COMP11120 Mathematical Techniques for Computer Science

Andrea Schalk
A.Schalk@manchester.ac.uk

7th September 2020

Organizational issues

Structure of the unit

COMP11120 is a 20 credit course unit that is taught over two semesters. You will receive one mark at the end of the unit, and it is that mark which decides whether you pass or fail.

You can find electronic copies of these notes in *Blackboard*[1], as well as past exam papers. Lecture podcasts, solutions to exercises and lecture slides (where they exist) will also be made available there as term progresses. Note that the electronic version of the notes shows colours, while the printed version is black and white, and that it has clickable links that allow you to follow references within the text easily.

When and where

The unit is taught with two lectures and one examples class per week.

Semester 1

Please refer to your personal timetable regarding the time and location of lectures.

For the examples classes you are split into four lab groups. Which group you are in is determined by the letter part of the name of your tutorial groups. Please refer to your personal timetable regarding location and timing of your example classes. In Semester 1, examples classes start in Week 1 and run through to Week 12, while in Semester 2 they go from Week 2 to Week 11.

Who

This course unit is delivered by a team:

- Andrea Schalk, `A.Schalk@manchester.ac.uk` (course leader).
- Renate Schmidt, `Renate.Schmidt@manchester.ac.uk`,
- Clare Dixon, `Clare.Dixon@manchester.ac.uk` and

Assessment

Overview

The assessment of this course unit[2] has the following components:

[1] This is the University's E-learning environment.

[2] Please note that this has changed for academic year 20/21.

80% Examination. If possible there will be two invigilated exam, one in January covering material from Semester 1 and one in May/June covering material from Semester 2. Otherwise there will be one invigilated exam in May/June covering the whole year.

20% Coursework. This consists of 1% each for solutions you submit each week for the exercises on the weekly exercise sheet.

Note that if you have to **resit** the unit because your original mark is not high enough to allow you to progress then this reassessment is based on **one paper covering all the material**.

Examinations

There usually are written examinations after both, Semester 1 and Semester 2. Note that this course unit has been substantially revised in recent years and reached its current format in 2015/16. As a consequence exam papers before that year do not accurately reflect the kind of paper you will sit. The exams are based mostly on questions which are similar to the core questions from the examples classes, with a few marks (at most 20%) available for questions beyond that.

There are three questions in each paper, one each for the major topics covered in that semester. This means that in **Semester 1** there will be one question each on

- statements and proofs,
- logic and
- probability,

while in **Semester 2** there will be one question each on

- recursion and induction,
- relations and
- linear algebra.

Either exam may contain some material from other chapters in the notes that have been covered to date. Note in particular that notions that were taught in Semester 1 may appear in the paper for Semester 2 (so a question on relations could involve, say, complex numbers). The assessed exercises, in particular the core ones, are good examples of the kind of thing you may be asked to do in an exam.

Mid-term test

In the past there has been an invigilated mid-term test in early November. This will not take place this year.

Exercise sheets

For each examples classes you are told to prepare a number of exercises (typically five exercises per week, see the sheets at the end of the notes). In some cases you have a choice regarding which part of an exercise you want to solve. This is to ensure that students who have seen similar material before still have something interesting to do. Whenever there is a choice all the parts of an exercise are concerned with the same abstract property, so no matter which ones you do you will familiarize yourself with that property. The exercises typically are about looking at a particular concept or technique.

There are two kinds of exercises on the sheets:

- **Core**. These belong to the core of the taught material, and there are typically three such exercises each week. They are labelled *CExercise* in the notes.
- **Extensional**. These questions extend the core material and they are labelled *EExercise* in the notes.[3]

If you are stuck on one exercise for ten minutes without making progress then *move on to the next one*—different exercises do not usually depend on each other. Also note that the different parts to each exercise are *not necessarily in rising order of difficulty*, and do not necessarily depend on each other.

The point of the examples classes are

- to give you feedback on whether you have understood the material and answered the exercises correctly at the right level of detail, and
- to help you with exercises you had difficulties solving, concentrating on the core exercises.

The feedback as well as the marking is carried out by *graduate teaching assistants* (GTAs). Please note that we have a limited number of qualified GTAs and you should use your time with them wisely. Make sure you have questions ready to ask them, or request that they look at a particular (part of an) exercise.If you need more help or feedback than a GTA can supply in the available time then note that there are two *academic members of staff* in each examples class in Semester 1, and one in Semester 2, to help with this.

Please note that on this course unit it is **not possible to get an extension**, in particular, students registered with DASS cannot have automatic one week extensions. This is because solutions to the marked exercises are published after the last examples class in a given week.

You have to submit a pdf with your written solutions on Blackboard by the given deadline. A GTA will look at it and assign a mark each week, and the examples class that week is used to give feedback to the group and to go over aspects of the solution that students struggled with.

The GTA may call on you to explain your solution to the rest of the group, so please make sure you are prepared to do that. You will be able to share your pdf with the rest of the group so that you can all look at it together, and your task is to tell everybody how you came up with your solution. Please note that preparing for this is part of the deliverable each week and that you may lose mark if you are unable to do so. For this reason we need you to attend the examples class each week even if you are happy with your submission.

Your marks will appear in *Spot*, the department's system that allows you to get an overview over your coursework marks. The numbers we use have the following meaning:

- 0: You have not submitted anything that looks like a credible attempt to solve the exercises for the week.

[3] Exercises named as *Exercise* do not appear on any sheets and are there to give you more material to practice on.

- 1: You have tried to solve some exercises, but you either did not do much work or your solutions were significantly lacking in quality or accuracy.

- 2: Your have tried to solve all the core exercises for the week and your solutions show that you have understood a good part of the material.

- 3: You have tried to solve all the core exercises; your solutions for the core exercises are largely solid, but there are some mistakes that go beyond being minor.

- 4: You tried to solve all of the core exercises and your solutions are correct with only very minor shortcomings.

- +1: An additional mark is available each week if you have made a substantial attempt at solving the extensional exercises.

These exercises are assessed work. If you copy somebody else's work you are committing a **plagiarism** offence. If you let somebody copy your work you are also committing plagiarism. This will result in a mark of 0 for the week, and if you commit repeated or particularly serious offences you may have to face Department or Faculty panels. See below for a detailed description of what you are allowed to share with other students.

GTAs record the marks each week, and there is a delay until you can view the mark in SPOT. If you think something has gone wrong with the recording process please contact the course leader, Andrea Schalk, at

`A.Schalk@manchester.ac.uk`

as soon as possible.

If there is a good reason that you cannot attend an examples class in a given week then you must email the unit staff to let them know.

The discipline of mathematics

Mathematics is a foundational subject for many sciences and engineering disciplines, and this holds true for computer science as well.

Please do have a look at the *FAQs* and the *Expectations* documents available on Blackboard.

School versus university level

Note that there are a number of differences between what you have been used to from school to what is expected at university level.

- There are a lot more people in your teaching group—as a consequence you have less access to those who teach you, and you have to become more self-reliant (but you can get a lot out of other students on the course if you go about it in the right way).

- The material is taught at a faster pace. That means that fewer teaching sessions are devoted to a piece of material, which means that you will typically see at most one relevant example before being expected to tackle a problem yourself.

- The material is more abstract—you are expected to cope with abstract definitions, whereas at school the emphasis is often on solving problems.

- When solving problems at school there is often a very recognizable pattern to coping with questions of a particular kind. At university level this pattern, if it exists, is of a much more general kind.

- Taken together you should not expect to be able to solve the exercises based on just looking at examples discussed in the lectures or spelled out in the notes. You are expected to develop your own approaches to solving problems rather than following ones you have been given.

This course unit

In order to master the subject sufficiently to be able to apply it it is not sufficient to treat mathematics as a series of recipes for various calculations.

Instead mathematics formally defines a number of abstract concepts, and then derives their properties (and so justifies calculations of various kinds). As a discipline mathematics is unique in that its theories may be proved or disproved conclusively. It is not necessary to acquire data to study whether the predictions of a theory match observations from experiments—instead one proves statements based on universally accepted derivation rules.

This course unit provides an introduction to those areas of (abstract) mathematics which are of particular importance in computer science. Its aim is to do so rigorously by introducing various concepts formally, and then to show what properties these have. For this purpose it is also necessary to talk about the notion of proof in order to make it clear how valid derivations are formed. Throughout the notes examples from applications within computer science are given wherever possible.

This is the only course unit teaching mathematics in the curriculum for computer science students in Manchester. It therefore has to lay the foundations for everything that is to follow, and in some cases you will not see the material introduced here until the second, or even the third year.

How to learn the material

Abstract mathematics requires a fairly lengthy learning process. Typically material has to be studied more than once over a longer period to be fully understood. The point of weekly assessed exercises is to encourage students to try their best to understand various abstract concepts and their uses. In the examples classes you get feedback regarding your understanding, and the revision period should provide another opportunity to improve your understanding before the exams.

Mathematics isn't a spectator sport—the only way to learn it is to do it yourself. Watching others carry out calculations or proofs may be instructive at first, but in the end the only way of properly understanding it is to make your own brain work through the problems given.

You should think of this kind of learning as happening on a spiral: Every time you revisit a topic you will find yourself having achieved a higher level of understanding, provided you really try to get your head around it. This kind of material is not suitable for postponing all the work until revision time. It is also very difficult to catch up once you have fallen behind. For this reason doing set work every week is rewarded by giving you marks, determining 20% of the mark for each semester.

Notes. You should think of the lecture notes as providing an account of the *examinable material.* The exercises are part of that material—usually not the specifics but the techniques. You are required to *read the notes independently.* Note that we expect students to know all the numbered definitions where a concept is defined in bold face (some have concepts defined in italics), although we don't ask many exam questions which are just about the definitions. Some of the material in the notes is there to signpost topics you will not meet until in your second or third year. To prepare for the exams we expect you to be familiar with the material around the core exercises; most 20% of the exam mark will rely on material other than that.

Materials. You will find materials on Blackboard to help you understand what is written in the notes, in the form of short videos that cover various ideas and show you how to solve examples. There are also quizzes that help you to check whether you have actively engaged with those ideas. Please note that not everything that is covered in the notes is covered here—you are expected to began to learn independently.

Exercises. You will have worked on some of the exercises as part of the assessed work, the others are available for revision purposes. You will receive solutions to most[4] of the exercises in the course of term. There are also *optional exercises* which are not part of the examinable material, and to which no solutions will be handed out.[5] Note that the exercises appear in the text where they are more appropriate. When working on an exercise for assessed work you will almost certainly have to (re-)read the material preceding the exercise.

Online sessions. Online sessions with unit staff give you an opportunity to ask questions. We may ask you to submit your questions prior to the session to give us a chance to prepare.

Blackboard Forum. There is a forum on Blackboard that allows you to ask questions.

Examples classes. The point of the examples classes is to give you feedback on exercises you have tried yourself, and to help you with concepts or ideas you are finding difficult.

In that way all the contact hours are dedicated to help you learn the material, but ultimately what you put in yourself will be the biggest decider in what you get out.

Additional help. If you find you are struggling with the material there are a number of actions you may take. You may want to consult an alternative text in case the explanations as given in the notes don't suit your way of understanding. You should ask lots of questions in the examples class—there will be two people in each examples classes whose only job is to answer questions. You should also talk to your friends on the programme, for example the other members of your tutorial group. It may make sense for some of you to form a study group and to tackle the material together. The weekly PASS sessions are an excellent way of getting help from students who are in their second or third year.

[4] At a minimum this will include solutions to all assessed exercises.

[5] The optional exercises are intended or students who wish to obtain a more complete understanding of various concepts. Some of them are intended to get those interested started on thinking some way beyond the material presented here.

Expected effort

The Department expects students to work around 40 hours a week (although some students put in more). This means that for every ten credit course unit you should expect to work roughly seven hours a week (including contact hours).

In COMP11120 you have two contact hours a week, which means you are expected to work **five hours a week** working through the material on Blackboard as well as reading the notes and solving the exercises (in particular the assessed ones).

Academic malpractice

Academic malpractice is a general term that covers a number of ways in which students may break regulations by submitting work as their own which in part is based on unattributed contributions by others.

Note that the rules for collaboration are **different for COMP11120** from almost all other units you may take in the Department of Computer Science. Please do not apply them to other units.

We are happy for you to

- try to understand the exercises in a group, including working out how to solve a particular exercise and

- compare your solution to somebody else's.

You **must not**

- copy somebody else's solution (even if you make some changes along the way) or

- give your solution to somebody (either to copy, or adapt, or to take a picture of it) or

- ask to look at somebody else's solution for an exercise which you haven't already solved.

If you are working as a group to understand a particular exercise, then each of you must **write down your own version** of your solution. If you aim to give a counterexample to a statement you should find different counterexamples based on the understanding you gained jointly. If you aim to give a proof then each of you should independently formulate your understanding, and you should not agree on using specific variable names, for example. Students who present identical (or very similar) answers will be penalized.

If you and another student have solved an exercise independently then we are happy for you to compare your solutions, and if you think the solutions are different you should try to work out between you whether they are both correct, and why. If one of you has an incorrect solution, and the other solution is deemed correct, then the first student may change their solution, but this **must not be done** by merely copying the correct version. Instead the student should start from their incorrect solution and adjust that. Similarly, you **must not let another student copy** your solution, even if you feel sorry for them. They will get at least one mark for their incorrect attempt, and if they've understood what was wrong they should be able to fix their solution for a higher mark.

Where one student has solved an exercise and another hasn't it is fine for the first to explain how to tackle the questions, to give tips for how to find the solution, and to explain the concepts involved. In a situation like this it is not okay to show your solution to the other student.

Remember also that you have to be able to explain your solution to get the marks, and that identical write-ups will be treated as copying.

These notes

You will receive a number of handouts over the course of the academic year, referred to here as 'these notes'. They are a text book in the making which we have written because there is currently none that covers the material required for our programmes at a suitable level of detail. They will cover the following topics:

- complex numbers,
- statements and proofs,
- logic,
- recursion and induction,
- relations,
- linear algebra (vectors and matrices).

These notes have been tailored specifically to the curriculum in computer science here at the University of Manchester. They can be read on a number of levels, and are meant to accompany you through your time here, allowing you to go back and reread material as it becomes relevant to your studies in one of the more advanced course units.

Programming languages. To make connections between the abstract ideas of mathematics and the more concrete implementations a number of references are made. These make connections to Python and Java the two main languages you encounter in the two introductory programming units. While Java is not taught until Semester 2 there is reference to the language in these notes for Semester 1.

The notes were new for the academic year 2014/15, and have been revised and expanded every year subsequently. Nonetheless there is doubtlessly room for improvement. We would like to invite you to email us if you have suggestions, for example

- if you think you have found a mistake,
- if you think that there is a passage that is misleading,
- issues that require more detailed explanations,
- issues that require more worked examples,
- exercises whose instructions are unclear,
- examples and exercises that we should think about adding,
- online resources you have found useful.

We will certainly incorporate such suggestions for future years, but if there is a particularly important issue, or one where we get several requests, we may hand out supplementary notes as term progresses. We also use interactions with you in examples classes and lectures to guide us, but we would very much appreciate receiving emails on this topic.

We would like to thank Graham Gough, Jonas Lorenz, Joe Razavi, Francis Southern, Yegor Guskov, Francisco Lobo, Sarah Nogueira, Chris Tedd, Luca Minciullo, David Paukstello, Sami Alabed, Anuj Vaishnav, Ruba Alassaf, Andrew Webb, Andrei Vintila, Leshna Shamloll, Robertas Maleckas, Gytautas Buivydas, Oliver Blanthorn, Toby Osborne, Nuno Nobre and Pouya Adrom for helping us improve these notes. Toby and Pouya also variously helped us improve marking schemes and the running of examples classes.

Some of the ideas for operations on lists come from notes originally written for COMP112 by David Lester, and the idea of including recursively defined functions for natural numbers is based on notes by Graham Gough for a previous version of this course. Some of the examples and exercises used have been inspired by a variety of sources, but most were specifically created for these notes. In particular some ideas for the probability part of this course where inspired by reading Kees van Schaik's notes for a previous version of this unit. Andrea Schalk would also like to thank Renate Schmidt for very careful proof-reading of her handouts.

Francisco Lobo provided the Latex package that allows me to compile the document in such a way that solutions to all exercises, and solutions to assessed exercises, form separate chapters at the end.

Joe Razavi has contributed to teaching on this unit over a number of years and he has given invaluable feedback and provided ideas.

Additional literature

There is *no one book* that covers all the material in these notes. They have been specifically written for computer science students going through the curriculum here in Manchester and they cover some material that is not present in most text books. Nonetheless it can be beneficial to look at an alternative presentation of material, and we suggest some books you may want to try for this purpose. These are all present in the Department's (ask at the SSO) as well as the University Library for you to borrow. There are certainly other books available that cover much of the material, and you will also find the internet a useful resource on specific topics.

Here is an account of what is covered in which of the suggested books.

- **Basics**. If you find the material in this chapter difficult you should use resources on the web to help you, and maybe also get some of the books below and read about the relevant part.

- **Complex numbers**. This is not covered in most text books aimed at computer scientists, but you will find relevant material in Jordan and Smith. All the operations described in the notes also appear in the Wikipedia article on complex numbers at https://en.wikipedia.org/wiki/Complex_number, and there are a number of webpages available that aim to explain the general ideas.

- **Statements and proofs**. The definitions and many of their properties are covered in Epp, some of them also in Truss. All the definitions have entries on Wikipedia, and in many other online sources. (for example, the notion of a commutative operation is explained at https://en.wikipedia.org/wiki/Commutative_property such sources provide an alternative point of view that you may find helpful.

- **Logic**. We will not be following any particular text books, but Truss and Epp contain chapters that cover formal logical systems, in particular, the basics of Boolean semantics and properties of propositional formulas. Again you can use a search engine to find additional material on each topic on the web.

- **Probability**. This subject is addressed to some extent in Epp, and also in Jordan and Smith, but less generally than in these notes.

- **Recursion and induction**. Recursion and induction are both covered in Epp as well as in Truss, but not at the same level of generality. There are many resources available on the web, for example the Wikipedia article on structural induction here https://en.wikipedia.org/wiki/Structural_induction.

- **Relations**. Relations are covered in both, Epp and Truss, and there are many online resources.

- **Linear algebra**. Again there is no text book which covers the material exactly in the way we do. Truss has some material on vectors and matrices, but it is quite condensed. Additional pointers to literature are given in that part of the notes.

Writing about mathematics is difficult, and in particular this holds for giving rigorous arguments. The following book gives a lot of good advice on that subject: Kevin Houston, **How to Think Like a Mathematician**, *Cambridge University Press*, 2009, ISBN: 052171978X.

Here are some text books on mathematics for computer scientists that you may want to consult in addition to these notes.
Susanna Epp. **Discrete Mathematics with Applications**, *Brooks/Cole* 2011, ISBN: 0495826162.

This book covers much of our material as well as significant parts of COMP11212. It is available online via the university library.

J.K. Truss. **Discrete Mathematics for Computer Scientists**, *Addison-Wesley*, 1999. ISBN: 0201360616

This book covers roughly the same material as the first, but with fewer applications, and in slightly less detail.

D.W. Jordan and P. Smith. **Mathematical Techniques: an Introduction for the Engineering, Physical, and Mathematical Sciences**, *Oxford University Press* 2008, ISBN: 9780199282012.

Much of this book is concerned with continuous mathematics, so if you need a refresher it may well be useful for that purpose beyond what is said above.

D. C. Montgomery and George C. Runger. **Applied Statistics and Probability for Engineers (5th edition)**, *Wiley* 2010, ISBN: 978047050578.

This is a very applied text which may help you with connecting the concepts taught here with applications.

E. Angel, **Interactive Computer Graphics**: **a Top-down Approach Using OpenGL**, *Pearson* 2008, ISBN: 9780321549433.

This book connects the material on matrices and vectors with the intended application in computer graphics.

All these books are available in the departmental Library, as well as in the University Library.

Contents

Chapter 0

Basics

This chapter explains some concepts most of which you should have encountered before coming to university; but you may not have been given formal descriptions previously. These notions give us a starting point so that we have examples for the formal development that follows from Chapter 1 onwards, but note that some of the concepts and properties that appear in this chapter are put on a formal footing subsequently.

Whenever you find concepts used in the notes that have familiar names you should check this chapter to ensure that you only use the fact provided here. There will be no lectures about the material in this chapter, but the examples classes in Week 1 are there to make sure you understand the ideas and the notation used here. Note that there is a universally accepted language described here that you will also encounter in other course units.

Note that we here assume that certain collections of numbers, with various operations, have already been defined. You will see formal definitions of most of these (real numbers being the exception) in Chapter 6 which we will study in Semester 2. The purpose of assuming they are present at the start is to allow us to use them as examples.

0.1 Numbers

Naively speaking, numbers are entities we often use when we wish to calculate something. Mathematically speaking, there is typically rather more going on: Numbers are sets with operations, and these operations have particular properties. Many of these properties are named and studied in Chapter 2.

0.1.1 Natural numbers

The **natural numbers** are often also referred to as *counting numbers*, and the collection of all of them is typically written as $\mathbb{N}$. For the time being we assume that you know what these numbers are; a formal definition appears as Definition 54 in Chapter 6.

Foreshadowing the formal definition, we point out that simplest way of formally describing the natural numbers is to say that

- there is a natural number 0 and
- given a natural number n there is another natural number Sn, the **successor of** n, more usually written as $n + 1$.

Every natural number can be generated in this way, although to reach 123456, for example, one has to apply the successor operation quite a few times! This also means that given a natural number n, we know that one of the following is the case:

- either $n = 0$ or
- there exists a natural number m with $n = Sm$ (or, if you prefer, $n = m + 1$).

This might seem like a trivial observation, but it is the basis of using the concept of *recursion* to define properties or functions for the natural numbers, and also for being able to prove properties by *induction*.

This is described in detail in Section 6.4 of these notes. Here we look at the informal notions you have met at school.

With the natural numbers come some operations we use; their properties are given below.

- Given natural numbers m and n we can add[1] these to get
$$n + m.$$
- Given natural numbers m and n we can multiply[2] these to get
$$m \cdot n.$$

You are allowed to use the following about natural numbers, except in Section 6.4 where we prove many of these facts formally.

Fact 1

Given x, y, and z in $\mathbb{N}$ we have[3]

$$\begin{aligned} x + y &= y + x && \text{commutativity of } + \\ (x + y) + z &= x + (y + z) && \text{associativity of } + \\ x + 0 &= x = 0 + x && 0 \text{ unit for } + . \end{aligned}$$

For the same variables we also have[4]

$$\begin{aligned} x \cdot y &= y \cdot x && \text{commutativity of } \cdot \\ (x \cdot y) \cdot z &= x \cdot (y \cdot z) && \text{associativity of } \cdot \\ x \cdot 1 &= x = 1 \cdot x && 1 \text{ unit for } \cdot . \end{aligned}$$

For the same variables we also have the property

$$x \cdot (y + z) = x \cdot y + x \cdot z \qquad \cdot \text{ distributes over } + .$$

For the same variables we also have[5]

$$x + z = y + z \qquad \text{implies} \qquad x = y.$$

[1] A formal definition of addition appears in Example 6.31.

[2] A formal definition of this operation appears in Example 6.36.

A mathematician might say that the natural numbers form a commutative monoid with unit 0 when looking at the addition operation, and a commutative monoid with unit 1 when looking at multiplication. In Section 2.5 we look formally at the properties given by these equalities.

There is one additional property we require. The following is used in *Euclid's algorithm, see Example 6.42*, but also to define *integer division*, see below, which appears in Chapter 2.

Fact 2

Given y in $\mathbb{N}$ and x in $\mathbb{N}$ with $x \neq 0$ there exist unique numbers k and l in $\mathbb{N}$ such that

- $0 \leq l < m$ and
- $y = kx + l$.

We use this fact to define a division operation on natural numbers, known as **integer division**[6]. We define[7]

$$y \operatorname{div} x$$

to be the unique number k in $\mathbb{N}$ in Fact 2. This is the number of times x divides y (leaving a remainder). We define the **remainder for integer division** by setting

$$y \bmod x$$

to be the unique l from Fact 2. This is the remainder y leaves when divided by x. See Code Examples 0.1 and 0.2 to see how these operations are implemented in Python and Java.

Example 0.1. For example, we have that

$$\begin{array}{rcr} 5 \operatorname{div} 2 = 2 & \text{and} & 5 \bmod 2 = 1 \\ 7 \operatorname{div} 3 = 2 & \text{and} & 7 \bmod 3 = 1 \\ 9 \operatorname{div} 3 = 3 & \text{and} & 9 \bmod 3 = 0 \\ 11 \operatorname{div} 4 = 2 & \text{and} & 11 \bmod 4 = 3. \end{array}$$

Example 0.2. We look at two particular cases to see the patterns which develop.

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$n \bmod 3$	0	1	2	0	1	2	0	1	2	0	1	2	0	1
$n \operatorname{div} 3$	0	0	0	1	1	1	2	2	2	3	3	3	4	4

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$n \bmod 7$	0	1	2	3	4	5	6	0	1	2	3	4	5	6
$n \operatorname{div} 7$	0	0	0	0	0	0	0	1	1	1	1	1	1	1

[3] Formal proofs of these properties appear in Example 6.35 as well as Exercise 159.

[4] Formal proofs of these as well as the final property are given in Exercise 160.

[5] Note that we cannot subtract within the natural numbers, so this property gives us the strongest statement we have. Mathematicians would say that addition is right (and also left) cancellable.

[6] Sometimes also called *Euclidean division*.

[7] See the following section to see that this idea can be extended to the integers.

Example 0.3. Note that it is not necessarily the case that

$$x \cdot (y \operatorname{div} x) = y,$$

for example

$$2 \cdot (3 \operatorname{div} 2) = 2 \cdot 1 = 2 \neq 3.$$

This is different from the way of dividing numbers you may be used to[8] and that is the reason that this kind of division has a different name, and a different symbol.

Lemma 0.1

For all natural numbers x and y, where $x \neq 0$, we have

$$y = x \cdot (y \operatorname{div} x) + (y \bmod x).$$

Exercise 1. Give an argument that Lemma 0.1 is valid using Fact 2.

Definition 1: divisible

Given natural numbers $x \neq 0$ and y, y **is divisible by** x or that x **divides** y if and only if there exists a natural number k such that

$$x \cdot k = y.$$

Note that x divides y if and only if it is the case that

$$y \bmod x = 0.$$

Definition 2: even/odd

An natural number x is **even** if and only if x is divisible by 2. Such a number is **odd** if and only if it is not divisible by 2.

This means that x is even if and only if

$$x \bmod 2 = 0,$$

and that x is odd if and only if

$$x \bmod 2 = 1.$$

Note in particular that 0 is an even number.

We might also want to think about which equations we can solve in the natural numbers. Assume that m and n are elements of $\mathbb{N}$.

For example, we can solve

$$m + x = n,$$

[8] See for example the discussion in the Section 0.1.3 on rational numbers below.

within $\mathbb{N}$, provided that[9] m is less than or equal to n, which we write as $m \leq n$.

We can also solve

$$mx = n,$$

within $\mathbb{N}$ provided that $n \bmod m = 0$. Because of the side conditions required we see that a lot of equations we can write down using the available operations do not have a solution.

We can use the natural numbers to count something, for example the number of instructions in a computer program, or the number of times a program will carry out the body of a loop. This is important to do when we are trying to estimate how long it may take a program to run on a large-size problem.

There are a lot of natural numbers, namely infinitely many. But by mathematical standards the natural numbers are the smallest infinite set, and there are substantially larger ones. Sets of this size are set to be *countably infinite*. This is formally defined in Section 5.2.

Computer languages do typically *not* implement the natural numbers—instead, a programming language will have support for all natural numbers up to a particular maximum. Nothing truly infinite can be implemented in any real-world computer (but there are theoretical computation devices which have infinite storage). Quite often programming languages have a built-in type for integers instead of natural numbers, as is the case with Python and Java.

0.1.2 Integers

A simple way of explaining the **integers** is that one wants to expand the natural numbers in order to make it possible for every number to have an *inverse* with respect to addition, that is, for every number x there is a number y, usually written as $-x$, with the property that

$$x + y = 0 = y + x.$$

Defining the integers formally in a way that supports the above idea is quite tricky. Such a description is given in Chapter 7, see Definition 62. It's fairly easy to describe the elements of this set, called[10] $\mathbb{Z}$, once one has the natural numbers, since one can[11] say

$$\mathbb{Z} = \mathbb{N} \cup \{-x \mid x \text{ in } \mathbb{N}, x \neq 0\},$$

but this does not tell us anything about how to calculate with these numbers. So this does not, mathematically speaking, define the integers with all the operations we customarily use for them.

The **absolute**, $|x|$, of an integer x is defined to be[12]

- x if x is greater than or equal to 0 and
- $-x$ if x is less than 0.

[9] The solution to such an equation would have to satisfy $x = n - m$ and this is not always defined.

[10] The notation $\mathbb{Z}$ for the set of integers is very common within mathematics, the letter coming from the German word 'Zahlen', or numbers. You may know this set under a different name, but that should not worry you.

[11] The following expression uses symbols explained in detail in Section 0.2.

[12] See Example 0.32 for a definition of this as a function, although that definition is for real numbers.

We (very rarely) use $\mathbb{Z}^+$ to refer to those integers[13] which are greater than or equal to 0.

Fact 3

The equalities from Fact 1 also hold[14] if the variables are elements of $\mathbb{Z}$. We have an additional property, namely,

$$\text{for every } x \text{ in } \mathbb{Z} \text{ there exists a unique } y \text{ in } \mathbb{Z} \text{ with } x + y = 0 = y + x.$$

We say that this number y is the **additive inverse for x with respect to addition**. The number $-x$ is defined to be the additive inverse of x.

A mathematician would say that $\mathbb{Z}$ forms a commutative ring with multiplicative unit 1.

Many people use *subtraction* as an operation. However, it is much preferable to think of this not as an operation, but as

$$y - x$$

being a shortcut for adding the additive inverse of x, $-x$, to y—in other words, this is merely a shortcut for

$$y + (-x).$$

Please do not talk about subtraction on this course unit, but about adding additive inverses. There are many many situations in mathematics where not all inverses exist,[15] and so you should pause to think whether the operation you wish to carry out is legal.

Fact 2 changes a bit when we use it for integers.

Fact 4

Given y in $\mathbb{Z}$ and x in $\mathbb{Z}$ with $x \neq 0$ there exist unique numbers k and l in $\mathbb{Z}$ such that

- $0 \leq l < |m|$ and
- $y = kx + l$.

Hence we may extend the definitions of the operations of mod and div, that come from **integer division** for natural numbers as defined above, to the integers. In other words, for integers x and y,

- $y \operatorname{div} x$ is the unique k, and
- $y \bmod x$ is the unique l,

from the above fact.

[13] This set of numbers is, of course, equivalent to $\mathbb{N}$

[14] The formal proof that addition satisfies these properties appears in Section 7.3.7 and Exercise 193 provides proof that multiplication satisfies them .

[15] For example, for the rational, real and complex (see Chapter 1) numbers, the number 0 has no multiplicative inverse. When you study matrices you will see that very few matrices have multiplicative inverses.

Example 0.4. We have that

$$\begin{array}{rcr}
-5 \operatorname{div} 2 = -3 & \text{and} & -5 \bmod 2 = 1 \\
7 \operatorname{div} -3 = -2 & \text{and} & 7 \bmod -3 = 1 \\
9 \operatorname{div} -3 = -3 & \text{and} & 9 \bmod -3 = 0 \\
-11 \operatorname{div} 4 = -3 & \text{and} & -11 \bmod 4 = 1.
\end{array}$$

Example 0.5. Once again we look at two particular cases to see the patterns which develop.

n	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
$n \bmod 3$	1	2	0	1	2	0	1	2	0	1	2	0	1
$n \operatorname{div} 3$	-2	-2	-1	-1	-1	0	0	0	1	1	1	2	2

n	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
$n \bmod 7$	2	3	4	5	6	0	1	2	3	4	5	6	0
$n \operatorname{div} 7$	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	1

Lemma 0.2

For all integers y, and all integers $x \neq 0$, we have

$$y = x \cdot (y \operatorname{div} x) + (y \bmod x).$$

The notions of evenness and oddness transfer with the same definitions as for natural numbers. Indeed, the definitions given below can be applied to natural numbers viewed as integers, and they will give the same result as the corresponding definition from the previous section.

Definition 3: divisible

Given integers $x \neq 0$ and y we say that y **is divisible by** x or that x **divides** y if and only if there exists an integer k such that

$$x \cdot k = y.$$

Note that x divides y if and only if $y \bmod x = 0$.

Exercise 2. Use Fact 4 and the formal definition of divisibility and mod to argue that the previous sentence is correct.

Definition 4: even/odd

An integer x is **even** if and only if x is divisible by 2. Such a number is **odd** if and only if it is not divisible by 2.

Exercise 3. Use Fact 4 and the formal definition of mod to argue that a natural number x is even if and only if $x \bmod 2 = 0$, and odd if and only if $x \bmod 2 = 1$.

How do the even numbers relate to those numbers which are a multiple of 2? Can you make your answer formal? Do your answers change if x is an integer?

The fact that every number has an additive inverse means that for m and n in $\mathbb{Z}$ we can solve all equations of the form

$$m + x = n$$

within $\mathbb{Z}$ without reservations. Indeed, every equation in one variable which involves addition and additive inverses has a unique solution. On the other hand, equations of the form

$$mx = n$$

are still not all[16] solvable within $\mathbb{Z}$.

