation of n elements with k cycles by $S_{n,k}$. We can then separate the n-th element and distinguish the following two cases:

1. n is a fix-point and therefore adds a cycle by itself. It remains to take into consideration the permutations of $n-1$ elements with $k-1$ cycles.
2. We can fit n into existing cycles. So for permutations of $n-1$ elements with k cycles, there are $n-1$ position where we could put n, essentially after each of the existing elements independent of which cycle they are in.

Therefore, we obtain the following recursive formula:

$$S_{n,k} = S_{n-1,k-1} + (n-1) \cdot S_{n-1,k} \ .$$

As there is no permutation of n elements without any cycles and merely one with n cycles, we obtain the base cases:

$$S_{0,0} = 1 \qquad S_{n,0} = 0 \qquad S_{n,n} = 1 \ .$$

This yields another Stirling's triangle.

We are now equipped with logical reasoning, mathematical objects, and tools for counting them. We now proceed to the next step: how to analyze relations and processes. In a nutshell, *to the graphs!*

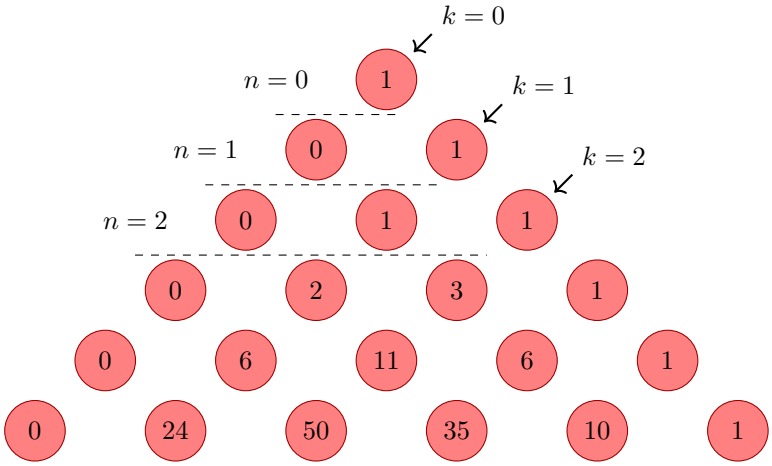


Figure 4.16: Stirling's triangle of the first kind.

4.7 Exercises

Exercise 4.1 (Binomial coefficients). Prove with a combinatorial argument that the following statements are true:

1. For all $n \geq 1$, $\binom{2n}{2} = 2\binom{n}{2} + n^2$.
2. For all $n \geq 1$, $\binom{2n}{n} = \sum_{k=0}^{n} \binom{n}{k}^2$.

Exercise 4.2 (Light bulbs).

1. Show that the sum of all numbers in the n-th row of the Pascal triangle is 2^n: $\sum_{k=0}^{n} \binom{n}{k} = 2^n$.
2. Show that, if we put, in the n-th row of the Pascal triangle, *alternating signs*, then the sum vanishes: $\sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0$.
   Conclude from this that when each one of n light bulbs fails with probability $1/2$, independently of the others, then the probability that an *even* number fails is $1/2$.

Exercise 4.3 (Inclusion/exclusion).

1. How many of the numbers from 1 to 300 are divisible by 4, 6, or 15? (Here, or refers to the *inclusive* OR)
2. In a village, there are three clubs. Only ten people of the village belong to no club. Club A has 20, club B 50, and C 40 members. If we pick any two clubs, there are always exactly five people in both. Finally, two people are in all clubs. How many people live in the village?

Exercise 4.4 (Strings). Determine the number of binary strings of length n in which the partial string 01 occurs *exactly twice*. An example (of length 10) is 1111**01**110.

Exercise 4.5 (Children – hard). A new family moved into the neighboring flat; your nosy flatmate already informed you that they have two kids. One day, you see the couple taking the elevator with their young daughter. What is the probability that the other child is a boy?

While making small-talk, they say that one of their children is 20 years old. That is definitely not the young girl. Does this new information change the probability that this second child is male?

Exercise 4.6 (The party).

1. You are at a party with n people. If everybody clinks glasses with everybody, how many times do you hear "ping!"?

2. Unfortunately, the party does not go that well, not everybody clinks with everybody. You are bored and think about parities, and you ask yourself whether the number of people who clinked with an odd number of others can be odd? What do you think?

3. After having found that answer, you ask yourself whether there exist two people at the party who clinked with exactly the same number of others. What do you think?

4. By the way, there were exactly 20 subjects people talked about at the party. Exactly 10 people spoke about each of these subjects. Every person has spoken about exactly four of the topics. How many people were at the party?

Exercise 4.7 (Power plugs). You have invited 5 of your friends for dinner. At some point in the evening, everyone has to charge their phone *at the same time*. There are 8 plugs in your house: 2 in the kitchen, 3 in the living room, and 1 each in the bathroom, the hall, and the bedroom.

1. Everyone (including you) would like to charge their phone in the living room, thus you decide to determine the three people who will. What is the probability that you will end up charging your phone in the living room?

2. In how many ways can you plug in the different phones in the house?

3. As you are the host, you decide not to charge your phone, and to plug every phone in a different room. How many different dispositions of phones can you come up with?

Exercise 4.8. How many triangles can one find in Figure 4.17?

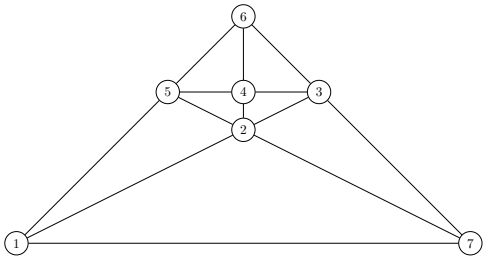


Figure 4.17: How many triangles do you see?

Chapter 5

Graph Theory

5.1 Motivation

Graphs are extremely useful models for many discrete problems, as they allow to focus the analysis on specific relations between objects.

Example 5.1 (Error-free communication over noisy channels). A channel is an abstract model of a device used to transmit some information (for instance, a wire, an optical fiber, or a radio signal). As shown in Figure 5.1, a channel can be depicted by a graph, where the input is connected with each of the possible outputs. Generally, channels are noisy, in that errors might occur while transmitting the signal. Therefore, a given input can potentially yield different outputs in different uses of the channel.[1]

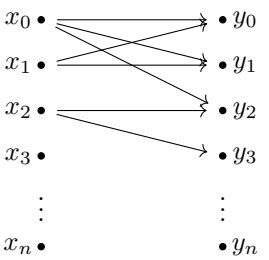


Figure 5.1: A graph for a noisy channel. An input $x \in \mathcal{X}$ can be mapped to more than one output $y \in \mathcal{Y}$.

[1] In a more complete model, one would assign a probability distribution over the outputs for each input. Typically the "right" output would occur with rather high probability. If the channel is noisy, there are other outputs that occur with probability larger than zero and cause the ambiguity for the receiver.

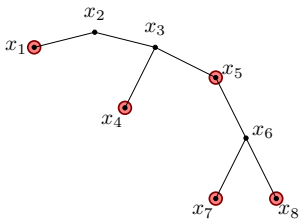


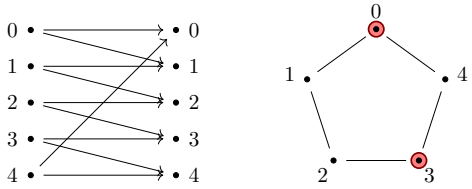
Figure 5.2: The ambiguity graph of a noisy channel. The red markers show an optimal coding.

Nodes representing the input in the graph might be connected to more than one element in the output alphabet $\mathcal{Y}$. As outputs can stem from different inputs, the receiver cannot tell which input the sender chose. Is there a way to transmit information without error through such a noisy channel? To answer the question, we first consider a different graph representation.

An *ambiguity graph* shows the conflicting inputs in a noisy channel. The nodes correspond to the inputs in $\mathcal{X}$. Two nodes are connected if the corresponding inputs might yield the same output.

In order to resolve the ambiguity of the inputs, the sender and the receiver agree on a *code*: The sender uses merely a subset of the input alphabet $\mathcal{X}$ without conflicts in the output. This subset has to be chosen such that none of the elements are neighbors in the ambiguity graph: This is called an *independent set*. The *optimal code* is an independent set of maximal size; to find one in a given general graph is a problem that is NP-complete. (In Figure 5.2, the codewords of the optimal code are highlighted in red.)

The noisy typewriter channel. We consider an explicit example of a noisy channel. Claude Shannon, in his seminal text from 1948, laying the basis for information theory, considered a noisy typewriter. For simplicity, we consider a reduced alphabet of 5 instead of 26 letters. Through some error the following letter is also hit every now and then; this error happens cyclically: When typing the *last* letter, sometimes the *first one* is hit. We obtain the following graphs:



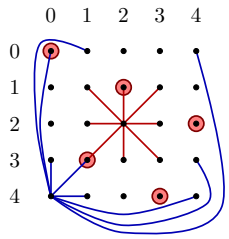


Figure 5.3: Ambiguity graph for the case of using the channel twice. Instead of showing all edges, the blue and the red lines give two examples of ambiguous pairs. The red circles indicate an optimal code for this new channel.

The corresponding optimal code has then two codewords and can, therefore, transmit *one* bit with each use of the channel: The *zero-error capacity* — the number of bits that can be sent unambiguously per use of the channel — is at least one. Is it equal to one? We are suspicious since one single bit can already be sent with an application of the same channel of smaller size 4.

The zero-error capacity is in fact the average number of bits set for *multiple* uses of the channel; can this help? Let us consider the case of using the channel *twice*. The new alphabet is then the Cartesian product $\mathcal{X}^2$ with 25 elements, as shown in Figure 5.3. We can now search for unambiguous codewords among these 25 ordered pairs. The optimal code indicated by the red circles allows then for 5 distinct codewords to be transmitted without error. Therefore the zero error capacity of the channel in bits is at least $\log_2 5/2$, which is larger than 1. It turns out that combining a larger number of channel uses does not help further: Lovasz showed that the zero-error capacity of this channel is equal to the lower bound obtained from pairs of channel uses.

5.2 Basic notions

Definition 5.1 (Graph)**.** A graph $G = (V, E)$ consists of

- a non-empty, finite vertex set V, i.e., $0 < |V| < \infty$. The elements of V are called vertices or nodes.
- an edge-set $E \subseteq V \times V$: E is a *relation on* V[2].

[2]This definition of graphs allows for loops, but not for more than two edges connecting the same pair of nodes. Graphs not having such a limitation are called *multigraphs* and appear in these notes only in reference to a historical example (cf. Section 5.5.1).

Example 5.2. The ambiguity graph in Figure 5.2 is formally defined by the set of vertices $V = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ and by the set of edges $E = \{\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}, \{x_3, x_5\}, \{x_5, x_6\}, \{x_6, x_7\}, \{x_6, x_8\}\}$.

Definition 5.2 (Undirected graph)**.** A graph G is undirected if

$$(u, v) \in E \quad \Leftrightarrow \quad (v, u) \in E \ .$$

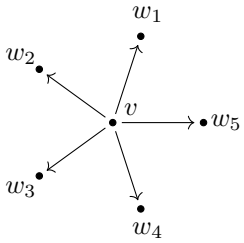
In this case, edges are sometimes taken to be unordered sets (instead of ordered pairs): $e = \{u, v\}$. When an undirected graph is drawn, a single line is normally drawn between u and v, instead of a pair of arrows.