If we accept that there are infinitely many natural numbers then it is clear that there are also infinitely many integers. Because the natural numbers are embedded inside the integers one might assume that there are more of the latter, but actually, this is not a sensible notion of size for sets. Mathematically speaking, $\mathbb{N}$ and $\mathbb{Z}$ have the same size, see Section 5.2 for details of what that statement means.

Many programming languages support a data type for the integers. However, only finitely many of them are represented. In Pythonor Java, for example, integers are given by the primitive type `int`, and range in Java they range from -2^{31} to $2^{31} - 1$. In Python there is a type `long` of *long integers* which are integers of unlimited size.

Code Example 0.1. In Python there is an implementation of integer division. However, it does **not implement our definition** when faced with negative numbers. There are Python commands n // m and n%m with the property that

```
m x (n//m) + (n%m) = m
```

However the implementation does not force n%m to be non-negative, and so if you use the Python commands to play with integer division you will see results that are misleading as far as the underlying mathematics is concerned. Also see the following example for Java showing that programmers prefer to implement something different from the mathematicians' definition.

Code Example 0.2. In Java integer division is also implemented. Here is a procedure that returns the result of dividing n by m (as integers).

```
public static int intdiv (int n, int m)
{
    return n/m;
}
```

Similarly there is an implementation of the remainder of dividing n by m.

```
public static int intmod (int n, int m)
{
    return n % m;
}
```

[16] The solution would have to satisfy $x = n/m$, and this is not defined for all m and n.

Note, however, that this does not return the numbers that appears in our definition: If n is negative then n%m is a *negative number*. The way Java implements the two operations ensures that they satisfy Lemma 0.2, that is

n = m∗(n/m) + n%m.

The result of the Java expression n%m is 'equivalent *modulo* m' to the result of $n \bmod m$, see Section 7.3.5. This means that for negative n you can get

$n \bmod m$ by adding m to n%m.

In the programming language C the language specification does not state what the smallest and greatest possible integers are—different compilers have different implementations here. You have to work out what is safe to use in your system.

0.1.3 Rational numbers

One can view the **rational numbers**, usually written[17] as $\mathbb{Q}$, as the numbers required if one wants to have a multiplicative inverse for every number other than 0. But again, giving a formal definition of these numbers is not straightforward if one wants to ensure that all the previous operations are available.

One way of talking about the rational numbers is to introduce the notion of a *fraction*, written as

$$x/y,$$

where x and y are integers.

But we cannot define the rational numbers to be the collection of all fractions since several fractions may describe the same rational number: We expect $2/4$ to describe the same number as $1/2$

Formally we have to define a notion of equality (or equivalence) on fractions, whereby

$$x/y = x'/y' \qquad \text{if and only if} \qquad xy' = x'y.$$

There is a formal definition of the rational numbers, and their addition and multiplication, in Chapter 7, see Definition 63.

We have quite a bit of structure on $\mathbb{Q}$. All the facts for integers still hold, but we get a new property.[18]

Fact 5

The statements from Fact 3 remain true if all variables are taken to be elements of $\mathbb{Q}$. In addition,

$$\text{for all } x \text{ in } \mathbb{Q} \text{ with } x \neq 0 \text{ there exists } y \text{ in } \mathbb{Q} \text{ such that } x \cdot y = 1 = y \cdot x.$$

We say that y is the **multiplicative inverse for** x. Every element $x \neq 0$ has a multiplicative inverse and the standard notation for this element is x^{-1}.

A mathematician would say that $\mathbb{Q}$ with addition and multiplication is a field.

[17] The name comes from the Italian 'quoziente', quotient. We look at why this is in Semester 2, see Section 7.3.

[18] Exercises 193 and 194 provide formal proofs of most of these properties.

In my experience many students do not worry sufficiently about potentially dividing by 0—Fact 5 makes it clear that only for numbers unequal to 0 are we allowed to divide. In a recent exam paper a number of students reasoned that

$$b = b' \qquad \text{and} \qquad ba = b'a'$$

imply that $a = a'$, but in making this claim they neglected the case where

$$b = b' = 0,$$

which makes that conclusion false.

Many people speak of *division* as an operation on rational (and real) numbers, but again, this is merely a shortcut: Writing

$$y/x$$

is an instruction to multiply y with the multiplicative inverse of x, that is, it is a shortcut for

$$y \cdot x^{-1}.$$

The number 0 does not have a multiplicative inverse, and that is why division by 0 is not allowed. In this course unit, please try not refer to division as an operation, and when you multiply with inverses, always check to ensure these exist.

Exercise 4. What properties would a multiplicative inverse for 0 have to satisfy? Argue that one such cannot exist.

The notion of the **absolute** can be extended to cover the rationals, using the same definition.

Given q and q' in $\mathbb{Q}$ we can now solve all equations of the form

$$q + x = q' \qquad \text{and} \qquad qx = q' \qquad (\text{if } q \neq 0)$$

within $\mathbb{Q}$, provided that,[19] for the second equation, $q \neq 0$. And indeed, every equation with one unknown involving addition, multiplication and inverses for these operations is solvable provided that it is not equivalent to one of the form $qx = q'$ where $q = 0$, $q' \neq 0$.

The rational numbers are sufficient for a number of practical purposes; for example, to measure the length, area, and volume of something to any given precision, and also to do calculations with such quantities.

There are infinitely many rational numbers, but mathematically speaking, $\mathbb{Q}$ has the same size as $\mathbb{N}$. See Section 5.2 for how to compare the size of sets.

Most mainstream programming languages do not have a datatype for the rationals (or for fractions), but those aimed at algebraic computations (such as Mathematica and Matlab) do.

[19] Note how the restrictions we have to make on equations to ensure they are solvable connect with where the operations involved are defined (or not) for the various sets of numbers discussed here.

0.1.4 Real numbers

The rational numbers allow us to measure anything up to arbitrary precision, we may add and subtract them, and there are additive and multiplicative inverses (the latter with the exception of 0), which allows us to solve many equations. Why do we need a larger set of numbers?

There are several approaches to this question. Here we give two. If we look at the rational numbers drawn on a line then there are a lot of gaps.

Mathematically speaking we may define a *sequence* (of rational numbers), that is a list of numbers

$$x_n \text{ in } \mathbb{Q}, \qquad \text{one for each } n \in \mathbb{N}.$$

Sometimes a sequence can be said to *converge to a number*, that is, the sequence gets arbitrarily close to the given number and never moves away from it.[20] If such a number exists it is called the *limit of the sequence*. For example, the limit of

$$1,\ 1/2,\ 1/4,\ 1/8,\ \dots \qquad \text{that is} \qquad 1/2^n \text{ for } n \text{ in } \mathbb{N}$$

is 0.

Let us consider the sequence defined as follows:

$$x_0 = 1$$
$$x_{n+1} = \frac{x_n^2 + 2}{2x_n}$$

We may calculate the first few members of the sequence to get

$$1,\ 3/2,\ 17/12,\ 577/408,\ \dots$$

and, if expressed in decimal notation,

$$1,\ 1.5,\ 1.41\overline{6},\ 1.4142568627451,\ \dots$$

One may show that

$$x_n^2$$

gets closer and closer to 2, so we may think of the above as approximating a number r with the property that $r^2 = 2$.

Optional Exercise 1. Show that there is no rational number x with the property that $x^2 = 2$. *Hint: Assume that you have $x = m/n$ for some natural numbers m and n and derive a contradiction.*

Hence there are numbers that are approximated by sequences of rational numbers which are not themselves rational. Or, if we draw the rational numbers as a line then it has a lot of gaps.

One can define the notion of a *Cauchy sequence*. One may think of this as a sequence that should have a limit (because the sequence contracts to a smaller and smaller part of the rational numbers), but where there is no suitable rational number for it to converge to. One can define the **real numbers** $\mathbb{R}$ as being all the limits for all the Cauchy sequences one can build from the rationals. This gives a 'complete' set of numbers in the sense that every Cauchy sequence built from

[20] This can be defined mathematically but would take up more space than we want to give it here.

elements of $\mathbb{R}$ has a limit in $\mathbb{R}$. We use $\mathbb{R}^+$ to refer to those real numbers which are greater than or equal to 0. The numbers in $\mathbb{R}$ which are not in $\mathbb{Q}$ are known as the *irrational numbers*.

We do not give a formal definition of the real numbers in these notes—the above outline should convince you that this is reasonably complicated to do rigorously. We may think of the rational numbers as being included in $\mathbb{R}$. The real numbers again come with the operations of addition and multiplication, and inverses for these (but 0 still does not have a multiplicative inverse), and we again have the previous distributivity law for these operations. Just like the rational numbers, the reals with these operations form a *field*, see Fact 6.

Fact 6

All statements from Fact 5 remain true if the variables are taken to be elements of $\mathbb{R}$.

A mathematician would say that the real numbers, with addition and multiplication, also form a field.

All the sets of numbers discussed so far are *ordered*, that is, given two numbers we may compare them. See Section 7.4.1 on how one generally talks about this idea. Here we are concerned with giving additional facts you may want to use in solving exercises.

The definition of the **absolute** again transfers to this larger set of numbers.

Fact 7

Let x, x', y and y' be elements of $\mathbb{R}$. Then the following hold:

For all x, x' in $\mathbb{R}$	we have	$x \leq x'$ or $x' \leq x$.
If $x \leq x'$ and $y \leq y'$	then	$x + y \leq x' + y'$.
If $x \leq x'$ and $y \geq 0$	then	$x \cdot y \leq x' \cdot y$.
If $x \leq x'$ and $y \leq 0$	then	$x \cdot y \geq x' \cdot y$.
If $x \leq y$	then	$-x \geq -y$.
If $x \leq y$	then	$x^{-1} \geq y^{-1}$
If $x \geq 1$, $y, y' \geq 0$ and $y \leq y'$	then	$x^y \leq x^{y'}$.
If $x > 1$, $y, y' \geq 1$ and $y \leq y'$	then	$\log_x y \leq \log_x y'$.

An alternative approach to introducing numbers beyond the rationals is as follows. Within the rational numbers we are able to solve all ‘sensible’ equations in one variable involving addition, multiplication and their inverses with rational numbers. We may even add multiples of that variable with each other.[21] But we may not multiply the unknown with itself: Equations of the form

$$xx = q \qquad\qquad \text{or} \qquad\qquad x^2 = q$$

are not all solvable with $\mathbb{Q}$. By moving from $\mathbb{Q}$ to $\mathbb{R}$ we add a lot of solutions to such equations to our set of numbers. For example, all equations of the form

$$x^n = r$$

are solvable for n in $\mathbb{N}$ and r in $\mathbb{R}$ with $r \geq 0$. Indeed, we may replace n in $\mathbb{N}$ by q in $\mathbb{Q}$ and we still have solutions.[22]

The situation becomes quite complicated. First of all we define a new symbol: We write

$$\sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

for a sum of a finite number of elements.[23]

Given a **polynomial equation**, that is one of the form

$$\sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0$$

where a_i in $\mathbb{Q}$ for $0 \leq i \leq n$, there may be up to n different solutions, or there may be none at all. Those real numbers that are solutions to such polynomial equations are known as *algebraic numbers*. Examples are $\sqrt{2}$, $\sqrt[5]{17}$ and $\sqrt[3]{3/2}$.

But not all elements of $\mathbb{R}$ can be written as solutions to such equations. Those that can not are the *transcendental numbers*; famous examples are e and π, and less well-known ones e^π and $2^{\sqrt{2}}$. We can therefore *not*[24] use the idea that $\mathbb{R}$ arises from $\mathbb{Q}$ by adding solutions to equations over $\mathbb{Q}$ to formally define $\mathbb{R}$.

[21] These equations are called *linear*.

[22] And we may even replace q in Q by r' in $\mathbb{R}$ and use the idea of the *continuity of a function* to define the operation of forming x to the power of r', and we can still find solutions.

[23] This idea is formally introduced on page 6.45 in Chapter 6 but we use the $\sum$ symbol in Chapter 4 as well.

[24] There is a way of algebraically defining the real numbers, but that requires a lot of mathematical theory to be set up that is fairly advanced.

Real numbers are often referred to using *decimal expansions*. Such an expansion is given by an integer together with a sequence of digits (one digit for each natural number). For the integer 0 for example one gets numbers typically written

$$0.d_1 d_2 d_3 \ldots,$$

where for i in $\mathbb{N}$ we know that d_i in $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. It is common not to write trailing 0s (that is 0s where there is no different digit occurring to the right), so we write 3.14 instead of $3.14\overline{0} = 3.14000000\ldots$. Note, however, that a number may have more than one decimal expansion, and $0.\overline{9} = 0.9999999\ldots$ refers to the same number as $1.0000000\ldots = 1.\overline{0} = 1$.

Exercise 5. If we change base from 10 we can still express numbers using pre- and post-decimal digits. This question asks you to think a little bit about this.

(a) Translate the number 1.1 in base 2 to base 10.

(b) Translate the number 1.75 in base 10 to base 2.

(c) Give an alternative representation for the number 1.0 in base 2.

We don't really need real numbers in the 'real world', but a lot of what we might want to describe becomes a lot smoother if we are allowed to use them (the trajectory of a ball is much easier thought of as a line than a sequence of points with rational coordinates), and they allow us to be precise when referring to the circumference of a circle, for example.

The set $\mathbb{R}$ is infinite in size—but mathematically speaking, it is strictly larger than $\mathbb{Q}$. It is *uncountably infinite*. See Section 5.2 for more details.

No real-world computer can implement all the real numbers. This is no surprise given that there are infinitely many of them. But more importantly every implementation of (some of) the real numbers will only allow limited precision.[25] Programming languages typically have some kind of *floating point number* type to approximate real (and so also) rational numbers, such as `float` in Python or Java. It's not unusual for there to be a more precise type, such as `double` in Java. Note that operations on such numbers typically incur *rounding errors* (for these operations to be precise it would be necessary to change the range of numbers which are representable by adding more digits—for example, $.5/2.0 = .25$, and we need to go from 1 digit after the decimal point to 2). In Java there is also *bignum* which allows for arbitrary precision (since the maximal allowable length of the number can be extended), provided the number has a finite decimal expansion, but these come at a price in memory and time performance (and a program that keeps adding digits will eventually run out of memory). Floating point numbers are given by a *significand* and an *exponent* (because this increases the range of numbers that can be represented), where for a given base, the number described is

$$\text{significand} \times \text{base}^{\text{exponent}}.$$

0.1.5 Numbers

We typically think of the sets of numbers introduced here as being subsets of each other, with

$$\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R}.$$

[25] There are some languages where it is possible to carry out calculations to a pre-defined precision, but these are not main stream, and significant overhead is required to make this work properly.

Mathematically speaking, this is not strictly correct, but instead we have a function that embeds the integers, say, in the rationals, in such a way that carrying out operations from the integers also works if we think of the numbers as rationals. See Section 7.3.7 for a formal definition of the integers and the rational numbers.

Sometimes in these notes we do not want to specify which set of numbers we mean, and then we assume there is a set

$$N \qquad \text{with} \qquad \mathbb{N} \subseteq N \subseteq \mathbb{R}$$

with an addition and a multiplication operation satisfying Fact 1.

Note that we can use the equalities given in the various Facts about sets of numbers to show general properties without knowing which set of numbers we are referring to.

Example 0.6. In this example we show that it is possible to establish facts about numbers just from the general properties given in the various Facts above.

Let N be a set of numbers from $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$. Then by one of Facts 3, 5 or 6 we have for all x, y and z in N the distributivity law

$$x(y+z) = xy + xz.$$

Further by the same fact we know that there exists a number 0 in N with the property that for all x in N we have

$$0 + x = x = x + 0,$$

and for all y in N we have an *additive inverse for* y, z in N with

$$y + z = 0 = z + y,$$

and the associativity law. Together these tell us that for all x in N we have

$$\begin{aligned} x \cdot 0 &= x \cdot (0+0) && \text{0 unit for } + \\ &= x \cdot 0 + x \cdot 0 && \text{distr law,} \end{aligned}$$

and if we set y to be the additive inverse of $x \cdot 0$ we may conclude from the previous equality, by adding y on both sides, that

$$\begin{aligned} 0 &= 0 \cdot x + y && y \text{ add inverse for } 0 \cdot x \\ &= (0 \cdot x + 0 \cdot x) + y && \text{prev equality} \\ &= 0 \cdot x + (0 \cdot x + y) && \text{associativity law} \\ &= 0 \cdot x + 0 && y \text{ add inverse for } 0 \cdot x \\ &= 0 \cdot x && \text{0 unit for } + . \end{aligned}$$

Of course you have known for a very long time that for all those sets of numbers, multiplying 0 with any other number gives 0 once again. But have you ever wondered whether there is a good mathematical reason for that fact? The answer is that addition and multiplication have general properties that force this equality upon us.x

More powerfully, if we have any set with operations we may call $+$ and $\cdot$ which satisfy the given equalities we can show that multiplying any element with the unit for addition has to again be the unit for addition. We look at the general properties of operations in Section 2.5.

0.2 Sets

Sets are very important in mathematics—indeed, modern mathematics is built entirely around the notion of sets.

A **set** is a collection of items. Collections are required in order to

- make it clear what one is talking about (ruling some things in and others out);
- precisely define various collections of numbers—and in general, much of algebra is concerned with structures given by
  - an *underlying set* (for examples see Section 0.1 for various sets of numbers which, however, aren't formally defined here),
  - operations on the set (such as addition and multiplication, for various collections of numbers) and
  - the properties of these operations (see for example Facts 1, 3, 5 and 6).[26]
- define *functions* (see following section)—instructions for turning entities of one kind into entities of another.

Sets have *members* and indeed a set is given by describing all the members it contains. We write

$$s \in S$$

if s is an member of the set S, for example

$$\pi \in \mathbb{R} \qquad \text{or} \qquad aa \in \{a, aa, aaa\}.$$

Members are often also referred to as *elements*. There is a set that contains no elements at all, the **empty set**, $\emptyset$.

0.2.1 Sets

Deciding which collections of entities may be considered sets is not as easy as it might sound. Originally mathematicians thought that there would not be any problems in allowing any collection to be considered a set, but very early into the 20th century Bertrand Russell described the *paradox* named after him:

If we are allowed to form the set of all sets which do not contain themselves as members then we have a contradiction.[27] Theories that contain contradictions are called *inconsistent*, and they are not very useful since (at least according to classical logic) *every statement* may be deduced in an inconsistent theory. But if every statement is valid then the theory is of no use.

This caused something of a crisis, and prompted the creation of **set theory** as a field within mathematics. Set theory is concerned with the question of how sets

[26] These properties are studied in more detail in Section 2.5.

[27] Ask yourself whether the given 'set' contains itself.

may be built in a way that does not lead to contradictions. Mathematicians need to build fairly complicated sets, and making sure that all their constructions are allowed in the underlying set theory is not easy. The sets we require on this course unit are nothing like as complicated and so we do not have to worry about proper set theory here (and you should not refer to what is described in this section as 'set theory').

0.2.2 Operations on sets

The most fundamental operations on sets we may use is to *compare*[28] them.

Definition 5: subset

A set S is a **subset** of the set T, written

$$S \subseteq T,$$

if and only if every member of S is also an member of T.

In this situation we have

$$s \in S \qquad \text{implies} \qquad s \in T,$$

or

$$\text{for all } s \in S \qquad\qquad s \in T.$$

Note that the usage of key phrases such as 'implies', 'there exists', 'for all' is described in detail in Chapter 2.2.1.

If $S \subseteq T$ and $T \subseteq S$ then $S = T$ because they contain precisely the same members.

We often define subsets of sets we already know by identifying some particular property. The notation used for this is

$$\{s \in S \mid s \text{ has property } P\}.$$

This notation is explained in more detail in Section 0.2.3.

Definition 6: proper subset

We say that a set S **is a proper subset of the set** T if and only if

- S is a subset of T, that is $S \subseteq T$ and
- there exists a member $t \in T$ with $t \notin S$,

Sometimes the notations

$$S \subset T \qquad \text{or} \qquad S \subsetneq T$$

are used in this situation.

When we have sets we may build new sets by putting their combined members into one set, or by considering only those members contained in both sets. Because constructing new sets is non-trivial and may lead to problems it is usually better first to find an 'ambient' set that contains both the given sets.

Given a set X, for S and T subsets of X, we define

[28] A more general notion of comparisons between sets is studied in Section 5.2.

- their **union**, $S \cup T$, to be

  $$\{x \in X \mid x \in S \text{ or } x \in T\},$$

  which means that

  $$x \in S \cup T \qquad \text{if and only if} \qquad x \in S \text{ or } x \in T;$$

- their **intersection**, $S \cap T$, to be

  $$\{x \in X \mid x \in S \text{ and } x \in T\},$$

  which means that

  $$x \in S \cap T \qquad \text{if and only if} \qquad x \in S \text{ and } x \in T.$$

Note that we may now define the union or intersection of finitely many subsets of X by applying the operation to two sets at a time, that is, for example,

$$S_1 \cup S_2 \cup S_3 \cup \cdots \cup S_n = (\cdots((S_1 \cup S_2) \cup S_3) \cup \cdots \cup S_n).$$

Again we have used $\cdots$ here, and to be more precise we should adopt the mathematical notation

$$\bigcup_{i=1}^{n} S_i$$

instead, which spells out that we are forming the union of all the sets from S_1 to S_n.

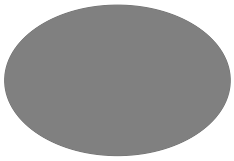
But in fact, given an arbitrary collection of subsets of X we may define their union and their intersection to obtain another subset of X. Let S_i be a subset of X for each $i \in I$, where I is an arbitrary set. Then

$$\bigcup_{i \in I} S_i = \{x \in X \mid \text{there is } i \in I \text{ with } x \in S_i\}$$

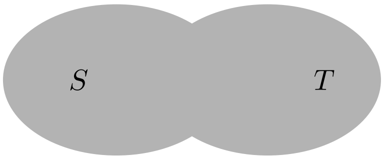
and

$$\bigcap_{i \in I} S_i = \{x \in X \mid \text{for all } i \in I \text{ we have } x \in S_i\}.$$

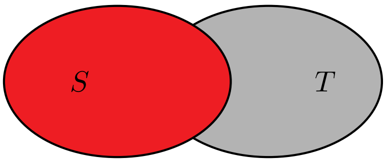
It is sometimes useful to draw such constructions in the form of a **Venn diagram**.



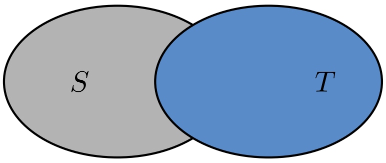
This is a picture of a generic set. The union of two generic sets, S and T, can then be drawn as follows.



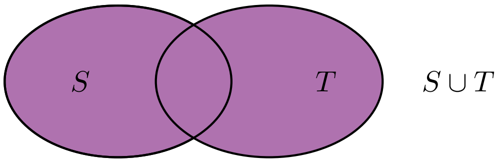
But this is a bit imprecise if we do not draw the boundaries of the sets, so it is more common to draw the boundaries of all the sets involved. We assume we have a set S, here shown in red,[29]



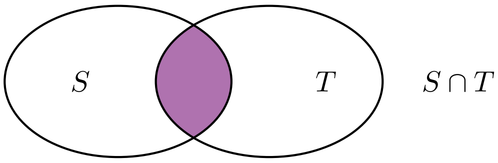
and a set T, here shown in blue



for which we form the union $S \cup T$ (here in purple).



The picture for the intersection, again drawn in purple.



Sometimes we care about the fact that two sets do not overlap.

Definition 7: disjoint

We say that two sets S and T are **disjoint** if and only if it is the case that

$$S \cap T = \emptyset.$$

There is one further important operation on sets.

Definition 8: complement relative to

Let S be a subset of a set X. The **complement of S relative to X**, $X \setminus S$, is given by

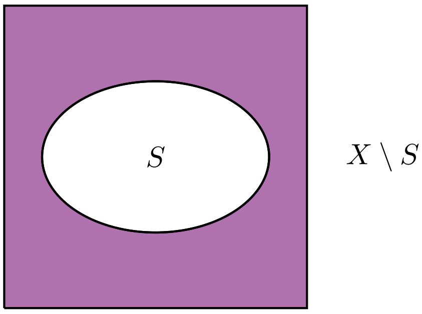
$$\{x \in X \mid x \notin S\}.$$

Some people write $X - S$ for this set, and some people write S' or $\overline{S}$. The latter two require that it is clearly understood which ambient set (here X) is meant. It has the advantage that some properties can be formulated very concisely in that notation. We do *not* use the primed version for complement in these notes—instead, we use it to give us variable names (so S, S' and S'' might be names for different sets).

[29] You will see the colours only in the electronic but not in the printed version.

Some of you have been taught that it is safe to write $\overline{S}$ for the complement of a set S because we somehow know in which set we are taking the complement. Always make it clear where you are taking your complements.

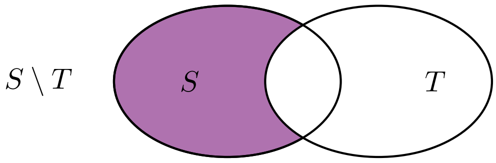
If we want to draw the complement then we *have* to draw the ambient set X. (We didn't have to do this for the examples so far.[30]) We do this by drawing a square, with S living inside the square.



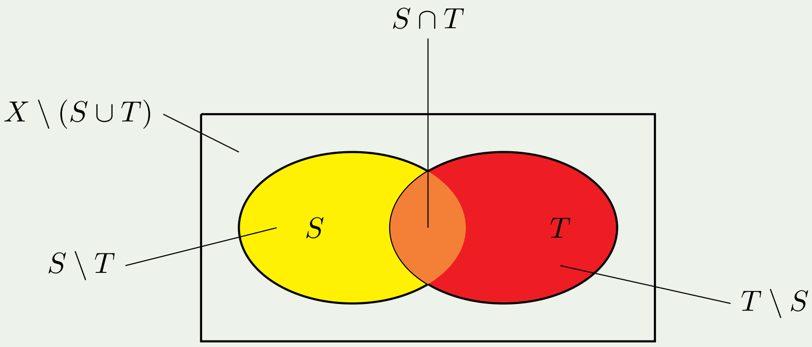
These are all the operations required to build new sets from given ones. It is now possible, for example, to define the **set difference**, $S \setminus T$, of all members of S that do not belong to T,

$$\begin{aligned} S \setminus T &= \{s \in S \mid s \notin T\} \\ &= S \cap (X \setminus T), \end{aligned}$$

drawn in purple below.

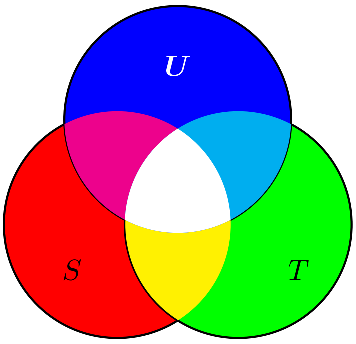


Example 0.7. We can use these operations to give names relative to S and T to all the regions in the following picture. This means we know how to determine the elements of all these regions, provided we know when an element is in S, and when it is in T.



[30] We could have drawn a box around the diagrams given above, but this doesn't really add anything.

If we have more than two sets to start with then there are many more sets one could describe, but we now have to tools to do so for all of them.



Exercise 6. Identify all regions in the above picture and give their description based on operations applied to S, T, and U.

Proofs involving sets are often quite simple. We give an example below.

Proposition 0.3

Let S, T and U be subsets of a set X. Then[31]

$$S \cap (T \cup U) = (S \cap T) \cup (S \cap U).$$

Proof. To show that two sets are equal we have to establish that all the elements of the first set occur in the second, and *vice versa*. Sometimes it is easier to give this as two separate proofs, and sometimes it can be done all in one go.

$$\begin{aligned} & S \cap (T \cup U) & \\ &= \{x \in X \mid x \in S \text{ and } x \in T \cup U\} & \text{def} \cap \\ &= \{x \in X \mid x \in S \text{ and } (x \in T \text{ or } x \in U)\} & \text{def} \cup \\ &= \{x \in X \mid (x \in S \text{ and } x \in T) \text{ or } (x \in S \text{ and } x \in U)\} & \text{see below} \\ &= \{x \in X \mid x \in S \cap T \text{ or } x \in S \cap U\} & \text{def} \cap \\ &= (S \cap T) \cup (S \cap U) & \text{def } \cup . \end{aligned}$$

The key step in the proof is the statement that, for $x \in X$, we have

$$x \in S \qquad \text{and} \qquad (x \in T \text{ or } x \in U)$$

if and only if

$$x \in S \text{ and } x \in T \qquad \text{or} \qquad x \in S \text{ and } x \in U.$$

We can make a case distinction: If

$$x \in S \qquad \text{and} \qquad x \in T \text{ or } x \in U$$

then at least one of

$$x \in S \qquad \text{and} \qquad x \in T$$

[31] This is known as a *distributivity law*, compare the last statement of Fact 1.

and

$$x \in S \qquad \text{and} \qquad x \in U$$

must hold, which justifies our original argument. Effectively we are applying here rules of logic which are explained in more detail in Chapter 3.

Alternatively we can show that the two sets are included in each other. We first show that $S \cap (T \cup U)$ is a subset of $(S \cap T) \cup (S \cap U)$.

$$\begin{aligned}
&x \in S \cap (T \cup U) && \\
&\text{implies } x \in S \text{ and } x \in T \cup U && \text{def} \cap \\
&\text{implies } x \in S \text{ and } (x \in T \text{ or } x \in U) && \text{def} \cup \\
&\text{implies } x \in S \text{ and one of } (x \in T \text{ or } x \in U) && \\
&\text{implies } (x \in S \text{ and } x \in T) \text{ or } (x \in S \text{ and } x \in U) && \text{see above} \\
&\text{implies } x \in S \cap T \text{ or } x \in S \cap U && \text{def} \cap \\
&\text{implies } x \in (S \cap T) \cup (S \cap U) && \text{def } \cup .
\end{aligned}$$

Next we show that $(S \cap T) \cup (S \cap U)$ is a subset of $S \cap (T \cup U)$.

$$\begin{aligned}
&x \in (S \cap T) \cup (S \cap U) && \\
&\text{implies } x \in S \cap T \text{ or } x \in S \cap U && \text{def} \cup \\
&\text{implies } (x \in S \text{ and } x \in T) \text{ or } (x \in S \text{ and } x \in U) && \text{def} \cap \\
&\text{implies in either case we have } x \in S, \text{ and we must also have} && \\
&\qquad\qquad \text{at least one of } x \in T \text{ or } x \in U && \\
&\text{implies } x \in S \text{ and } x \in T \cup U && \text{def} \cup \\
&\text{implies } x \in S \cap (T \cup U) && \text{def } \cap .
\end{aligned}$$

EExercise 7. Assume that S and T are subsets of a set X.

(a) Show that the complement relative to X of the union of S and T is the intersection of the complements (relative to X) of S and T. *Hint: Turn the sentence into an equality of sets. Look at the proof of Proposition 0.3 for an example how to prove that two sets are equal.*

(b) Show that the union of two sets may be written using only the complement and the intersection operations. *Hint: Use your equality from the previous part.*

(c) Give an argument that we may describe precisely the same sets using ($\cup$, $\cap$ and $\setminus$) as using ($\cap$ and $\setminus$).

A useful operation assigns to a finite set the number of elements in that set, which is written as[32][33]

$$S \longmapsto |S|.$$

For all sets of numbers we have useful operation that allows us to extract the smallest/largest number from a set, provided it exists, which is always the case if the set is finite.

[32] Some texts may use $\#S$ instead.

[33] If you are not familiar this notation to describe a function come back to this once you have read Section 0.3.

Given a set S of numbers we write

$$\min S$$

for the smallest number in S if it exists, and

$$\max S$$

for the largest number in S if it exists.

Example 0.8. We have that

$$\min\{1, 2, 3, \pi\} = 1$$

and

$$\max\{1, 2, 3, \pi\} = \pi,$$

and

$$\min[0, 1] = 0,$$

while

$$\max[0, 1] = 1.$$

Note, however, that

$$\min(0, 1) \qquad \text{and} \qquad \min \mathbb{R}$$

are not defined, and that the same is true for

$$\max(0, 1) \qquad \text{and} \qquad \max \mathbb{R}.$$

0.2.3 Describing sets

Describing sets precisely is harder than it may sound. If a set has finitely many elements then, in principle, we could list them all. But if there are a lot of them this is rather tedious and time-consuming. People often resort to using ... to indicate that there are members that are not explicitly named, and they hope that it is clear from the context what those members are. Take for example

$$\{0,\ 1,\ 2,\ \ldots,\ 100,000\}.$$

But whenever this notation is used there is room for confusion. It is *much* better to give a more precise description such as

$$\{n \in \mathbb{N} \mid\ n \leq 100,000\}.$$

The idea behind this kind of description is that one describes the set in question as a *subset of a known set* (here $\mathbb{N}$), consisting of all those members satisfying a particular *property* (here being less than or equal to $100,000$). In logic such a property is known as a *predicate*. It is almost inevitably the case that any set we might want to describe is a subset of a set already known, so this technique works remarkably often.

In general the format is to have a known set S and to define

$$\{s \in S \mid\ s \text{ has property } P\}.$$

Example 0.9. Let's assume we want to describe the set of even natural numbers. We could write

$$\{0,\,2,\,4,\,6,\ldots\},$$

but this leaves it to the reader to make precise which elements belong to the set and which ones don't. This is strongly discouraged. Instead we could write the preferable

$$\{n\in\mathbb{N}\mid\ n \text{ even}\},$$

but that assumes that the reader knows how the even property is defined. If we want to leave no room for doubt we could apply the definition (see Definition 4) and write

$$\{n\in\mathbb{N}\mid\ n \bmod 2=0\}.$$

This makes it precise which members belong to our set—indeed, it gives us a test that we can apply to some given natural number to see whether it belongs to our set.