Definition 5.3 (Simple graph)**.** A graph G is called *simple* if it does not contain loops:

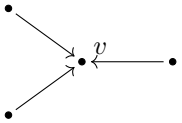
Definition 5.4 (Neighborhood of a vertex)**.** For a vertex $v \in V$ we call the set

$$\Gamma(v) := \{w \in V \mid (v, w) \in E\}$$

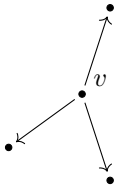
the *neighborhood* of v:



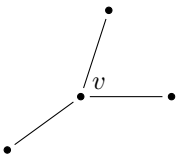
Definition 5.5. The number of edges towards a node v is called the *in-degree* $\deg^-(v)$:



The number of edges from a node v is called the *out-degree* $\deg^+(v)$:



For undirected paths the number of edges ending in a node v is called the *degree* $\deg(v)$:



In a directed graph, any ingoing edge of a node is an outgoing edge of another node. Further, any edge is an ingoing (respectively outgoing) edge for some node. Thus, we obtain for directed paths the following equality:

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E| \ .$$

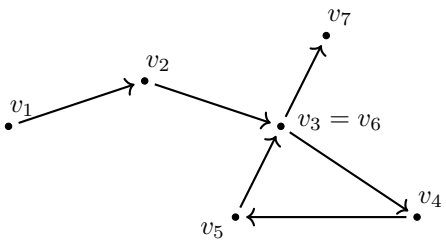
Similarly, it holds for undirected paths

$$\sum_{v \in V} \deg(v) = 2\,|E| \ .$$

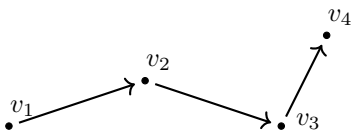
5.2.1 Basic notions for simple undirected graphs

Definition 5.6 (Way)**.** A *v_1-v_l-way* with length l is a sequence of vertices $w = (v_1, \ldots, v_l)$ with

$$(v_i, v_{i+1}) \in E \qquad \forall i \in \{1, \ldots, l-1\}$$



Definition 5.7 (Path)**.** A *v_1-v_l-path* is a v_1-v_l-way for which all vertices are distinct (i.e., a way without loops):



Definition 5.8 (Circuit or cycle)**.** A *circuit* or *cycle* is a closed path, namely, a sequence of pairwisely distinct vertices $c = (v_1, \ldots, v_l)$ satisfying

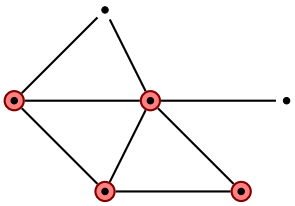
$$(v_i, v_{i+1}) \in E \ , \qquad \forall i = 1, \ldots, l-1 \ ,$$
$$(v_l, v_1) \in E \ .$$

Definition 5.9 (Subgraph)**.** The pair $H = (V', E')$ is a *subgraph* of $G = (V, E)$ if

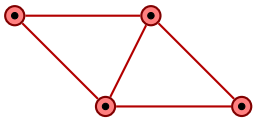
$$V' \subseteq V \qquad E' \subseteq E \qquad E' \subseteq V' \times V' \ .$$

Given a vertex set $\tilde{V} \subseteq V$, the *subgraph of G induced by $\tilde{V}$*, denoted by $G[\tilde{V}]$, is the subgraph of V with vertex set $\tilde{V}$ and all possible edges which are also in G:

$$\forall u, v \in \tilde{V} : \quad (u, v) \in E \quad \Rightarrow \quad (u, v) \in \tilde{E} \ .$$



(a) Graph G with highlighted set $\tilde{V}$

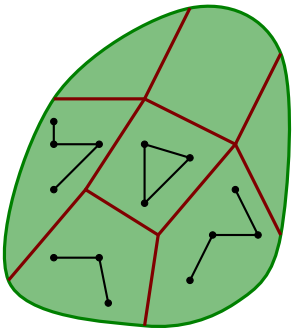


(b) Subgraph $H = G[\tilde{V}]$

Definition 5.10 (Connected components)**.** Let $(V_i)_{i \in I}$ be a partition of the vertex set of a graph G such that all elements within each partition V_i are connected by a path. That is

$$\exists (u, v)\text{-path} \quad \Leftrightarrow \quad \exists i \in I : \ u, v \in V_i \ .$$

Then the induced subgraphs $G[V_i]$ are called the *connected components* of G:



Note that the existence of a path between vertices leads to an *equivalence relation* on the set of vertices. Correspondingly, it can also be seen as a partition. The graphs induced by the parts are the connected components.

Definition 5.11 (Bridge)**.** An edge $e \in E$ is called a *bridge* if the graph $G' := (V, E \setminus \{e\})$ has one more connected component than G.

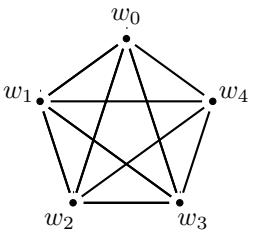
Theorem 5.1. *A graph $G = (V, E)$ has at least $|V| - |E|$ connected components.*

Proof. Observe first that the graph $G = (V, \emptyset)$ has $|V|$ connected components, each containing just one vertex. Any edge that is inserted into the graph reduces the number of connected components by at most one. If it connects two previously separate connected components, it reduces the number by one. If the edge connects two vertices within a connected component the number of connected components remains the same. This implies the statement. □

Corollary 5.2. *If a graph G is connected, i.e., all components are connected with one another by paths and there is just one connected component, then the number of edges and vertices is related by*

$$|V| - |E| \leq 1$$

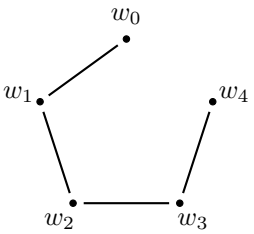
The following is an example of a connected graph:



The number of vertices is $|V| = 5$, whereas the number of edges is $|E| = 10$. Therefore, the difference is well below the bound from the corollary above, as

$$|V| - |E| = -5 < 1.$$

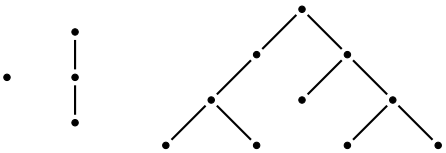
Indeed, we could remove all internal edges and one of the outer ones and still obtain a connected graph:



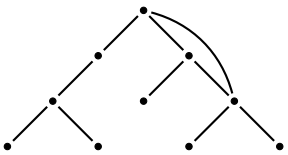
In this minimally connected graph, the difference of the number of edges and the number of vertices reaches the bound from the corollary above. All edges are bridges. These minimally connected graphs are called *trees*.

5.3 Trees

A *tree* is a minimally connected graph. A graph containing trees as its connected components is called *a forest*. The following graph is a forest with 3 trees:



The following graph is *not* a tree, as it contains a cycle:



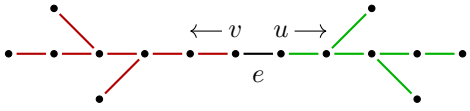
Definition 5.12 (Forest)**.** An undirected simple graph without cycles is a *forest*.

Definition 5.13 (Tree)**.** A *connected* forest is a *tree*.

Definition 5.14 (Leaf)**.** A node $v \in V$ with $\deg(v) = 1$ is a *leaf*.

Theorem 5.3. *Every tree with at least two vertices has at least two leaves.*

Proof. As there are at least two vertices, and the tree is connected, there is at least one edge $e \in E$, connecting two vertices $u, v \in V$. From both these vertices, we can walk in opposite directions away from one another. As long as the $\deg(v_i) > 1$ of the nodes along the way, one can choose another edge for the next step:



As there are no cycles and merely finitely many vertices, one eventually reaches a leaf. □

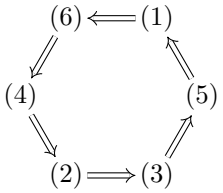
The following are different ways of implying that a graph is a tree.

Theorem 5.4. *Given an undirected simple graph $G = (V, E)$ the following statements are equivalent:*

1. *G is a tree, i.e., a connected graph without cycles.*
2. *G is connected and $|V| = |E| + 1$.*
3. *G has no cycles and $|V| = |E| + 1$.*
4. *G is connected and every edge is a bridge.*
5. *G has no cycles; if an additional edge is added to the graph, it obtains a cycle.*
6. *For all vertices $v, u \in V$, there* exists a unique *$u - v$-path.*

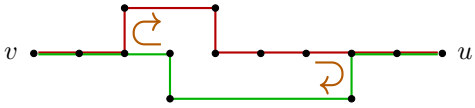
In total, $2 \cdot \binom{6}{2} = 30$ implications are claimed. By transitivity of implication, it is, however, sufficient to show a loop of implications. Any statement then follows from any other by just following the implications in the loop.

Proof. We will show the implications in the following cycle.



(1) ⇒ (6). By the definition of connectedness, there exists a $u - v$-path for all pairs of vertices $u, v \in V$. It remains to show that this path is unique. We show this by contradiction. More precisely we show that if the path is not unique, we can construct a cycle in the graph.

Let us assume that there are 2 different paths connecting the vertices u and v:



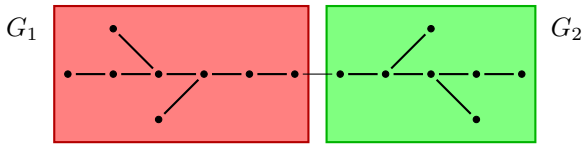
Where the two paths split, we can start constructing a loop taking one path until we reach the vertex where they rejoin and return on the other path.

(6) ⇒ (4). As there exists a path for any pair $u, v \in V$, we obtain immediately that G is connected. It remains to show that every edge is a bridge. Again, we show this by contradiction. Assume there exists an edge $e \in E$ that is not a bridge. Then, we can remove e and G is still connected, i.e., there is a path connecting the vertices adjacent to e. In other words, with e there are two paths connecting these two vertices. This contradicts the uniqueness of the path.

(4) ⇒ (2). As connectedness is already given, it remains to show that $|V| = |E| + 1$ if every edge is a bridge. We show this by induction over the number of edges.

Base case If $|E| = 1$ then there are two vertices and the equation is satisfied.

Induction step As all edges are bridges, we can simply split the graph into two connected components by removing any of the edges:



Then, we are left with two connected graphs. By induction hypothesis (and considering that all remaining edges are still bridges), we have

$$|E_1| + 1 = |V_1| \qquad |E_2| + 1 = |V_2|$$

Therefore, we obtain the equality we were looking for by adding these two equations:

$$\underbrace{|E_1| + |E_2|}_{=|E|-1} + 2 = |V_1| + |V_2| = |V|$$

(2) ⇒ (3). We show by contradiction that a connected graph G satisfying the equality $|V| = |E| + 1$ has no cycles. Let us assume that G has a cycle. Then, there exists an edge that is not a bridge. We can, therefore, remove that edge to obtain another connected graph with one edge less, $G' = (V, E \setminus \{e\})$. For this graph it holds then

$$1 \geq |V| - (|E| - 1) = |V| - |E| + 1 = 2 \ ,$$

a contradiction.

(3) ⇒ (5). We have to show that adding an edge e' to the graph creates a cycle. Thus we have from (3) that $|V| = |E| + 1 = |E'|$ for $E' := E \cup \{e'\}$. We show by induction over the number of vertices that whenever $|V| \leq |E'|$, then there is a cycle.

Base case The first interesting case is $|V| = 3$. If the number of edges is also 3, we obtain the graph

which obviously contains a cycle.

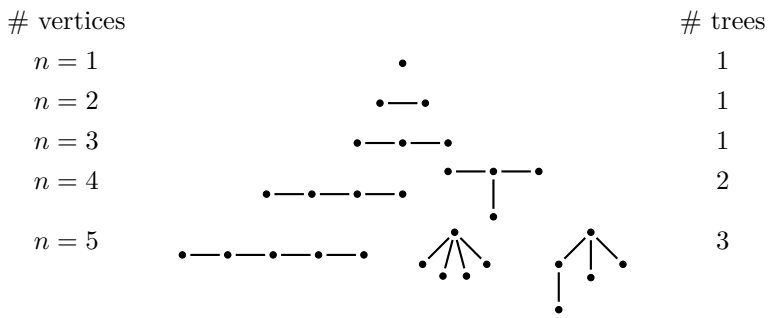
Induction step We distinguish two cases. If there exists a leaf in the graph, we merely remove that vertex and the corresponding edge. This reduces the number of vertices and the number of edges by one. If $|V| \leq |E|$ then also $|V| - 1 \leq |E| - 1$ and the graph contains a cycle by induction hypothesis. So, also the graph containing the leaf has a cycle.
If there is no leaf, then all vertices have $\deg(v) \geq 2$. Then we can simply construct a cycle by going from one vertex to the next. As there is no leaf, there is no dead-end. And as the number of vertices is finite, we have to end-up at the initial vertex sooner or later.

(5) ⇒ (1). The graph G does not contain any cycles. So it remains to show that G is connected. Again we use an indirect proof. Let us assume that G has two connected components. Then, we could add a vertex e' to G without creating cycles by linking the two connected components. This yields a contradiction with (5), as this insertion of an edge did not create a cycle. □

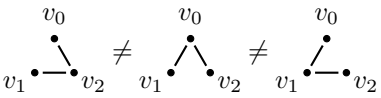
5.3.1 Counting trees: Cayley's theorem

After having derived different characterizations of a tree, we now turn to the following question: How many trees are there with n vertices?

The number of different graphs for $n = 1, \ldots, 5$ is given in the last column.



Unfortunately, the general question, i.e., the same question for a general number n of vertices, is hard to answer. We replace it by an easier one: How many *marked* trees exist with n vertices? Here, the nodes are numbered from 1 to n, and a different numbering leads a priori to a different tree. In this view, the following trees are *not* equal:



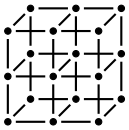
while there corresponding unmarked trees are equal:

The question in our focus now can be rephrased, asking for *the number of different "spanning trees" of the complete graph (clique) with n vertices, K_n.*

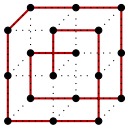
Definition 5.15 (Spanning Tree)**.** Given a connected (undirected, simple) graph $G = (V, E)$, a graph $H = (V, E')$ with the same vertex set is called a *spanning tree* of G if

- H is a tree;
- $E' \subseteq E$.