Example 0.10. Because we may form intersections and unions of sets we may also specify sets consisting of all those elements which have more than one property. All even numbers up to $100,000$ could be described as an intersection, namely

$$\{n\in\mathbb{N}\mid\ n \bmod 2=0\}\cap\{n\in\mathbb{N}\mid\ n\leq 100,000\},$$

but it is more customary instead to combine both properties by using 'and', that is

$$\{n\in\mathbb{N}\mid\ n \bmod 2=0\quad\text{and}\quad n\leq 100,000\}.$$

When looking at the real numbers there is a standard way of defining subsets which give a contiguous part of the real line:

$$[x,y]=\{r\in\mathbb{R}\mid\ x\leq r\leq y\}$$

and

$$(x,y)=\{r\in\mathbb{R}\mid\ x<r<y\},$$

or

$$[x,y)=\{r\in\mathbb{R}\mid\ x\leq r<y\},$$

but also

$$(-\infty,y]=\{r\in\mathbb{R}\mid\ r\leq y\}.$$

Sets of this form are known as 'real intervals'. Note that we use the notation

$$\mathbb{R}^+=[0,\infty),$$

for non-negative real numbers.

We may also use the idea of defining sets using properties to describe all those elements of a given set which satisfy at least one of several properties.

Example 0.11. An example of this idea is given by

$$\{n\in\mathbb{N}\mid\ n \bmod 2=0\quad\text{or}\quad n \bmod 2=1\},$$

which is the union of two sets, namely

$$\{n \in \mathbb{N} \mid n \bmod 2 = 0\} \cup \{n \in \mathbb{N} \mid n \bmod 2 = 1\},$$

and this set is equal to $\mathbb{N}$.

Example 0.12. It is also possible to use this idea to specify the elements that *do not* have a particular property. The odd natural numbers are those that are not even.

$$\begin{aligned}\{n \in \mathbb{N} \mid n \bmod 2 \neq 0\} &= \mathbb{N} \setminus \{n \in \mathbb{N} \mid n \bmod 2 = 0\} \\ &= \{n \in \mathbb{N} \mid n \bmod 2 = 1\}.\end{aligned}$$

Example 0.13. If we want to describe the rational numbers as a subset of $\mathbb{R}$ we may use

$$\{r \in \mathbb{R} \mid \text{there exists } m \text{ and } n \text{ in } \mathbb{Z} \text{ such that } r = m/n\},$$

Example 0.14. Nothing stops us from specifying

$$\{n \in \mathbb{N} \mid n \bmod 2 = 0 \quad \text{and} \quad n \bmod 2 = 1\},$$

which is a rather complicated description of the empty set.

It is possible to use infinitely many restricting properties.

Example 0.15. Given a natural number k, the multiples of k can be written as

$$\{n \in \mathbb{N} \mid n \bmod k = 0.\}.$$

So the set of natural numbers which are *not* multiples of k is

$$\{n \in \mathbb{N} \mid n \bmod k \neq 0\}.$$

Example 0.16. A more complicated question is how to describe the set of all prime numbers.

For that it helps to consider the set of elements which are *not* a multiple of any number other than one and themselves, which is equivalent to saying that they are not a multiple of *any* number with a factor of at least 2.

The set of all multiples of k in $\mathbb{N}$, with a factor of two or greater, is given by

$$\{n \in \mathbb{N} \mid n \bmod k = 0,\, n \operatorname{div} k \geq 2.\},$$

and the set of all numbers which are *not* such a multiple is

$$\mathbb{N} \setminus \{n \in \mathbb{N} \mid n \bmod k = 0,\, n \operatorname{div} k \geq 2,\}.$$

which is the same as

$$\{n \in \mathbb{N} \mid n \bmod k \neq 0 \text{ or } (n \bmod k = 0 \text{ and } n \operatorname{div} k = 1.\},$$

which is the same as

$$\{n \in \mathbb{N} \mid n \bmod k \neq 0 \text{ or } n = k\}.$$

Note that all these sets contain the number 1, which we would like to exclude from the set of prime numbers.

This suggests that we can use the intersection of all these sets of non-multiples of k, where $k \in \mathbb{N} \setminus \{0, 1\}$, to express the prime numbers. This set is given by

$$\bigcap_{k \in \mathbb{N} \setminus \{0,1\}} \{n \in \mathbb{N} \setminus \{1\} \mid n \bmod k \neq 0 \text{ or } n = k\}$$

Instead of restricting the elements of a known set to describe a new set it is sometimes possible instead to provide instructions for *constructing the elements* of the new set. This is the second important technique for describing sets.

Example 0.17. An alternative way of describing the even numbers is to recognize that they are exactly the multiples of 2, and to write

$$\{2n \mid n \in \mathbb{N}\}.$$

The odd numbers may then be described as

$$\{2n + 1 \mid n \in \mathbb{N}\}.$$

But for a better answer, we should add something here. Read on to find out what.

We can think of this as *constructing* a new set, but usually this only makes sense when describing a subset of a previously known set. Certainly the notation assumes that we know what we mean by $2n$, or $2n + 1$—this implies we know where the addition and multiplication operations that appear in these expressions are to be carried out. In this examples it is in $\mathbb{N}$, so it would be better to write

$$\{2n + 1 \in \mathbb{N} \mid n \in \mathbb{N}\}.$$

This may seem obvious, since $\mathbb{N}$ is explicitly named as the set n belongs to, but assume that in order to describe the rational numbers we wrote

$$\{m/n \mid (m, n) \in \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})\}.$$

But what does this mean? Where do we take m/n? This is not defined in the collection $\mathbb{Z}$ of numbers where m and n live, and so it is not clear what we mean here. Maybe this is a set of formal fractions? We could clarify this by writing

$$\{m/n \in \mathbb{R} \mid (m, n) \in \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})\},$$

from which it is clear that we mean to collect all the results of calculating $m \cdot n^{-1}$ within the real numbers.

Example 0.18. To describe the integer multiples of π (for example if we want to have all the points on the real line for which sin takes the value 0) we might write,

$$\{n\pi \mid n \in \mathbb{Z}\}.$$

Again we have to deduce from the context where $n\pi$ is meant to be carried out. If we write

$$\{n\pi \in \mathbb{R} \mid n \in \mathbb{Z}\},$$

then everything is made explicit.

In general what we have done here is to assume that we have two known sets, say S and T, and a way of producing elements from the second set from the first, using a function

$$f\colon S \longrightarrow T.$$

We then write

$$\{fs \in T \mid s \in S\}$$

for the set of all elements of T which are 'generated' by elements of S using the function f.

We are using the notation $\{\ldots \mid \ldots\}$ in two ways that look different, but we can think of the statement $s \in S$ as a property so this notation is not inconsistent. One could even combine the two ideas.

You should think of the vertical line as saying 'such that', so

$$\{n \in \mathbb{Z} \mid n \text{ is even}\}$$

can be pronounced as

the set of all n in $\mathbb{Z}$ such that n is even

and

$$\{2n \in \mathbb{Z} \mid n \in \mathbb{Z}\}$$

can be pronounced as

the set of all those $2n$ in $\mathbb{Z}$ for which n is in $\mathbb{N}$.

Note that these two sets are not equal!

CExercise 8. For the sets given below, give a description using a predicate (as in Example 0.9), and also give a description where you generate the set (as in Example 0.17).

(a) Describe the set of all integers that are divisible by 3.

(b) Describe the set of all integers that are divisible by both, 2 and 3.

(c) Describe the set of all integers that are divisible by 2 or by 3. To generate this set you need to use the union operation.

(d) Describe the set of all integers that are divisible by 2 or by 3 but not by 6. To generate this set you need to use the union, and the relative complement, operations.

(e) Describe the set of all real numbers r for which $\cos r = 0$.

0.2.4 Constructions for sets

There is one other fairly common construction for sets.

Definition 9: product of two sets

Given sets X and Y their[34] **product**,

$$X \times Y,$$

is the set

$$\{(x, y) \mid x \in X \text{ and } y \in Y\}.$$

This means that the elements of the product are pairs whose first component is an element of X and whose second component is an element of Y. Products of sets appear in many places, and the examples we give below barely scratch the surface.

Example 0.19. The product of the set $\{0, 1\}$ with itself is the set with the elements

$$(0, 0),\ (0, 1),\ (1, 0),\ (1, 1),$$

so

$$\{0, 1\} \times \{0, 1\} = \{(0, 0),\ (0, 1),\ (1, 0),\ (1, 1)\}.$$

Example 0.20. A more familiar example is a deck of cards: You have four suits, clubs ♣, spades ♠, hearts ♡ and diamonds ♢, and you have standard playing cards, say 7, 8, 9, 10, J, Q, K, A in a 32-card deck. Each of those cards appears in each of the suits, so you have four Queens, one each for clubs, spaces, hearts and diamonds. In other words, your 32 card deck can be thought of as the product

$$\{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\} \times \{7, 8, 9, 10, J, Q, K, A\}.$$

We can picture the result as all combinations of elements from the first set with elements from the second set. The accepted standard for describing cards is to first give the value and then the suit, so in the table below

$$9\diamondsuit$$

is the notation used for the element

$$(\diamondsuit, 9)$$

of our product set.

	7	8	9	10	J	Q	K	A
♣	7♣	8♣	9♣	10♣	J♣	Q♣	K♣	A♣
♠	7♠	8♠	9♠	10♠	J♠	Q♠	K♠	A♠
♡	7♡	8♡	9♡	10♡	J♡	Q♡	K♡	A♡
♢	7♢	8♢	9♢	10♢	J♢	Q♢	K♢	A♢

Whenever you draw the graph of a function from $\mathbb{R}$ to $\mathbb{R}$ you do so in the product of the set $\mathbb{R}$ with itself: You use the x-axis to give the source of the function, and the y-axis for the target, and you then plot points with coordinates (x, fx), where x varies through the source set.

[34] This is also known as their **Cartesian product**.

Example 0.21. A very important set that is a product is the *real plane*

$$\mathbb{R} \times \mathbb{R} = \mathbb{R}^2 = \{(r, r') \mid r, r' \in \mathbb{R}\}.$$

This is the set we use when we draw the graph of a function from real numbers to real numbers (see Section 0.3.4), where we use the first coordinate to give the argument, and the second argument to give the corresponding value.

Example 0.22. Similarly, the n-dimensional vector space based on $\mathbb{R}$ has as its underlying set the n-fold product of $\mathbb{R}$ with itself,

$$\underbrace{\mathbb{R} \times \mathbb{R} \times \cdots \times \mathbb{R}}_{n \text{ times}} = \mathbb{R}^n = \{(r_1, r_2, \dots, r_n) \mid r_1, r_2, \dots r_n \in \mathbb{R}\}.$$

Note that it is possible to recover the components of an element of a product: We have two functions,[35]

$$\pi_1 \colon X \times Y \longrightarrow X$$

and

$$\pi_2 \colon X \times Y \longrightarrow Y,$$

known as the *projection functions* with the behaviour that, for all $(x, y) \in X \times Y$ we have

$$\pi_1(x, y) = x \qquad \text{and} \qquad \pi_2(x, y) = y.$$

In general, if S is a set, people often write

$$S^2$$

for

$$S \times S$$

and more generally,

$$S^n$$

for the n-fold product of S with itself. The elements of this set can be described as n-tuples of elements of S, that is

$$S^n = \{(s_1, s_2, \dots s_n) \mid s_i \in S \text{ for } 1 \leq i \leq n\}.$$

Note that here we construct a new set, and we define what the elements of the set are (namely pairs of elements of the given sets) and we do not have to identify an ambient set.

In Section 2.5 we describe operations on sets as functions (see Sections 0.3) and for that we require the product construction. A **binary operation**[36] is one that takes two elements from a set, and returns one element of the same set.

Example 0.23. Addition for the natural numbers $\mathbb{N}$ is a binary operation on the set $\mathbb{N}$. As a function (see Section 0.3) it takes two elements, say x and y, of $\mathbb{N}$, that is

$$\text{an element } (x, y) \text{ of } \mathbb{N} \times \mathbb{N},$$

[35] You may want to come back to this after reading Section 0.3.

[36] That is for example an operation which takes two numbers and returns a number.

and returns

$$\text{an element } x + y \text{ of } \mathbb{N}.$$

The type of this operation is

$$\mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}.$$

But, of course, we may also consider addition for different sets of numbers, giving operations, for example

$$\mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{Z} \qquad \mathbb{Q} \times \mathbb{Q} \longrightarrow \mathbb{Q} \qquad \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}.$$

Another general operation sometimes applied to sets is the **disjoint union** but we do not describe this here.

Definition 10: powerset

Given a set X, its powerset $\mathcal{P}X$, is given by the set of all subsets of X,.

$$\mathcal{P}X = \{S \mid S \subseteq X\}.$$

All our operations on sets were defined for elements of such a powerset. For example, given an element S of $\mathcal{P}X$, which is nothing but a subset of X, we have $X \setminus S$, the complement of S with respect to X, which is another element of $\mathcal{P}X$.

Example 0.24. We may think of the union operation as taking two elements of $\mathcal{P}X$, and returning[37] an element of $\mathcal{P}X$, so we would write that as

$$\cup \colon \mathcal{P}X \times \mathcal{P}X \longrightarrow \mathcal{P}X,$$

with the assignment given by

$$(S, T) \longmapsto S \cup T \ ,$$

that is, given the arguments S and T in $\mathcal{P}X$ the function returns their union, $S \cup T$.

Because there are so many operations on the powerset it turns out to be a useful model for various situations. In the material on logic we see how to use it as a model for a formal system in logic.

Sometimes we care only about the finite subsets of a set, that is

$$\{S \subseteq X \mid S \text{ is finite}\}.$$

People sometimes call this ‘the finite powerset’, but that is a bit problematic since this often isn’t itself a finite set.

0.3 Functions

One could argue that sets are merely there to allow us to talk about functions, and while this is exaggerated sets wouldn’t be much use without the ability to move between them.

[37] You may want to come back to this example after reading Section 0.3.

0.3.1 Function, source, target, range

A function is a way of turning entities of one kind into those of another. Formally a **function**

$$f\colon S \longrightarrow T$$

is given by

- a **source set** S
- a **target set** T and
- an instruction that turns every element s of S into an element fs of T, often[38] written as

  $$s \longmapsto fs.$$

Many people allow giving functions without specifying the source and target sets but this is sloppy. Every function has a type, and for our example f here the type is $S \to T$.

Some instructions can be used with multiple source and target sets. For example

$$x \longmapsto 2x$$

may be used to define a function

- $\mathbb{N} \to \mathbb{N}$,
- $\mathbb{Z} \to \mathbb{Z}$,
- $\mathbb{Q} \to \mathbb{Q}$,
- $\mathbb{R} \to \mathbb{R}$.

and

$$x \longmapsto x^2$$

could have the types (among others)

- $\mathbb{N} \to \mathbb{N}$,
- $\mathbb{Q} \to \mathbb{Q}$ or $\mathbb{Q} \to \mathbb{Q}^+$,
- $\mathbb{R} \to \mathbb{R}$ or $\mathbb{R} \to \mathbb{R}^+$.

0.3.2 Composition and identity functions

Which functions we can define from one set to another depends on the structure of the sets, and on any known operations on the sets. Only one (somewhat boring) function is guaranteed to exist for every set S.

[38] It is quite often standard to write $f(s)$ but as long as the argument is not a complicated expression this is unnecessary.

Definition 11: identity function

The identity function id_S on a set S given by the assignment

$$\begin{aligned}\mathsf{id}_S\colon S &\longrightarrow S\\ s &\longmapsto s.\end{aligned}$$

An important operation on functions is given by carrying out one function after another.

Definition 12: composite of two functions

Given two functions

$$f\colon S \longrightarrow T \qquad \text{and} \qquad g\colon T \longrightarrow U$$

where the target of f is the source of g, the composite **of f and g**

$$\begin{aligned}g\circ f\colon S &\longrightarrow U\\ a &\longmapsto g(fa),\end{aligned}$$

is defined by first applying f to s and then g to the result, that is

$$\begin{tikzcd} s \arrow[r, mapsto] & fs \arrow[r, mapsto] & gfs \\ S \arrow[r] & T \arrow[r] & U, \end{tikzcd}$$

and so overall we map $s \in S$ to $gfs \in U$.

Composition allows us to build more complicated functions from simple ones.

Example 0.25. One may think of a linear function on $\mathbb{R}$, of the form

$$x \longmapsto mx + b$$

to be the result of composing the following two functions $\mathbb{R} \longrightarrow \mathbb{R}$:

$$f\colon x \longmapsto mx \qquad \text{and} \qquad g\colon x \longmapsto x + b\ ,$$

since this amounts to calculating

$$\begin{tikzcd} x \arrow[r, mapsto, "f"] & mx \arrow[r, mapsto, "g"] & mx + b \\ \mathbb{R} \arrow[r] & \mathbb{R} \arrow[r] & \mathbb{R}. \end{tikzcd}$$

In other words, if we apply the function f to x, and the function g to the result, we find that overall x is mapped to $mx + b$.

Example 0.26. You probably have used the notion of a composite already. You may find it easier to realize this by looking at the assignment

$$x \longmapsto \sqrt{|\sin x|}$$

from $\mathbb{R}$ to $\mathbb{R}$. This tells you to first apply the sine function to x, and to apply the square root function to the absolute of the result. The idea of composing functions just makes this explicit, and it also forces you to ensure that the output of the first function is always a valid input to the second function.

Hence we may express the given function as the composite of the following functions:

$$\begin{array}{rcl} f\colon \mathbb{R} & \longrightarrow & \mathbb{R} \\ x & \longmapsto & \sin x \end{array} \qquad \begin{array}{rcl} g\colon \mathbb{R} & \longrightarrow & \mathbb{R}^+ \\ x & \longmapsto & |x| \end{array} \qquad \begin{array}{rcl} h\colon \mathbb{R}^+ & \longrightarrow & \mathbb{R} \\ x & \longmapsto & \sqrt{x} \end{array}$$

in the sense that the given function is

$$h \circ g \circ f$$

which means that the assignment given is the same as

$$x \longmapsto h(g(fx)).$$

Note that we could have specified a different target for the sine function, such as the real interval $[-1, 1]$, and made that the source of the following function.

In order to define a function you *have to* specify its source and target. Don't forget to do this.

CExercise 9. Define three functions such that their composite is a function $\mathbb{R} \longrightarrow \mathbb{R}$ which maps an input to the logarithm (for base 2) of the result of adding 2 to the negative of the square of the sine of the input. *Hint: To define a function you need to give its source and target. You need to make sure that your functions can be composed.*

0.3.3 Basic notions for functions

It can sometimes be useful to determine which part of the target set is reached by a function.

Definition 13: image, range of a function

Given a function $f\colon S \longrightarrow T$, for $s \in S$ we say that fs is **the image of** s **under** f, and the set

$$\{fs \in T \mid s \in S\}$$

is the **range of f**. It is also known as the **image of the set** S, and written $f[S]$.

Note that we may also write the range of f, which can also be thought of as image of the set S under the function f, by using a property of elements of T as

$$\{t \in T \mid \text{ there exists } s \in S \text{ with } fs = t\}.$$

Example 0.27. For the sine function

$$\sin\colon \mathbb{R} \longrightarrow \mathbb{R},$$

the image of 0 under $\sin$ is $0 = \sin 0$, and the range of $\sin$ is the set

$$[-1, 1].$$

Example 0.28. If we formally want to define the notion of a **sequence** of, say, real numbers, then we should do so as a function a from $\mathbb{N}$ to $\mathbb{R}$. The nth member of the sequence is given by an. In such cases an is often written a_n. For example, the sequence given on page 13 would have the first few values

argument	0	1	2	3	4
value	1	$1/2$	$1/4$	$1/8$	$1/16$,

and the formal definition of this function is

$$\begin{aligned} \mathbb{N} &\longrightarrow \mathbb{R} \\ n &\longmapsto \frac{1}{2^n}. \end{aligned}$$

We may also think of a function as translating from one setting to another. In Java *casting* allows us to take an integer, `int` and cast it as a floating point number, `float`. This is effectively a function which takes an `int` (which amounts to a number of bits) and translates it into what we think of as the same number, but now expressed in a different format. Similarly in Python it is possible to 'convert' numbers of one type into another, for example `float(x)` takes a number in a different format, for example an integer, and converts it into a floating point number.

For a mathematical example, we note that we have functions connecting all our sets of numbers since

$$\mathbb{N} \text{ is embedded in } \mathbb{Z} \text{ which is embedded in } \mathbb{Q} \text{ which is embedded in } \mathbb{R}.$$

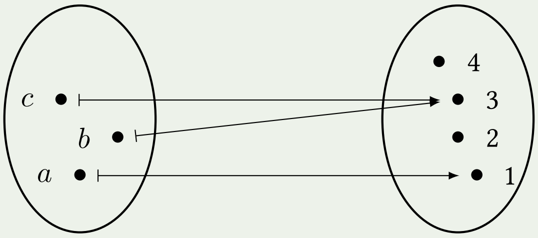
All these embeddings are functions, but they are so boring that we don't usually bother to even name them. For a slightly more interesting example take the set of all fractions. From there we have a function that maps a fraction to the corresponding rational number (and so $1/2$ and $2/4$ are mapped to the same number), allowing us to translate from the presentation as fraction to the numbers we are really interested in.

If you have a customer database you could print a list of all of your customers. You have effectively constructed a function that takes an entry in your database and maps it to the name field. Note that if you have two customers called John Smith then that name will be printed twice, so thinking of a 'set of names' is not entirely appropriate here.

If we have small finite sets then one can define a function in a graphical way, by showing which element of the source set is mapped to which element of the target set. We give an example of this below.

Example 0.29. We draw a function

$$\{a, b, c\} \longrightarrow \{1, 2, 3, 4\}.$$



This function maps a to 1 and b and c to 3. Note that in order for such a diagram to describe a function, every element of the source set must be mapped to precisely one element of the target set.

0.3.4 The graph of a function

It can be useful to think of a function via its graph.

Definition 14: graph of a function

The **graph of a function** is the set of pairs consisting of an element of the source set with its image under the function,[39] that is, given

$$f\colon S \longrightarrow T$$

its graph is the set

$$\{(s, fs) \in S \times T \mid s \in S\}.$$

We can see what this definition means by assuming we are given a function

$$f\colon S \longrightarrow T$$

and noting that this definition tells us that is graph is the set

$$\{(s, fs) \in S \times T \mid s \in S\}$$

which is a subset of the product of S and T.

See Proposition 2.1 for a characterization of those subsets of $S \times T$ which are the graph of a function from S to T.

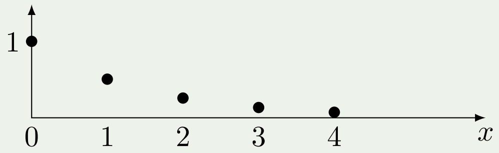
When we have functions between sets of numbers we can draw a picture of their graph.

Example 0.30. Let's return to the function from Example 0.28, which is given by

$$\begin{aligned} \mathbb{N} &\longrightarrow \mathbb{R} \\ x &\longmapsto \frac{1}{2^x}. \end{aligned}$$

[39] And indeed, a standard way of defining functions in set theory is via their graphs.

Its graph can be drawn as follows.



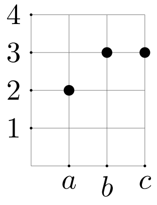
More examples appear in the following section.

For functions between finite sets drawing the graph in this way is usually not particularly useful. The graph of the finite example

above is

$$\{(a,2),\ (b,3),\ (c,3)\},$$

and one might draw it as follows:



This does not really show anything that is not visible in the previous diagram.

0.3.5 Important functions

When we are interesting in judging how long a computer program will take we typically count the number of instructions that will have to be carried out. How many instructions these are will, of course, depend on the program, but also on the *particular input* we are interested in. Often the inputs to a program can be thought of as having a particular size: For example, sorting five variable of type **int** will be quite different from doing so for one million such variables.

Typically the number of instructions a program has to carry out depends on the *size* of the input rather than the actual input, and so we can think of this as defining a function from $\mathbb{N}$ to $\mathbb{N}$ which takes the size of the input to the number of instructions that are carried out.

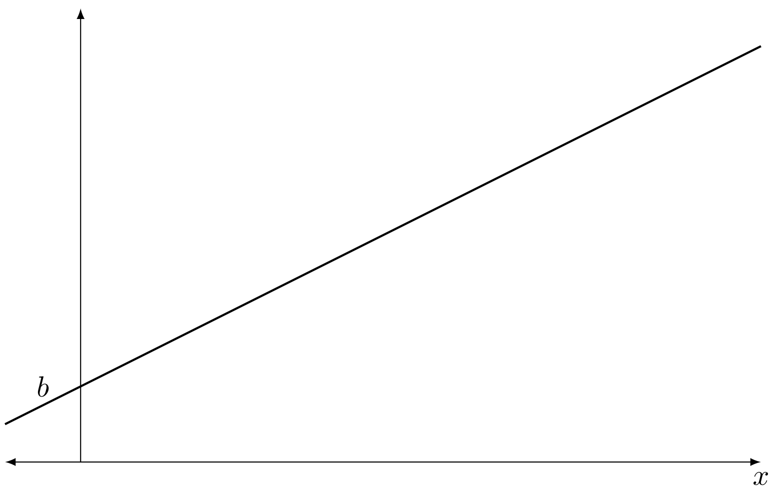
There are a number of functions that typically appear in such considerations.[40] In computer science it would be sufficient for these purposes to consider these functions as going from $\mathbb{N}$ to $\mathbb{N}$, but it is often more convenient to draw their graphs as functions from $\mathbb{R}^+$ to $\mathbb{R}^+$. In what follows we consider functions that commonly appear in that setting, and where possible we draw their graph as functions from $\mathbb{R}$ to $\mathbb{R}$.

There are linear functions, which are of the form

$$\begin{aligned} \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto mx+b \end{aligned}$$

and their graphs look like this.

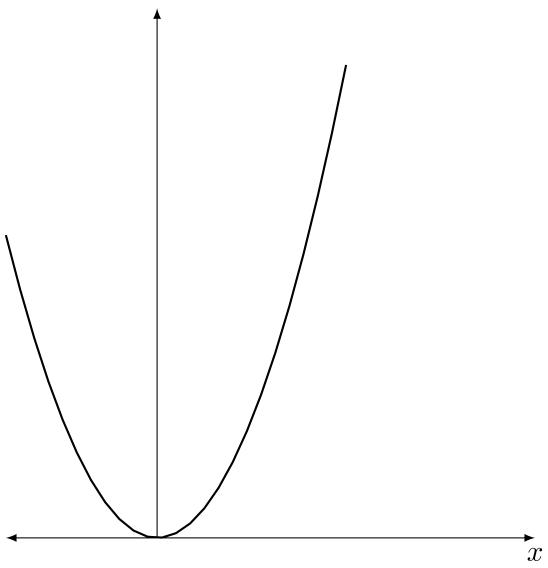
[40] You will meet them again when you look at this in more detail in COMP11212 and COMP26120.



A typical quadratic function is given by

$$\begin{aligned} \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto ax^2 + bx + c \end{aligned}$$

and (for some values of a, b and c) its graph looks like this:



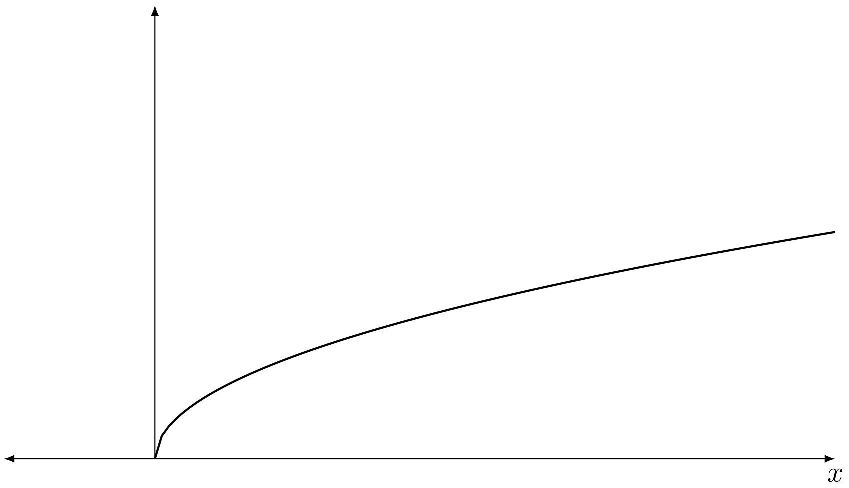
Other polynomial functions, that is functions of the form

$$\sum_{i=1}^{n} a_i x^i$$

may also feature.

Sometimes we wish to consider functions which involve the argument being taken to a power other than a natural number, for example

$$\begin{aligned} \mathbb{R}^+ &\longrightarrow \mathbb{R}^+ \\ x &\longmapsto x^{1/2} = \sqrt{x}. \end{aligned}$$



Some of these functions are defined for non-negative numbers only, so their source is $\mathbb{R}^+$, rather than all of $\mathbb{R}$. Note that for fixed $x \in \mathbb{R}^+$ this function only gives the *positive* solution of the equation

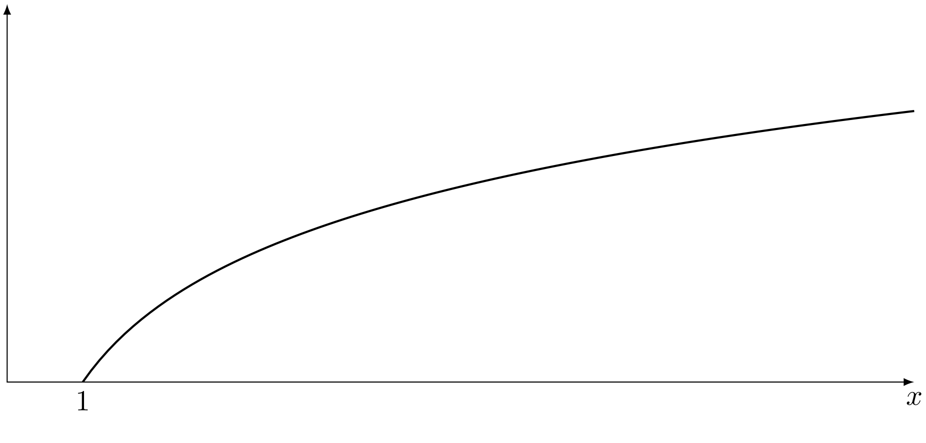
$$y = x^2.$$

If you want to refer to both of these[41] you have to write $\pm\sqrt{x}$.

Apart from these polynomial functions, important examples that come up in computer science are concerned with logarithmic functions. In computer science one typically wishes to use logarithms to base 2. They are typically written as

$$\begin{array}{rcl} [1, \infty) & \longrightarrow & \mathbb{R}^+ \\ x & \longmapsto & \log x \end{array}$$

and look like this.



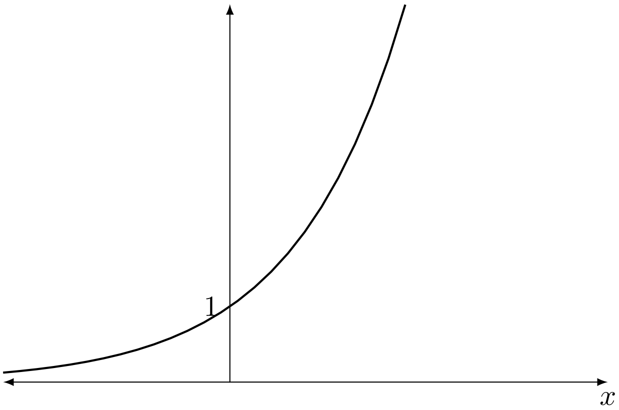
And then there are exponential functions. Because of the speed with which these grow having a program whose number of instructions is exponential in the size of the problem is a serious issue since it means that it is not feasible to

[41] If you have been taught otherwise then this is at odds with notation used at university level and beyond.

calculate solutions for larger problem sizes using this program. It is fairly usual to use 2 as a base once again. The function in question is

$$\begin{aligned} \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto 2^x \end{aligned}$$

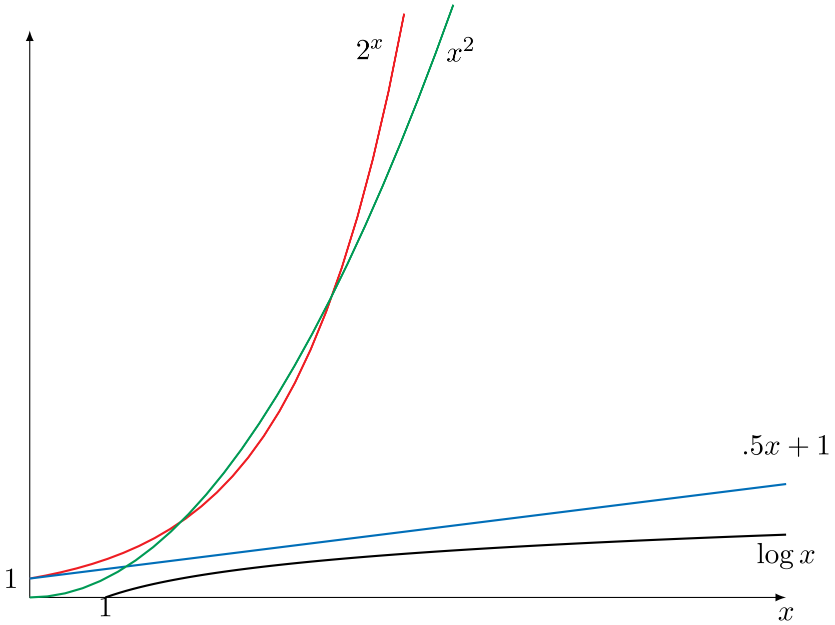
and its graph is even steeper than that of the quadratic curve above.