Example 5.3. The connected graph



has the spanning tree

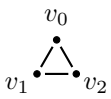


Generally, there exist many different spanning trees for a given graph. It is an interesting *algorithmic* problem to find the one that optimizes, for instance, the total weight, given that each vertex has such a weight; the most famous algorithms are greedy and due to *Kruskal* and *Prim*.

Let us now return to the question: How many spanning trees are there for complete graphs K_n with n vertices? Let us first consider some cases with small n.

Example 5.4. In a completely connected graph, any vertex is connected with any other. We will now consider the spanning trees for $n = 1, 2, 3, 4$:

- K_1: There is merely one vertex and thus only one spanning tree.
- K_2: There is merely one connected graph with two nodes. The single spanning tree is just the graph itself.
- K_3: The completely connected graph K_3

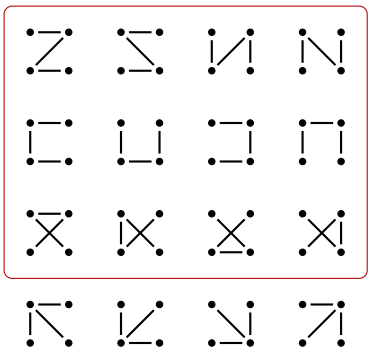
  has three spanning trees

- K_4: The completely connected graph K_4

  has 16 spanning trees.

  It is natural to group them into to sets: The "snake" type and the "star" type. The first 12 of these, the "snakes," are actually the same if we considered them as unmarked graphs: We call them *isomorphic*. By rearranging the vertices of a given tree among these 12, we can obtain

any other of the 12. An example of such a rearrangement is this

It preserves neighbor-relations. Any pair in the edge set $(u, v) \in E$ is also in the edge set E' of the graph obtained after the re-arrangement. Functions with this property are called *isomorphisms* and are formally defined below.

Definition 5.16 (Isomorphism)**.** Two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic*, $G \cong G'$, if there exists a bijective function $f : V \to V'$ such that

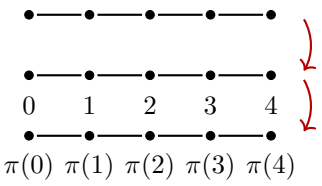
$$\forall u, v \in V : \quad (u, v) \in E \quad \Leftrightarrow \quad (f(u), f(v)) \in E' \ .$$

The bijective function f that preserves the edge relations is called an *isomorphism*.

As the number spanning trees of completely connected graphs grows pretty quickly, it is rather tedious to write down all of them for K_5. We can now take a different approach employing isomorphism. First, we consider the different unmarked trees with n vertices. Then, we derive the number of marked trees isomorphic to it, i.e., the number of possibilities to put the numbers 1 to 5 for obtaining (isomorphic but) different graphs.

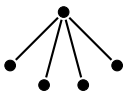
Example 5.5. We consider the unmarked trees with 5 vertices and how many marked graphs are isomorphic to each of them.

- The "snake" graph



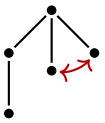
corresponds for instance to the marked graph with the vertices in increasing order. For any permutation of the 5 vertices, we obtain another isomorphic graph — except when we simply reverse the order, the graph stays the same. Therefore, there are $5!/2 = 60$ different marked graphs corresponding to this unmarked graph.

- The "star" graph



corresponds to 5 different marked graphs. Swapping the leaves does not change the graph but merely changes the center of the star. So, for each of the 5 vertices being the center, we obtain a different graph.

- The third type is the graph



Again, the permutations of the vertices give the isometries. But swapping the two leaves as indicated by the red arrow does not change the graph. We, therefore, have a similar symmetry as above and have to divide again the number of permutations 5! by 2 to obtain 60 graphs.

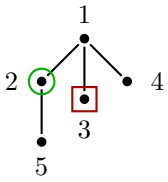
So, in total we are left with $125 = 5^3$ different spanning trees (or different marked trees with 5 vertices).

Considering the examples above we might guess that in general the number of spanning trees of completely connected graphs K_n is n^{n-2}. Indeed, this is true.

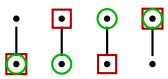
Theorem 5.5 (Cayley)**.** *The number of different marked trees with n nodes (i.e., the number of spanning trees of the completely connected graph K_n) is*

$$n^{n-2} .$$

Proof. We count marked trees *with two additional marks*, a green circle and a red square, both put to an arbitrary node, possibly also the same for both:



There are n^2 different ways to place these marks. If, for instance, $n = 2$, then we have the following 4 possible placements:



We show that there exist n^n different marked trees with n vertices and two additional marks. This then implies what we want, that there are

$$\frac{n^n}{n^2} = n^{n-2}$$

marked trees without the additional marks.

For counting the number of trees with marks, we construct a bijection from these trees to the functions from a set of n elements to itself:

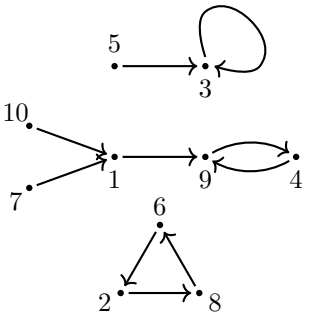
$$\{f : \{1, \ldots, n\} \to \{1, \ldots, n\}\} \ .$$

There are n^n such functions.

We illustrate the bijection we have in mind with an example:

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 9 & 8 & 3 & 9 & 3 & 2 & 1 & 6 & 4 & 1 \end{pmatrix}$$

We represent this function by a directed graph:



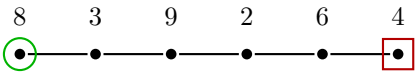
In every component, there must exist a cycle, as there is only a finite number of nodes, and there is no "end point" if one applies the map over and over again (i.e., follows the arrows). Restricting the function to the vertices that are contained in cycles yields a *permutation*, i.e., a bijective map from that set to itself. The set of vertices contained in some cycle is

$$M = \{2, 3, 4, 6, 8, 9\}$$

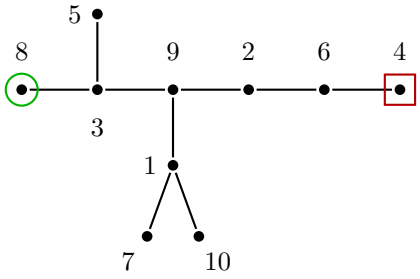
and the corresponding permutation is the restriction

$$f|_M = \begin{pmatrix} 2 & 3 & 4 & 6 & 8 & 9 \\ 8 & 3 & 9 & 2 & 6 & 4 \end{pmatrix} \ .$$

We encode this permutation in a "snake" tree:

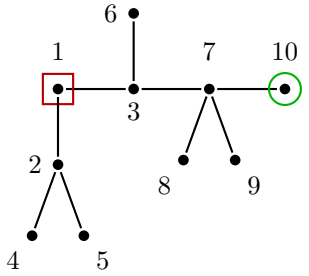


This is usually compared to an animal with the green circle being the "head," the red square the "tail," and the encoding of the permutation the "spinal cord." We add the remaining elements to the tree as follows:



The edges are to be read as arrows towards the spine. This procedure yields a tree with head and tail for any function $f : \{1, \dots, n\} \to \{1, \dots, n\}$. It remains to show that we have an inverse map from trees to functions.

Inverse direction. Let us construct a function for the following tree with head and tail.



From the spine, we directly obtain the permutation, as the first row is given by convention by the numbers in increasing order:

$$f|_M = \begin{pmatrix} 1 & 3 & 7 & 10 \\ 10 & 7 & 3 & 1 \end{pmatrix}$$

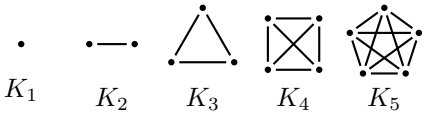
Reading the other edges as arrows towards the spine, we obtain the entire map:

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 10 & 1 & 7 & 2 & 2 & 3 & 3 & 7 & 7 & 1 \end{pmatrix}$$

This procedure works similarly for any tree. □

5.4 Some special graphs

Complete graphs or *cliques* are simple, undirected graphs with an edge between any pair of vertices:

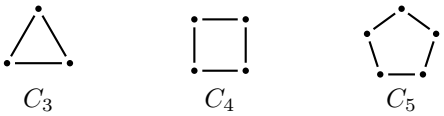


For a complete graph with n vertices, denoted K_n, there are then

$$|E| = \frac{n(n-1)}{2} = \binom{n}{2}$$

edges.

Cycles (also circles or circuits) are graphs with all nodes contained in one single cycle without additional edges. The smallest cycle is the triangle:



The cycle C_n with n vertices has n edges.

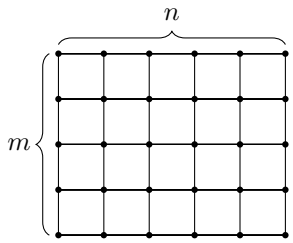
Mesh graphs are graphs with a node set:

$$V = \{(i,j) | 1 \leq i \leq m, 1 \leq j \leq n\} \ .$$

The edge set contains merely the closest neighbors of each vertex

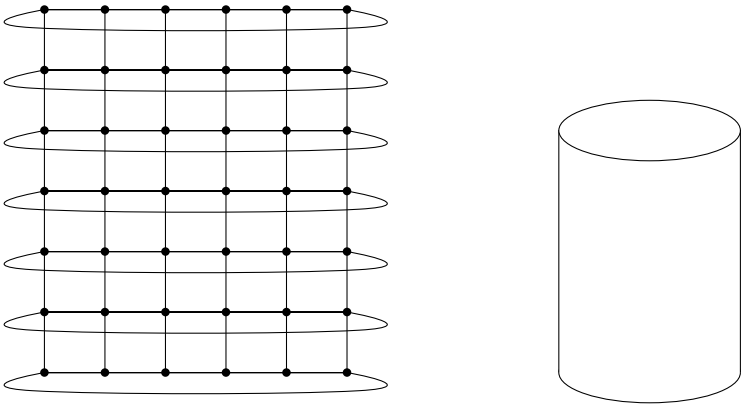
$$((i_1,j_1),(i_2,j_2)) \in E \quad \Leftrightarrow \quad |i_1 - i_2| + |j_1 - j_2| = 1$$

and no diagonals:

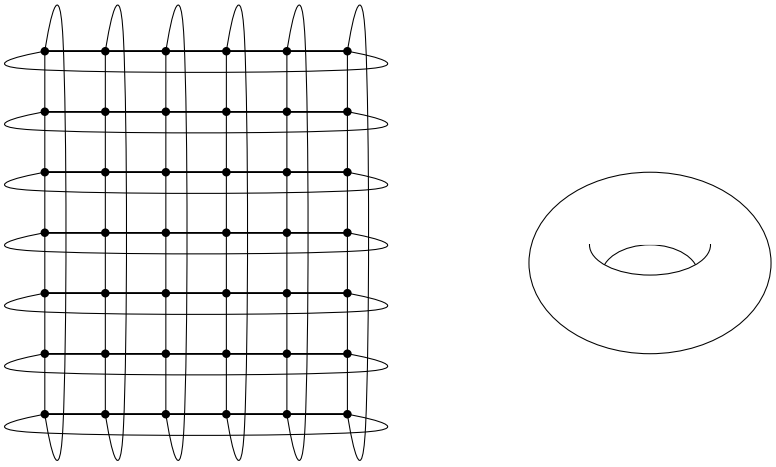


Sometimes, the mesh is interpreted in a cyclic way. Depending on how we connect the boundaries, we obtain graphs that resemble different topologies.

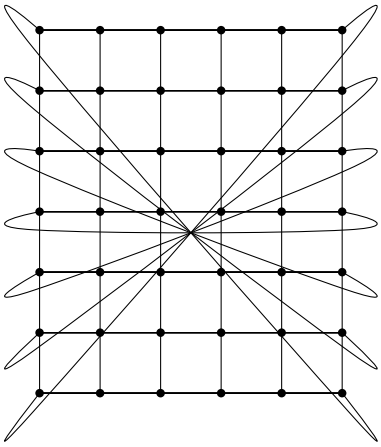
For instance, the nodes on the very right can be connected to the ones on the very left as follows:

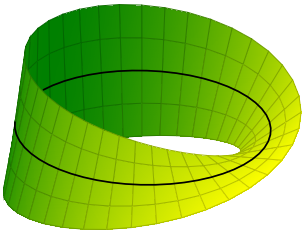


This graph has the shape of a cylinder. One might not only connect the nodes horizontally but also vertically and obtain a torus shaped graph:

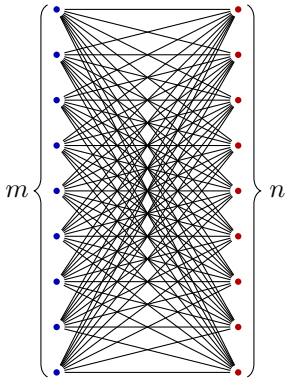


Another variant would be to invert the order to obtain a Moebius strip.