In all these cases typically the shape of the curve is more important than any parameters involved in defining it—so knowing that we have a quadratic function is very useful, whereas there is little added benefit in knowing a, b and c in $ax^2 + bx + c$.

If one has a problem size of 1,000,000, for example, then it is important to know how fast the function grows to see how many instructions will have to be carried out for that size (and so how long it will take for the program to finish, or it if is possible for this program to finish at all).

If we draw the functions from above in the same grid (note that we have compressed the y-axis here) we can compare them.



The issue of how to compare functions when we are only interested in how they do for large inputs is discussed in Section 5.1, and this is relevant for calculating the *complexity* of a programme or algorithm.

Note that Fact 7 gives us a lot of material when it comes to comparing numbers that we can use to also compare functions:

Example 0.31. Let us consider the following functions:

$$\begin{aligned} f\colon [1,\infty) &\longrightarrow \mathbb{R} & \qquad g\colon [1,\infty) &\longrightarrow \mathbb{R} \\ x &\longmapsto x^2 & x &\longmapsto x^3. \end{aligned}$$

We can show that for all $x \in [1,\infty)$ we have that

$$x^2 \le x^3,$$

and so

$$fx \le gx:$$

Given such an x we have that

$$\begin{aligned} x^2 &= 1 \cdot x^2 && 1 \text{ unit for mult} \\ &\le x \cdot x^2 && 1 \le x, \text{ Fact 7} \\ &= x^3. \end{aligned}$$

When we come to comparing functions in Section 5.1 you will find the following comparison for functions helpful.

Fact 8

We have that for all $n \in \mathbb{N}$ that

$$2^n \ge n+1$$

as well as

$$n \ge \log(n+1).$$

This statement is formally shown in Exercise 170.

Two functions which are useful when we need to convert results which are real or rational into integers.

The *floor function*

$$\begin{aligned} \mathbb{R} &\longrightarrow \mathbb{Z} \\ x &\longmapsto \lfloor x \rfloor \end{aligned}$$

maps a real number x to the greatest integer less than or equal to it. See Example 4.70 if you want to find out how to draw a graph for a function like this.

The *ceiling function*

$$\begin{aligned} \mathbb{R} &\longrightarrow \mathbb{Z} \\ x &\longmapsto \lceil x \rceil \end{aligned}$$

maps a real number x to the smallest integer greater than or equal to it.

0.3.6 Functions with several variables

You may have been taught about functions with several variables as being somehow more general then functions with one variable. However, this is not really the case.

If we have a function whose source set is a product set, for example

$$f\colon \mathbb{R}^2 \longrightarrow \mathbb{R}$$

then every argument for this function is a *pair*, because every element of $\mathbb{R}^2$ is a pair. It may be useful to have access to the two components of the argument, and so it is fairly common to write something like

$$f(x,y) = x + y$$

to describe the behaviour of the function f.

If we had insisted of using

$$z \in \mathbb{R}^2$$

to describe the argument of f then we would have to write[42]

$$fz = \pi_1 z + \pi_2 z,$$

which is much less clear.

So, a function with several arguments is a function whose source set is a product, and where we have written the argument to have as many components as the product set has factors. Examples 0.23 and 0.24 talk about functions with two arguments and you should go back to them and look at them once more now.

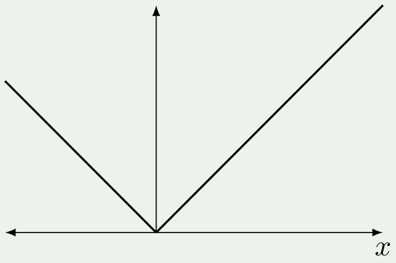
0.3.7 Constructions for functions

An important way of constructing new functions from old ones is what is known as **definition by cases**. What this means is that one pieces together different functions to give a new one.

Example 0.32. Assume we want to give a proper definition of the 'absolute' function

$$|\cdot|\colon \mathbb{R} \longrightarrow \mathbb{R}^+$$

for real numbers. The value it returns depends on whether the input is negative, or not. The graph of this function is depicted here.



We can write the corresponding assignment as

$$x \longmapsto \begin{cases} x & x \geq 0 \\ -x & \text{else.} \end{cases}$$

[42] Recall the projection functions π_1 and π_2 from Section 0.2.4.

Example 0.33. If you want to give an alternative description of the function

$$\begin{aligned} \mathbb{Z} &\longrightarrow \mathbb{Z} \\ x &\longmapsto x \bmod 2, \end{aligned}$$

which maps even numbers to 0, and odd numbers to 1, you could instead write

$$x \longmapsto \begin{cases} 0 & x \bmod 2 = 0 \\ 1 & \text{else} \end{cases}$$

or, if you don't want to put the mod function into the definition, you could write

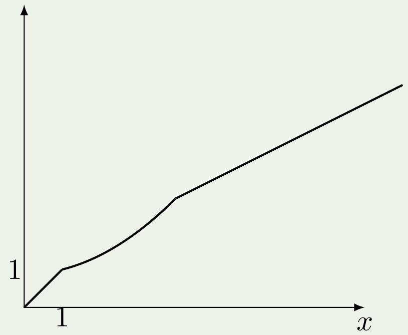
$$x \longmapsto \begin{cases} 0 & x \text{ even} \\ 1 & \text{else.} \end{cases}$$

What is important is that

- you give a value for each element of the source and
- you don't give more than one value for any element of the source.

In other words, on the right you must split your source sit into disjoint parts, and say what the function does for each of those parts.

Example 0.34. You might need this when you are trying to describe the behaviour of an entity which changes. For example, assume you are given the following graph:



This function $\mathbb{R}^+ \longrightarrow \mathbb{R}^+$ is given by the assignment

$$x \longmapsto \begin{cases} x & x \in [0,1] \\ \frac{1}{8}x^2 + \frac{7}{8} & x \in (1,4] \\ \frac{1}{2}x + \frac{7}{8} & \text{else.} \end{cases}$$

CExercise 10. Write down formal definitions for the following functions.

(a) The function which takes two integers and returns the negative of their product.

(b) The function from $\mathbb{R} \times \mathbb{R}$ to $\mathbb{R}$ which returns its first argument.

(c) The function from $\mathbb{Z} \times \mathbb{Z}$ to $\{0, 1\}$ which is equal to 1 if and only if both arguments are even.

(d) The function from $\mathbb{R}$ to $\mathbb{R}$ which behaves like the sine function for negative arguments, and like the exponential function for base 2 for non-negative arguments.

(e) Draw a picture of the set

$$\{Abdul, Bella, Clara, Dong\} \times \{red, blue, green\}.$$

Define a function from that set to $\{0, 1\}$ which is 1 if and only if its first argument has more letters than its second.

Apart from this, the constructions we have for sets are also meaningful for functions.

If we have functions $f: S \longrightarrow S'$ and $g: T \longrightarrow T'$. Then we can define a function

$$S \times T \to S' \times T',$$

which we refer to as

$$f \times g$$

by setting

$$(s,t) \longmapsto (fs, gt).$$

Optional Exercise 2. Can you think of something that would allow you to extend the powerset construction to functions?

The following exercises draws on functions, as well as on the definition of the powerset from the previous section.

EExercise 11. Given a set X, define the following functions. Don't forget to write down their source and target.

(a) A function f from X to its powerset with the property that for every $x \in X$ we have $x \in fx$.

(b) A function from the product of the powerset of X with itself, to the powerset of X, with the property that a pair of sets is mapped to the set consisting of all those elements of X which is either in the first set, or in the second set, but not in both.

(c) Define a function from the product of X with its powerset to the set $\{0, 1\}$ which returns 1 if and only if the first component of the argument is an element of the second component.

(d) Define a function from the set of finite subsets of $\mathbb{N}$ to $\mathbb{N}$ which adds up all the elements in the given set.

0.4 Relations

We study relations in detail in Chapter 7. Prior to that chapter, however, relations play a (minor) role in Chapter 3 and we give the basic ideas here for that reason.

Sometimes we have connections between two sets S and T which do not take the form of a function. We might have some set of pairs of the form (s, t), where $s \in S$ and $t \in T$. Such a set is known as a **binary relation**. Note that relations of other arities exist, but it is customary to drop the 'binary' part and just speak of a relation.

Example 0.35. Consider the set S of all the first year students in the School of Computer Science, and the set U of all course units on offer in the university. We may then define a relation as

$$\{(s,u) \in S \times U \mid s \text{ is enrolled on } u\}.$$

This set is encoded in a database somewhere in the student system.

Relations are very flexible when it comes to capturing connections between various entities. A number of examples are given in Chapter[43] 7, but here are some ideas for the kind of thing that one can do:

[43] Note that this chapter is studied in Semester 2.

- Sometimes a set of interest may contain a number of elements one wishes to consider 'the same', for example when using fractions to describe the rational numbers. One may use an *equivalence relation* (between the set and itself) to partition the set into *equivalence classes* and use those instead of the original elements. An example of this is the relation which connects two students if and only if they are in the same lab group.
- One may wish to compare the elements of a set with each other, indicating that one is below another. This is done using a relation between the set and itself known as a *(partial) order*. Examples of these are the usual orders on $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$, but more interesting options exist.

How does one describe a relation? The most common description is that of a subset of the product as in the example above, similar to the graph of a function. This is a set, so the usual suggestions for describing sets apply.

Quite often it is possible to describe a relation using a predicate.

Example 0.36. The relation which connects the integers m and n if m divides n is

$$\{(m, n) \in \mathbb{Z} \times \mathbb{Z} \mid n \bmod m = 0\}.$$

We may apply more complicated conditions to pick out the set of pairs we want to describe.

Example 0.37. The equality of fractions as rational numbers provides another example. This relation is defined as

$$\{(x/y, x'/y') \mid x, x' \in \mathbb{Z},\ y, y' \in \mathbb{Z} \setminus \{0\} \text{ and } xy' = x'y\}.$$

It is less often the case that one can use the idea of generating the relation as a set. This typically only works if there is a way of expressing one of the pair in terms of the other.

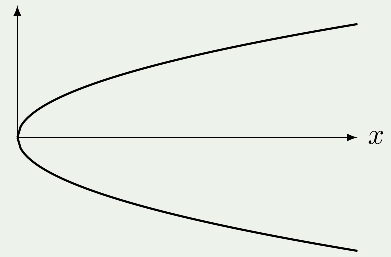
Example 0.38. The relation (x, y) in R^2 with $x^2 = y$ can be generated as

$$\{(x, x^2) \mid x \in \mathbb{R}\}.$$

Note that in this particular case it is also possible to describe the same relation as the union of two sets, namely as

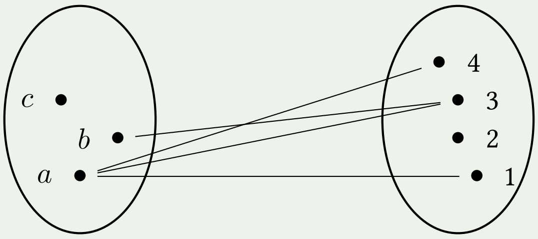
$$\{(\sqrt{x}, x) \in \mathbb{R} \times \mathbb{R} \mid x \in \mathbb{R}^+\} \cup \{(-\sqrt{x}, x) \in \mathbb{R} \times \mathbb{R} \mid x \in \mathbb{R}^+\}.$$

In a case like this it is easy to show a picture of the set in question.



If the relation is finite (and small) then it may be possible to list all the elements it contains. In this case it is also possible to draw a graph to indicate which elements are related.

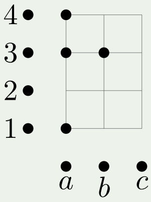
Example 0.39. Here is the kind of graph one might draw for a small relation.



This is the relation which relates a to 4, 3 and 1, it relates b to 3, and it relates c to nothing at all. Its set description is

$$\{(a, 1),\ (a, 3),\ (a, 4),\ (b, 3)\}.$$

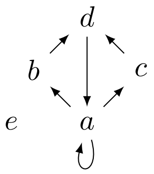
Alternatively one could draw those pairs in the product set that belong to the relation in this way, similar to the graph of a function (see Section 0.3.4):



You may think of the grid as giving all the possible combinations when picking one element from $\{a, b, c\}$ and one from $\{1, 2, 3, 4\}$. The dot tells you whether the corresponding pair belongs to the relation, or not.

Note that every function defines a relation between its source and its target via its graph. These are very special relations, described in more detail in Section 7.1.

If we have a binary relation from one set to itself then we can picture this by drawing connections between the elements of the given set. Typically we would say that we have 'a (binary) relation on the set S' instead of 'a (binary) relation from S to S'.



This is a picture of the following relation on the set $\{a, b, c, d, e\}$:

$$\{(a, a), (a, b), (a, c), (b, d), (c, d), (d, a)\}.$$

Note that relations do not have to be *binary*, they can have a higher arity. A ternary relation for sets S, T and U, for example, is a subset of $S \times T \times U$. This kind of relation is difficult to picture in two dimensions, so typically no pictures are drawn for these.

Chapter 1

Complex Numbers

The real numbers allow us to solve many equations, but equations such as

$$x^2 = -1$$

have no solutions in $\mathbb{R}$.

One way of looking at the complex numbers is that they remedy this problem. But assuming this is all they do would sell them far short. We here give a short introduction to the set of complex numbers, addition and multiplication operations for them, and their basic properties.

Note that in order to solve exercises in this chapter you should only use properties given by Facts 1 to 7 in Chapter 0.

1.1 Basic definitions

We begin by giving some basic definitions.

Definition 15: complex numbers

The **set of complex numbers** $\mathbb{C}$ consists of numbers of the form

$$a + bi,$$

where a, b in $\mathbb{R}$. Here a is known as the **real part** and b as the **imaginary part** of the number.

At first sight it is not entirely clear what exactly we have just defined. One may view $a + bi$ as an expression in a new language.

If one of a or b is 0 it is customary not to write it, so the complex number a is equal to $a + 0i$ and the complex number bi is equal to $0 + bi$. Similarly, if $b = 1$ then it is customary to write $a + i$ instead of $a + 1i$.

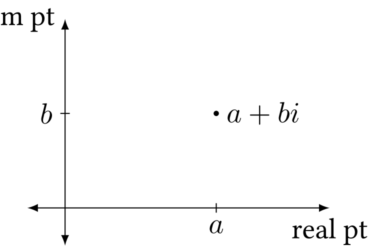
We may think[1] of a real number r as being a complex number whose imaginary part is 0, so it has the form $r + 0i$. In that way the complex numbers can be thought to include the real numbers (just as we like to think of the real numbers as including the rational numbers).

This gives a function from $\mathbb{R}$ to $\mathbb{C}$ defined by

$$r \longmapsto r + 0i.$$

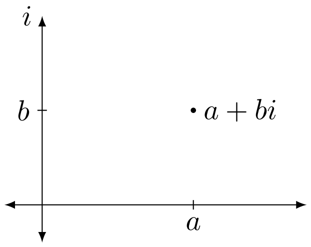
[1] Compare this with casting a value of one datatype to another in Java.

Complex numbers are usually drawn as points within the plane, using the horizontal axis for the real and the vertical axis for the imaginary part.

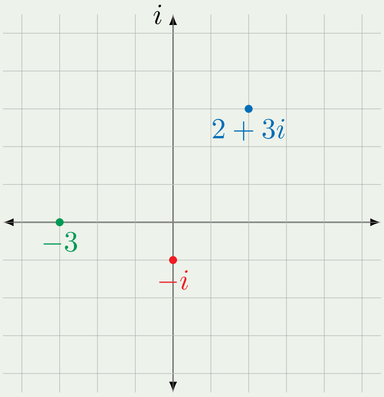


Above we have added labels for orientation, but usually this is done a bit differently.

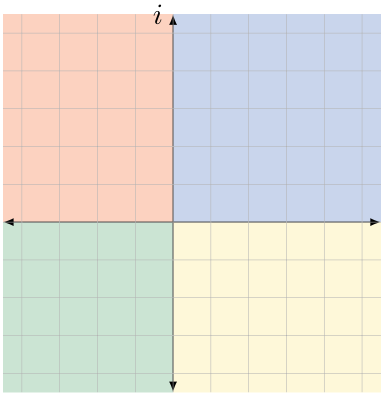
Instead of marking the real and the imaginary part on the axes it is more common to mark the 'imaginary axis' with i, giving a picture in the *complex plane*.



Example 1.1. We show how to draw the numbers $2 + 3i$, -3 and $-i$ in the complex plane.



The complex plane is naturally divided into four quadrants.



1.2 Operations

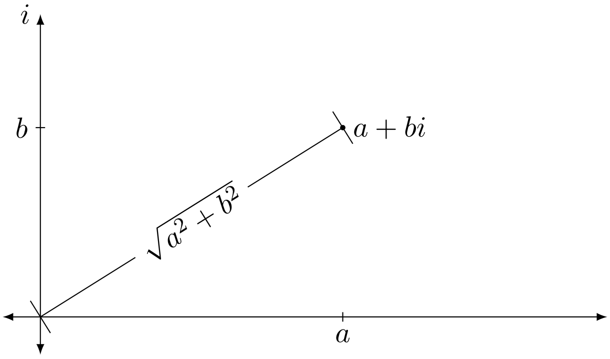
There are quite a few operations one defines for complex numbers.

1.2.1 The absolute

The[2] **absolute** $|a + bi|$ **of a complex number** $a + bi$ is given by

$$\sqrt{a^2 + b^2}.$$

We may think of this as the length of the line that connects the point 0 with the point $a + bi$:



Example 1.2. The absolute of the complex number $1 + 2i$ is calculated as follows.

$$|1 + 2i| = \sqrt{1^2 + 2^2} = \sqrt{5}.$$

Note that this extends the notion of absolute for real numbers in the sense that

$$|a + 0| = \sqrt{a^2 + 0} = \sqrt{a^2} = |a|$$

where we use the absolute function for real numbers on the right.[3]

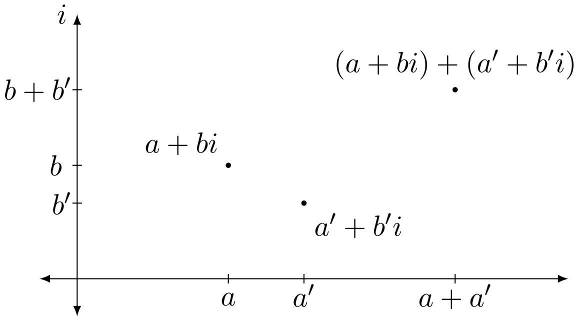
One can calculate with the complex numbers based on the following operations.

1.2.2 Addition

Addition of two complex numbers is defined as follows.

We set

$$(a + bi) + (a' + b'i)$$
$$= (a + a') + (b + b')i.$$

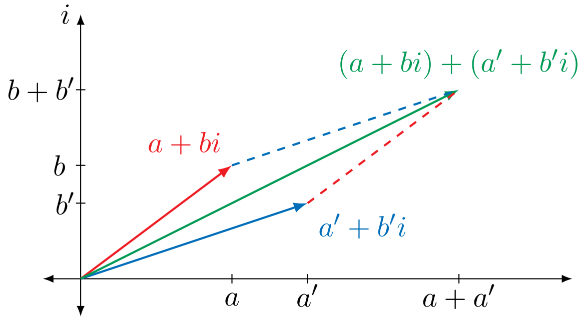


To understand addition it is useful to think of the numbers in the complex plane as[4] *vectors*, then addition is just the same as the addition of vectors:

[2] This is also known as the *modulus* of a complex number.

[3] And indeed note that we could use $\sqrt{r^2}$ as a definition of the absolute for a real number r.

[4] Vectors will be taught in detail in the second half of Semester 2.



If you prefer, you may think of this as taking the vector for $a'+b'i$ and shifting it so that its origin coincides with the end point of the vector for $a+bi$.[5]

Example 1.3. We calculate the sum of $1+2i$ and $-1+i$ as follows.

$$(1+2i)+(-1+i)=(1-1)+(2+1)i=3i.$$

We note that if we have two complex numbers whose imaginary part is 0, say a and a', then their sum as complex numbers is $a+a'$, that is their sum as real numbers.

Important properties of this operation are established in Exercise 27 and 28, which establish two equalities from Fact 6 for the complex numbers.

Note that 0 is the unit for addition[6], that is adding 0 to a complex number (on either side) has no effect.[7] In other words we have

$$\begin{aligned}
0+(a+bi) &= (0+a)+(0+b)i && \text{def addition}\\
&= a+bi && \text{Fact 6}\\
&= (a+0)+(b+0)i && \text{Fact 6}\\
&= (a+bi)+0. && \text{def addition}
\end{aligned}$$

For the real numbers every element r has an *inverse* for addition in the form of $-r$: This is the unique number[8] which,[9] if added to r on either side, gives the unit for addition 0.

For addition of complex numbers we can find an inverse by making use of the inverse for addition for the reals. The following lemma explains how to calculate the additive inverse, and it also establishes that such an inverse exists for all complex numbers.

Lemma 1.1

The additive inverse of the complex number $a+bi$ is

$$-a-bi,$$

[5] Note that you may just as well think of shifting the vector for $a+bi$ such that its origin coincides with the end point of the vector for $a'+b'i$.

[6] Look at the unit for addition given by Facts 1, 3, 5 and 6.

[7] For a formal definition of the unit of an operation see 20.

[8] Again, compare Facts 3. 5 and 6 from the previous chapter.

[9] For a formal definition of the inverse for a given element with respect to a given operation see 21 in the following chapter.

which we often write as

$$-(a+bi).$$

This establishes that every element of $\mathbb{C}$ has an additive inverse and that means we may define *subtraction for complex numbers* by setting

$$(a+bi)-(a'+b'i)=(a+bi)+-(a'+b'i),$$

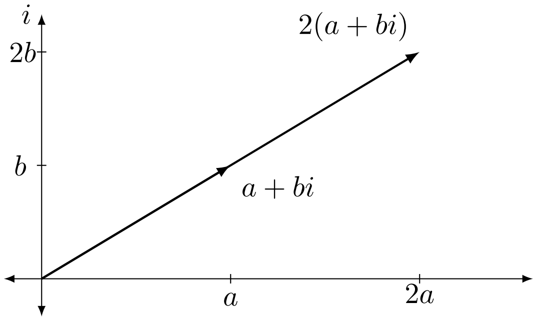
so as usual this is a shortcut to deducting the additive inverse of the second argument from the first argument.

Exercise 12. Prove Lemma 1.1. *Hint: The paragraph above this exercise tells you what you need to check, or look ahead to Definition 21.*

Note that there is no easy connection between the absolute and addition; the best we may establish for complex numbers z and z' is that

$$|z+z'|\leq|z|+|z'|.$$

Note that $(a+bi)+(a+bi)=2a+2bi$, and that we may think of this as stretching $a+bi$ to twice its original length, and write it as $2(a+bi)$:



In general, given a real number r and a complex number $a+bi$ we may define

$$r(a+bi)=ra+rbi.$$

Note that this means that our definition of the negative of a complex number works in the expected way in that

$$-(a+bi)=(-1)(a+bi)=(-1)a+(-1)bi=-a-bi.$$

Example 1.4. We see that $3(5+i) = 15+3i$, and we further calculate $-\sqrt{2}(\sqrt{2}-\sqrt{2}i)=-2+2i$.

CExercise 13. Draw the following numbers in the complex plane: 2, -2, $2i$, $-2i$, $3+i$, $-(3+4i)$, $(-1+2i)+(3+i)$, $(1+2i)+(3-i)$, $(1+2i)-(3+i)$. For each quadrant of the complex plane pick one of these numbers (you may pick at most two numbers lying on an axis, and the axes have to be on different ones), and calculate its absolute.

Assume your friend has drawn a complex number z on a sheet that you cannot see. Instruct them how to draw the following.

(a) $-z$,

(b) $2z$,

(c) $3z$,

(d) rz, where r is an arbitrary real number.

Exercise 14. Consider the function f from $\mathbb{R}^2$ to $\mathbb{C}$ which is defined as follows:

$$(a, b) \longmapsto a + bi.$$

Show that $f(a, b) + f(a', b') = f((a, b) + (a', b'))$ for all $(a, b), (a', b') \in \mathbb{R}^2$. Here we use the *componentwise* addition for elements of $\mathbb{R}^2$, that is

$$(a, b) + (a', b') = (a + a', b + b')$$

for all a, a', b and b' in $\mathbb{R}$.

1.2.3 Multiplication

We define the multiplication operation on complex numbers by setting

$$(a + bi)(a' + b'i) = aa' - bb' + (ab' + ba')i.$$

Example 1.5. We calculate

$$(1 + 2i)(2 - 3i) = (2 + 6) + (4 - 3)i = 8 + i.$$

Exercise 15. Show that 1 is the unit for multiplication. *Hint: Check the calculation carried out above which shows that 0 is the unit for addition. Also look at Fact 1 which tells you what it means for 1 to be the unit for multiplication of natural numbers.*

Note that if one of the numbers has imaginary part 0 then we retain the multiplication with a real number defined above, that is

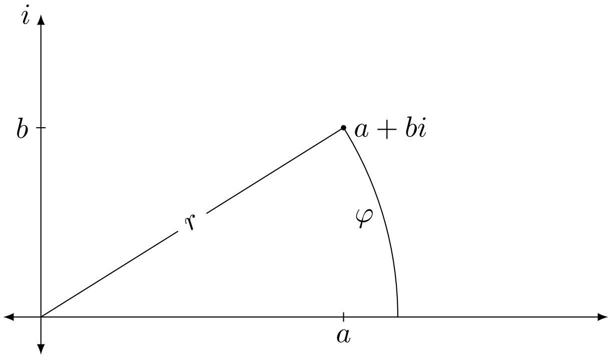
$$a(a' + b'i) = aa' + ab'i.$$

There is a geometric interpretation of multiplication, but it is a bit more complicated than that for addition. We here only give a sketch of this.

Instead of giving the coordinates a and b to describe a point in the complex plane one also could give an **angle** and a **length**.

Definition 16: polar coordinates

The description in polar coordinates **of a complex number** or its **polar form** consists of a non-negative real number, known as the absolute and an angle in $[0, 360)$ (or in $[0, 2\pi)$) known as the argument.



Note that the absolute of a complex number in this sense is nothing but the absolute $|a+bi|$ from above.

Example 1.6. The complex number $1+i$ has the absolute $\sqrt{2}$ and the argument $45°$ or, if you prefer, $\pi/4$. One might use a notation such as $(\sqrt{2}, 45°)$ (or $(\sqrt{2}, \pi/4)$) for complex numbers given in this way.

This means there are two ways of describing a complex number:

via		via
• the real part a and	or	• the absolute r and
• the imaginary part b		• the argument φ.

To move from polar coordinates to the standard form there is a simple formula: The complex number given by r and φ is

$$r(\cos\varphi + (\sin\varphi)i).$$

In the other direction one can use the arctangent function $\arctan$, the partial inverse of the tangent function, to calculate φ given a and b, but a few case distinctions are required.

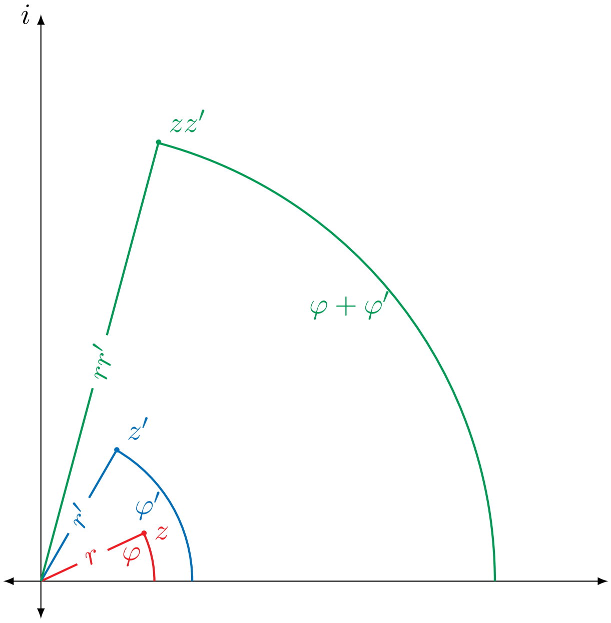
Optional Exercise 3. Write out the definition of the function that gives the argument, for a complex number $a+bi$. Then prove that, starting from a complex number, calculating the argument and the absolute, and then calculating the real and imaginary part from the result, gives back the number one started with. Use these calculations to show that the argument of zz' is the argument of z plus the argument of z'.

Describing multiplication is much easier when we do it with respect to these polar coordinates:

Fact 9

Assume that (r, φ) and (r', φ') are two complex numbers whose first component defines the absolute, and whose second component gives the argument. Their product (in the same format) is given by the number

$$(rr', \varphi + \varphi').$$



Note that there is a nice connection between the absolute and multiplication.

Lemma 1.2

For complex number z and z' we have

$$|zz'| = |z||z'|.$$

Exercise 16. Prove Lemma 1.2.

We note that according to the definition of multiplication we have

$$ii = (0+1i)(0+1i) = 0 - 1\cdot 1 + (0\cdot 1 + 1\cdot 0)i = -1,$$

so in the complex numbers we may solve the equation

$$x^2 = -1$$

which has no solution in $\mathbb{R}$.

CExercise 17. Pick four numbers in at least three different quadrants of the complex plane. Calculate, and then draw, their product with the number i.

Your friend has drawn the number z on the complex plane, but you can't see what they are doing. Instruct them how to draw iz without referring to any coordinates.

Optional Exercise 4. What happens if we keep multiplying i with itself? What does that tell you about solutions to the equation $x^4 = 1$? What about solutions for $x^n = 1$ more generally?

Exercise 18. Consider the function f from $\mathbb{R}^2$ to $\mathbb{C}$ which is defined as follows:

$$(a, b) \longmapsto a + bi.$$

Define addition and multiplication on $\mathbb{R}^2$ based on these operations for complex numbers. *Hint: You may want to consult Exercise 14.*

We have seen that with regards to addition, every complex number z has an inverse in the form of $-z$. What about inverses for multiplication?

Lemma 1.3

For every complex number $a + bi \neq 0$ the multiplicative inverse is given by[10]

$$\frac{a}{a^2 + b^2} - \frac{b}{a^2 + b^2}i.$$

Sometimes the notation

$$(a + bi)^{-1}$$

is used for this number. More generally, if we have a complex number z then it's inverse, if it exists, is written as z^{-1}. Just as for real numbers the expression

$$z/z'$$

is a shortcut for

$$z(z')^{-1}.$$

Do not divide by complex numbers in your work. The correct operation is to multiply with the multiplicative inverse, and for full marks you have to include an argument that this exists in the case you are concerned with. In particular if you would like to multiply with the multiplicative inverse of a variable you have to explicitly consider the case where that variable happens to be equal to 0.

Whenever you use the number z^{-1} you have to include an argument that this exists, that is, that $z \neq 0$ (and indeed you should do this whenever you use r^{-1} for real or rational numbers).

Recall that we avoid talking about division as an operation on this unit, so if you want to remove a factor from an equation please try to talk about multiplying with the multiplicative inverse, and think about whether this exists!

EExercise 19. Prove Lemma 1.3 *Hint: Check Fact 5 from the previous chapter to see what you have to prove, or look ahead to Definition 21. Note: We have not defined $1/z$ for a complex number z and you should not use this expression.*

Calculate the inverse of the complex number $z = a + 0i = a$. How does that compare with the multiplicative inverse for a when viewed as an element of $\mathbb{R}$?

[10] Note that our condition means that $a^2 + b^2 \neq 0$ and therefore we may form the fractions given here.

EExercise 20. Assume you have a complex number in polar form (r, φ). What is the polar form of its multiplicative inverse? *Hint: What does multiplication with the inverse have to give? Look at the picture on page 57 which explains multiplication in terms of absolute and argument to find a number that satisfies the requirement for an arbitrary absolute r and argument φ.*

In summary we have seen that just as for the real numbers, we may define addition and composition for complex numbers, and in such a way that if we treat real numbers as particular complex ones, then the operations agree. Indeed, it is also possible to define exponentiation and logarithms for complex numbers but this idea leads us too far afield.

1.2.4 Conjugation

There is a further operation that you may find in texts that deal with complex numbers, namely **conjugation**. The **conjugate** $\overline{z}$ of a complex number $z = a + bi$ is given by $a - bi$.

Example 1.7. We give sample calculations.

$$\overline{-2+i} = -2 - i, \qquad \overline{3} = 3, \qquad \overline{i} = -i.$$

Exercise 21. Assume you have a complex number given by its absolute and argument. What are the absolute and argument of its conjugate? *Hint: If you find this difficult draw a few examples in the complex plane.*

CExercise 22. Show that $z\overline{z} = |z|^2$.

1.3 Properties

The complex numbers have various properties which make them a nice collection of numbers to work with. You are allowed to use the following in subsequent exercises on complex numbers.[11]

Fact 10

Addition and multiplication of the complex numbers have all the properties of the real numbers as given in Fact 6.

Optional Exercise 5. Prove the statements of Fact 10.