Complete bipartite graphs. The maximal elements, in terms of connectivity, among the set of bi-partite — two-colorable, such that no vertices of the same color are connected — graphs are the *complete bipartite graphs* $K_{m,n}$: The graph has $m+n$ vertices, m in one color and n in the other, and all nodes of different colors are connected. We have $|E| = mn$:



Hypercubes. The vertices of the d-dimensional hypercube are the d bit strings, i.e.,

$$V = \{0,1\}^d = \{d\text{-bit strings}\} \ .$$

There are 2^d vertices. The edges are then given by the pairs with *Hamming distance* 1.

Definition 5.17 (Hamming distance)**.** The *Hamming distance* between two bit strings of same length, $d_H(x, y)$, with $x, y \in \{0,1\}^d$, is the number of bits in which x and y differ.

For instance, $d_H(0110, 1010) = 2$ and

$$d_H(0101, 1010) = d_H(0000, 1111) = 4.$$

The edges of the d-dimensional hypercube are

$$(u, v) \in E \quad :\Leftrightarrow \quad d_H(u, v) = 1 \ .$$

Every vertex has d neighbors, i.e., $\forall v \in V : \ \deg(v) = d$. The total number of edges is

$$|E| = \frac{d \cdot 2^d}{2} = d \cdot 2^{d-1} \ .$$

as each of the 2^d degrees is d.

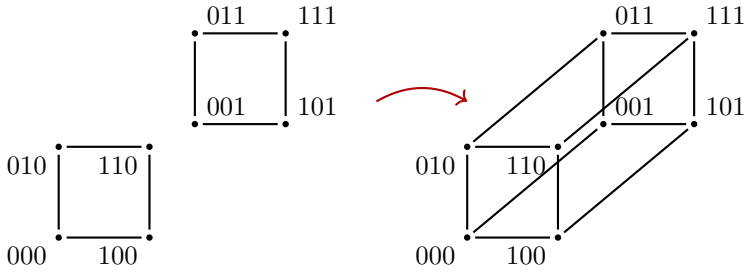
We draw the graph as follows: For obtaining the graph Q_{d+1} we draw twice the graph Q_d. To one of the two we add a 0 after all bit strings labelling the vertices, to the second respectively a 1. Finally, we add edges to connect the corresponding vertices of the two copies of Q_d.

We start from Q_0, containing merely the empty word ϵ

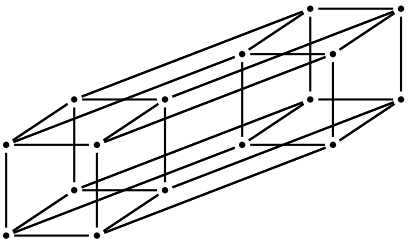
Then the hypercube Q_1 is constructed following the procedure above:

Repeating the same process yields Q_d for larger d. For Q_2 we get

Q_3 is the three dimensional cube.



Similarly, we obtain the 4-dimensional hypercube.

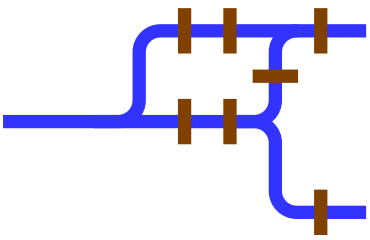


This is how we can imagine four dimensions. Or 17, for that matter.

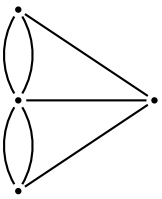
5.5 Euler tours and Hamilton cycles

5.5.1 Bridges of Königsberg

In 1736, Leonhard Euler considered the question of whether there is a tour crossing every of the seven bridges in Königsberg



exactly once. This question is today often said to mark the beginning of graph theory. The situation is reflected in the following (multi-)graph:



We are looking for a closed way that passes each edge exactly once — a so-called *Euler tour*.

Definition 5.18 (Euler Tour)**.** An Euler tour is a closed sequence of edges of a graph $G = (V, E)$ that contains each edge of the graph exactly once.

As the degree of any of the vertices in the graph above is 3 or 5, thus odd, there does not exist an Euler tour.

Theorem 5.6 (Euler)**.** *A connected graph has an Euler tour if and only if all degrees are even. That is*

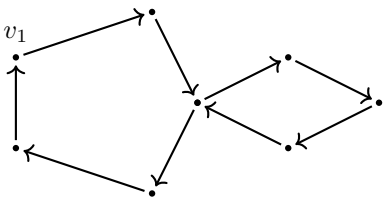
$$G \text{ has an Euler tour} \quad \Leftrightarrow \quad \forall v \in V: \quad \deg(v) \text{ is even.}$$

Proof. We first show that the condition

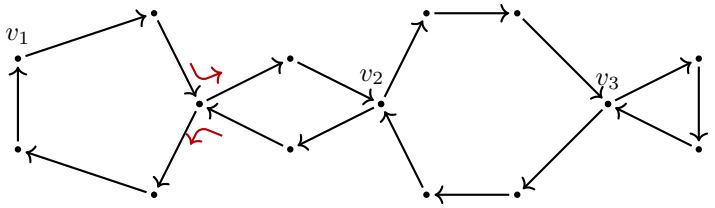
$$\forall v \in V: \quad \deg(v) \text{ is even}$$

is necessary for the existence of an Euler tour. This follows from the observation that any vertex is reached and left the same number of times. Whenever we reach or leave the edge, we have to use an other edge to form an Euler tour. Thus, the number of available edges has to be even if we finally have to have passed all edges. Note that the same holds for the starting vertex.

It remains to show that the condition is sufficient. We construct an Euler tour for a graph satisfying the condition above. First, we choose an initial vertex $v_1 \in V$. Following any yet unused edge one now proceeds to other vertices. As there is an even number of edges ending at each of the vertices, and as edges are always used (i.e., removed) in pairs (arrive + leave), we always finds such an unused edge for continuing, except at v_1. As the number of edges and the number of vertices are finite, we *must* eventually end up in v_1, and thus obtain a first closed way:

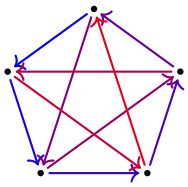


This way may fail to contain all edges already. Then, as the graph is connected, there must be a vertex v_2 in the already-found way with still (at least two) unused edges. We use this vertex v_2 as the starting point of an additional way, using the same procedure of following the unused edges. (Note that it is still true that all degrees inn the graph are even since, again, we removed edges always in pairs with respect to any vertex: arrive and leave.) The red arrows show how to interpret this as just one cycle:



Repeating this iteratively until we used all edges yields the desired Euler tour. □

Example 5.6. All vertices in the complete graph K_5 have degree 4. Therefore, K_5 has an Euler tour. An example is this: We start from the vertex at the top, follow the outer circle, and then the inner edges:



Example 5.7.

- The complete graph K_n has an Euler tour if and only if n is odd:

$$\deg(v) = n - 1 \quad \forall v \in V \ .$$

- The hypercube Q_d has an Euler tour if and only if d is even:

$$\deg(v) = d \quad \forall v \in V \ .$$

The given proof above uses a simple *greedy* algorithm[3], that finds an Euler tour in *linear time* in the number of edges $|E|$. Finding on Euler tour is, hence, among the computationally easiest problems (you simply have to go through the entire graph, and you have it). If we modify the problem a little — replacing "edge" by "vertex," we end up with a hard problem, the *Hamilton cycles*.

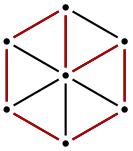
Definition 5.19 (Hamilton Cycles)**.** A cycle that visits every vertex exactly once is called a *Hamilton cycle*. A graph containing such a cycle is called *Hamiltonian*.

Example 5.8. The following graphs are Hamiltonian.

- The cycles C_k are Hamiltonian for all $k \geq 3$.
- The complete graphs K_n contain the cycles C_n for all $n \geq 3$, and therefore a Hamilton cycle. Thus these are Hamiltonian:

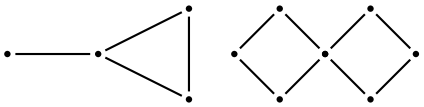
[3] At each vertex we choose any unused edge, independent of any previous choice. The choice does not depend on any global properties.

- The wheel graph:



Example 5.9. These graphs are *not* Hamiltonian.

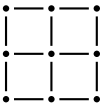
- Any tree is not Hamiltonian as it does not contain any cycles at all.
- The following graphs do neither contain any Hamilton cycles:



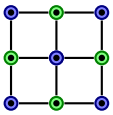
We would now like to address the question whether *mesh graphs* are Hamiltonian. The graph $M_{1,2}$

is a tree and thus not Hamiltonian. The two following two mesh graphs, $M_{2,2}$ and $M_{2,3}$,

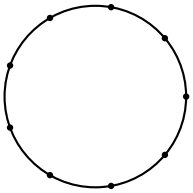
are Hamiltonian as the outer cycles connect all nodes. The mesh graph $M_{3,3}$



is not Hamiltonian. To see this, note first that the graph is two-colorable, i.e., bipartite:



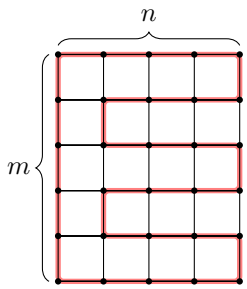
A bipartite graph can be Hamiltonian only if it contains the same number of nodes of each of the two colors: Along the (closed) Hamiltonian cycle, the colors must switch in every step:



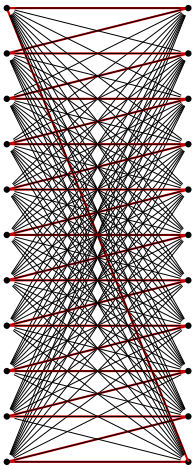
In consequence, we have that

$$M_{m,n} \text{ is Hamiltonian} \quad \Leftrightarrow \quad m \cdot n \text{ is even}$$

This can be seen as follows. From the argument above, we directly obtain that $m \cdot n$ being even is a *necessary* condition: This proves "⇒". It remains to show that the condition is also *sufficient*: "Whenever the product is even, there exists a Hamilton cycle." Now, if the product is even, then at least one of the two, m and n, is even. Without loss of generality we can assume that it is m. We construct a Hamilton cycle as follows:

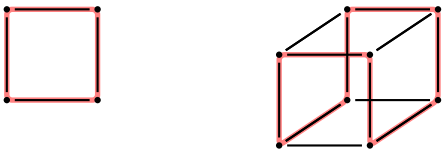


For the complete bipartite complete graphs $K_{m,n}$, the condition $m = n$ is necessary and sufficient for the graph to be Hamiltonian:

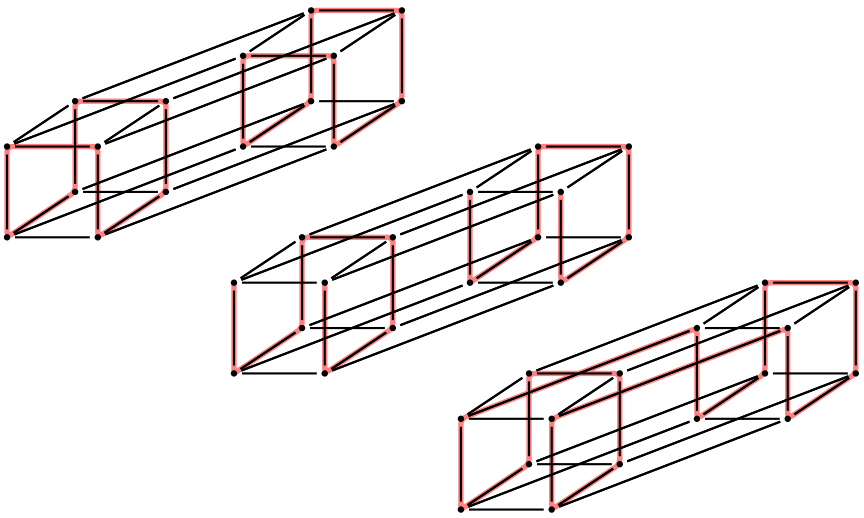


What about the hypercubes? Note first that Q_0 and Q_1, are not Hamiltonian: They are trees, and thus cycleless. The square Q_2 and the cube Q_3 are

Hamiltonian:



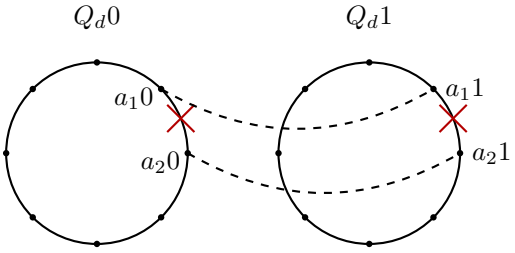
Using the Hamilton cycle on Q_3, we can now construct a Hamilton cycle on Q_4. We remove two corresponding edges and link the two cycles at the corresponding vertices:



This procedure can be applied iteratively to obtain a Hamilton cycle for any Q_d, as soon as $d \geq 2$, as the following inductive proof illustrates:

Base case. As we have seen above, the Q_2 is Hamiltonian.

Induction step. By induction hypothesis, there exists a Hamilton cycle for Q_d. We use it to construct a Hamilton cycle for Q_{d+1}:



The process removes two corresponding edges, $(a_1 0, a_2 0)$ and $(a_1 1, a_2 1)$, and then adds $(a_1 0, a_1 1)$ and $(a_2 0, a_2 1)$.

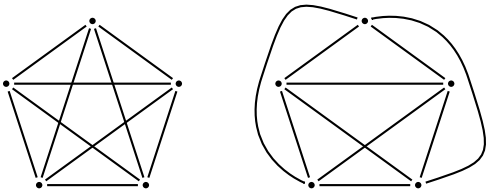
The problem whether a general graph is Hamiltonian is believed to be hard: It is NP-complete. We have seen the two extrema of computational hardness (linear time vs. NP-complete) in two very similar-sounding problems: *Euler vs. Hamilton.*

5.6 Planar graphs

Definition 5.20 (Planar graphs)**.** A graph $G = (V, E)$ is planar if it can be drawn in the plane such that no edges cross.

Example 5.10. The complete graph K_4 is planar:

The complete graph K_5 is not planar: One can shift two of the inner edges outside, but there remains a crossing. Shifting further edges outside introduces crossings outside:



At the beginning of the course we said that the bipartite graph $K_{3,3}$ is not planar (without proof):

To actually that the latter two graphs are not planar, we derive properties all planar graphs share. If a graph does not have one of these properties, it is not planar.

Theorem 5.7 (Euler's polyhedron formula)**.** *Let $G = (V, E)$ be a planar connected graph which divides the plane into f regions (including the region outside the graph). Then, the number of regions, of vertices and of edges satisfy the following equation:*

$$|V| + f - |E| = 2. \tag{5.1}$$

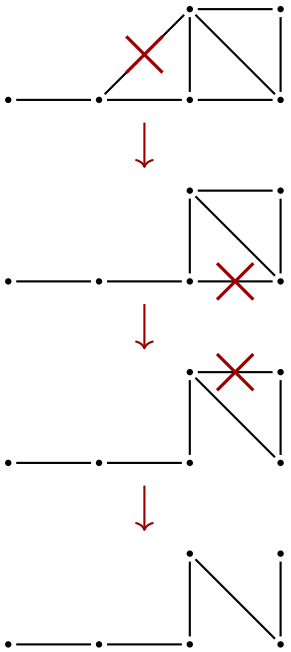
Note that the inverse implication does not hold. A graph satisfying equation (5.1) is not necessarily planar.

In the Introduction to this course, we saw a proof of this statement using spanning and dual trees. We will now consider a different, inductive one.

Proof. Note first of all that equation (5.1) holds for trees. As there are no cycles, there is merely one region, i.e., $f = 1$. Furthermore, the number of edges is one less than the number of vertices, $|E| = |V| - 1$. Thus one obtains

$$|V| + f - |E| = |V| + 1 - (|V| - 1) = 2 \; .$$

We can now reduce any other connected planar graph to a tree by removing edges in cycles, as shown for the following graph:



Removing an edge of a cycle reduces the number of regions and the number of edges by one, while the number of vertices remains the same:

$$f \to f_1 := f - 1 \qquad |E| \to |E_1| := |E| - 1 \qquad |V| \to |V_1| = |V| \, .$$

Thus, the value of the left-hand side for the new graph $G_1 = (V_1, E_1)$ with f_1 regions is the same as before:

$$|V_1| + f_1 - |E_1| = |V| + f - |E| \, .$$

The same holds for any other graph $G_i = (V_i, E_i)$ resulting from removing further edges to break apart cycles. Finally we end up with a tree that satisfies Euler's formula as seen above. Thus, Euler's formula also holds for the initial general planar graph. □

Unfortunately, Euler's formula is of no use to prove that K_5 or $K_{3,3}$ are not planar: The notion of a *region* is not defined if there are crossings, and f is undefined. We find bounds on the number of regions f, in terms of the number of *edges*. In a simple graph, a region is bounded by three edges:

On the other hand, each edge bounds at most two regions. (This latter point is why we argue with respect to the edges, not the vertices: It is also true that every region is surrounded by three nodes, but a node can be a limit to arbitrarily many regions; for instance, 100 edges can meet in a vertex.) We obtain

$$3 \cdot f \leq 2\,|E| \quad \Rightarrow \quad f \leq \frac{2}{3}\,|E| \ .$$

Inserting this into Euler's formula yields

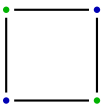
$$|V| - |E| + \frac{2}{3}|E| \leq 2 \ ,$$

and the following bound on the connectivity of planar graphs.

Corollary 5.8. *Any planar graph with $|V| \geq 3$ satisfies the following inequality:*

$$|E| \leq 3\,|V| - 6 \ . \tag{5.2}$$

The complete graph K_5 violates this equation, as $|V| = 5$ and $|E| = 10$. The complete bipartite graph $K_{3,3}$ with $|E| = 9$ and $|V| = 6$, however, does *not* violate the equation (5.2). We fix this issue by improving the bound on the number of regions for bipartite graphs. A region in a bipartite graph is bounded by at least 4 edges:



We conclude, as above,

$$4f \leq 2\,|E| \quad \Rightarrow \quad f \leq \frac{1}{2}\,|E| \ .$$

Thus,

$$|V| - |E| + \frac{1}{2}|E| \leq 2 \ .$$

Corollary 5.9. *A bipartite planar graph $G = (V, E)$ satisfies*

$$|E| \leq 2\,|V| - 4\ . \tag{5.3}$$

The bipartite planar graph $K_{3,3}$ with $|E| = 9$ and $|V| = 6$ violates this inequality.

Average degree. The inequalities (5.2) and (5.3) imply for the average degree:

$$\overline{\deg(v)} := \frac{1}{|V|}\sum_{v\in V}\deg(v) = \frac{2\,|E|}{|V|}\ .$$

For all planar graphs, we have

$$|E| \leq 3\,|V| - 6 < 3\,|V| \quad \Rightarrow \quad \overline{\deg(v)} < 6\ .$$

For *bipartite* graphs,

$$|E| \leq 2\,|V| - 4 < 3\,|V| \quad \Rightarrow \quad \overline{\deg(v)} < 4\ .$$

Starting from the other end, let us ask what graphs are *not* planar:

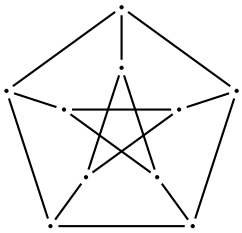
- K_5, $K_{3,3}$.
- A graph that contains the K_5 or the $K_{3,3}$ as a subgraph.
- *Subdivisions* of such graphs. For a given graph $G = (V, E)$ we add a vertex v' to the vertex set V and replace one of the edges $(u, v) \in E$ by the two edges (u, v') and (v', v). The new graph is then called a *subdivision* of G.

The surprising fact was proven by *Kuratowsky*: This list is complete. In a sense, this means that every non-planar graph "contains" in one way or another our two specimens K_5 and $K_{3,3}$.

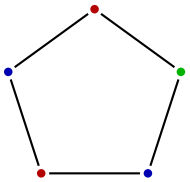
5.7 Graph colorings

Example 5.11. Imagine the following problem: An airline wants to find the minimal number of planes to operate some given flights. Each flight is represented by a node. Two nodes are connected if the flights cannot be operated by the same plane. Then the minimal number of colors required to color the vertices, such that any two neighboring vertices have a different color, yields

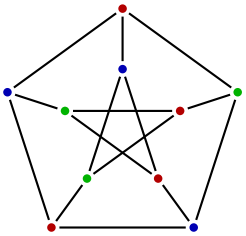
the number of planes required. As an example, let us try finding a colorings of the so-called *Petersen graph*:



Two colors do not suffice, as can be seen from the outer cycle. It contains an odd number of vertices and thus requires a coloring with three colors:



We can extend this coloring to the entire Petersen graph:



Therefore, in this case of ten flights, we would need at least three planes.

Definition 5.21 (Coloring)**.** A k-coloring of a graph $G = (V, E)$ is a function

$$c : V \to \{1, \ldots, k\},$$

such that any neighbours are assigned a different value, i.e.,

$$(u, v) \in E \quad \Rightarrow \quad c(u) \neq c(v).$$

Definition 5.22 (Chromatic number)**.** The chromatic number $\chi(G)$ of a graph G is the minimal k such that a k-coloring of G exists.

Example 5.12. We consider the chromatic number of some special graphs:

- Planar graphs G_p have a chromatic number less than or equal to 6 as we have shown in the Introduction 1.4:

$$\chi(G_p) \leq 6 \ .$$

  It can be shown that the chromatic number is actually smaller than or equal to 4.

- Complete graphs K_n have $\chi(K_n) = n$ as every vertex needs its own color.

- The complete bipartite graphs have a chromatic number $\chi(K_{m,n}) = 2$ by definition.

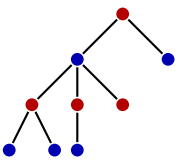
- Mesh graphs are bipartite: $\chi(M_{m,n}) = 2$ (except for $m = n = 1$).

- Hypercubes are bipartite: $\chi(Q_d) = 2$ for $d \geq q$. This follows from an inductive argument: Obviously, Q_1 is two-colorable. If Q_d is two-colorable, then we can color the second Q_d with flipped colors before connecting the two to construct Q_{d+1}.

- For the cycles, it depends on the parity of their length:

$$\chi(C_n) = \begin{cases} 2 & n \text{ even} \\ 3 & n \text{ odd} \end{cases} \ .$$

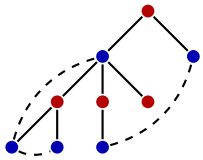
- Trees are two-colorable, $\chi(\text{tree}) = 2$ (for any tree with at least an edge), as we can arrange the nodes in "levels." Then all levels with odd parity (depth) are colored in one color and those with even parity in another:



Theorem 5.10. *A graph $G = (V, E)$ is bipartite, i.e. two-colorable, if and only if it contains no odd cycle, i.e., no cycle of odd length.*

Proof. The implication $\Rightarrow$ follows directly from the observation above. If G is bipartite, it cannot contain an odd cycle, since then $\chi(G) \geq 3$. It remains to show that, if G contains no odd cycle, then it is also bipartite. Let's consider a spanning tree of G. We can then color the levels with two colors as above. We now need to show that there are no edges in G that connect two levels of the same color. This follows from the observation that adding an edge connecting

two vertices of levels with same parity – examples are indicated by dashed lines below – always introduces an odd cycle, and thus a contradiction:



□

The theorem yields an efficient way, i.e., in *linear time*, to decide whether a graph is two-colorable. To decide however whether a graph is 3-colorable is an NP-complete problem. Again, we see the two extremes of computational hardness in two very similar-sounding problems.

5.8 Exercises

Exercise 5.1. Let $n \geq 3$. Find the following:

1. What is *the smallest* m for which it is true that every graph with n nodes and m edges has a cycle?
2. What is *the greatest* m for which it is true that every graph with n nodes and m edges has no cycle?
3. What is *the smallest* m for which it is true that every graph with n nodes and m edges is connected?
4. What is *the greatest* m for which it is true that no graph with n nodes and m edges is connected?

Exercise 5.2 (Leaves). Show that every tree with a vertex of degree k has at least k leaves.

Exercise 5.3 (Trees and Degrees). Let $n \in \mathbf{N}$ and $k_1 \geq k_2 \geq \cdots \geq k_n \geq 1$, $k_i \in \mathbf{N}$. Show that there exists a tree with n nodes v_i, $i = 1, 2, \ldots, n$ and $\deg(v_i) = k_i$ for all i if and only if it holds

$$\sum_{i=1}^{n} k_i = 2n - 2 \ .$$

Exercise 5.4 (Cayley). Let K_n be the complete undirected graph with n vertices: Every pair of nodes is connected by an edge. Theorem 5.5 shows that K_n possesses exactly n^{n-2} different spanning trees. We now form a new graph from K_n by leaving away exactly one edge, i.e., the connection between the vertices v_1 and v_2. How many different spanning trees does the resulting graph possess?

Hint. How many of the spanning trees of K_n contain the eliminated edge? Exploit the symmetry of K_n.

Exercise 5.5 (Platonic Solids). In a *platonic solid*, all faces are the same regular n polygon, the same number of which meet in every corner:

1. Decide which of the solids are Eulerian.
2. Decide which are Hamiltonian.
3. Show that all the platonic solids are planar graphs.
4. (hard) Prove that there are no further Platonic solids.