Note that when it comes to solving equations, the complex numbers are even better behaved than the real ones: Every polynomial equation, that is an equation of the form

$$\sum_{i=0}^{n} c_i x^i = c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0 = 0,$$

[11] You are not allowed to use them in exercises where you are specifically asked to prove them!

where the c_i are complex numbers has at least one solution[12] in $\mathbb{C}$, whereas this is not true in $\mathbb{R}$ even if the c_i are all elements of $\mathbb{R}$. This means that the complex numbers are particularly suitable for various constructions that depend on having solutions to polynomials.

Note that it does not make any sense to use the square root operation for complex numbers. While for a positive number r, we use the symbol

$$\sqrt{r}$$

to refer to the positive of the two solutions to the equation

$$x^2 = r$$

for a complex number z there is no sensible way of of picking out one of the possible solutions of

$$x^2 = z.$$

Example 1.8. Consider the equation

$$x^2 = i.$$

We may check that there are two solutions,

$$\frac{1}{\sqrt{2}}(1+i) \qquad \text{and} \qquad \frac{1}{\sqrt{2}}(-1-i).$$

Which of those should be the number we mean by $\sqrt{i}$? You may think there's still a sensible choice, namely the one where both, real and imaginary part are positive.

Now consider the equation

$$x^2 = -i.$$

It has the solutions

$$\frac{1}{\sqrt{2}}(-1+i) \qquad \text{and} \qquad \frac{1}{\sqrt{2}}(1-i),$$

and picking one over the other does not make sense.

If we go to equations involving powers higher than two then the number of solutions increases.

Example 1.9. Let us consider the following equation:

$$x^4 = 1.$$

If x is supposed to be a real number then there are two solutions, namely 1 and -1.

If, on the other hand, we are allowed to pick solutions from $\mathbb{C}$ we note that there are at least four:

$$1, \qquad -1, \qquad i, \qquad -i.$$

[12] In fact, one can show that there are n solutions, but this requires counting some solutions more than once.

Certainly there is no good way of picking out one of these solutions to determine which number we might mean by $\sqrt{41}$.

For this reason there are no root operations on the complex numbers.

Do not use square root symbols if you are interested in a solution of the equation $z^2 = z'$, where z' is given. This is not a valid operation for complex numbers.

An important difference between the complex numbers and the other sets of numbers discussed in Chapter 0 is that we are used to thinking of the latter as being *ordered*, which means we can compare two elements. There is no way of turning $\mathbb{C}$ into an ordered set so that the statements in Fact 7 are true for that order.

In analysis, which includes the study of functions, their derivatives and their integrals, the theory of functions of complex numbers is much smoother than its counterpart for the reals. In order to calculate various (improper) integrals for functions of real variables one may apply methods that require functions of complex variables.

The fact that complex numbers may be thought of as having two parts, and that we have various operations for these, means they are particularly suited to a number of application areas where these operations may be interpreted.

1.4 Applications

Complex numbers may appear artificial, and having numbers with an 'imaginary' part may suggest that these are merely figments of some mathematicians' imagination.

It turns out, however, that they are not merely some artefact whose only use it is to deliver a number which is a square root of -1. Because complex numbers can effectively be thought of as vectors, but vectors which allow multiplication as well as addition, they are very useful when it comes to talking about quantities that need a more complex structure to express them than just one number.

In physics and various areas of engineering quantities which may be described by just one number are known as 'scalars'. Examples are distance, speed (although to describe movement one might want to include direction with speed) and energy. In a direct-current circuit, voltage, resistance and current are treated as scalars without problem. In alternating current circuits, however, there are notions of frequency and phase shift which have to be taken into account, and it turns out that using complex numbers to describe such circuits results in a very useful depiction. Moreover some calculations become much simpler when one exploits the possibilities given by modelling the circuit with complex numbers.

Signal analysis is another area where complex numbers are often employed. Again the issue here are periodically varying quantities. Instead of describing these using a sine or cosine function of some real variable, employing the extensions of these functions to the complex numbers makes it possible to describe the amplitude and phase at the same time.

There you will see for example, that by using complex numbers for a Fourier transform calculations that look complicated can be carried out via matrix multiplication.[13] When you meet this material you should remind yourself of what you know about complex numbers from these notes.

There are other areas where applications arise, such as fluid dynamics, control theory and quantum mechanics.

[13] The latter will be treated towards the end of Semester 2 of this course unit.

Chapter 2

Statements and Proofs

Mathematics is a discipline that relies on rigorous definitions and formal proofs. As a consequence, in mathematics statements hold, or they do not (but we may not know which it is).[1] This is very different from the situation in the natural sciences, for example. Here a theory may be falsified by observations that contradict it, but there is no way of formally verifying it.

How does a system that seeks to provide such certainty work? In principle the thought is that it is possible to define a theory strictly from first principles (typically starting with a formal theory of sets), with rules for deriving statements from existing ones. Such a system is very rigid and syntactic[2] in nature, much like a computer language (and indeed there are computer programs that implement at least aspects of this). Statements that may be formally derived in the system are known as theorems. In principle it should be possible to fit all of mathematics into a formal system like this.[3]

But in practice this is not what mathematicians do. There are two reasons for this. Starting from first principles it takes a very long time to build up the apparatus required to get to where one may even talk about entities such as the real numbers with complete rigour. Secondly the resulting statements are very unwieldy and not human-readable. Hence mathematicians carry out their work in some kind of *meta-language* which in principle can be translated into a formal system. Increasingly there are computer-verified proofs in some formal system in various areas, in particular in theoretical computer science.

In this and the following chapter of the notes we look at both these ideas—proofs as they are customarily carried out by mathematicians and a formal system.

2.1 Motivation

You are here to study computer science rather than mathematics, so why should you worry about proving statements? There are two reasons one might give here.

For one there is the area of *theoretical computer science* which arguably is also an area in mathematics. The aim of this part of computer science is to make formal

[1] There are also issues to do with whether a given formal system allows us to construct a proof or a counterexample.

[2] This means concerned with symbols put together according to some rules without any concern what they might mean.

[3] But there is a famous result by the logician Kurt Gödel, his *Incompleteness Theorem*, which tells us that any system sufficiently powerful for most of mathematics cannot prove its own consistency.

statements and to prove them. Here are some examples of the kind of statement that are of concern in this area.

- This abstract computational device has the same computational power as another.
- This computation is equivalent to another.[4]
- This abstract computational system behaves in a particular manner over time.
- This problem cannot be solved by a computer, or, equivalently, there is no algorithm (or decision procedure) for it (see COMP11212).
- The best possible algorithm for this problem requires a number of steps that is a quadratic function in the size of the problem (see COMP26120).
- This program will terminate and after it has done so its result will satisfy a particular condition.
- This circuit implements a particular specification.

You can see that while the first few statements sound fairly abstract the latter two look as if they might be closer to real-world applications.

Secondly, under certain circumstances it is important to make absolute statements about the behaviour of a computational device (a chip or a computer program for example). Formally proving that programs behave in a particular way is labour-intensive (and creating a formal model of the real world in which the device lives is potentially error-prone).

In safety-critical systems, however, the benefits are usually thought to outweigh the cost. For example in an aircraft it is vital that the on-board computer behaves in a particular way. Emergency course corrections have to be made promptly and correctly or the result may be fatal for those on board. When NASA sends an explorer onto Mars, or the Voyager space craft to fly through the solar system (and to eventually leave it) then it is vital that a number of computer-controlled manoeuvres are correctly implemented. Losing such a craft, or rendering it incapable of sending back the desired data, costs large amounts of money and results in a major setback.

But even outside such applications computing is full of statements that are at least in part mathematical. Here are some examples.

- The worst case complexity of this algorithm is $n \log n$ (the kind of statement that you will see in COMP26120).
- This recursive procedure leads to exponential blow-up.
- A simple classification rule is to choose the class with the highest posterior probability (in artificial intelligence or machine learning).
- Time-domain samples can be converted to frequency domain using Fourier Transforms, which are a standard way of representing complex signal $g(t)$ as a linear sum of basic functions $f_g(t))$ (from COMP28512).

[4] What it might mean for two computations to be equivalent is a whole branch of theoretical computer science.

The aim of this course unit is to prepare you for both these: studying areas of theoretical computer science and making sense of mathematical statements that appear in other parts of the field.

2.2 Precision

Something the language of mathematics gives us is precision. You need to become familiar with some aspects of this. In particular, there are some key phrases which sound as if they might be parts of every-day language, which have a precise meaning in a mathematical context.

2.2.1 Key phrases

Vocabulary that helps us with this are phrases such as

- 'and',
- 'or',
- 'implies' (and the related 'if and only if'),
- 'there exists' and
- 'for all'.

The aim of this section is to introduce you to what these phrases mean, and how that is reflected by by proving statements involving them.

Keyword And

Formally we use this word to connect several statements, or properties, and we demand that all of them hold.

When you enter several words into the Google search box you ask it to return pages which contain *all* the listed words—you are demanding pages that contain $word$ 1 and $word$ 2 and …

If you are running database queries you are often interested in all entries that combine several characteristics, for example, you might want all your customers from a particular country for whom you have an email address so that you can make a special offer to them.

These are all informal usages, but they have fundamentally the same meaning as more mathematical ones.

Example 2.1. A very simple example is the definition of the intersection of two subsets S and T of a set X.

$$S \cap T = \{x \in X \mid x \in S \text{ and } x \in T\}.$$

In order to prove that an element x is in this intersection we have to prove both, that

x is in S and that x is in T.

It is a good idea to structure proofs so that it is clear that these steps are carried out. Here is an example of this idea.

Example 2.2. To show that 6 is an element of

$$\begin{aligned}&\{n \in \mathbb{N} \mid n \bmod 2 = 0 \text{ and } n \bmod 3 = 0\}\\ &= \{n \in \mathbb{N} \mid n \bmod 2 = 0\} \cap \{n \in \mathbb{N} \mid n \bmod 3 = 0\}\end{aligned}$$

one splits the requirement into the two parts connected by *and*.

- To show that 6 is in the first set we have to show that $6 \bmod 2 = 0$, and we may conclude this from

  $$6 = 3 \cdot 2 + 0,$$

  Fact 2 and the definition of mod.

- To show that 6 is in the second set we have to show $6 \bmod 3 = 0$, and we may conclude this in the same way from $6 = 2 \cdot 3 + 0$.

Overall this means that 6 is in the intersection as required.

This usage of 'and' may also be observed in every-day language: If I state 'it is cloudy and it is raining' then I am claiming that both of the following statements are true:

It is cloudy. It is raining.

Example 2.3. In order to check that a first year student in the School satisfies the degree requirement, and is enrolled on COMP10120 as well as being enrolled on COMP16321 I have to do both,

- check that the student is enrolled on COMP10120 and
- check that the student is enrolled on COMP16321.

Example 2.4. In order to establish

$$x \notin S \cap T$$

it is sufficient to show *one* of

$$x \notin S \qquad\qquad x \notin T.$$

In general in order to argue that a statement of the form (Clause 1 and Clause 2) does not hold it is sufficient to show that one of the two clauses fails to hold.

Example 2.5. In order to argue that it is *not* the case that

3 is a prime number and 3 is even

it is sufficient to be able to state that since 3 leaves the remainder of 1 when divided by 2 it is not even by Definition 4.

Keyword Or

We connect two statements or properties with ‘or’ if at least one (but possibly both of them) hold.

To use the Google search box as an example once again, if you type two entries separated by ‘OR’ it will look for pages which contain one of the two words.

This is also a fairly standard database query: You might be interested in all the customers for whom you have a landline or a mobile phone number, or all the ones who have ordered product X or product Y because you have an accessory to offer to them.

Example 2.6. Again a simple example is given by sets, namely by the definition of the union of two subsets S and T of a set X, which is given by

$$\{x \in X \mid x \in S \text{ or } x \in T\}.$$

For a concrete version of this see the following example.

Example 2.7. In order to show that 6 is an element of

$$\{n \in \mathbb{N} \mid n \bmod 2 = 0 \text{ or } n \bmod 3 = 0\}$$

it is sufficient to prove *one* of the two parts. It is therefore sufficient to state that

- Since $6 = 3 \cdot 2 + 0$ we have that $6 \bmod 2 = 0$ by the definition of $\bmod$.

It is not necessary to check the other clause.

This suggests a proof strategy: Look at both cases separately, and stop when one of them has been established.

Again this usage is well established in informal language (although usage tends to be less strict than with ‘and’). If I say ‘Tomorrow I will go for a walk or a bicycle ride’ then I expect (at least) one of the following two sentences to be true:

Tomorrow I will go for a walk. Tomorrow I will go for a bicycle ride.

There is no information regarding which one will occur—and indeed I may find the time to do both!

Example 2.8. If the degree rules state that a student on the Computer Science with Mathematics programme must take one of COMP11212 and COMP13212 and COMP15212 then I can stop checking once I have seen that the student is enrolled on COMP11212.

Note that in informal language ‘or’ often connects incompatible statements, which means that at most one of them is true. In mathematics two statements connected with ‘or’ may be incompatible, but they may well not be, and typically they aren’t. If we wanted to express the idea of two statements being incompatible we would have to say ‘Exactly one of Statement 1 and Statement 2 holds’.

In order to show that a statement of the form (Clause 1 or Clause 2) does not hold we have to show that *neither* of the clauses holds.

Example 2.9. In order to show that

$$x \notin S \cup T$$

we have to establish *both*.

$$x \notin S \qquad \text{and} \qquad x \notin T.$$

Example 2.10. In order to show that 7 is not an element of

$$\{n \in \mathbb{N} \mid n \bmod 2 = 0 \text{ or } n \bmod 3 = 0\}$$

I have to argue in two parts:

- Since $7 \bmod 2 = 1 \neq 0$ we see that 7 does not satisfy the first condition.
- Since $7 \bmod 3 = 1 \neq 0$ we see that 7 does not satisfy the second condition.

Hence 7 is not in the union of the two sets.

Keyword Implies

This is a phrase that tells us that if the first statement holds then so does the second (but if the first statement fails to hold we cannot infer anything about the second).

For this notion it is harder to find examples outside of mathematics, but you might want to ensure that in your database, the existence of an address entry for a customer implies the existence of a post code.

Again a simple formal example can be found by looking at sets.

Example 2.11. If S and T are subsets of some set X then

$$S \subseteq T$$

means that given $x \in X$,

$$x \in S \qquad \text{implies} \qquad x \in T.$$

In order to establish that $S \subseteq T$ given $x \in X$ one only has to show something in case that $x \in S$, so usually proofs of that kind are given by assuming that $x \in S$, and then establishing that $x \in T$ also holds. We look at a concrete version of this.

Example 2.12. To show that

$$\{n \in \mathbb{N} \mid n \bmod 6 = 0\} \subseteq \{n \in \mathbb{N} \mid n \bmod 3 = 0\}$$

we pick an arbitrary element n in the former set. Then $n \bmod 6 = 0$, and by the definition of mod this means that there is

$$k \in \mathbb{N} \qquad \text{with} \qquad n = k6 + 0.$$

But that means that

$$n = k6 = k(2 \cdot 3) = (k2)3$$

using Fact 1, and so picking $l = 2k$ we can see that $n = 3l + 0$, which means that $n \bmod 3 = 0$.

Typically we do not use 'implies' in informal language, but we have constructions that have a similar meaning. I might state, for example, 'if it rains I will stay at home'. So if on the day it is raining you should not expect to meet me, you should expect me to be at home. Note that this does *not* allow you to draw any conclusions in the case where it is not raining, although many people tend to do so. ('But you said you wouldn't be coming only if it was raining ...'.) If I say 'if it is raining I will definitely stay at home' I've made it clearer that I reserve the right to stay at home even if it is not raining. Note that in the formal usage of 'implies' the meaning is completely precise.

Example 2.13. One might use implication to state that if a student is in a tutorial group Wn,[a] where n is a natural number, then the student is in lab group W.

In order to show that this is true one has to check all the tutorial groups that start with W and make sure[5] their members are in labgroup W.

[a] Similar statements hold for tutorial groups Xn, Yn and Zn.

In order to show that a claimed implication does not hold one has to find an instance where the first clause holds while the second does not. So in order to catch me out as having said an untruth in the above example, you've got to find me out of the house when it is raining at the appointed time.

Example 2.14. In order to *refute* the claim that, for a natural number n,

n divisible by 2 implies n divisible by 6

we need to find a number that

- fulfils the first part, that is, it must be divisible by 2, but
- does not fulfil the second part, that is, is not divisible by 6.

Since $4 = 2 \cdot 2$ the number 4 is divisible by 2 according to Definition 3. Since there is no number $k \in \mathbb{N}$ with $4 = 6k$. the number 4 is not divisible by 6, which establishes that the claimed implication is false.

Key phrase If and only if

This phrase is merely a short-cut. When we say that

Statement 1 (holds) if and only if Statement 2 (holds)

then we mean by this that both,

Statement 1 implies Statement 2

[5] But this is the definition of labgroup W so we don't actually have to check this in reality!

and

$$\text{Statement 2} \qquad \text{implies} \qquad \text{Statement 1.}$$

To prove for two sets $S, T \subseteq X$ that

$$S = T$$

is equivalent to showing that for all $x \in X$, we have

$$x \in S \qquad \text{if and only if} \qquad x \in T,$$

which is equivalent to showing that

$$S \subseteq T \qquad \text{and} \qquad T \subseteq S.$$

Example 2.15. To show that

$$S = \{n \in \mathbb{N} \mid n \bmod 6 = 0\}$$

is equal to

$$T = \{n \in \mathbb{N} \mid n \bmod 2 = 0\} \cap \{n \in \mathbb{N} \mid n \bmod 3 = 0\}$$

we show that $S \subseteq T$ and that $T \subseteq S$.

It is a good idea to optically structure the proof accordingly.

$S \subseteq T$. Given $n \in S$ we know that $n \bmod 6 = 0$ which means that we can find $k \in \mathbb{N}$ with $n = k6$. This means that both

- $n = (k3)2$ and so $n \bmod 2 = 0$ and
- $n = (k2)3$ and so $n \bmod 3 = 0$

and so $n \in T$.

$T \subseteq S$. Given $n \in T$ we know that both

- $n \bmod 2 = 0$, which means that there is $k \in \mathbb{N}$ with $n = k2$ and
- $n \bmod 3 = 0$, which means that there is $l \in \mathbb{N}$ with $n = l3$.

This means that 2, which is a prime number, divides $n = l3$. By Definition 17 this means that

- 2 divides 3 (which clearly does not hold) or
- 2 divides l, which means that there exists $j \in \mathbb{N}$ with $l = j2$.

Altogether this means that $n = l3 = j2 \cdot 3 = j6$, and so n is divisible by 6.

Quite often when proving an ‘if and only if’ statement the best strategy is to prove the two directions separately. The only exception is when one can find steps that turn one side into the other, and every single step is reversible.

In order to show that an if and only if statement does not hold it is sufficient to establish that one of the two implications fails to hold.

Key phrase For all

Again a phrase that is very common in mathematical definitions or arguments, but there are other uses. For example you might want to ensure that you have an email address for every customer in your database.

Example 2.16. Consider the following statement.

$$\text{for all elements } k \text{ of } \{4n \mid n \in \mathbb{N}\} \text{ we have that } k \text{ is divisible by } 2.$$

In order to show that this is true I have to assume that I have an arbitrary element k of the given set. In order for k to be in that set it must be the case that there exists $n \in \mathbb{N}$ such that $k = 4n$. But now

$$k = 4n = 2(2n)$$

and according to Definition 3 this means that k is divisible by 2.

Since an arbitrary element of the given set satisfies the given condition, they must all satisfy it.

A 'for all' statement should have two parts.

- For which elements are making the claim? There should be a set associated with this part of the statement (this is $\mathbb{N}$ in the previous example).
- What property or properties do these elements have to satisfy? There should be a statement which specifies this. In the previous example it is the statement that the number is divisible by 2.

Typically when proving a statement beginning with 'for all' one assumes that one has an unspecified element of the given set, and then establishes the desired property.

Example 2.17. Looking back above at the statement that one set is the subset of another, we have, strictly speaking, suppressed a 'for all' statement.

Given subsets S and T of a set X the statement

$$S \subseteq T$$

is equivalent to

$$\text{for all } x \in X \qquad x \in S \text{ implies } x \in T.$$

In the proof in Example 2.12 we did indeed pick an arbitrary element of the first set, and then showed that it is an element of the second set.

Example 2.18. A nice example of a 'for all' statement is that of the equality of two functions with the same source and target. Let $f\colon S \longrightarrow T$ and $g\colon S \longrightarrow T$ be two functions. Then

$$f = g$$

if and only if

$$\text{for all } s \in S \text{ we have } fs = gs.$$

We look at a concrete version of this idea.

Example 2.19. Consider the following two functions.

$$\begin{aligned} f\colon \mathbb{N} &\longrightarrow \mathbb{N} \\ x &\longmapsto 2(x \operatorname{div} 2) \end{aligned} \qquad \begin{aligned} g\colon \mathbb{N} &\longrightarrow \mathbb{N} \\ x &\longmapsto \begin{cases} x & x \text{ is even} \\ x-1 & \text{else.} \end{cases} \end{aligned}$$

To show that the two functions are equal, assume we have an arbitrary element n of the source set $\mathbb{N}$. The second function is given in a definition by cases, and usually it is easier to also split the proof into these two cases.

- Assume that n is even, which means that $n \bmod 2 = 0$ by Definition 4. In this case

$$\begin{aligned} fn &= 2(n \operatorname{div} 2) && \text{def } f \\ &= 2(n \operatorname{div} 2) + 0 && 0 \text{ unit for addition} \\ &= 2(n \operatorname{div} 2) + n \bmod 2 && n \text{ even} \\ &= n && \text{Lemma 0.1} \\ &= gn && \text{def } g. \end{aligned}$$

- Assume that n is not even, which means that $n \bmod 2 = 1$ by Definition 4. In this case

$$\begin{aligned} fn &= 2(n \operatorname{div} 2) && \text{def } f \\ &= 2(n \operatorname{div} 2) + 0 && 0 \text{ unit for addition} \\ &= 2(n \operatorname{div} 2) + 1 - 1 && 1 - 1 = 0 \\ &= 2(n \operatorname{div} 2) + n \bmod 2 - 1 && n \text{ not even} \\ &= n - 1 && \text{Lemma 0.1} \\ &= gn && \text{def } g. \end{aligned}$$

In every-day language you are more likely to find the phrase ‘every’ instead of ‘for all’. Mathematicians like to use phrases that are a little bit different from what is common elsewhere to draw attention to the fact that they mean their statement in a formal sense.

Example 2.20. ‘Every first year computer science student takes COMP11120’ is a claim that is a ‘for all’ statement. In order to check whether it is true you have to go through all the first year students in the School and check whether they are enrolled on this unit.

In order to show that a statement beginning ‘for all’ does not hold it is sufficient to find *one element* of the given set for which it fails to hold.

Example 2.21. In the previous example, if you can find *one* student in computer science who is not enrolled on COMP11120 then you have shown that the statement given above does not hold.

Example 2.22. In order to refute the claim that for all natural numbers x and y it is the case that

$$x - y = y - x,$$

it is sufficient to find *one counterexample*, so by merely writing

$$2 - 1 = 1 \neq -1 = 1 - 2,$$

we have proved that the claim does not hold.

Key phrase There exists

This is a phrase that is frequently found both in mathematical definitions and arguments.

Example 2.23. The definition of divisibility (compare Definition 3)

$$\begin{array}{lll} y \text{ is divisible by } x & & \\ & \text{if and only if} & \text{there exists } k \in \mathbb{N} \\ & & \text{such that } y = kx. \end{array}$$

is an example of a 'there exists' statement.

Whenever a statement is made about existence there should be two parts to it:

- Where does the element exist? There should always be a set associated with the statement. In the above example, k had to be an element of $\mathbb{N}$ (and indeed the existence of some $r \in \mathbb{R}$ with the same property would completely change the definition and make it trivial).
- What are the properties that this element satisfies? There should always be a statement which specifies this. In the example above the property is $m = kn$ (for the given m and n).

One proves a statement of this form by producing an element which satisfies it, which is also known as a *witness*.

Example 2.24. To show that 27 is divisible by 9, by Definition 3, I have to show that

$$\text{there exists } k \in \mathbb{N} \text{ with } 27 = k9.$$

To show this I offer $k = 3$ as a witness, and observing that

$$9 \cdot 3 = 27$$

verifies that this element has the desired property.

To go back to the database example you might wonder whether you have a customer in your database who lives in Italy, or whether you have a customer who is paying with cheques (so that you can inform them that you will no longer accept these as a payment method).

Again in every-day language the phrase 'there is' is more common than 'there exists'. The latter serves to emphasize that a statement including is should be considered a precise mathematical statement.

Example 2.25. 'There is a student who is enrolled on both, COMP25212 as well as MATH20302' may have implications for the timetable.

Example 2.26. In order to show that there exists a number which is both, even and prime, it is sufficient to supply the witness 2 together with an argument that it is both, even and prime.

Often the difficulty with proving a 'there exists' statement is *finding* the witness, rather than with proving that it has the required property.

Sometimes instead of merely demanding the existence of an element we might demand its *unique existence*. This is equivalent to a quite complex statement and is discussed below.

In order to show that a statement beginning 'there exists' does not hold one has to establish that it fails to hold for *every* element of the given set. So to demonstrate that the statement above regarding students does not hold you have to check every single second year student.

Key phrase Unique existence

We sometimes demand that there exists a *unique* element with a particular property. This is in fact a convenient shortcut.

There exists a unique $s \in S$ with the property P

holds if and only if

there exists $s \in S$ with property P and
for all $s, s' \in S$, if s and s' satisfy property P then $s = s'$.

Example 2.27. If $f\colon S \longrightarrow T$ is a function from the set S to the set T then we know that the function assigns to every element of S an element of T, and this means that

for every $s \in S$ there exists a unique[6] $t \in T$ with $fs = t$.

Uniqueness is important here: We expect that a function, given an input value, produces precisely one output value for that input. So if we have valued t and t' in T which both satisfy the statement then we have $t = fs = t'$, and so $t = t'$. This idea is used in characterizing graphs of functions, see Definition 14.

[6] Note that this is a different statement from either Definition 22 or Definition 23—in exams students sometimes get confused about this.

Key phrases: Summary

We have the key ingredients that formal statements are made of, namely the key phrases which allow us to analyse their structure. Analysing the structure of a statement allows us to construct a blueprint for a proof of that statement. The key ideas are given in the text above; we give a summary in form of Table 2.1. By 'counterproof' we mean a proof that the statement does not hold. In the table S, $S1$ and $S2$ are statements, possibly containing further key phrases.

statement	**proof**	**counterproof**
$S1$ and $S2$	proof of $S1$ and proof of $S2$	counterproof for $S1$ or counterproof for $S2$
$S1$ or $S2$	proof of $S1$ or proof of $S2$	counterproof for $S1$ and counterproof for $S2$
if $S1$ then $S2$	assume $S1$ holds and prove $S2$	find situation where $S1$ holds and $S2$ does not
for all x, S	assume an arbitrary x is given and prove S for that x	give a specific x and show S does not hold for that x
there is x such that S	find a specific x and show that S holds for that x	assume you have an arbitrary x and show S does not hold for that x

Table 2.1: Key phrases and proofs

Tip

Every statement we might wish to prove, or disprove, is constructed from the key phrases. In order to find a blueprint for a proof, or counterproof, all we have to do is to take the statement apart, and follow the instructions from Table 2.1. We give a number of additional examples for more complex statements in the following sections. Note in particular the proof of Proposition 2.1 as an example of a lengthy proof of this kind.

One shouldn't think of the above as mere 'phrases'—they allow us to construct formal statements and come with a notion of how to establish proofs for these. This is what mathematics is all about. We look at an even more formal treatment of these ideas in the material on **logic**, which is taught after we are finished with the current chapter.

In the following sections we look at examples for such statements which give definitions which are important in their own right. The aim is for you to become familiar with the logical constructions as well as learning about the given example.

2.3 Properties of numbers

We begin by giving examples within some sets of numbers. You will need to use the definitions and properties from Chapter 0 here.

In the examples that follow on the left hand side we give running commentary on how to construct the proof that appears on the right hand side.

Example 2.28. We prove the following statement for integers x, y and z:

If x divides y then x divides $y \cdot z$.

This is an 'if … then' statement. Table 2.1 above tells us we should assume the first statement holds.

Assume that x and y are integers and that x divides y.

Sooner or later we have to apply the formal definition of divides to work out what this means.

By Definition 3 this means that there exists an integer m such that $x \cdot m = y$.

It is usually a good idea to write down what we have to prove, again expanding the definition of 'divides'.

We have to show that x divides $y \cdot z$, and by Definition 3 we have to show that there exists an integer n such that $x \cdot n = y \cdot z$.

We have to establish a 'there exists' statement, and Table 2.1 tells us we have to find a witness for which the statement is true. At this point one usually has to stare at the statements already written down to see whether there is an element with the right property hidden among them.

We have

$$\begin{aligned} y \cdot z &= (x \cdot m) \cdot z && \text{assmptn} \\ &= x \cdot (m \cdot z) && \text{Fact 1.} \end{aligned}$$

and so we have found an integer, namely $m \cdot z$ with the property that we may multiply it with x to get $y \cdot z$.

Example 2.29. Assume we are asked to prove

for all $x \in \mathbb{N}$, $2x$ is even,

Table 2.1 says to show a statement of the form 'for all …' we should assume we have a natural number n.

Let n be in $\mathbb{N}$.

So far so good. What about the statement $2x$ is even? At this point one should always look up the formal definition of the concepts used in the statements.

We have to show that $2x$ is even, by Definition 4 this means we have to show that 2 divides $2x$.

So now we have put in the definition of evenness, but that leaves us with divisibility, so we put in that definition.

By Definition 3 we have to show that there is $m \in \mathbb{N}$ with $2x = 2m$. We pick $m = x$ and so the claim is established.

Sometimes you are not merely asked to prove or disprove a statement, but you first have to work out whether you should do the former or the latter. This changes the workflow a little.

Example 2.30. Assume we are asked to prove or disprove the following statement for integers x, y and k.

If x divides z, and y divides z, then $x \cdot y$ divides z.

Now we first have to work out whether we want to prove the statement, or find a counterproof. Usually it's a good idea to do some examples. 2 divides 6 and 3 divides 6, and $2 \cdot 3 = 6$ divides 6, but 2 divides 2 and 2 divides 2, whereas $2 \cdot 2 = 4$ does not divide 2, so this statement is false.

But what does a formal argument look like in this case? Table 2.1 tells us that it is sufficient to find one way of picking x, y, and z which makes the claim false. We show how to use the counterexample we found informally to formally establish that the statement does not hold.

We note that

$$2 \cdot 1 = 2,$$

and so 2 divides 2 by Definition 3. We pick

$$x = y = z = 2.$$

For those choices, the above establishes that x divides z and that y divides z.
But $x \cdot y = 4$ and this number does not divide $z = 2$, hence the statement is false.

Example 2.31. Assume we are given the statement

there is an $x \in \mathbb{Z} \setminus \{0, 1\}$ such that $x + x = -(x \cdot x)$.

and are asked to prove or disprove it.

Do we believe the statement? If we try $2 + 2$ we get 4, but $-(2 \cdot 2) = -4$, and clearly we get a sign mismatch if we use any positive integer. But what about $x = -2$?

Table 2.1 tells us that all we have to do to give a proof for a 'there exists' statement is to find one witness for which the claim is true.

If we set $x = -2$ then we have

$$\begin{aligned} x + x &= -2 + (-2) && \text{def } x \\ &= -4 && \text{arithmetic} \\ &= -(2 \cdot 2) && \text{arithmetic} \\ &= -(x \cdot x) && \text{def } x. \end{aligned}$$

as required.

Exercise 23. Prove or disprove the following statements about divisibility for integers making sure to use Definition 3. Assume that x, y and z and w are integers. Follow the examples above in style (you don't have to give the running commentary).

(a) If x divides z and y divides w then $x \cdot y$ divides $z \cdot w$.

(b) If x^2 divides $y \cdot z$ then x divides y and x divides z.

(c) If x divides y and y divides z then x divides z.

(d) If x divides y and y divides x then $x = y$.

Here is a definition of a number being prime that will look different from the one you have seen before. The aim of this is to encourage you to follow the given formal definition, and not your idea of what it should mean.

Definition 17: prime

An element $x \neq 1$ of $\mathbb{N}$ (or $x \neq \pm 1$ in $\mathbb{Z}$) is **prime** if and only if for all elements y and z of $\mathbb{N}$ (or $\mathbb{Z}$) it is the case that

$$x \text{ divides } yz \qquad \text{implies} \qquad x \text{ divides } y \quad \text{or} \quad x \text{ divides } z.$$

Example 2.32. Assume that we have the statement

$$\text{for all } x \in \mathbb{N} \setminus \{0, 1\}, x \text{ is prime or } x \text{ is a multiple of } 2.$$

Do we believe the statement? Well, 0 and 1 have been excluded, so let's look at the next few numbers. We have that 2 is prime, 3 is prime, 4 is a multiple of 2, 5 is prime...

This looks good, but do we really believe this? Are there really no odd numbers which are not prime? The number 9 comes to mind.

So we want to give a counterproof. Table 2.1 tells us that we are looking for one x such that the statement does not hold.

Let $x = 9$.

To give a counterproof we have to show that 9 does not satisfy the claim. The two statements are connected with ‘or’, so according to Table 2.1 we have to show that *neither* holds.

We note that 9 is not prime since $9 = 3 \cdot 3$, so 9 divides $3 \cdot 3$ but 9 does not divide 3, which means that 9 does not satisfy Definition 17.

If 9 were even it would have to be divisible by 2 according to Definition 4. but since $9 \bmod 2 = 1$, Definition 3 tells us that this is not the case.

Hence this is a counterexample to the claim.

Exercise 24. Establish the following claims for prime numbers using Definition 17, and definitions and facts from Chapter 0.

(a) Show that if an element x of $\mathbb{N}$ is prime then

$$y \text{ divides } x \qquad \text{implies} \qquad y = 1 \text{ or } y = x.$$

(b) Show that if x and y are prime in $\mathbb{N}$, $x \neq y$, and z is any natural number then

$$x \text{ divides } z \qquad \text{and} \qquad y \text{ divides } z \qquad \text{implies} \qquad xy \text{ divides } z.$$

Compare this statement and its proof with Example 2.30.

(c) Show that if x is prime in $\mathbb{Z}$ then

$$y \text{ divides } x \qquad \text{implies} \qquad y = \pm 1 \text{ or } y = \pm x$$

Note that the converse of (b) and (c) are also true, that is, our definition of primeness is equivalent to the one you are used to. However, the proof requires a lot more knowledge about integers than I want to ask about here.

Example 2.33. Assume we are given the statement

There exists $x \in \mathbb{Z}$ such that x is a multiple of 3 and x is a power of 2.

Do we believe the statement? A bit of thinking convinces us that powers of 2 are only divisible by powers of 2, so they cannot be divisible by 3 and therefore they cannot be a multiple of 3. But how do we show that such a number cannot exist? This is a situation where what counts as a formal proof very much depends on what properties one may use.

The cleanest proof is via the prime factorization of integers (or corollaries thereof), but that is more than I want to cover in these notes.

In a situation where you cannot see how to write down a formal proof you should write something along the lines of the first paragraph written here. Never be afraid of expressing your thoughts in plain English!

If you were to start a formal proof it would look like something on the right.

Assume that x is a power of 2, that is, there exists $k \in \mathbb{N}$ such that $x = 2^k$.

If x is a multiple of 3 then there is $m \in \mathbb{Z}$ such that $x = 3m$.

Hence $2^k = x = 3m$.

This is where you would like to use that 3 cannot divide 2^k, but this requires a fact that is not given in Chapter 0. So the best you can do is to write what I wrote on the right.

This means that 3 divides 2^k, which is impossible.

CExercise 25. Which of the following statements are valid? Try to give a reason as best you can, following the previous examples. You should use the definitions from Chapter 0 for the notions of evenness and divisibility (and there is a formal definition of primeness above, but for this exercise you may use the one you are familiar with).

(a) For all $x \in \mathbb{N}$, x is even or x is odd.

(b) There exists $x \in \mathbb{N}$ such that x is even and x is a prime number.

(c) There exists a unique $x \in \mathbb{Z}$ such that x is even and x is a prime number.

(d) For all $x \in \mathbb{Z}$, x is divisible by 4 implies x is divisible by 2.

(e) For all $x \in \mathbb{Z}$, n is odd implies $x \bmod 4 = 1$ or $x \bmod 4 = 3$.

(f) There exists $x \in \mathbb{N}$ such that x is even implies x is odd.

(g) For all $x \in \mathbb{Z} \setminus \{-1, 0, 1, 3\}$ there exists y in $\mathbb{Z}$ such that $x \operatorname{div} y = 2$.

Examples for treating more complex statements, and giving more formal proofs, are given in the following sections.

2.4 Properties of Sets and their Operations

We use this opportunity to give more sample proofs for sets, but also note in particular the proof of Proposition 0.3 and Examples 2.12 and 2.15.

Example 2.34. Let S, S' and T be subsets of a set X. We show that

$$\text{if} \qquad S \subseteq S' \qquad \text{then} \qquad S \cup T \subseteq S' \cup T.$$

In order to show that one set is a subset of another we have to show that every element of the first set is one of the second. So, as suggested by Table 2.1 we begin by assuming we have an arbitrary element of the first set.

Let $x \in S \cup T$. By definition of $\cup$ this means that

$$x \in S \qquad \text{or} \qquad x \in T.$$

In the first case we know that $x \in S \subseteq S'$, so $x \in S'$, and in the second case we stick with $x \in T$. Hence the statement above implies that

$$x \in S' \qquad \text{or} \qquad x \in T,$$

which is equivalent to $x \in S' \cup T$ by the definition of $\cup$.

Exercise 26. Let S, S' and T be subsets of a set X. Assume that

$$S \subseteq S'.$$

Show the following statements.

(a) $S \cap T \subseteq S' \cap T$.

(b) $X \setminus S \supseteq X \setminus S'$.

More proofs involving sets and their operations are given below, see in particular Examples 2.36 and 2.39.

2.5 Properties of Operations

Functions that appear very frequently are *operations* on a set. Usually we are interested in *binary operations on a set* S, that is functions

$$S \times S \longrightarrow S.$$

Examples of such functions are

- addition and multiplication for $\mathbb{N}$,
- addition, and multiplication for $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ or $\mathbb{C}$, as well as the derived operation of subtraction,
- union and intersection of sets as functions from $\mathcal{P}X \times \mathcal{P}X$ to $\mathcal{P}X$,
- concatenation of strings in Python;

- concatenation of lists (see Section 6.1) over some set.

Note that we cannot define a division operation for rational, real or complex numbers, in the way that we define subtraction. We may not divide by 0 and so we can only define division as a function where the source has been adjusted, for example,

$$\begin{aligned} \mathbb{R} \times (\mathbb{R} \setminus \{0\}) &\longrightarrow \mathbb{R} \\ (x, y) &\longmapsto x \cdot y^{-1}, \end{aligned}$$

where

$$y^{-1}$$

is our notation for the multiplicative inverse of y.

These are operations we use all the time, and they are deserving of further study. Note that we typically write binary operations in *infix notation*, that is, we write the operation between its two arguments, such as $r + r'$, $c \cdot c'$.

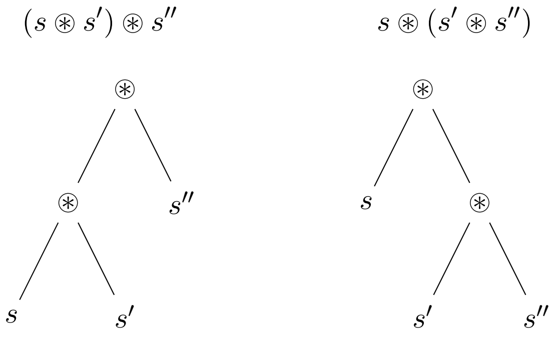
For what follows we need an *arbitrary* binary operation, where we make no assumptions about the kind of operation, or the set it is defined on. For that we use the symbol $\circledast$.

Definition 18: associative

A binary operation $\circledast$ on a set S is **associative** if and only, for all s, s', s'' in S it is the case that

$$(s \circledast s') \circledast s'' = s \circledast (s' \circledast s'').$$

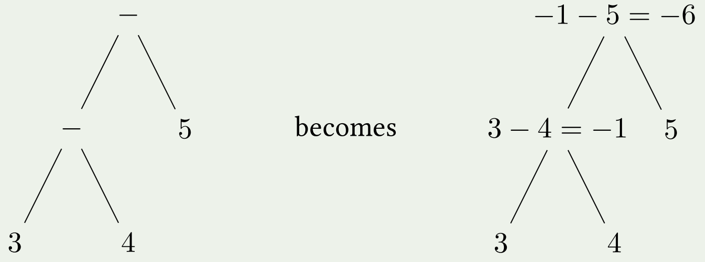
Why is this important? We use brackets to identify in which order the operations should be carried out. We can think of the two expressions as encoding a tree-like structure (known as a *parse tree*[7]), which tells us in which order to carry out the operations present in the expression.



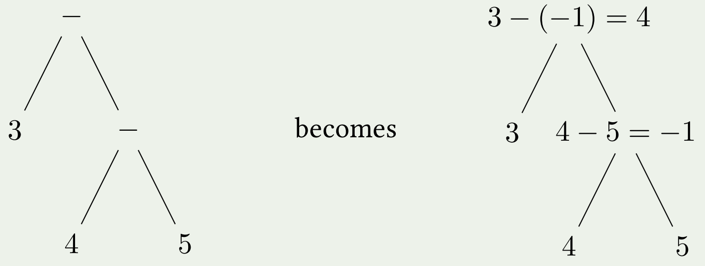
$(s \circledast s') \circledast s''$ $\qquad$ $s \circledast (s' \circledast s'')$

Example 2.35. Recall that $m - n$ is a shortcut for calculating $m + (-n)$, Using that derived operation as an example, we illustrate how one can think of this as allowing the filling in of the various steps of the calculation:

[7] Parse trees are studied in detail in COMP11212.



whereas



Knowing that an operation is associative means that both trees evaluate *to the same number* and therefore we *may leave out brackets* when using such an operation. It is safe to write

$$s \circledast s' \circledast s''$$

for such an operation.

This is important to computer scientists for two main reasons:

- When writing a program, leaving out brackets in this situation makes the code more readable to humans.
- When writing a compiler for a programming language, knowing that an operation is associative may allow significantly faster ways of compiling.

Note that if we write our operation as a binary function

$$f \colon S \times S \longrightarrow S$$

where we use prefix notation then associativity means that the following equality holds:

$$f(f(s, s'), s'') = f(s, f(s', s''))$$

Example 2.36. Assume we are given a set X. Recall from Section 0.2.4 that we may think of the union operation as a function

$$\cup \colon \mathcal{P}X \times \mathcal{P}X \longrightarrow \mathcal{P}X \ .$$

We show that this operation is associative.

The statement we wish to show is a 'for all' statement. Following Table 2.1 we assume that S, S' and S'' are (arbitrary) elements $\mathcal{P}X$. We calculate

$$(S \cup S') \cup S'' = \{x \in X \mid x \in S \text{ or } x \in S'\} \cup S'' \qquad \text{def union}$$

$$\begin{aligned}
&= \{x \in X \mid (x \in S \text{ or } x \in S') \text{ or } x \in S''\} && \text{def union}\\
&= \{x \in X \mid x \in S \text{ or } x \in S' \text{ or } x \in S''\} && \text{common sense}\\
&= \{x \in X \mid x \in S \text{ or } (x \in S' \text{ or } x \in S'')\} && \text{common sense}\\
&= S \cup \{x \in X \mid x \in S' \text{ or } x \in S''\} && \text{def union}\\
&= S \cup (S' \cup S'') && \text{def union}
\end{aligned}$$

Note that we have justified each step in the equalities used above—this ensures that we check we only use valid properties, and tells the reader why the steps are valid.

Note that we had to invoke 'common sense' in the example—usually this means that we are relying on definitions that are not completely rigorous mathematically speaking. What we have done in the definition of the union of two sets is to rely on the meaning of the English language. Only when we are down to that is it allowable to use 'common sense' as a justification (you might also call it 'the semantics of the English language'). In formal set theory there is formal logic to define the union of two sets, but we do not go to this level of detail here.

Example 2.37. Example 2.35 establishes that the derived operation of subtraction is *not* associative for the integers since it shows that

$$(3-4)-5 \neq 3-(4-5),$$

and to refute a 'for all' claim we merely need to give *one* counterexample.

Since we so far do not have formal definitions of addition and multiplication for $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ it is impossible to formally prove that these are indeed associative. You may use this as a fact in your work on this unit, apart from when you are asked to formally prove them in Chapter 6.[8]

CExercise 27. Work out whether the following operations are associative.

(a) Intersection for sets.

(b) Addition for complex numbers.

(c) Subtraction for complex numbers.

(d) Multiplication for complex numbers.

(e) Define the average ave of two real numbers r, r' as

$$\text{ave}(r, r') = \frac{r+r'}{2}.$$

Is this operation associative? Would you apply it to calculate the average of three numbers? If not, can you think of a better averaging function?

[8] Note that formal definitions, and proofs, of these properties for the natural numbers are given in Section 6.4.

(f) Multiplication of real numbers where every number is given up to *one* post-decimal digit, and where rounding takes place every time after a multiplication has been carried out.[9]

(g) The concatenation operator for strings (as, for example, implemented as + in Python).

(h) The and operator for boolean expressions in Python.

(i) Let S be a set and let $\mathsf{Fun}(S,S)$ be the set of all functions with source S and target S. Show that composition is an associative operation on the set $\mathsf{Fun}(S,S)$.

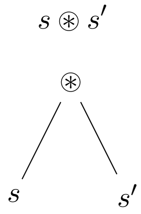
Some operations allow us even greater freedom: Not only is it unnecessary to provide brackets, we may also change the *order* in which the arguments are supplied.

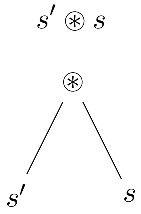
Definition 19: commutative

A binary operation $\circledast$ on a set S is **commutative** if and only if, for all s and s' in S we have

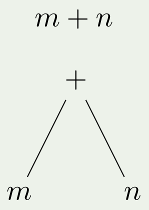
$$s \circledast s' = s' \circledast s.$$

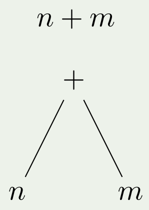
If an operation is commutative then it does not matter in which order arguments are supplied to it. Hence the two trees below will evaluate to give the same result.





Example 2.38. We know that when we have natural numbers m and n then





have the same number at the root of the tree, and so addition is a commutative operation.

Example 2.39. As in Example 2.36 we look at the union operation on the powerset $\mathcal{P}X$ for a given set X. We show that this operation is commutative.

Once more this is a statement of the 'for all ...' kind. Following Table 2.1

[9] When programming there is usually limited precision, and rounding has to take place after each step of the computation. While a computer has more precision, say for floating point numbers, the problems that occur are the same as here.

once again we assume that we have (arbitrary) elements S and S' of $\mathcal{P}X$. The union of S and S' is defined as follows:

$$S \cup S' = \{x \in X \mid x \in S \text{ or } x \in S'\}$$

and, once again invoking 'common sense', this is the same as

$$\{x \in X \mid x \in S' \text{ or } x \in S\} = S' \cup S.$$

Alternatively we can argue with more of an emphasis on the property of elements of the given sets:

$$\begin{array}{rlll} x \in S \cup S' & \text{if and only if} & x \in S \text{ or } x \in S' & \text{def } \cup \\ & \text{if and only if} & x \in S' \text{ or } x \in S & \text{logic} \\ & \text{if and only if} & x \in S' \cup S. & \end{array}$$

Example 2.40. Consider the following[10] operation for complex numbers: Given z and z' in $\mathbb{C}$ we set

$$z \circledast z' = \overline{z}z'.$$

The question is whether this operation is commutative.

First of all we have to work out whether we think it is true, and should try to prove it, or whether we should aim for a counterproof.

There are two approaches here: You can write down what this operation does in terms of real and imaginary parts which approach we follow in the following example, or you can think for a moment about what the conjugate operation does. It affects the imaginary part only, so if we have the product of one number with imaginary part 0, and one with imaginary part other than 0, there should be a difference. This suggests we should try a counterproof, that is, we should find one choice for z, and one for z', such that the statement becomes false.

The simplest numbers fitting the description given above, and which are distinct from 0, are 1 and i. We check

$$i \circledast 1 = \overline{i} \cdot 1 = -i \cdot 1 = -i,$$

and

$$1 \circledast i = \overline{1} \cdot i = 1 \cdot i = i.$$

Since $-i \neq i$ we have established that the given operation is not commutative.

Example 2.41. We give an alternative solution to the previous example. We calculate that

$$\begin{aligned} (a + bi) \circledast (a' + b'i) &= (a + bi)\overline{a' + b'i} \\ &= (a + bi)(a' - b'i) \\ &= (aa' + bb') + (-ab' + ba')i, \end{aligned}$$

[10] This appeared in a past exam paper.

whereas

$$\begin{aligned}(a'+b'i)\circledast(a+bi) &= (a'a+b'b)+(-a'b+b'a)i\\ &= (aa'+bb')+(ab'-a'b)i\\ &= (aa'+bb')-(-ab'+ba')i.\end{aligned}$$

So the two resulting numbers will have the same real part, but their imaginary parts will be the negatives of each other. Now it is important to remember that it is sufficient to find *just one* counterexample, and it is best to keep that as simple as possible. We pick

$$a=0 \qquad b=1 \qquad\qquad a'=1 \qquad b'=0,$$

and verify that this means

$$(a+bi)\circledast(a'+b'i)=i$$

and

$$(a'+b'i)\circledast(a+bi)=-i.$$

CExercise 28. Work out whether the following operations are commutative. If you think the answer is 'yes', give a proof, if 'no' a counterexample.

(a) Multiplication for complex numbers.

(b) Subtraction for integers.

(c) Division for real numbers different from 0.

(d) Set difference on some powerset.

(e) The ave function from the previous exercise.

(f) The concatenation operator for strings as for example implemented by + in Python.

(g) The and operator for boolean expressions in Python.

Some operations have an element which does not have any effect when combined with any other.

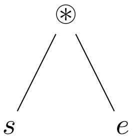
Definition 20: unit

Let $\circledast$ be a binary operation on a set S. An element e of S is a[11] **unit for** $\circledast$ if and only if it is the case that for all elements s of S we have

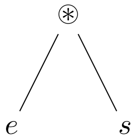
$$s\circledast e=s=e\circledast s.$$

If we want to picture this using a tree then it is saying that

[11] This is sometimes also known as the identity for the operation, but that terminology might create confusion with the identity function for a set.



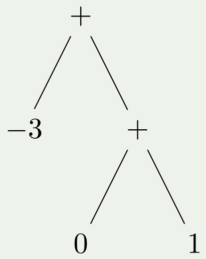
and

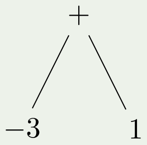


become s

This looks odd, but if you think of the first two trees as being part of a larger tree then this becomes a useful simplification rule.

Example 2.42. Knowing that 0 is the unit for addition for the integers we may simplify the tree on the left to become the tree on the right.





Example 2.43. We have already seen a number of examples of units. The number 0 is the unit for addition on all the sets of numbers we cover in these notes. This is one of the statements from Fact 1 (and corresponding facts about the other sets of numbers), since for all $x \in \mathbb{N}$ we have

$$x + 0 = x = 0 + x.$$

Example 2.44. If we look at the intersection operation for subsets of a given set X,

$$\cap\colon (S, S') \longrightarrow S \cap S',$$

we can show that X is the unit of this operation. For that we have to calculate, given an arbitrary subset S of X,

$$\begin{aligned} S \cap X &= \{x \in X \mid x \in S \text{ and } x \in X\} && \text{def} \cap \\ &= \{x \in X \mid x \in S\} \\ &= S. \end{aligned}$$

and

$$\begin{aligned} X \cap S &= \{x \in X \mid x \in X \text{ and } x \in S\} && \text{def} \cap \\ &= \{x \in X \mid x \in S\} \\ &= S. \end{aligned}$$

Hence the claim is true.

Working out whether a unit exists for some operation can be tricky. The existence of a unit is equivalent to the statement

$$\text{there exists } e \in S \text{ such that for all } s \in S \;\; s \circledast e = s = e \circledast s.$$

By Table 2.1 to refute such a statement we have to show that

$$\text{for all } e \in S \text{ there exists } s \in S \ \ (s \circledast e \neq s \text{ or } s \neq e \circledast s).$$

Statements like this are quite tricky to prove. The next two examples show how one might argue in such a situation. The strategy is to deduce properties that e would have to have (if it existed), and to then argue that an element with such properties cannot exist.

Example 2.45. Consider subtraction for integers, where $n - m$ is a shortcut for $n + (-m)$. Does this operation have a unit? Once again, we first have to decide whether we should try to give a proof or a counterproof.

The statement in question is of the kind 'there exists ...'. To prove such a statement we have to give an element with the required property. In a situation where we're not sure what such an element might look like, it is often possible to derive properties it needs to have. This is the strategy we follow here.

If the number we have e were a unit for subtraction we would require

$$n - e = n$$

for all elements n of $\mathbb{Z}$. The only number which satisfies this is $e = 0$, but if we calculate

$$0 - 1 = -1,$$

we see that this element cannot be the unit since we would require that number to be equal to 1 to satisfy $e - n = n$ for all $n \in \mathbb{N}$. Hence the given operation does not have a unit.

Note that the subtraction operation satisfies none of our properties! For this reason it is quite easy to make mistakes when using this operation, and that is why it is preferable to not to consider subtraction a well-behaved operation.

It is usually harder to establish that an operation does not have a unit, so we give another example for this case.

Example 2.46. Let us recall the set difference operation from Section 0.2 on $\mathcal{P}X$ for a given set X which, for S, S' in $\mathcal{P}X$, is defined as

$$S \setminus S' = \{s \in S \mid s \notin S'\}.$$

Does this operation have a unit? As in the previous example we derive properties that such a unit would have to have.

In order for $S \setminus S' = S$ to hold it must be the case that none of the elements of S occurs in S'. In particular if U were the unit we must have, instantiating S as X,

$$X \setminus U = \{x \in X \mid x \notin U\} = X,$$

which means that U must necessarily be empty. But for the empty set we have

$$\emptyset \setminus X = \{x \in \emptyset \mid x \notin X\} = \emptyset,$$

but for $\emptyset$ to be the unit this would have to be equal to X.

This means that no element of $\mathcal{P}X$ can satisfy the requirements for a unit for this operation.

The following exercise asks you to identify units for a number of operations, if they exist.

EExercise 29. Identify the unit for the following operations, or argue that there cannot be one:

(a) Union of subsets of a given set X.

(b) Multiplication for integers, rational, real and complex numbers.

(c) The operation from Example 2.40.

(d) The ave operation for the preceding two exercises.

(e) The concatenation operation for strings as, for example, implemented by + in Python.

(f) The and operator for boolean expressions in Python.

Note that mathematicians call a set with an associative binary operation which has a unit a **monoid**.

Exercise 30. Prove that there is at most one unit for a binary operation $\circledast$ on a set S. *Hint: Assume you have two elements that satisfy the property defining the unit and show that they must be equal.*

Exercise 31. Consider the set $\mathsf{Fun}(S, S)$ of all functions from some set S to itself. This has a binary operation in the form of function composition. If you have not already done so in Exercise 27 then show that this operation is associative. Find the unit for the operation. Conclude that we have a monoid. Further show that the operation is not commutative in general.

Definition 21: inverse element

Let $\circledast$ be an associative binary operation with unit e on a set S. We say that the element s' is an **inverse for $s \in S$ with respect to $\circledast$** if and only if we have

$$s \circledast s' = e = s' \circledast s.$$

Note that if

$$s^{-1} \qquad \text{is the inverse for } s \text{ with respect to } \circledast$$

then

$$s \text{ is the inverse of } s^{-1} \text{ with respect to } \circledast$$

since this definition is symmetric.

It is standard[12] to write s^{-1} for the inverse of s, but that convention changes if one uses the symbol + for the operation. In that case one writes $-s$ for the inverse of the element s with respect to the operation +.

Example 2.47. For addition on the integers the the inverse of an element n is $-n$, since

$$n + (-n) = 0 = -n + n,$$

and 0 is the unit for addition. The same proof works for the rationals and the

[12] This is the usual notation for $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{Z}$.

reals. For the complex numbers we have *defined* $-(a+bi) = -a - bi$, and shown that this is the additive inverse for $a + bi$ in Exercise 1.1.

Example 2.48. For addition on the natural numbers 0 is the unit for addition, but inverses do not exist in general. The number 0 is the only number that has an inverse.[13]

Exercise 32. Show that if $\circledast$ is a binary operation on the set S with unit e then e is its own inverse.

Example 2.49. For the rational or real numbers the multiplicative inverse of an element $r \neq 0$ is $r^{-1} = 1/r$. Note that when you use r^{-1}, or divide by r you must include an argument that r is not 0!

Example 2.50. In Chapter 1 we have proved that inverses exist for both, addition and multiplication for complex numbers, and we have shown how to calculate them for a given element. Recall that if you want to use z^{-1} you must include an argument that this exists,[14] that is, that $z \neq 0$.

Example 2.51. The proof that inverses for addition exist for integers, rationals, or reals, is very short: Given such a number r, we are so used to the fact that $r + (-r) = 0 = -r + r$ that it hardly feels as if this is a proof!

Example 2.52. To show that a given operation does not have inverses for every element one has to produce an element which does not have an inverse.

Assume that X is a set. Consider the intersection operation,

$$\begin{aligned} \cap \colon \mathcal{P}X \times \mathcal{P}X &\longrightarrow \mathcal{P}X \\ (S, S') &\longmapsto S \cap S'. \end{aligned}$$

The unit for this operation is given by X as established in Example 2.44. We show that the empty set does not have an inverse:

If S were an inverse for $\emptyset$ with respect to $\cap$ it would have to be the case that $S \cap \emptyset = X$. But $S \cap \emptyset = \emptyset$, and so as long as X is non-empty, an inverse cannot exist.

Exercise 33. For the following operations, give an argument why inverses do not exist.

(a) Union of subsets of a given set.

(b) The ave function from the previous exercises.

(c) The concatenation operation for strings.

[13] Think about why that is.

[14] Students have lost marks in exams for just dividing by some number z without comment.

(d) The and operation for boolean expressions in Python.

EExercise 34. Let S be a set with an associative binary operation $\circledast$, and assume that $e \in S$ is the unit for that operation.

(a) Show that if s_1 and s_2 have inverses then the inverse for the element $s_1 \circledast s_2$ is given by $s_2^{-1} \circledast s_1^{-1}$.

(b) Show that every element has at most one inverse. *Hint: Assume that there are two inverses and prove that they have to be the same.*

Note that mathematicians call a set with an associative binary operation with a unit, and where element has an inverse, a **group**. Groups are very nice mathematical entities, but most of the sets with a binary operation you will see will not have the full structure of a group (typically lacking inverses).

Optional Exercise 6. Assume that A is a set with a binary operation $\circledast$ which is associative and has a unit. Consider the set $\mathsf{Fun}(X, A)$ of all functions from some set X to A. Given two elements, say f and g of $\mathsf{Fun}(X, A)$, we define a new function which we call $f \circledast g$ in $\mathsf{Fun}(X, A)$ by defining for, $x \in X$,

$$(f \circledast g)x = fx \circledast gx,$$

(in other words the result of applying the new function to the argument x is to apply both, f and g to x and to combine the results by using the binary operation on A. This is known as defining an operation *pointwise* on as set of functions. Find the unit for this operation and show that it is one. If the operation on A is commutative, what about the one on $\mathsf{Fun}(X, A)$?

2.6 Properties of functions

Functions allow us to transport elements from one set to another. Section 0.3 gives a reminder of what you should know about functions before reading on. Recall Definition 14 which says that the graph of a function

$$f \colon S \longrightarrow T$$

is defined as

$$\{(s, fs) \in S \times T \mid s \in S\}.$$

This is the set we typically draw when trying to picture what a function looks like, at least for functions from sets of numbers to sets of numbers. The typical case for that is for S and T to be subsets of $\mathbb{R}$.

We can characterize all those subsets of $S \times T$ which are the graph of a function of the type $S \to T$.

Proposition 2.1

A subset G of $S \times T$ is the graph of a function from S to T if and only if

$$\text{for all } s \in S \qquad \text{there exists a unique } t \in T \text{ with} \qquad (s, t) \in G.$$

This statement requires a proof. We give one here as another example for how to use the key phrases in the statement to structure the proof.

We have an 'if and only if' statement, and we split the proof into two parts accordingly.

- Assume that[15] G is the graph of a function. We would like to have a name for that function, so we call it f, and note that if G is its graph then
$$G = \{(s, fs) \mid s \in S\}.$$
We have to show that G has the given property. This is a statement of the form 'for all ...', so following Table 2.1 we assume that we have an arbitrary $s \in S$. We now have to establish the remainder of the given statement. This is a 'unique existence' property, which means we have to show two things:
  - **Existence**. In order to show a 'there exists' statement Table 2.1 tells us we must find a witness for the variable, here t, with the desired property. We know that (s, fs) is in the graph G of f, and so we have found a witness in the form of $t = fs$ for the existence part.
  - **Uniqueness**. A uniqueness proof always consists of assuming one has two elements with the given property and showing that they must be equal. Assume we have t and t' in T so that (s, t) and (s, t') are both elements of G. We can see from the equality for G given above that the only element with first component s in G is the element (s, fs), and so we must have $t = fs = t'$ and we have established the uniqueness part.
- Assume that[16] G is a subset of $S \times T$ satisfying the given condition. We have to show that G is the graph of a function, and the only way of doing this is to
  - define a function f and
  - show that G is the graph of f.

  We carry out those steps in turns.
  - We would like to define a function $f\colon S \longrightarrow T$ by setting
$$f\colon s \longmapsto t \qquad \text{if and only if} \qquad (s, t) \in G.$$
We have to check that this definition produces a function, that is that there is precisely *one* output in T for every input from S. By the existence part of the assumed condition we know that for every $s \in S$ there is at least one element t of T with $(s, t) \in G$ and so there is indeed an output for every input. But by uniqueness we know that if (s, t) and (s, t') are in G then $t = t'$, so there is at most one element for every $s \in S$. Hence given an input our function creates the unique output required.

- It remains to check that G is the graph of f, and for that we note that by Definition 14 the graph of a function is given as follows.

$$\begin{aligned}
&\{(s, fs) \in S \times T \mid s \in S\} \\
&= \{(s,t) \in S \times T \mid (s,t) \in G\} && \text{def } f \\
&= G
\end{aligned}$$

This completes the proof.

Our concept of function from Chapter 0 says that a function

$$f\colon S \longrightarrow T$$

produces an output in T for every input from S. The proposition above tells us that this means that for every element of s we have a unique element of T, namely fs, which is associated with s.

Some functions have particular properties that are important to us.

Definition 22: injective

A function $f\colon S \longrightarrow T$ is[17] **injective** if and only if

$$\text{for all } s \text{ and } s' \text{ in } S \qquad fs = fs' \qquad \text{implies} \qquad s = s'.$$

Under these circumstances we say that f is an **injection**.

One way of paraphrasing[18] this property is to say that two different elements of S are mapped to two different elements of T. This means that knowing the result $fs \in T$ of applying f to some element s of S is sufficient to recover s.

Example 2.53. The simplest example of an injective function is the identity function

$$\mathsf{id}_S\colon S \longrightarrow S$$

for any set S. To prove this formally, note that we have a 'for all' statement, so we assume that s and s' are elements of S. We have to prove an implication, so assume the first part holds, that is, we have $\mathsf{id}_S s = \mathsf{id}_S s'$. But this implies that

$$s = \mathsf{id}_S s = \mathsf{id}_S s',$$

and so we have $s = s'$ as required.

Example 2.54. The function d from $\mathbb{N}$ to $\mathbb{N}$ given by

$$d\colon x \longmapsto 2x$$

is injective. To show this we have to show a 'for all' statement, so according to Table 2.1 we should assume that we have n, n' in $\mathbb{N}$. To show an implication,

[15] This direction is sometimes known as the 'forward' (in the sense that it shows that the first statement implies the second) or 'only if' direction.

[16] This direction is sometimes known as the 'backwards' or 'if' direction, in that it shows that the second given statement implies the first.

[17] Note that some people call such functions 'one-on-one' instead.

[18] But usually not a good way of attempting a proof.

the same table tells us we should assume the first part holds, so we assume that

$$dn = dn'.$$

But by inserting the definition of d this means that

$$2n = dn = dn' = 2n',$$

and by multiplying both sides with the multiplicative inverse of 2 we may conclude that $n = n'$.

Example 2.55. On the other hand the function from $\mathbb{R}$ to $\mathbb{R}$ given by

$$f\colon x \longmapsto 1$$

is not injective. In order to refute a 'for all' statement by Table 2.1 it is sufficient to produce a counter-example. This means we have to find two elements of $\mathbb{R}$. say r and r', such that the given implication does not hold.

The same table tells us that for the implication not to hold we must ensure that the first condition is true, which here means that $fr = fr'$, while the second condition is false, that is, we must have $r \neq r'$.

This is quite easy for our function: The numbers $r = 0$ and $r' = 1$ are certainly different, but we have

$$f0 = 1 = f1,$$

so we have indeed found a counterexample.

Example 2.56. Typical examples from the real world are unique identifiers, for example, student id numbers. We would expect every student to have a unique id number which is not shared with any other student.

This is certainly a desirable property, but to prove formally that it holds we would have to know how exactly the university assigns these numbers, and then we could check that. Nonetheless you should be able to work out whether real world assignments ought to be injective, and you should be able to come up with ways of testing this realistically, or write a program that confirms it.

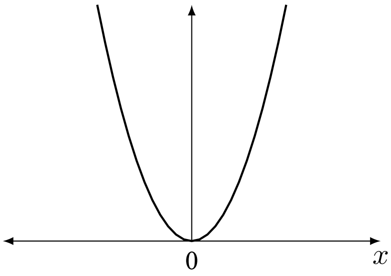
There are many other situations where we have to ensure this—for example, in a database we often want to have a unique key for every entry (for example the customer number).

Also, when casting an element of some datatype to another we expect that if we cast an `int` to a `double` in Java that two different `int` values will be cast to different `double` values. This operation should be performable without losing any information.