Chapter 6

Cryptography

6.1 Diffie-Hellman key exchange

For millennia, information has been selectively disclosed through cryptography. A first, Roman example is the *Caesar cipher*, which relied on permuting the letters of a message according to a fixed permutation. The permutation, i.e., the *key*, was known to those sending and receiving the clandestine information. All ciphers used before the 1970s relied on such a *secret key* that was shared among the communicating parties and had to be kept hidden from possible adversaries. This required the parties to meet, or send a trusted courier, before sending the secure communication, and exchange the key.

In 1976, *Whitfield Diffie and Martin Hellman* published a protocol to establish a key using merely *authenticated but public* communication over a public channel. Two parties, usually referred to as *Alice* and *Bob*, exchange unencrypted messages. Thus, even a potential adversary Eve can read the messages. Nonetheless, Alice and Bob can generate a secret key Eve cannot retrieve. Imagine: Two people talk publicly in a room, everyone can hear everything, and in the end, they share a secret... This sounds like magic!

First of all, Alice sends Bob a large prime number p ($\sim$ 200 digits). They can use this number p to construct a trapdoor one-way function, i.e., a function that is easy to compute but hard to invert (*one-way*) without knowing a particular piece of information called a *trapdoor*. Using this sort of function, it is possible to establish the key. We now take a closer look at one particular one-way function.

One-way function. Let us consider the function

$$x \mapsto R_p(2^x) \equiv 2^x \mod p \ .$$

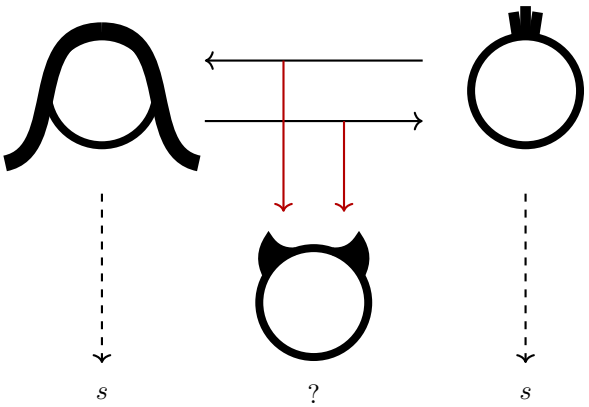


Figure 6.1: Alice and Bob communicate through a public channel. Even though Eve receives all their messages, she does not know their shared secret.

Here, $R_p(a)$ denotes the remainder of a divided by p. How can one compute this function efficiently? Multiplying

$$\underbrace{2 \cdot 2 \cdot \ldots \cdot 2}_{x \text{ times}}$$

x times (where the "reduction modulo p" can be made after every step; so at least the numbers never outgrow p) before taking the modulo requires x multiplications. For large exponents, this is totally inefficient.

A more efficient algorithm: Let us first assume that x is a power of 2, i.e., $x = 2^k$. Then

$$2^{(2^k)} = \underbrace{\left(\left(\left(2^2\right)^2\right)^2 \cdots\right)^2}_{k \text{ times}}$$

one can compute the power by $k = \log_2 x$ multiplication, each time multiplying the result with itself. To generalize this to general x, we write the argument in its binary expansion:

$$x = (x_r x_{r-1} x_{r-2} \cdots x_0)_2 = x_0 + 2^1 \cdot x_1 + 2^2 \cdot x_2 + \ldots + 2^r \cdot x_r \ .$$

The computation one has to perform turns into *repeated squaring and multiplying*:

$$2^x = 2^{x_0 + 2^1 \cdot x_1 + 2^2 \cdot x_2 + \ldots + 2^r \cdot x_r} = \left(\left(\left(2^{x_r}\right)^2 \cdot 2^{x_{r-1}}\right)^2 \cdots 2^{x_1}\right)^2 \cdot 2^{x_0} \ .$$

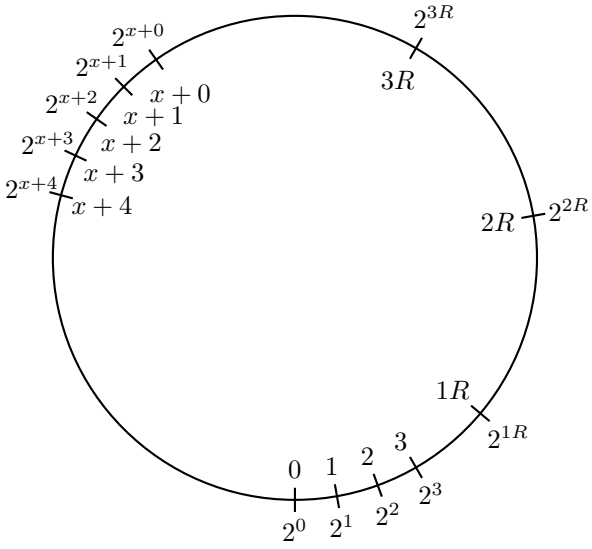


Figure 6.2: This illustrates the *giant-step-baby-step* algorithm.

Thus, we are left with $2r \approx 2 \cdot \log x \leq 2 \cdot \log p$ computational steps.

Computing the inverse

$$R_p(2^x) \mapsto x$$

seems harder: One way is to multiply repeatedly $2 \cdot 2 \cdot 2 \cdots$ and compare at each step with $R_p(2^x)$ until equality holds. There is, however, a slightly more efficient protocol. For some fixed number $R < p$, one calculates the *giant steps* $2^R, R^{2R}, 2^{3R} \ldots$ and the *baby steps* $2^{x+1} = 2^x \cdot 2, 2^{x+2} = 2^x \cdot 2^2, \ldots, 2^{x+R} = 2^x \cdot 2^R$, as shown in Figure 6.2, until one encounters equality $2^{j \cdot R} = 2^{x+i}$. Then, $x = j \cdot R - i$. The number of computational steps is then

$$R + \frac{p}{R} \ .$$

The optimal choice of R is roughly $\sqrt{p}$: Then, there are therefore roughly $\Theta(\sqrt{p})$ steps. Note that

$$\sqrt{p} = 2^{1/2 \cdot \log p}$$

is still *exponential* in the input size, $\log p$: Even the baby-step-giant-step procedure is by far less efficient than the computation of $x \mapsto R_p(2^x)$. Therefore, the function is a good candidate for a one-way function.

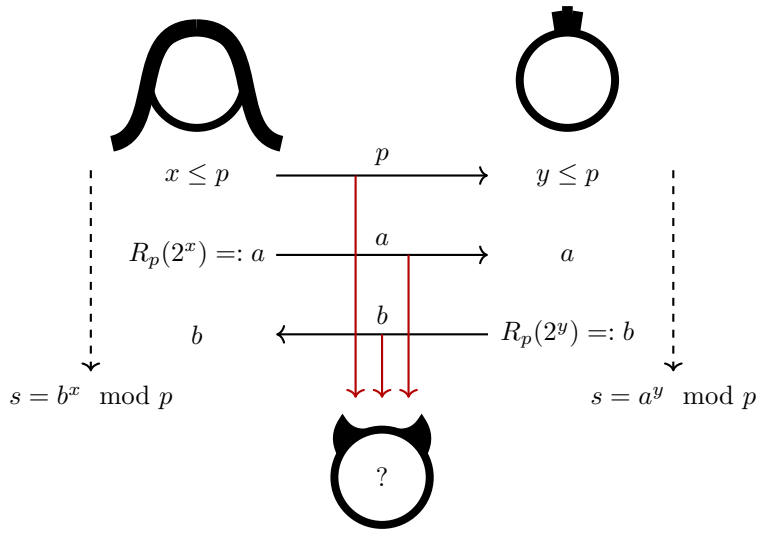


Figure 6.3: The Diffie-Hellman key exchange protocol.

The Diffie-Hellman key exchange protocol. Employing the one-way function from above we can now devise a protocol for Alice and Bob to agree on a shared key:

- Alice chooses a large prime p and sends it to Bob.
- Alice chooses a random integer $x \leq p$ and sends $a := R_p(2^x)$ to Bob.
- Bob chooses a random integer $y \leq p$ and sends $b := R_p(2^y)$ to Alice.
- Alice computes $R_p(b^y)$ and Bob $R_p(a^y)$. These are the same numbers, and thus their shared secret:

$$s := a^y \equiv (p^x)^y \equiv (p^y)^x \equiv b^x \pmod{p} .$$

Schematically the protocol is depicted in Figure 6.3.

Eve knows a, b, and p but neither x nor y. There is no more efficient way known to compute s from a, b, and p than to first invert $R_p(2^x)$ and compute x (or y). This inversion is believed to take a long time if the number x, y, and p are sufficiently large. (*Not* on a quantum computer, however.)

There is an intuitive analogy with padlocks, depicted in Figure 6.4: Alice and Bob have each two open padlocks of the same kind. (You can even imagine them to be, a little impractically, without a key: They can be closed, but they cannot be opened again. This is in fact a mechanical analog to a one-way

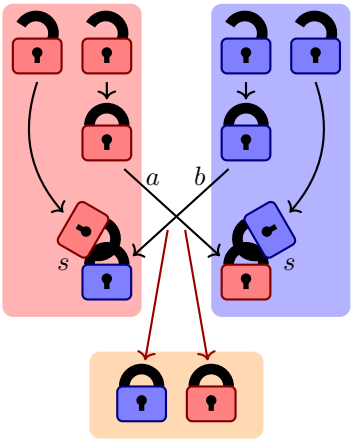
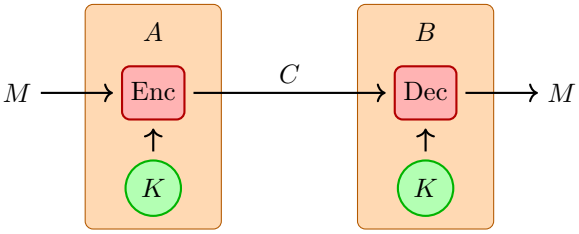


Figure 6.4: Eve can copy the two closed locks sent by Alice and Bob, but she cannot intertwine them to obtain the secret s.

function.) Both close one of the padlocks and send it to the other party. With the remaining padlock, they can lock the two padlocks together. Even Eve can intercept the messages and "copy" the closed locks. But as she cannot open any of the two, she cannot lock them together to obtain the secret s.

6.2 The RSA cryptosystem

Symmetric cryptosystems. Until the late 1970s, publicly known cryptosystems relied on Alice and Bob sharing the same key.[1]

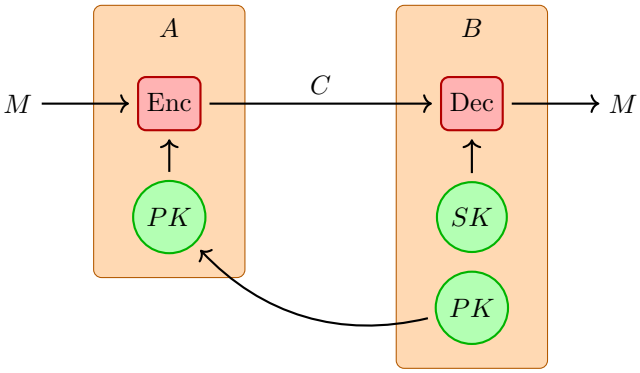


Alice first uses the key K to encrypt the message. Bob eventually decrypts the message using the very key K again. Therefore, such cryptosystems are called *symmetric*.

[1]Except for this: Two entities had invented public-key encryption before that: The British secret service, who kept it as a secret, of course, and *Ralph Merkle*, whose ideas were beautiful, less practical, and totally unrecognized.

Diffie and Hellman solved the issue of distributing the secret key K using merely a public authenticated channel as we have seen above. They also envisioned a completely new scheme employing different keys for encryption and decryption, without having to rely on a shared secret key.

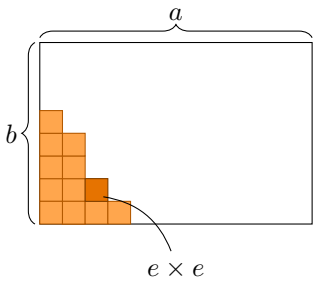
Public-key cryptosystems rely on a publicly known key PK to encrypt a message. This key does, however, not allow decryption of the message. The recipient knows the *private* (or secret) key SK required to decrypt the message.



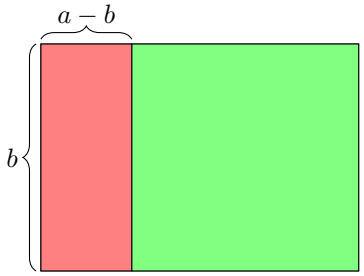
Diffie and Hellman could not provide a realization of such an *asymmetric* or *public-key* cryptosystem. In 1977, however, Ron Rivest, Adi Shamir, and Leo Adleman came up with a public cryptosystem employing the factorizing problem. Before turning to the protocol itself, we introduce the required mathematical basis.

6.2.1 Euclid's algorithm

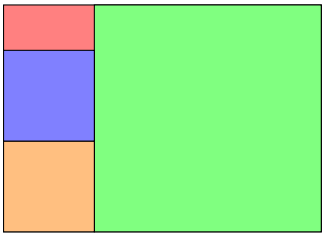
Given a terrace of dimensions $a \times b$, what is the length of the *largest* square tile that can be used to exactly cover the terrace?



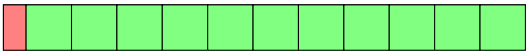
We are actually looking for the greatest common divisor, $\gcd(a, b)$. One could compute the gcd from the prime factors of a and b. There is, however, no efficient algorithm known to compute the prime factors of a number. (Ironically, RSA's security depends on just that.) Instead, the "terrace picture" allows deriving an efficient way to compute the gcd (without computing first all prime factors). One can reduce the question to the same for smaller rectangles (à la "divide et impera"): We can "subtract" from the full terrace the square $b \times b$ and ask for the largest tile exactly covering what remains, $(a - b) \times b$:



The answer is the same as for the rectangle $a \times b$. We can then repeat this over and over until we get to a square:



What if a is much larger than b, $a \gg b$? Then, we have to "subtract" the same square $b \times b$ over and over again:



It is more efficient to directly reduce to the *remainder* $R_b(a)$. In particular, we have $0 \leq R_b(a) < a$. The algorithm consists of repeatedly reducing to smaller rectangles $a_k \times b_k$ with alternately $a_k := R_b(a_{k-1}), b_k = b_{k-1}$ or $a_k = a_{k-1}, b_k = R_a(b_{k-1})$. The longer side is replaced by the remainder of the longer modulo the shorter (which becomes then the longer). The algorithm terminates when one of the remainders becomes zero. This happens at the latest when one reaches a square. This yields *Euclid's algorithm* – an efficient algorithm to compute the gcd of two numbers a and b.

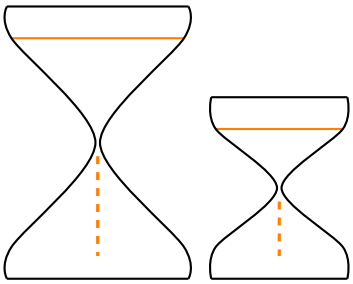
Example 6.1. Let us consider the case $a = 17$ and $b = 10$. First we reduce a to the remainder $R_{10}(17) = 7$, then b to the remainder $R_7(10) = 3$ and so on and so forth. We therefore obtain the sequence

$$(17, 10) \to (7, 10) \to (7, 3) \to (1, 3) \to (1, 0).$$

Thus, we obtain $\gcd(17, 10) = 1$.

6.2.2 The *extended* Euclidean algorithm

Given two hour-glasses of different sizes, i.e., with different time intervals of $a = 17$ and $b = 10$ minutes ($a, b \in \mathbb{N}$), can one measure one minute?



$a = 17$ $\qquad$ $b = 10$

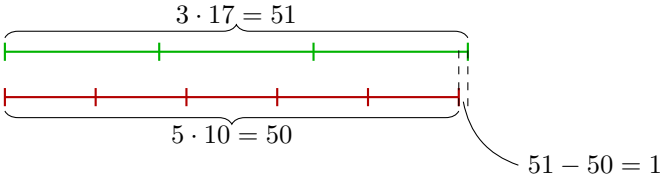
As the gcd of 17 and 10 is 1, we can actually measure an interval of 1 minute: We consider an iterative method to measure smaller time intervals with the two hour glasses. We derive for each of the remainders in the Euclid algorithm how to measure the corresponding time interval until we reach 1.

To measure an interval of 17 minutes, we use the first hour glass once and the second not at all. Similarly, for 10 min, we use merely the second one. To measure an interval of 7 minutes, we turn both hour glasses at the same time. But the interval we are interested starts only after the sand in the second hour glass has fallen through. Thus, we delay the start of the interval by 10 minutes.

Indicating delays in this way with minus signs, we obtain the following number of uses of the hour glasses:

	17	10
17	1	0
10	0	1
7	1	−1
3	−1	2
1	3	−5

The first two lines can be filled easily. Each of the following lines can be computed from the previous two. Take, for instance, the last line: From integer division in the first column, we obtain the factor $k := 7/3 = 2$. We obtain the last line (l_5) from the previous two, (l_4) and (l_3) by computing $(l_3) - k \cdot (l_4) = (l_5)$. To measure an interval of one minute, we use the hour glasses as follows.



This means

$$1 = 3 \cdot 17 + (-5) \cdot 10 \ ,$$

or in terms of modulo[2]

$$1 \equiv 3 \cdot 17 \pmod{10} \ .$$

Therefore, this yields an algorithm to compute inverses in modular arithmetic; the method is called the *extended Euclidean algorithm*.

$$3 \equiv 17^{-1} \pmod{10}$$

6.2.3 The Chinese remainder theorem

In order to count large groups if people, the ancient Chinese came up with the following algorithm about 3000 years [sic!] ago: People had to form blocks with rows of (relatively prime) number, such as 2, 3, 5, etc. For each prime number, one would then record the remainder. As, for instance, shown in Figure 6.5. From the first block we conclude that the number of soldiers x is odd. From the second we obtain, that it is a multiple of 3. We can write this as

$$\begin{aligned} x &\equiv 1 \pmod{2} \\ x &\equiv 0 \pmod{3} \\ x &\equiv 2 \pmod{5}. \end{aligned}$$

Then, the only solution to this equation smaller than 30 is 27. More precisely, the triplet of equivalences given is equivalent to the single equivalence

$$x \equiv 27 \pmod{30} \ .$$

[2]Recall that $a \equiv b \pmod{m}$ is equivalent to $R_m(a) = R_m(b)$ or m dividing the difference $(a - b)$, i.e., $m|(a - b)$.

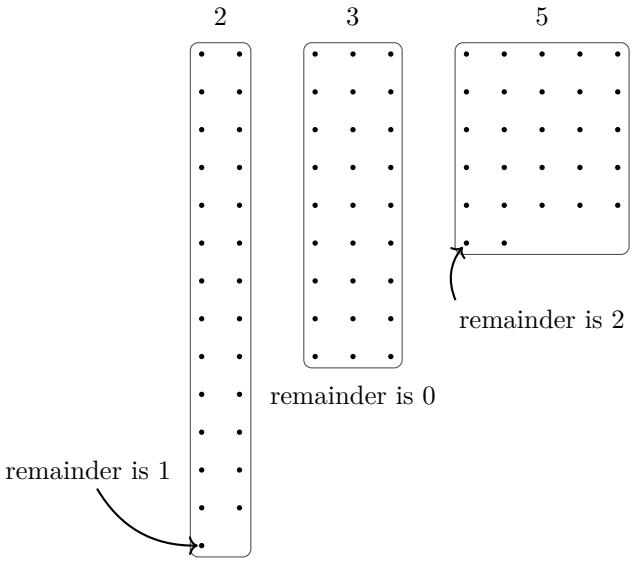


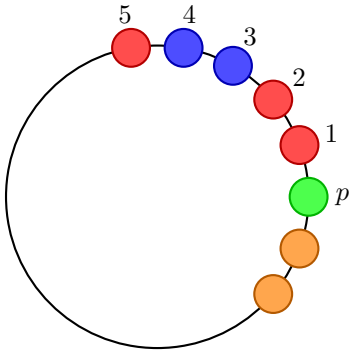
Figure 6.5: Counting in blocks.

We are in particular interested in the following special case. For two prime numbers p and q it holds

$$x \equiv a \pmod{pq} \quad \Leftrightarrow \quad \begin{cases} x \equiv a \pmod{p} \\ x \equiv a \pmod{q} \end{cases} . \tag{6.1}$$

6.2.4 Fermat's little theorem

Finally, we revise Fermat's theorem, treated at the beginning of the course. Imagine we want to make a necklace with p pearls, each having one of a colors:



How many different necklaces are there? A priori we can choose for each of the pearls among a colors. So there are a^p different necklaces. There are, however, some that differ merely by a rotation. Thus, we have to divide by the number of rotations. This would reduce the number of necklaces too much, as we initially counted the a necklaces with all pearls having the same color merely once. So we have to subtract those before the division and add them afterwards again. Thus we obtain

$$\#\text{necklaces} = \frac{a^p - a}{p} + a \in \mathbb{N}.$$

Therefore, p divides $a^p - a$

$$p \mid a^p - a \quad \Leftrightarrow \quad a^p \equiv a \pmod{p} \quad \Leftrightarrow \quad a^{p-1} \equiv 1 \pmod{p}\ , \tag{6.2}$$

where the last equivalence requires that p does not divide a, i.e., $\gcd(a, p) = 1$.

6.2.5 Chinese Euclid

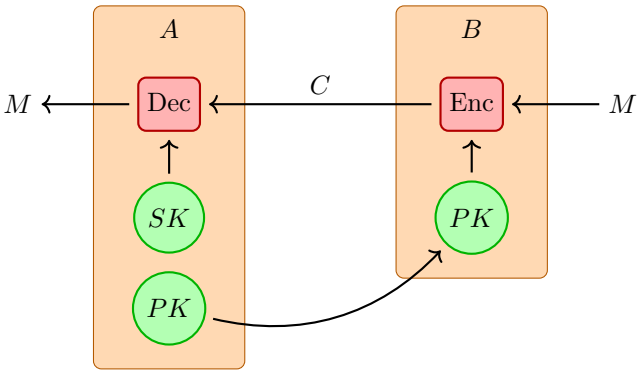
We can now combine the Chinese Remainder Theorem (6.1) and Fermat's theorem (6.2) to obtain for two primes p, q, with $p \neq q$, and an integer number a with $\gcd(a, pq) = 1$

$$\begin{aligned} & a^{(p-1)(q-1)} \equiv 1 \pmod{p} \\ & a^{(p-1)(q-1)} \equiv 1 \pmod{q} \\ \Rightarrow \quad & a^{(p-1)(q-1)} \equiv 1 \pmod{pq} \end{aligned}$$

Equipped with these tools, we introduce the RSA protocol.

6.2.6 The RSA protocol

Let Bob be the sender and Alice the receiver:



The protocol consist of these steps:

- Alice generates two large prime numbers p, q and computes $n := p \cdot q$. She then chooses a number e that is relatively prime to $(p-1)(q-1)$, i.e., $\gcd(e, (p-1)(q-1)) = 1$. For instance $e = 3$, may be a possible choice.

- Alice computes

  $$d := e^{-1} \pmod{(q-1)(p-1)} \qquad \Rightarrow \qquad d \cdot e \equiv 1 \pmod{(q-1)(p-1)}$$

  using the extended Euclidean algorithm.

- Alice now sends the public key $PK = (n, e)$ to Bob while she keeps the secret key $SK = (n, d)$ secure with herself.

- Bob can now encrypt his message $M \leq n$ by computing $C = M^e$ employing the square and multiply algorithm. Then Bob sends C to Alice.

- Alice then decrypts the message as follows.

$$\begin{aligned} C^d &\equiv (M^e)^d \\ &\equiv M^{e \cdot d} \\ &\equiv M^{1+k(p-1)(q-1)} \quad \text{for some } k \in \mathbb{N} \\ &\equiv M^1 \cdot \big(\underbrace{M^{(p-1)(q-1)}}_{=1 \mod n} \big)^k \\ &\equiv M \pmod{n} \end{aligned}$$

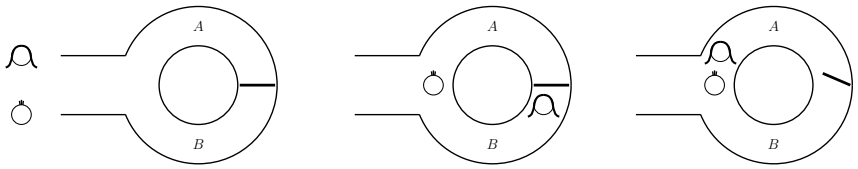
Security. RSA's confidentiality is based on the hardness of factoring. To compute p and q from n seems hard (it is not on a quantum computer — so come to that course, any spring term!). An upper bound on the number of computational steps is

$$\sqrt{n} = 2^{1/2 \log n} \ ,$$

i.e., exponential in the size of the input. The fastest algorithm publicly known today is the so-called *number field sieve* and requires

$$2^{c(\log n)^{1/3}}$$

computational steps. This is less than exponential but not polynomial in the input size, thus called *subexponential*. It suffices to choose two large (e.g., 2048 bit) integers p, q to make RSA secure (assuming that the NSA does not have secretly developed a substantially more efficient algorithm for factorizing or possess a properly working quantum computer).



(1) P and V find the cave (2) P chooses B (3) V chooses A

Figure 6.6: The cave zero-knowledge proof.