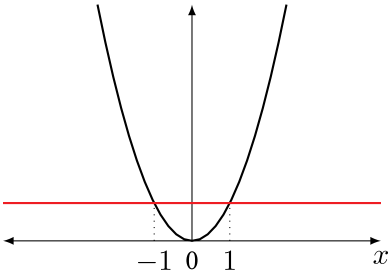
Example 2.57. Showing that a real world assignment is not injective has to be done by producing two witnesses. For example, the assignment that maps students to tutorial groups is not injective. To prove that all we have to do is to find two (different) students who are in the same tutorial group. You all know students like that.

We can also think of an injection as a ‘unique relabelling’ function: Every element from the source set S is given a new label from the target set T in such a way that no two elements of S are given the same label.

The graph of a function can be useful when determining whether a function is injective. For the squaring function described above the graph looks like this.



Whenever we can draw a horizontal line that intersects the graph of our function in more than one place then the function is not injective:



The x-coordinates of the two intersection points give us two different elements of $\mathbb{R}$ where the function takes the same value, namely here for 1 and -1, see example 2.58.

Note that one has to be careful when using the graph to determine whether a function is injective: Since most examples have an infinite graph it is impossible to draw all of it, so one has to ensure that there isn't any unwanted behaviour in the parts not drawn. Further note that a graph cannot provide a *proof* that a function is injective (or not), but it can help us make the decision whether we want to give a proof or a counterproof.

Example 2.58. We show that the function whose graph is given above is not injective. Consider the function

$$\begin{aligned} f\colon \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto x^2. \end{aligned}$$

Injectivity is a ‘for all’ statement. To give a counterproof by Table 2.1 all we have to do is to find witnesses s and s' such that the implication in the definition of injectivity does not hold. So as in Example 2.55 we are looking for two elements, say r and r' of $\mathbb{R}$, which are different, but which are mapped by the given function f to the same element.

As suggested by the graph, let $r = -1$, and let $r' = 1$. Then $fr =$

$(-1)^2 = 1$, and $fr' = 1^2 = 1$, and since $r = -1 \neq 1 = r'$ we have found a counterexample.

In the following example we show how a failing proof for injectivity can be turned into a counterexample for that property. This is a good strategy to follow if you cannot see from the definition of the given function whether it is injective or not.

Example 2.59. Assume we have the function

$$\begin{aligned} f\colon \mathbb{C} &\longrightarrow \mathbb{C} \\ x + yi &\longmapsto 2x - 2xyi. \end{aligned}$$

We would like to work out whether or not it is injective. We do this by starting with a proof to see if we can either complete the proof, or whether that leads us to a counterexample.

Assume we have $a+bi$, $a'+b'i$ in $\mathbb{C}$ which are mapped to the same element by f, that is

$$2a - 2abi = f(a+bi) = f(a'+b'i) = 2a' - 2a'b'i.$$

Since two complex numbers are equal if and only if their real and imaginary parts are equal this implies that

$$2a = 2a' \qquad \text{and} \qquad -2ab = -2a'b'.$$

From the first equality we may deduce that $a = a'$. However, the second equality says that

$$\begin{aligned} -2ab &= -2a'b' && \text{from above} \\ &= -2ab' && \text{since } a = a'. \end{aligned}$$

This implies that

$$ab = ab',$$

but that does *not* allow us to conclude that $b = b'$ since a might be 0. We can use the reason that this proof fails to help us construct a counterexample: We are unable to show that our two numbers have the same imaginary part if their real parts are 0. So if we use

$$a + bi = 0 + i \qquad \text{and} \qquad a' + b'i = 0 - i$$

we can see that

$$f(a+bi) = fi = 0 - 2 \cdot 0 \cdot 1 = 0$$

and

$$f(a'+b'i) = f(-i) = 0 - 2 \cdot 0 \cdot (-1) = 0,$$

and we have established that f is *not injective.*

Using something other than the original definition of injectivity is often problematic.

I sometimes see students paraphrase injectivity as 'for all elements of the source set there is a unique element of the target set which the function maps to'. This is *not* the property of injectivity, this is merely the definition of a function (compare Proposition 2.1). Sticking with the given definition is simpler.

If you do want to paraphrase injectivity using unique existence, then the valid formulation for a function

$$f\colon S \longrightarrow T$$

is:

for all t in the range of f
there exists a unique s in S
such that $fs = t$.

But this is more complicated than the original definition.

Exercise 35. Show that the statement above is equivalent to f being injective.

If there is an injection from some set S to some set T then we may deduce that T is at least as large as S. See the Section 5.2 for more detail, in particular Definition 47.

Exercise 36. Show that the following functions are injective or not injective as indicated.

(a) Injective: The function $x \longmapsto x + 0i$ from $\mathbb{R}$ to $\mathbb{C}$ defined on page 50.

(b) Not injective: The function $x \longmapsto 2x^2 - 4x + 1$ from $\mathbb{R}$ to $\mathbb{R}$.

(c) Not injective: The function from the set of first year CS student to lab groups $M + W$, $B + X$, Y and Z.

(d) Injective: The function from $\mathbb{C}$ to $\mathbb{C}$ which given by $x \longmapsto \overline{x}$.

CExercise 37. Determine which of the following functions are injective. You have to provide an argument with your answer. You should not use advanced concepts such as limits or derivatives, just basic facts about numbers. Where the function is not injective can you restrict the source set to make it injective?

(a) The $\sin$ function from $\mathbb{R}$ to $\mathbb{R}$.

(b) The $\log$ function from $[1, \infty)$ to $\mathbb{R}^+$.

(c) The function $x \longmapsto 2^x$ from $\mathbb{N}$ to $\mathbb{N}$, or from $\mathbb{R}$ to $\mathbb{R}$, you may choose.

(d) The function used by the School from the set of first year CS students to the set of tutorial groups.

(e) The function used by the University from the set of first year CS students to the set of user ids.

(f) The function $x \longmapsto x(-i)$ from $\mathbb{C}$ to $\mathbb{C}$.

(g) The function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$ which maps (n, m) to $2^n 3^m$.

(h) The function $x \longmapsto \{x\}$ from a set S to the powerset $\mathcal{P}S$.

EExercise 38. Establish the following properties.

(a) If S is a one-element set then every function which has it as a source is injective.

(b) The composite of two injective functions is injective.

(c) If $f\colon S \longrightarrow T$ and $g\colon T \longrightarrow U$ are two functions such that $g \circ f$ is injective then f is injective.

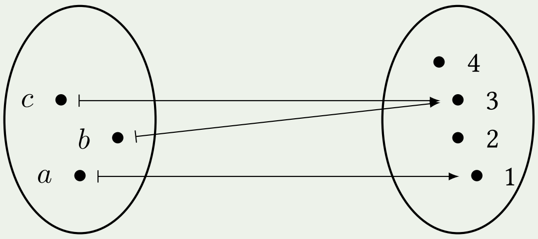
(d) Show for the previous statement that g need not be injective by giving an example.[19]

(e) Assume that $f\colon S \longrightarrow S'$ and $g\colon T \longrightarrow T'$ are both injections. Show that this is also true for

$$\begin{aligned} f \times g\colon S \times T &\longrightarrow S' \times T' \\ (s,t) &\longmapsto (fs, gt). \end{aligned}$$

In the case where we have a function from one small finite set to another we can draw a picture that makes it very clear whether or not the function is injective.

Example 2.60. Consider the function f defined via the following picture.



We can see immediately that b and c are mapped to the same element, 3, and so this function is not injective. Formally, we have found two elements with $fb = fc$, but $b \neq c$.

If a function is given by a picture like this, then all one has to do to check injectivity is to see whether any element in the target set has more than one arrow going into it. Exercise 43 invites you to try this technique for yourself.

Exercise 39. Show that if S is a set with finitely many elements, and $f\colon S \longrightarrow T$ is an injective function from S to a set T, then the image of S under f has the same number of elements as S.

The connection between injective functions and the sizes of sets is further explored in Section 5.2. Here is a second important property of functions.

[19] The smallest example concerns sets with at most two elements. You may want to read the next two paragraphs to help with finding one.

Definition 23: surjective

A function $f\colon S \longrightarrow T$ is[20] **surjective** if and only if

$$\text{for all } t \in T \qquad \text{there exists } s \in S \qquad \text{with} \qquad t = fs.$$

We also say in this case that f is a **surjection**.

In other words a function is surjective if its range is the whole target set, or, to put it differently, if its image reaches all of the target set.

We care that a function is surjective if we are using the source set to talk about members of the target set. It means that we can use it to access *all* the elements of the target set. If you are writing code that has to do something with all the elements of an array, for example, you must make sure that you write a loop that really does go through all the possible indices of the array. If you have programmed a graph, and you want to write an algorithm that visits each element of the graph, you must make sure that your procedure does indeed go to every such node.

Example 2.61. Once again, when we have real world example it is impossible to formally prove that a given assignment is a surjective function *unless* we know how it is defined. However, you should be able to tell whether the assignment ought to be surjective, and you should be able to come up with ways of testing this, and write a program that confirms it. If you construct a mailing list that emails all undergraduate students on a specific course unit you must make sure that your list contains all the students on that course.

Example 2.62. The simplest example of a surjective function is the identity function

$$\mathsf{id}_S\colon S \longrightarrow S$$

on a set S. We give a formal proof.

We have to show a 'for all' statement, so let s in the target of the function, which is S. We have to find a witness in the form of an element of the source set of the function which is mapped to s. For this we can pick s itself, since $\mathsf{id}_S s = s$.

Example 2.63. Consider the function

$$\begin{aligned} f\colon \mathbb{N} &\longrightarrow \{2n \in \mathbb{N} \mid n \in \mathbb{N}\} \\ x &\longmapsto 2x. \end{aligned}$$

In order to show that this function is surjective we have to show a statement of the 'for all there is' kind. By Table 2.1 we may do this by assuming that we have an arbitrary element for the 'for all' part, and then we have to find a witness so that the final part of the statement hold.

So let

$$m \in \{2n \in \mathbb{N} \mid n \in \mathbb{N}\}.$$

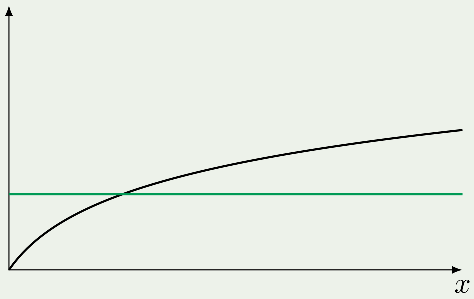
[20] Note that some people call such functions 'onto' instead.

By definition this means that there is $n \in \mathbb{N}$ with $m = 2n$. This n has the desired property since $m = 2n = fn$, and so is the required witness.

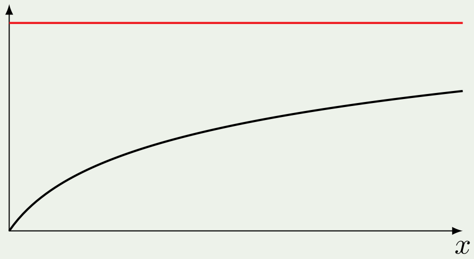
One can again take the graph of a function to help decide whether a given function is surjective. It can be tricky, however, to determine the answer from looking at the graph. Instead of looking whether there is a horizontal line which intersects the graph in at least two points we now have to worry about whether there is a horizontal line that intersects the graph not at all. For some functions this can by quite difficult to see.

Example 2.64. Consider for example the function from $\mathbb{R}^+$ to $\mathbb{R}^+$ given by

$$x \longmapsto \log(x + 1).$$



It is really difficult to judge whether some horizontal line will have an intersection with this graph or not. The picture above tells us that there is a number (namely 3) whose image is 2. But for the picture below it is far less clear whether there is an intersection between the line and the graph of the function.



You might argue that the problem would be solved if we drew a larger part of the graph, but then we could also move the horizontal line higher up (remember that one has to show that one can find an intersection for *every* horizontal line).

Example 2.65. We show formally that a surjective function is given by the previous example,

$$\begin{aligned} f\colon \mathbb{R}^+ &\longrightarrow \mathbb{R}^+ \\ x &\longmapsto \log(x + 1). \end{aligned}$$

We proceed following the same blueprint as in Example 2.63. Let r be an arbitrary element of the target set $\mathbb{R}^+$. We have to find an element x of the source set which is mapped to r, that is we are looking for $x \in \mathbb{R}^+$ such that

$$\log(x+1) = r.$$

We can solve this as an equation where r is given and x is unknown: This equation is true if and only if

$$x + 1 = 2^{\log(x+1)} = 2^r,$$

which holds if and only if

$$x = 2^r - 1.$$

Note that it is very easy, in a case like the above, to write something that is *not* a valid proof. The statement we need is that *if* we define

$$x = 2^r - 1 \qquad\qquad \textit{then} \qquad\qquad fx = r.$$

I have seen many student answers which say

$$\log(x+1) = r \qquad\qquad \text{so} \qquad\qquad x = 2^r - 1.$$

The important thing to note here is that the two statements are connected by an 'if and only if', that is, x satisfies the left hand equality if and only if it also satisfies the right-hand one. But in general, when students start with $fx = r$ and perform a number of steps to arrive at some statement for x, they typically have derived a necessary condition for x. Only when all these steps are reversible will defining x in the given way guarantee that it satisfies the original equation.

Otherwise it is necessary to take x defined in the given way and to check that it really does give a solution to the original problem.

Tip

A correct argument starts with a correct statement, and then applies a number of valid rules to get to the target statement. Implicitly this means that we read such arguments as the current line implying the next one.[21]

If you are constructing an argument which you intend the reader to interpret 'backwards', that is, the current line implies the *previous* line you have to indicate this in your text (and make sure your justifications work in the intended direction).

Example 2.66. The function

$$\begin{tikzcd} f\colon \mathbb{Z} \arrow[r] & \mathbb{Z} \\ x \arrow[r, mapsto] & x+1 \end{tikzcd}$$

[21] The fact that you can derive a valid statement from the given one does *not* imply anything about the validity of the given statement.

is surjective. Surjectivity is a statement of the 'for all …' kind, so following Table 2.1 we assume we are given $n \in \mathbb{Z}$.

The remainder of the surjectivity property is a 'there exists' statement, so by to the same table we have to find a witness, say $x \in \mathbb{Z}$. This witness has to satisfy $fx = n$. Inserting the definition of f, this means we need to pick x such that $x + 1 = fx = n$, so we pick $x = n - 1$ and this has the required property since

$$fx = f(n-1) = (n-1) + 1 = n,$$

which establishes that f is surjective.

Example 2.67. The function

$$\begin{aligned} f\colon \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto x^2 \end{aligned}$$

is not surjective.

To show this we want to find a counterproof to a statement of the 'for all …' kind. According to Table 2.1 means we have to find a witness r in the target $\mathbb{R}$ of the function that does not satisfy the remainder of the property.

Which property is this? It's a property of the 'there exists' kind, so following the same table we have to show that no x in $\mathbb{R}$ satisfies that $x^2 = r$.

Putting it like this should give us the right idea: We choose $r = -1$, and then no real number x can be squared to give r.

Alternatively, looking at the graph of this function, see Example 2.58, we can see that any negative number would work as the required witness.

In the following example we illustrate how a failing proof of surjectivity can be turned into a counterexample for that property.

Example 2.68. We again use the function from Example 2.59,

$$\begin{aligned} f\colon \mathbb{C} &\longrightarrow \mathbb{C} \\ x + yi &\longmapsto 2x - 2xyi, \end{aligned}$$

and look at the question of whether it is surjective. Once again we see how far we can get with a proof of that property.

For that we assume that we have an element of the target set, say $a + bi$. We have to find an element of the source set, say $x + yi$, with the property that

$$f(x + yi) = a + bi.$$

If such an $x + yi$ exists then it must be the case that

$$2x - 2xyi = f(x + iy) = a + bi.$$

Since two complex numbers are equal when both, their real and their imaginary parts, are equal we know that for this to be valid we must have

$$2x = a \qquad \text{and} \qquad -2xy = b.$$

We may think of these as equations in x and y that we are trying to solve. We can solve the first equation by setting

$$x = \frac{1}{2}a.$$

However, the second equation then becomes

$$\begin{aligned} b &= -2xy && \text{second equation} \\ &= -ay && \text{x} = a/2 \end{aligned}$$

and $-ay = b$ is an equation that we cannot solve when $a = 0$. Once again we can use this information to find a counterexample: If $a = 0$ and b is a number other than 0, say 1, then there is *no* element[22] of the source set that is mapped to $a + bi = i$:

Given an element of the source set $x + yi$, if

$$2x - 2xyi = f(x + iy) = a + bi = 0 + i$$

then it must be the case that

$$x = 0$$

to make the two real parts equal, but in that case we have that

$$2xy = 2 \cdot 0 \cdot y = 0,$$

which is not equal to the given imaginary part 1.

Tip

Proving that a function $f\colon S \longrightarrow T$ is surjective amounts to solving an equation: given $t \in T$ we have to find $x \in S$ with $fx = t$. You can think of x as the variable in that equation, and t as a parameter that is unknown but fixed. It may be a good idea to make sure that you give typical variable names, like x, y and z to the quantity you are trying to find, and typical 'parameter' names, like letters earlier in the alphabet, to the quantity which is given (but unknown).

If there is a surjection from a set S to a set T then we may deduce that T is at most as large as S. See Lemma 134 in Section 5.2.

Exercise 40. Show that the following functions are surjective or not surjective as indicated.

(a) Surjective: The function $x \longmapsto |x|$ from $\mathbb{Z}$ to $\mathbb{N}$.

(b) Surjective: The function used by the School from the set of first year CS students to the set of tutorial groups.

(c) Not surjective: The function used by the University from the set of all students currently in the university to the set of valid student id numbers.

[22] The argument given above already establishes that this is the case but I spell it out here again to make it clearer to see why that is.

(d) Not surjective: The function $x \longmapsto x + 0i$ from $\mathbb{R}$ to $\mathbb{C}$ given on page 50.

CExercise 41. For the following functions determine whether they are surjective and support your claim by an argument. You should not use advanced concepts such as limits or derivatives, just basic facts about numbers.

(a) The function from $\mathbb{Q}$ to $\mathbb{Q}$ given by

$$x \longmapsto \begin{cases} 0 & x = 0 \\ 1/x & \text{else.} \end{cases}$$

(b) The function from $\mathbb{R}$ to $\mathbb{R}$ given by $x \longmapsto x^4 - 100$.

(c) The function from $\mathbb{C}$ to $\mathbb{C}$ given by $x \longmapsto xi$.

(d) The function from $\mathbb{C}$ to $\mathbb{R}$ given by $x \longmapsto |x|$.

(e) The function that maps each first year CS student to their labgroup $W+M$, $B+X$, Y or Z.

(f) The function that maps each member of your tutorial group to one of the values E and W, depending on whether they were born in Europe (E) or in the rest of the world (W).

(g) The function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$ given by $(x, y) \longmapsto x$.

(h) The function from the *finite powerset* of $\mathbb{N}$,

$$\{S \subseteq \mathbb{N} \mid S \text{ has finitely many elements}\},$$

to $\mathbb{N}$ that maps S to the number $|S|$ of elements of S.

Exercise 42. Establish the following statements

(a) The composite of two surjections is an surjection.

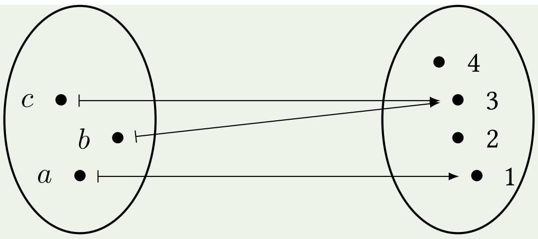
(b) If $f\colon S \longrightarrow T$ and $g\colon T \longrightarrow U$ are functions such that $g \circ f$ is surjective then g is surjective.

(c) Establish that in the previous statement f need not be surjective by giving an example.

(d) Assume that $f\colon S \longrightarrow S'$ and $g\colon T \longrightarrow U$ are both surjections. Show that this is also true for $f \times g$.

Again, if we are looking at functions between small finite sets then we can easily work out whether a function is surjective by drawing a picture.

Example 2.69. Consider the function given by the following diagram.



This function is not surjective since there is no element of the source set that is mapped to the element 4 of the target set.

For a function to be surjective all one has to check is that every element of the target set (on the right) has at least one arrow going into it. This example is not surjective.

CExercise 43. For the following functions draw a picture analogous to the above and determine whether or not it is injective and/or surjective.

(a) The function from $\{0, 1, 2, 3, 4\}$ to itself which maps the element i to $i \bmod 3$.

(b) The function from $\{0, 1, 2, 3, 4, 5, 6, 7\}$ to $\{0, 1, 2, 3\}$ which maps x to $x \bmod 4$.

(c) The function from $\{0, 1, 2, 3, 4\}$ to $\{n \in \mathbb{N} \mid n \leq 9\}$ which maps i to $2i$.

(d) The function from the set of members of your tutorial group to the set of letters from A to Z, which maps a member of the group to the first letter of their first name.

(e) The function that maps the members of your tutorial group to the set $\{\mathsf{M}, \mathsf{F}\}$ depending on their gender.

We need two further notions for functions. First of all there is a name for functions which are both, injective and surjective.

Definition 24: bijective

A function $f\colon S \longrightarrow T$ is **bijective** if and only if it is both, injective and surjective. We say in this case that it is a **bijection**.

Example 2.70. The simplest example of a bijective function is the identity function on a set S. Examples 2.53 and Example 2.62 establish that this function is both, injective and surjective.

Example 2.71. Consider the function f from $\mathbb{Z}$ to $\mathbb{Z}$ given by

$$x \longmapsto x + 1.$$

It is shown in Example 2.66 that this function is surjective and so it remains to show that it is also injective.

Following Table 2.1 to show a 'for all' statement we have to assume that we have arbitrary elements n and m in $\mathbb{Z}$, and that these have the property on the left hand side of the 'implies' statement, that is

$$fn = fm.$$

From this we wish to prove $n = m$. If we insert the definition of f then the given equality means that

$$n + 1 = fn = fm = m + 1,$$

and by deducting 1 on both sides we deduce

$$n = m.$$

This establishes that f is also injective, and so it is bijective.

Exercise 44. Determine which of the following functions are bijections. Justify your answer.

(a) The function from $\mathbb{Q}$ to $\mathbb{Q}$ given by

$$x \longmapsto \begin{cases} 0 & x = 0 \\ 1/x & \text{else} \end{cases}$$

(b) The function from $\mathbb{C}$ to $\mathbb{C}$ given by $x \longmapsto xi$.

(c) The function from $\mathbb{Z}$ to $\mathbb{N}$ given by $x \longmapsto |x|$.

(d) The function from $\mathbb{C}$ to $\mathbb{R}$ given by $x \longmapsto |x|$.

Exercise 45. Show that if $f\colon S \longrightarrow T$ and $g\colon T \longrightarrow U$ are two functions and $g \circ f$ is a bijection then f is an injection and g is a surjection.

Recall that we may think of a function that attaches to every element from the source set S a label from the target set T. A bijection is a very special such function.

- If the function is *injective* then we know that the label attached to each element of the source set is unique, that is, no other element of that set gets the same label.
- If the function is *surjective* then we know that all the labels from the target set are used.

If we have two sets with a bijection from one to the other then these sets have the same size—this idea is developed in Section 5.1, see in particular Exercise 133.

Whenever we have a bijection f there is a companion which undoes the effect of applying f. In other words, we get a function from the target set to the source set which reads the label and gives us back the element it is attached to.

Definition 25: inverse function

A function $g\colon T \longrightarrow S$ is the **inverse of the function** of $f\colon S \longrightarrow T$ if and only if

$$g \circ f = \mathrm{id}_S \qquad \text{and} \qquad f \circ g = \mathrm{id}_T.$$

Note that if g is the inverse function of f then f is the inverse function of g since the definition is symmetric.

Example 2.72. Consider the function

$$\begin{aligned} f\colon \mathbb{Z} &\longrightarrow \mathbb{Z} \\ x &\longmapsto x+1. \end{aligned}$$

In Example 2.71 it is shown that this function is a bijection. This function has an inverse (and indeed, Theorem 2.4 tells us that every bijection has an inverse).

To give this inverse we need to find a function which 'undoes' what f does, and the obvious candidate for this is the function g given by

$$x \longmapsto x-1.$$

We show that g is indeed the inverse function for f. Assume that $n \in \mathbb{Z}$. We calculate

$$\begin{aligned} (g \circ f)n &= g(fn) && \text{Definition 12} \\ &= g(n+1) && \text{def } f \\ &= (n+1)-1 && \text{def } g \\ &= n && \text{arithmetic} \end{aligned}$$

and so we know that $g \circ f = \mathrm{id}_{\mathbb{Z}}$. We also have to show that the other composite is the identity, so again assume we have $n \in \mathbb{Z}$. We calculate

$$\begin{aligned} (f \circ g)n &= f(gn) && \text{Definition 12} \\ &= f(n-1) && \text{def } g \\ &= (n-1)+1 && \text{def } f \\ &= n && \text{arithmetic,} \end{aligned}$$

so we also have $f \circ g = \mathrm{id}_Z$, and both equalities together tell us that g is indeed the inverse function for f.

We illustrate how the properties of a function from S to T say something about a function one may construct going from T to S: Note that the proposition tells us that for an injective function we can find a function which satisfies one of the two equalities required for inverse functions.

Proposition 2.2

The function $f\colon S \longrightarrow T$ is an injection if and only if either S is empty or we can find a function $g\colon T \longrightarrow S$ such that $g \circ f = \mathrm{id}_S$.

Proof. We begin by assuming that the function f is an injection.

If S is empty then the function f satisfies the definition of injectivity.

If S is non-empty we pick an arbitrary element $s_\bullet$ of S. Now given $t \in T$

we would like to define g as follows:

$$g\colon t \longmapsto \begin{cases} s & \text{if there is } s \in S \text{ with } fs = t \\ s_\bullet & \text{else} \end{cases}$$

First of all we have to worry whether this does indeed define a function—we need to ensure that in the first case, only one such s can exist. But since f is an injection we know that $fs = fs'$ implies $s = s'$ and so s is indeed unique. Hence our definition does indeed give us a function g.

Secondly we have to check that the equations for f and g holds as promised. Given $s \in S$ we calculate

$$\begin{aligned} (g \circ f)s &= g(fs) && \text{def composition} \\ &= s && \text{def } g \\ &= \mathsf{id}_S s && \text{def identity function,} \end{aligned}$$

which completes the proof.

Now assume that we have f, and a function g as given. We want to show that f is injective. Assume that we have s and s' in S such that

$$fs = fs'.$$

We apply g on both sides and obtain that

$$\begin{aligned} s &= \mathsf{id}_S s && \text{def } \mathsf{id}_S \\ &= (g \circ f)s && g \circ f = \mathsf{id}_S \\ &= g(fs) && \text{def } \circ \\ &= g(fs') && fs = fs' \\ &= (g \circ f)s' \text{def } \circ \\ &= \mathsf{id}_S s' && g \circ f = \mathsf{id}_S \\ &= s' \text{def } \mathsf{id}_S. \end{aligned}$$

If we have a surjective function we get a function that satisfies the other inequality for an inverse function:

Proposition 2.3

A function $f\colon S \longrightarrow T$ is surjective if and only if there exists a function g going in the opposite direction, that is $g\colon T \longrightarrow S$, such that $f \circ g = \mathsf{id}_T$.

Proof. We assume first that the function f is surjective. This means that for every t in T we can find $s \in S$ such that $fs = t$. Of course there may be many potential choices of such an s, depending on the function f. We define[a] g to be the function which gives us such an s for each input t. Then by definition we have for each t in T that

$$f(gt) = t$$

by the construction of g.

The proof that if we have a function g as described then f is surjective is given in the solution to Exercise 47.

[a] Strictly speaking this uses some non-trivial set theory, but we don't have time to worry about that here. Exactly what is needed depends on the proof that f is surjective.

Theorem 2.4

A function $f\colon S \longrightarrow T$ is a bijection if and only if it has an inverse function.

Proof. We carry out the proof in two parts to reflect the two directions of 'if and only if'.

Assume that f is a bijection. This means that, in particular, it is a surjection, so ' for every t in T there is $s \in S$ with $fs = t$. We would like to define $g\colon T \longrightarrow S$ by

$$t \longmapsto s,$$

for this s. A priori it is not clear that this defines a function—how do we know that there exists precisely one such s for each t?

Existence follows from surjectivity of f. Uniqueness comes from injectivity of that function: Assume we have s and s' in S with $fs = t = fs'$. This implies $s = s'$, and so we have indeed defined a function.

We next show that g is indeed the inverse of f.

To show that $g \circ f = \mathsf{id}_S$ let $s \in S$. Then

$$\begin{aligned} (g \circ f)s &= gfs && \text{def function comp} \\ &= s && \text{def } g \\ &= \mathsf{id}_S s && \text{def } \mathsf{id}_S \end{aligned}$$

The last but one step requires further elaboration. Recall that the definition of g is to map $t \in T$ to the unique $s \in S$ with $fs = t$. But this means that when g is applied to an element of the form fs it returns s.

To show that $f \circ g = \mathsf{id}_T$, let $t \in T$. Then

$$\begin{aligned} (f \circ g)t &= f(gt) && \text{def function comp} \\ &= t && \text{def } g \\ &= \mathsf{id}_T t && \text{def } \mathsf{id}_T \end{aligned}$$

Again the last but one step requires further justification. We have defined gt to be the unique element of S with $fs = t$, so by applying f on both sides we get $fgt = t$.

Now assume that we have an inverse function g for f. We have to show that f is both, an injection and a surjection. For the former, let $s, s' \in S$ with $fs = fs'$. Then

$$\begin{aligned} s &= \mathsf{id}_S s && \text{def identity function} \\ &= (g \circ f)s && g \circ f = \mathsf{id}_S \\ &= g(fs) && \text{def function composition} \\ &= g(fs') && fs = fs' \\ &= (g \circ f)s' && \text{def function composition} \end{aligned}$$

$$
\begin{aligned}
&= \mathsf{id}_S s' && g \circ f = \mathsf{id}_S \\
&= s' && \text{def identity function}
\end{aligned}
$$

To see that f is also surjective, let $t \in T$. Then $f(gt) = t$ since $f \circ g = \mathsf{id}_T$, so gt is an element with the property that applying f to it results in t.

Note that the proof given above combines the proofs of Propositions 2.2 and 2.3 with minor alterations.

Note that if we wish that a function is bijective we may use this result and instead produce an inverse function.

Example 2.73. Consider the function

$$
\begin{aligned}
f\colon \mathbb{C} &\longrightarrow \mathbb{C} \\
x &\longmapsto x + i.
\end{aligned}
$$

We show that this function has an inverse. We need to find a function that 'undoes' the action of f, which takes a complex number and moves it 'up' one unit by increasing the imaginary part by 1. To reverse that effect all one has to do is to move it 'down' by one unit, so we claim that

$$
\begin{aligned}
g\colon \mathbb{C} &\longrightarrow \mathbb{C} \\
x &\longmapsto x - i
\end{aligned}
$$

is the inverse of f.

The formal proof of this is not long. Based on Definition 25 we have to establish that

$$f \circ g = \mathsf{id}_{\mathbb{C}} \qquad \text{and} \qquad g \circ f = \mathsf{id}_{\mathbb{C}},$$

which by definition of the equality of two functions (compare Example 2.18) means establishing the two equalities that follow. Let $z \in \mathbb{C}$.

$$
\begin{aligned}
(f \circ g)(z) &= f(gz) && \text{def funct comp} \\
&= f(z - i) && \text{def } g \\
&= (z - i) + i && \text{def } f \\
&= z.
\end{aligned}
$$

$$
\begin{aligned}
(g \circ f)(z) &= g(fz) && \text{def funct comp} \\
&= g(z + i) && \text{def } f \\
&= (z + i) - i && \text{def } g \\
&= z.
\end{aligned}
$$

Hence we may conclude that the function f is bijective, as is the function g.

We give another example for a function that is injective and surjective, and show how to find its inverse.

Example 2.74. Assume we have the function

$$\begin{aligned} f\colon \mathbb{C} &\longrightarrow \mathbb{C} \\ x+yi &\longmapsto 2x-y+(x+2y)i. \end{aligned}$$

We want to know whether it is injective and/or surjective.

Injectivity

Assume we have two elements of the source set, say $a+bi$ and $a'+b'i$ which are mapped by f to the same element of the target set, that is

$$2a-b+(a+2b)i = f(a+bi) = f(a'+b'i) = 2a'-b'+(a'+2b')i.$$

This means that the real and imaginary parts of these two numbers must be equal, so we must have

$$2a-b = 2a'-b' \qquad \text{and} \qquad a+2b = a'+2b'.$$

The first equality gives us that

$$b' = 2(a'-a)+b,$$

and inserting that into the second equality gives

$$a+2b = a'+2(2(a'-a)+b) = 5a'-4a+2b.$$

We add $4a$ and subtract $2b$ on both sides to obtain

$$5a = 5a',$$

from which we may deduce, by dividing by 3 on both sides, that

$$a = a'.$$

Inserting this back into the equality for b' we get that

$$b' = 2(a'-a)+b = 2(a-a)+b = b,$$

and so we have established that overall,

$$a+bi = a'+b'i,$$

which means that our function is injective.