6.3 Zero-knowledge proofs

Secure communication is not the only goal of cryptography. In fact, cryptography can be used to achieve a variety of seemingly impossible tasks. Consider, for example, the problem of protecting sensitive data. The last few years have seen a dramatic increase of database breaches, so much so that the word "data breach" is now part of our daily lexicon. One idea to mitigate this risk is *data minimization*, easily summarized as "only collect the data you *really* need." For example, in most countries, unsupervised social media use is allowed only if the user is older than 13. As a consequence, service providers usually ask for the user's age during the sign in phase. This is unnecessary data: It would be enough for the service provider to know that the user's age is greater than 13, not its actual value. A solution is to use a *zero-knowledge proof*, a cryptographic protocol that allows one party, called the *prover*, to convince a *verifier* that she knows a solution to some problem (age ≥ 13), without revealing such information (her age) to the verifier. The *zero-knowledge* property allows to minimize the data that the service provider has to store: The verifier gains no information from the protocol execution beyond the certainty that the prover's secret satisfies the verifier's requirement.

6.3.1 The magic cave

A great example of a zero-knowledge proof is the following story, published at an important conference in 1989 by Jean-Jacques Quisquater and others.

Two friends, Peggy and Victor, one day uncover a magic cave. The cave has only one entrance and it is shaped as a ring, as shown in Figure 6.6 (1). In the middle of the ring, opposite from the entrance, there is a magic door that can only be opened by whispering a magic word. Peggy knows the word, but does not want to reveal it to Victor. To convince him that she can open the door without Victor learning the magic word, Peggy proposes a challenge: If she can win it, Victor can be sure she knows the word. They label the left and right path as side A and B, respectively. Then, Peggy enters the cave

and chooses a side unbeknownst to Victor. Once she reaches the door, Victor enters the cave (Figure 6.6 (2)) and asks Peggy to return from a side of his choice. If Peggy knows the word, she can always return from the selected side, independently from Peggy's and Victor's choices. At the same time, Victor cannot learn the word, as he is too far away to hear it. However, if she lied, she has only a 50% chance to satisfy Victor's requirement; if they repeat the experiment many times in a row, her success probability becomes close to zero. Thus, if Peggy constantly reappears from the correct side, Victor can safely conclude that she in fact knows the magic word.

6.3.2 Zero-knowledge proof of discrete logarithm

In cryptography, this can be used to prove knowledge of a particular value, e.g., of a password $x \in \mathbb{Z}$. Assume Victor is a email provider, and Peggy is one of the users. When Peggy creates her account and generates her password $x \in \mathbb{Z}$, she also publishes an information S that will help her to later prove that she knows x. The public S consists of three elements: a prime $p > x$, a randomly chosen integer $g < p$, and the value $y = R_p(g^x) = g^x \pmod{p}$. As we already saw for the Diffie-Hellman key-exchange (cf. Section 6.1), extracting x from y seems to be computationally hard, so Peggy's password is safe even if she publishes $S = (p, g, y)$. Now, to prove to Victor that she knows the password, Peggy and Victor run the following protocol: She selects a random r in $\{0, 1, \ldots, p-1\}$, and sends Victor the number $t = R_p(g^r)$. Upon receiving t, he replies with a bit c chosen at random in $\{0, 1\}$. Depending on the value that she receives, Peggy computes her answer z as follows:

- If $c = 0$, she sets $z = r$,
- If $c = 1$, she sets $z = R_{p-1}(x + r) = x + r \pmod{(p-1)}$.

When Victor receives z, he checks the value of c and,

- If $c = 0$, he checks that $R_p(g^z) = t$.
- If $c = 1$, he checks that $R_p(g^z) = R_p(t \cdot y)$.

The protocol is summarized in Figure 6.7.

First, let us check whether the protocol works when Peggy is honest. In case $c = 0$, this is trivially true. When $c = 1$, it works thanks to Fermat's Little Theorem (cf. Section 6.2.4). To see why, let us write $x + r$ as $x + r = (x + r) \mod (p-1) + (p-1) \cdot n$, where n is the result of the integer division of $x + r$

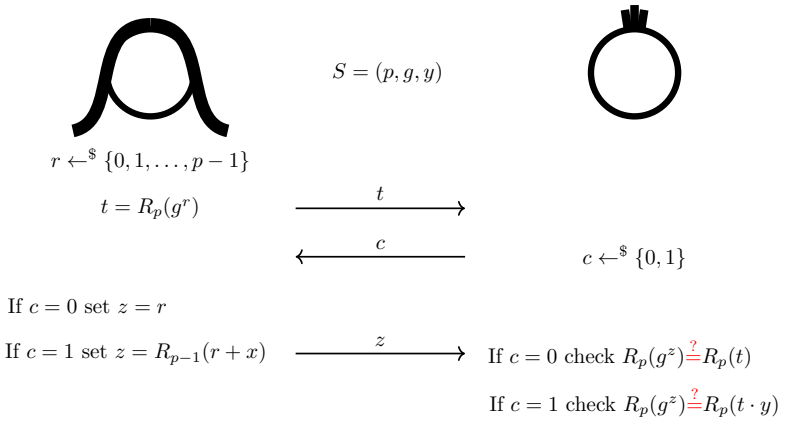


Figure 6.7: Zero-knowledge proof for the login problem.

by $p - 1$. Then Fermat's Little Theorem yields:

$$\begin{aligned} R_p(t \cdot y) &= g^{x+r} \pmod{p} \\ &= g^{(x+r) \pmod{(p-1)}} \cdot g^{(p-1)\cdot n} \pmod{p} \\ &= g^{(x+r) \pmod{(p-1)}} \cdot 1 \pmod{p} \\ &= g^z \pmod{p} . \end{aligned}$$

Thus, the protocol works. However, can Victor be sure that Peggy actually knows x whether the checks are satisfied? It is clear that Peggy can always send r, thus she can always win the case $c = 0$. However, she cannot send $R_{p-1}(r+x)$ unless she knows x. Analogously, if she chooses a random r' as the value of $R_{p-1}(r+x)$ before the game, she still cannot compute t, as she cannot extract r from r' without knowing x. Therefore, Peggy can successfully cheat only with 50% probability. As before, Victor can repeat the test multiple times, to lower the cheating probability, and be sure that Peggy knows the value x.

6.4 Cryptography and the quantum computer

In the last two centuries computers went from being huge mechanical machines powered by steam (like those built by Charles Babbage in the 19$^{\text{th}}$ century) to complex, small machines relying on microscopic technology powered by electricity. With each technological advancement came faster algorithms, implementations exploiting the larger computational power and storage of the

new machines. From the point of view of cryptography, this is both a menace and a blessing: Faster computers allow for faster attacks to existing cryptosystems, but also for practical implementation of more complex protocols. Thus, part of cryptographic research focuses on designing cryptography tailored to the threads and possibilities that future computers will yield.

Quantum computers are one of the possibilities for future computers architecture. Such machines run exploiting phenomena from quantum physics and thus approach computing in a completely new way. While they might not be faster than classical computer on average, they definitely outperform current supercomputers on some specific tasks. As it was hinted already when we talked about RSA, factoring large numbers is one of them. Peter Shor published a fast quantum algorithm for factoring in polynomial-time already in the 1990s, and since then cryptographers have been concerned with finding cryptographic schemes that cannot be broken even using a quantum computers. This part of cryptography is called *post-quantum cryptography*.

But what is exactly the difference between classical and post-quantum cryptography? In a nutshell, post-quantum cryptography includes all protocols that still run on classical computers whose security relies on mathematical problems that would take centuries to solve using either classical or quantum computers (or both!). At the same time, quantum computers can be used to run cryptographic protocols, too! These protocols are grouped under the label of *quantum cryptography*.

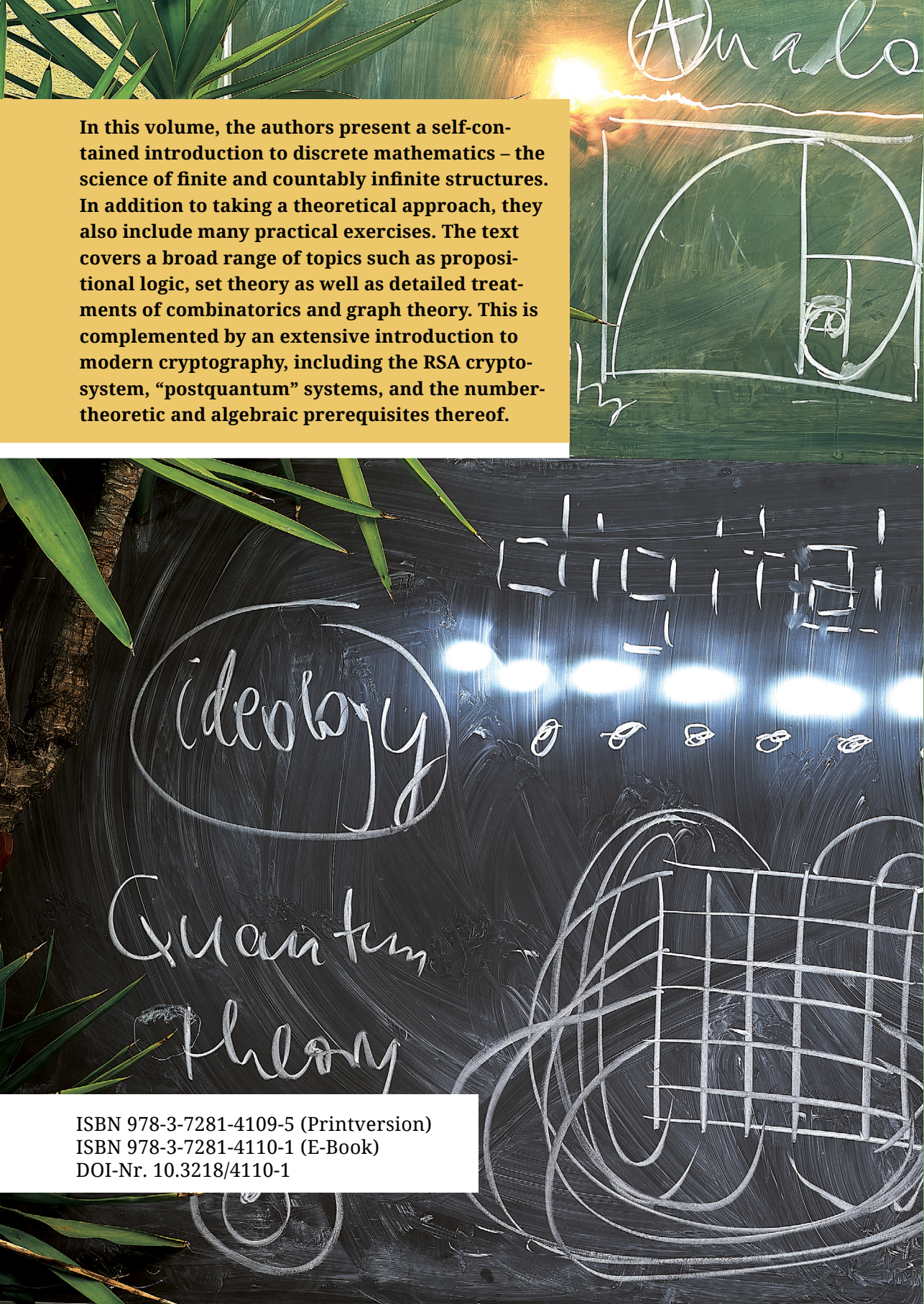
To summarize, one can classify cryptographic protocols in three main categories depending on how they perform in a quantum world:

Classical Cryptography includes protocols that run on classical computers and are secure when attacked using classical computers, but can be broken using quantum computers. RSA and Diffie-Hellman key-exchange are unfortunately in this category, alongside all protocols whose security relies on the hardness of factoring over elliptic curves (sounds weird, but one can actually do algebraic computations on the points of a curve!).

Post-Quantum Cryptography includes protocols that run on classical computers and are secure when attacked using both classical and quantum computers. Symmetric key primitives are in this category, alongside code-based protocols, and others.

Quantum Cryptography includes protocols that run on quantum computers and are secure when attacked using classical and quantum computers. In fact, as they rely on peculiarity of quantum computers that cannot be simulated using classical computers, there is no need to worry about classical attacks.

Bear in mind that this is not a comprehensive classification: If a new type of computer appears in the future, this list has to be revised.



In this volume, the authors present a self-contained introduction to discrete mathematics – the science of finite and countably infinite structures. In addition to taking a theoretical approach, they also include many practical exercises. The text covers a broad range of topics such as propositional logic, set theory as well as detailed treatments of combinatorics and graph theory. This is complemented by an extensive introduction to modern cryptography, including the RSA cryptosystem, "postquantum" systems, and the number-theoretic and algebraic prerequisites thereof.

ISBN 978-3-7281-4109-5 (Printversion)
ISBN 978-3-7281-4110-1 (E-Book)
DOI-Nr. 10.3218/4110-1