Surjectivity

Let us assume we have an element $a+bi$ of the target set. We want to find an element $x+yi$ of the source set with the property that

$$2x-y+(x+2y)i = f(x+yi) = a+bi,$$

so we try and find solutions for x and y. Again we know that the real and imaginary parts must be equal, so we may deduce that

$$2x - y = a \qquad \text{and} \qquad x + 2y = b.$$

We can see that for the first equation to hold it is sufficient that

$$y = 2x - a,$$

and inserting this into the second equation we get

$$5x - 2a = x + 2(2x - a) = x + 2y = b,$$

so if we set

$$x = \frac{b + 2a}{5},$$

and

$$y = 2x - a = \frac{2(b + 2a)}{5} - a = \frac{2b + 4a - 5a}{5} = \frac{2b - a}{5}$$

we have found x and y that solve our equation. It's a good idea to check that we haven't made a mistake, so we calculate

$$\begin{aligned}
&f\left(\frac{b+2a}{5} + \frac{2b-a}{5}\right) \\
&= \frac{2(b+2a) - (2b-a) + (b + 2a + 2(2b-a))i}{5} && \text{def f} \\
&= \frac{2b + 4a - 2b + a + (b + 2a + 4b - 2a)i}{5} && \text{calcs in } \mathbb{R} \\
&= \frac{5a + 5bi}{5} && \text{calcs in } \mathbb{R} \\
&= a + bi && r(a + bi) = ra + rbi.
\end{aligned}$$

Hence our function is indeed surjective.

Inverse function

Since we have established that f is bijective we know that it has an inverse function. That means that we want to define a function

$$g\colon \mathbb{C} \longrightarrow \mathbb{C}$$

with the property that

$$f \circ g = \mathsf{id}_{\mathbb{C}} \qquad \text{and} \qquad g \circ f = \mathsf{id}_{\mathbb{C}}.$$

The second equality tells us that g has to undo the effect of f, and can use the work we did to show that f is surjective to help us. There we answered the question of which element $x + yi$ is mapped by f to a given element $a + bi$ of the target set, which amounts to also answering the question of how to undo the effect f had on its input to give the output $a + bi$.

In other words we want to write an assignment that maps $a + bi$ to $x + yi$, where $x + yi$ is the solution we worked out above. The real part x of the result has to be equal to $(b + 2a)/5$, while the imaginary part y of the result has to be equal to $(2b - a)/5$, so we set

$$g\colon a + bi \longmapsto \frac{1}{5}\left(b + 2a + (2b - a)i\right).$$

We formally show that this g is indeed the inverse function for f.
Let $a + bi \in \mathbb{C}$. Then

$$\begin{aligned}
&g(f(a+bi)) \\
&= g(2a - b + (a+2b)i) && \text{def } f \\
&= \frac{(a+2b) + 2(2a-b) + (2(a+2b) - (2a-b))i}{5} && \text{def } g \\
&= \frac{a + 2b + 4a - 2b + (2a + 4b - 2a + b)i}{5} && \text{calcs in } \mathbb{R} \\
&= \frac{5a + 5bi}{5} && \text{calcs in } \mathbb{R} \\
&= a + bi && r(a+bi) = ra + rbi,
\end{aligned}$$

while also

$$\begin{aligned}
&f(g(a+bi)) \\
&= f\left(\frac{b+2a}{5} + \frac{2b-a}{5}\right) && \text{def } g \\
&= \frac{2(b+2a) - (2b-a) + (b + 2a + 2(2b-a))i}{5} && \text{def f} \\
&= \frac{2b + 4a - 2b + a + (b + 2a + 4b - 2a)i}{5} && \text{calcs in } \mathbb{R} \\
&= \frac{5a + 5bi}{5} && \text{calcs in } \mathbb{R} \\
&= a + bi && r(a+bi) = ra + rbi.
\end{aligned}$$

Note how the second proof is almost identical to the one at the end of the surjectivity argument. So when we do a surjectivity proof then if our function is also injective we get

- the assignment that gives us the inverse function and
- one of the two proofs that it is indeed the inverse function.

Sometimes giving an inverse function can be easier than doing separate injectivity and surjectivity proofs. If you can give an inverse function to a given function then you may use Theorem 2.3 to argue that the given function is bijective.

Exercise 46. Let $f\colon S \longrightarrow T$ be a function. Let $f[S]$ be the image of S under f in T (also known as the range of f, see Definition 13). We may define a function f' as follows.

$$\begin{aligned}
f'\colon S &\longrightarrow f[S] \\
s &\longmapsto fs.
\end{aligned}$$

Show that if f is injective then f' is a bijection.

EExercise 47. Calculate the inverse for the function from $\mathbb{C}$ to $\mathbb{C}$ given by

$$x + yi \longmapsto 2x - y^3 i$$

and show it is the required inverse.

Without using Theorem 2.4 show how you can use the inverse function to give a surjectivity proof. You can either do that for the function given, or in general which completes the proof of Proposition 2.3. Use the inverse function to show that the given function is surjective.

Exercise 48. Recall from Exercise 31 the set $\mathsf{Fun}(S, S)$ of all functions from a set S to itself. We define a subset of this set

$$\mathsf{Bij}(S, S) = \{f \in \mathsf{Fun}(S, S) \mid f \text{ is a bijection}\}.$$

Show that the composite of two bijections is a bijection. This means that we can use function composition to define a binary operation

$$\mathsf{Bij}(S, S) \times \mathsf{Bij}(S, S) \longrightarrow \mathsf{Bij}(S, S),$$

which is again a monoid. Show that the inverse function of an element of $\mathsf{Bij}(S, S)$ which is known to exist by Theorem 2.4 is its inverse with respect to the function composition operation. Conclude that $\mathsf{Bij}(S, S)$ is a group (under the composition operation).

Chapter 3

Formal Systems

The previous chapter concentrates on statements as they are commonly made in mathematics, using key phrases from the English language, and how to prove such statements.

Much of the discipline of mathematical logic is concerned with *formal systems*, which use a language of particular symbols to mimic (and formalize) this reasoning. Variations of such systems are often used in computer science to make precise statements.

This chapter introduces a formal system, and then discusses how to model it, as well as the question of which formulae should be considered equivalent.

In programming languages a number of the rules that are derived in Section 3.3 have been implemented, giving another reason why this material is important from a computer science perspective.

3.1 A system for propositional logic

We first outline a comparatively simple system known as propositional logic. The entities in the formal system we are about to describe are known as **propositions**.[1] Propositions are built using **connectives** (which serve to connect propositions). See Section 6.3.2 for a formal definition of how propositions are built.

3.1.1 Connectives

Conjunction, $\wedge$. Assuming we have propositions A and B we may form a new proposition, namely

$$A \wedge B.$$

The intended meaning is that this is a formalization of 'and' from the previous chapter, but nothing we have done so far indicates this.

Disjunction, $\vee$. If we have propositions A and B we may form the new proposition

$$A \vee B.$$

The intended meaning for this is a formalization of 'or' from the previous chapter.

[1] Sometimes these are referred to as formulae.

Implication, $\rightarrow$. If we have propositions A and B we may form the new proposition[2]

$$A \rightarrow B.$$

The intended meaning for this is ‘implies’ from the previous chapter.

So how do we make it clear that these symbols have an intended meaning? There are a number of ways of doing this. One of them is to give rules that govern reasoning with propositions. We here give (parts of) an **inference system** known as **natural deduction**.

Rules are of the form

$$\frac{\text{proposition } 1 \quad \text{proposition } 2 \quad \dots \quad \text{proposition } n}{\text{proposition } n+1}$$

Here the propositions[3] above the line are the *premises* and the proposition below the line is the *conclusion* of the rule. We think of this as saying

> ‘under the assumption that we have the premises we get the conclusion’.

For conjunction we have the following *introduction rule*:

$$\frac{A \quad B}{A \wedge B}$$

Think of this as ‘if we know A and we know B, then we have $A \wedge B$’.

We also have two *elimination rules*:

$$\frac{A \wedge B}{A} \qquad \frac{A \wedge B}{B}$$

Think of this as ‘if we have $A \wedge B$ then we have A (or B)’.

Contrast this with the rules for disjunction. Introduction:

$$\frac{A}{A \vee B} \qquad \frac{B}{A \vee B}$$

Think of this as ‘if we have A (or B) then we have $A \vee B$’.

But this is where our simple system runs into problems. With what we have done so far we cannot give an elimination rule for $A \vee B$—if we know $A \vee B$ we may not deduce anything that we can express. This is because what we have so far is not expressive enough.[4]

3.1.2 Propositions, judgements, inferences

First of all we have to revisit our notion of proposition. In the above we only say how we may build new propositions from once we already have (using the connectives), but not how to create a proposition from nothing.

Propositional variables

In order to ensure that our propositions are interesting, and to get started, we have to allow them to contain *variables*. We therefore assume that there is a list of propositional variables, Z_1, Z_2, and so on. Then the definition of a proposition is as follows.

[2] You may find a few other notations for this, such as $A \Rightarrow B$ or $A \supset B$.

[3] Note that some people prefer to refer to *propositional formulae* rather than proposiitons.

[4] It also does not allow us to give rules for implication.

- If Z is a propositional variable then Z is a proposition.
- If A is a proposition, then $\neg A$ is a proposition.[5]
- If A and B are propositions[6]
  - then $(A \wedge B)$ is a proposition,
  - also $(A \vee B)$ is a proposition
  - and $(A \to B)$ is a proposition.

This is an example of a *recursive definition*. We study such definitions formally in Chapter 6, and Section 6.3.2 talks about how propositions are defined recursively.

.

Note that we have two kinds of variables,

$$Z,\ Z',\ Z'',\ \ldots \qquad \text{or} \qquad Z_1,\ Z_2,\ Z_3,\ \ldots$$

which range over propositional variables and

$$A,\ B,\ C,\ \ldots\ldots \qquad \text{or} \qquad A_1,\ A_2,\ A_3,\ \ldots$$

which range over propositions. Do not confuse the two—their meaning is quite different.

Judgements

In order to express[7] rules for the connectives we build a *more complicated system*, one which has inferences for **judgements**[8] of the form

$$A_1, A_2, \ldots, A_n \vdash B.$$

We have a *list of propositions* on the left of the turnstile symbol $\vdash$ and one proposition on the right. We may think of this as saying:

'If we have A_1, A_2, …, A_n then we have B.'

The A_i are known as the *antecedents* and B as the *consequent* of the given judgement. It makes sense to think of the A_i as *assumptions*,

'if we may assume A_1,…, A_n then we get B'

being another way to think about the judgement above. Because writing lists of propositions can be a bit tedious it is customary to use capital Greek letters such as Γ, Δ instead.

[5] We look at the meaning of this symbol, known as negation, when we talk about its rules on page 120.

[6] Note that we often leave out brackets where they are not required to parse the term.

[7] There are other solutions for natural deduction systems but these are more liable to lead to confusion.

[8] This is also known as a *sequent*.

Inference rules

In this new system our rules from above become:

Conjunction introduction $\wedge$I:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,\wedge\mathrm{I}$$

‘If given Γ we have A and given Γ we also have B, then from Γ we have $A \wedge B$.’

Conjunction elimination $\wedge\mathrm{E}_l$ and $\wedge\mathrm{E}_r$:

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\,\wedge\mathrm{E}_l \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\,\wedge\mathrm{E}_r$$

‘If given Γ we have $A \wedge B$ then given Γ we have A (or B).’

Disjunction introduction $\vee\mathrm{I}_l$ and $\vee\mathrm{I}_r$:

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\,\vee\mathrm{I}_l \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\,\vee\mathrm{I}_r$$

‘If given Γ we have A (or B) then given Γ we have $A \vee B$.’

Disjunction elimination $\vee$E:

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\,\vee\mathrm{E}$$

‘If given Γ we have $A \vee B$, and given Γ and A, or Γ and B, we have C, then from Γ we have C.’

For implication we have the introduction rule $\to$I:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\,\to\mathrm{I}$$

‘If given Γ and A we have B, then given Γ we have $A \to B$.’

The elimination rule for implication[9] $\to$E:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash A \to B}{\Gamma \vdash B}\,\to\mathrm{E}$$

‘If given Γ we have A and given Γ we have $A \to B$, then from Γ we have B.’

Note that each rule has a name, given to the right of the horizontal line. These rules are known as (some of) the **inference rules** of our system.

All the rules we have seen so far only tell use how to build new judgements assuming we already have one, inviting the question: How does one get started? In other words, which judgements do we accept without requiring any premises?

There is a very obvious rule for this purpose, known as the ‘identity rule I:

$$\frac{}{A \vdash A}\,\mathrm{I}$$

‘Given A we have A.’

There are a couple of rules known as ‘structural rules’.

[9] This is also known as *modus ponens*.

$$\frac{\Gamma \vdash A}{\Gamma, B \vdash A}\,\mathrm{W} \qquad \frac{\Gamma, B, B \vdash A}{\Gamma, B \vdash A}\,\mathrm{C} \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\,\mathrm{P}$$

The first of these, W for *weakening*, tells us that we may always add antecedents (and surely if we know that Γ allows us to deduce A, then Γ and an extra assumption B will allow us to deduce A). The second, C for *contraction*, says that if we have an assumption once then repeating it does not give us anything more. The final rule, P for *permutation*, says we may change the order of the assumptions.

There is one more logical construct missing, namely ‘negation’. In our ‘key phrases’ in Chapter 2 this did not appear, but note that for every phrase we discussed how to prove that a statement involving it is *not* valid. That is the equivalent of the logical $\neg$. So if A is a proposition then $\neg A$ is a proposition, the **negation of** A. We again have an introduction and an elimination rule.

Negation introduction $\neg$I:

$$\frac{\Gamma, A \vdash \neg B \quad \Gamma, A \vdash B}{\Gamma \vdash \neg A}\,\neg\mathrm{I}$$

‘If given Γ and A we have both $\neg B$ and B, then given Γ we have $\neg A$.’

Negation elimination $\neg$E:

$$\frac{\Gamma, \neg A \vdash \neg B \quad \Gamma, \neg A \vdash B}{\Gamma \vdash A}\,\neg\mathrm{E}$$

‘If given Γ and $\neg A$ we have $\neg B$ as well as B then from Γ we have A.’

Derivations

In this system we may build formal **derivations**.

Example 3.1. A derivation is a way of getting to propositions from the axioms using the derivation rules given above.

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\mathrm{I}}{A, B \vdash A}\,\mathrm{W} \qquad \dfrac{\dfrac{\dfrac{}{B \vdash B}\,\mathrm{I}}{B, A \vdash B}\,\mathrm{W}}{A, B \vdash B}\,\mathrm{P}}{A, B \vdash A \wedge B}\,\wedge\mathrm{I}$$

This tells us that from nothing at all we may deduce that if we have A and B then have $A \wedge B$. This may appear terribly obvious, but once one adds more rules to the system and studies longer derivations it becomes much less clear what may be derived in such a system.

A derivation is then a tree where the inference rules have been employed on each step down the tree. Note that in the derivation each step gives the name of the rule used. You can see two examples of significantly larger derivations on the following two pages.

$$
\dfrac{
  \dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\,\mathrm{I}}{B, A \to (B \to C) \vdash B}\,\mathrm{W}}{A \to (B \to C), B \vdash B}\,\mathrm{P}}{A \to (B \to C), B, A \vdash B}\,\mathrm{W}
  \qquad
  \dfrac{
    \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\mathrm{I}}{A, A \to (B \to C) \vdash A}\,\mathrm{W}}{A, A \to (B \to C), B \vdash A}\,\mathrm{W}}{A \to (B \to C), A, B \vdash A}\,\mathrm{P}}{A \to (B \to C), B, A \vdash A}\,\mathrm{P}
    \qquad
    \dfrac{\dfrac{\dfrac{}{A \to (B \to C) \vdash A \to (B \to C)}\,\mathrm{I}}{A \to (B \to C), B \vdash A \to (B \to C)}\,\mathrm{W}}{A \to (B \to C), B, A \vdash A \to (B \to C)}\,\mathrm{W}
  }{A \to (B \to C), B, A \vdash B \to C}\,{\to}\mathrm{E}
}{
  \dfrac{
    \dfrac{A \to (B \to C), B, A \vdash C}{A \to (B \to C), B \vdash A \to C}\,{\to}\mathrm{I}
  }{A \to (B \to C) \vdash B \to (A \to C)}\,{\to}\mathrm{I}
}\,{\to}\mathrm{E}
$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\mathrm{I}}{A, (A \lor B) \to C \vdash A}\,\mathrm{W}}{(A \lor B) \to C, A \vdash A}\,\mathrm{P}}{(A \lor B) \to C, A \vdash A \lor B}\,{\lor}\mathrm{I} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{(A \lor B) \to C \vdash (A \lor B) \to C}\,\mathrm{I}}{(A \lor B) \to C, A \vdash A}\,\mathrm{W}}{A, (A \lor B) \to C \vdash (A \lor B) \to C}\,\mathrm{P}}{(A \lor B) \to C, A \vdash (A \lor B) \to C}\,\mathrm{P}}{\dfrac{(A \lor B) \to C, A \vdash C}{(A \lor B) \to C \vdash A \to C}\,{\to}\mathrm{I}}\,{\to}\mathrm{E} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\,\mathrm{I}}{B, (A \lor B) \to C \vdash B}\,\mathrm{W}}{(A \lor B) \to C, B \vdash B}\,\mathrm{P}}{(A \lor B) \to C, B \vdash A \lor B}\,{\lor}\mathrm{I_r} \quad \dfrac{\dfrac{}{(A \lor B) \to C \vdash (A \lor B) \to C}\,\mathrm{I}}{(A \lor B) \to C, B \vdash (A \lor B) \to C}\,\mathrm{W}}{\dfrac{(A \lor B) \to C, B \vdash C}{(A \lor B) \to C \vdash B \to C}\,{\to}\mathrm{I}}\,{\to}\mathrm{E}}{(A \lor B) \to C \vdash (A \to C) \land (B \to C)}\,{\land}\mathrm{I}$$

EExercise 49. Give derivations for the following judgements.

(a) $\neg\neg A \vdash A$,

(b) $A \vdash \neg\neg A$,

(c) $\neg(A \wedge B) \vdash \neg A \vee \neg B$,

(d) $\neg A, \neg B \vdash \neg(A \vee B)$.

Instead of only considering derivations which have only instances of the identity rule at the top we may consider 'partial' derivations, where we assume we already have some judgements. I

Example 3.2. Assume both,

$$\Gamma \vdash \neg A \qquad \text{and} \qquad \Gamma \vdash B \to A.$$

Then we can build a derivation using these judgements as starting points.

$$\dfrac{\dfrac{\Gamma \vdash \neg A}{\Gamma, B \vdash \neg A}\,\text{w} \qquad \dfrac{\dfrac{\dfrac{}{B \vdash B}\,\text{I}}{\Gamma, B \vdash B}\,\text{W,P sev times} \qquad \dfrac{\Gamma \vdash B \to A}{\Gamma, B \vdash B \to A}\,\text{W}}{\Gamma, B \vdash A}\,{\to}\text{E}}{\Gamma \vdash \neg B}\,\neg\text{E}$$

EExercise 50. This exercise is about giving partial derivations.

(a) Derive, from the judgements $\Gamma \vdash B$ and $\Gamma \vdash \neg B$ the judgement $\Gamma \vdash A$ for arbitrary A.

(b) Derive, from the judgement $\Gamma \vdash A$, the judgement $\Gamma \vdash \neg\neg A$.

(c) Derive, from the judgement $\Gamma \vdash \neg\neg A$, the judgement $\Gamma \vdash A$.

(d) Derive, from $\Gamma \vdash A \to B$ and $\Gamma \vdash \neg B$, the judgement $\Gamma \vdash \neg A$.

(e) Derive, from $\Gamma \vdash A \vee B$ and $\Gamma \vdash \neg A$, the judgement $\Gamma \vdash B$.

The system we have given here is one for **classical logic**.[10] It aims to only use principles that are generally accepted as valid in its rules, so the idea is that everything that can be derived in such a system is valid in some sense.

A judgement of the form

$$\vdash A$$

tells us that the proposition A is valid without requiring any assumptions. If we can derive this judgement in our system then this means it is a this is a 'true' proposition in some sense—at least if we believe the rules of our system to be valid. This gives rise to an obvious query.

Question. Which judgements of the form $\vdash A$ have a derivation in our system?

We call such propositions **derivable**. There is a theorem which connects arbitrary derivable judgements with those where the list of assumptions is empty.

[10] Technically speaking it gives the rules for natural deduction in a sequent calculus with a single consequent in each judgement.

Theorem 3.1

The judgement

$$A_1, A_2, \dots, A_n \vdash B$$

is derivable in this system if and only if the same is true for the judgement

$$\vdash (A_1 \wedge A_2 \wedge \dots \wedge A_n) \to B.$$

The original aim of systems like this were to formalize the kind of reasoning that occurs in mathematical proofs. The idea was to write down rules which 'make sense', with a hope to translating proofs in mathematics into such a formal system, and with the more modern hope of *verifying* proofs carried out by humans through machines.

Moreover, one could then study the system in its own right, with a view to finding out which judgements are derivable within it, and that one might be able to prove formally that in such a system, nothing false (or contradictory) may ever be deduced. It turns out that the latter is still very much unfinished.[11]

Meanwhile the associated formalisms have been widely used in other disciplines, including computer science, to make precise statements. Examples of this are given in Section 3.4 once the system has been expanded further.

While in what follows we look at this system through models it is worth pointing out that there is a branch of mathematical logic known as proof theory where such systems are studied in their own right: Different derivations of the same propositions are thought to play a similar role to different programs for the same problem.

3.2 Models

All that is defined above is purely syntactic, that is, we are manipulating symbols, and while we may have a meaning in mind for these symbols, nothing we have done so far gives any meaning to a judgement. The fact that each connectives has particular inference rules does provide some meaning, but not in a way that can be made explicit very easily.

What should a proposition like $A \wedge B$ mean, and how should that connect with the judgement $\vdash A \wedge B$ being derivable (or not)? We want to give a model which tells us something about derivability in our system.

Because our propositions are built using propositional variables we have to worry about how to model those variables. We cannot make any assumptions about the meaning of those variables, and so we employ a trick that is also found in the formal semantics of programming language.

The way we assign meaning to a proposition is to consider *all* the possible meanings a variable could have, one at a time. This leads to the concept of a *valuation*:

Assume we have a set S in which propositions take their values. For a proposition containing n propositional variables, say Z_1, Z_2, …, Z_n a valuation v is a function that assigns a value in S to every propositional variable, that is,

$$\{Z_1, Z_2, \dots, Z_n\} \xrightarrow{v} S.$$

[11] We know thanks to the logician Kurt Gödel that a system which is powerful enough to express the kinds of things mathematicians are interested in cannot prove its own consistency.

If we look a *all* possible valuations, that is at all the functions with the source and target given above, then we look at all the possible meanings the proposition could have in our model.

As a consequence we assign a value to each proposition *relative to a valuation.* We may state properties of this model by quantifying over all possible valuations. This process is described in more detail for each of the models given below.

3.2.1 Model 1: Boolean

The Boolean algebra $\mathbb{B}$

We want to interpret, or model, propositions within the set $\{0, 1\}$. You may think of 0 as denoting 'false', and of 1 as denoting 'true'.[12] Then we can interpret the logical connectives as operations on this set.[13]

Note that the *same symbols* are used for the operations on $\{0, 1\}$ as in our formal system. Indeed we intend to model, for example, the connective $\wedge$ in our system by the operation $\wedge$ on $\{0, 1\}$.

Conjunction:

$\wedge$	0	1
0	0	0
1	0	1

Disjunction:

$\vee$	0	1
0	0	1
1	1	1

Implication:

$\to$	0	1
0	1	1
1	0	1

Negation:

$\neg$	
0	1
1	0

We use $\mathbb{B}$ to refer to the set $\{0, 1\}$ with these operations. It is also known as the **two element boolean algebra**.

Note that there is a different way of displaying what these operations do by giving a table that lists the possible values of the two arguments in the first two columns and the result in the third.

Conjunction:

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

Disjunction:

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

[12] You sometimes find people using different names for the elements of this set, for example $\{\mathsf{F}, \mathsf{T}\}$. You should have no difficulty translating between these.

[13] Compare these with the rules for logical operators on page 128 of *Java: Just in Time*, and with those in the first handout for COMP12111—they all talk about the same operations, only that implication is omitted there. The reason for this is explained in Section 3.3.

Implication:

x	y	$x \to y$
0	0	1
0	1	1
1	0	0
1	1	1

Negation:

x	$\neg x$
0	1
1	0

Certainly from a common-sense point of view these make sense: If I have two true statements S and S' then the statement we obtain by connecting these with ‘and’ is true, but if one of them is false then the combined statement should be false. You may want to go back and read the text for the key phrases from Section 2.2.1 to compare what is said there with the above definitions for these operations.

Mathematically the above tables define four functions, three (namely $\wedge$, $\vee$ and $\to$) from $\{0,1\} \times \{0,1\}$ to $\{0,1\}$, and one (namely $\neg$) from $\{0,1\}$ to $\{0,1\}$. The way to interpret the above tables as functions of type

$$\{0,1\} \times \{0,1\} \longrightarrow \{0,1\}$$

is to

- take the value in the first column as the first input,
- take the value in the top row as the second input and
- use the corresponding entry in the table as the outcome of applying the function to these arguments.

An alternative way of using a table to define these functions is to give all the combinations of inputs in the first two columns,[14] for example for $\wedge$:

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

Exercise 51. There are three binary operations defined on $\{0,1\}$. Determine which of them are commutative and associative, and which of them have a unit, and what it is.

Valuations

But how do we interpret a proposition such as $A \wedge B$? As indicated above, we have to work *relative to valuations*[15] *for the propositional variables.* We give an example here.

Assume we have a proposition that contains two variables Z and Z', such as

$$((Z \vee Z') \wedge \neg Z) \vee \neg Z'.$$

[14] Note that the safest way of ensuring that all combinations are listed for the inputs is to ‘count in binary’.

[15] Some people call these ‘interpretations’ or ‘truth assignments’ for the boolean model.

In this example a valuation v is a function

$$\{Z, Z'\} \longrightarrow \{0, 1\}.$$

Our interpretation is relative to a valuation, and to really understand the given proposition we have to consider *all possible valuations*.

Exercise 52. If you have a proposition with n propositional variables, how many possible valuations are there?

Interpretation of propositions

Given the set $\{Z, Z'\}$ assume we have a valuation v that takes its values in $\{0, 1\}$. The interpretation of $((Z \vee Z') \wedge \neg Z) \vee \neg Z'$ *relative to* v is then given by

$$((vZ \vee vZ') \wedge \neg vZ) \vee \neg vZ,$$

where $\wedge$, $\vee$ and $\neg$ are the respective operations for $\mathbb{B}$ defined above.

Example 3.3. Assume that we have a valuation v which maps

$$Z \text{ to } 0 \qquad\qquad \text{and} \qquad\qquad Z' \text{ to } 1.$$

Then the value of $((Z \vee Z') \wedge \neg Z) \vee \neg Z'$ relative to that v is

$$\begin{aligned} ((vZ \vee vZ') \wedge \neg vZ) \vee \neg vZ' &= ((0 \vee 1) \wedge \neg 0) \vee \neg 1 && \text{def } v \\ &= ((0 \vee 1) \wedge 1) \vee 0 && \text{def } \neg \\ &= (1 \wedge 1) \vee 0 && \text{def } \vee \\ &= 1 \vee 0 && \text{def } \wedge \\ &= 1. && \text{def } \vee \end{aligned}$$

Without showing the calculation we give below the value for this proposition relative to the other valuations. We do this by filling in a table[16] that helps make the calculations faster.

vZ	vZ'	$vZ \vee vZ'$	$(vZ \vee vZ') \wedge \neg vZ$	$((vZ \vee vZ') \wedge \neg vZ) \vee \neg vZ'$
0	0	0	0	1
0	1	1	1	1
1	0	1	0	1
1	1	1	0	0

In general we use the following definition. Given a proposition A with variables $\{Z_1, Z_2, \dots, Z_n\}$, and a valuation v of $\{Z_1, Z_2, \dots, Z_n\}$ in $\mathbb{B}$ we get the **boolean interpretation of A relative to v** by the following steps:

- In the expression A, replace every occurrence of Z_i by vZ_i,
- calculate the value of the resulting expression in $\mathbb{B}$ by using the operations for $\mathbb{B}$.

[16] This is the same idea as the truth tables in Chapter 8 of *Java: Just in Time*.

Exercise 53. Calculate the boolean interpretation of

$$((Z \vee Z') \wedge \neg Z) \vee \neg Z'$$

relative to the valuation which maps Z to 1 and Z' to 0.

The boolean interpretation of a proposition is sometimes called its *truth value*.

Definition 26: tautology

A proposition A is a **tautology** if and only if for all valuations v for A it is the case that the boolean interpretation of A relative to v is equal to 1. It is **satisfiable** if and only if there is a valuation such that the boolean interpretation of A relative to v is equal to 1.

This interpretation is very strongly connected with what is derivable in our system.

Theorem 3.2

A judgement of the form $\vdash A$ is derivable if and only if A is a tautology.

This is a very important theorem in mathematical logic. We don't have the time here to give even a sketch of a proof. Note that it tells us that our system is very closely related to the model. Or, in other words, the model tells us exactly which judgements can be derived in our system. It's usually much easier to check what happens in the model than to find a derivation.

Our example proposition is satisfiable since there is a valuation[17] relative to which its boolean interpretation is 1. It is not a tautology since there is a valuation[18] relative to which its interpretation is 0. This means that

$$\vdash ((Z \vee Z') \wedge \neg Z) \vee \neg Z'.$$

cannot be derived in our system, but if we replace the propositional variables Z and Z' by suitable propositions then the resulting judgement may be derivable.

3.2.2 Model 2: Powersets

Reducing the interpretation of every proposition to just one value, 0 or 1, is somewhat restrictive, and not very subtle.

Assume we are given a set X, and from that we construct the powerset, $\mathcal{P}X$. We may give interpretations to the logical connectives in $\mathcal{P}X$ as follows:

- conjunction $\wedge$: use intersection $\cap$;
- disjunction $\vee$: use union $\cup$;
- negation $\neg$: use complement $X \setminus -$;
- implication $\rightarrow$: for $S \rightarrow S'$ use $(X \setminus S) \cup S'$;

[17] Any of the first three in our table works.

[18] The last one from our table.

Again we have a notion of *valuation*, but this time to each variable a valuation has to assign a subset of X. So a valuation v for a proposition with variables $\{Z_1, Z_2, \dots, Z_n\}$ is a function

$$\{Z_1, Z_2, \dots, Z_n\} \xrightarrow{v} \mathcal{P}X.$$

The powerset interpretation **of a proposition A relative to a given valuation** v is given by a *subset of X*, calculated as follows.

- In the expression A, replace every occurrence of Z_i by vZ_i,
- replace the connectives according to the instructions given above and then
- calculate the value of the resulting expression in $\mathcal{P}X$ by computing the resulting set.

Example 3.4. We go back to proposition considered in Example 3.3, $((Z \vee Z') \wedge \neg Z) \vee \neg Z'$. Assume that $X = \{a, b, c\}$ and that the valuation v maps

$$Z \text{ to } \{a, b\} \qquad\qquad \text{and} \qquad\qquad Z' \text{ to } \{b, c\}.$$

Then the value of $((Z \vee Z') \wedge \neg Z) \vee \neg Z'$ relative to v is

$$\begin{aligned}
&((vZ \cup vZ') \cap \neg vZ) \cup \neg vZ' \\
&= ((\{a, b\} \cup \{b, c\}) \cap (X \setminus \{a, b\})) \cup (X \setminus \{b, c\} && \text{def } v \\
&= (\{a, b, c\} \cap \{c\}) \cup \{a\} && \text{def } \cup, \setminus \\
&= \{c\} \cup \{a\} && \text{def } \cap \\
&= \{a, c\}. && \text{def } \cup
\end{aligned}$$

For the powerset interpretation a tautology is that of a proposition which is mapped to X for every valuation. Satisfiability becomes the existence of a valuation such that the interpretation of the proposition in question is non-empty. The theorem connecting derivability and and this adaptation of tautology remains true.

Hence in this interpretation we can see that $((Z \vee Z') \wedge \neg Z) \vee \neg Z'$ is not a tautology after making calculations for just one valuation. On the other hand there are significantly more valuations available, so checking all of them takes more time.

EExercise 54. Calculate the powerset interpretation in $X = \{a, b, c\}$ of

$$((Z \vee Z') \wedge \neg Z) \vee \neg Z'$$

relative to the valuation which maps Z to $\{a, b, c\}$ and Z' to $\{b\}$.

Exercise 55. Give an argument that this model subsumes Model 1. *Hint: Consider the powerset for a one-element set.*

Exercise 56. Assume that every valuation may assign only the values $\emptyset$ or X to a propositional variable. What does that mean for this model?

Optional Exercise 7. In general, is there a way of relating the interpretation in Model 2 for arbitrary X (and arbitrary valuations) to the interpretation in Model 1?

3.2.3 Model 3: Logic Gates

We may build yet another model for our formal system, using logic gates. Each propositional variable for a given proposition becomes an *input wire* for our circuit. We use the following to connect these wires according to the connectives in the proposition:

- conjunction $\wedge$: use an AND gate;
- disjunction $\vee$: use an OR gate;
- negation $\neg$: use a NOT gate;
- implication $\rightarrow$: for $A \rightarrow B$ use an OR gate where the first input has been negated;

The resulting circuit has one output wire. A *valuation* means deciding, for each input wire, whether to put a voltage onto that input wire or not. Given a valuation we get two possibilities for the output wire: Either it carries a voltage, or it doesn't.

The **circuit interpretation** of a proposition is the resulting circuit. The notion of tautology is then adjusted to this model by demanding that for every valuation (that is, for every way of putting a voltage, or not, onto the input wires) the output wire carries a voltage, and satisfiability means that there is at least one way of assigning voltages to the input wires such that the output wire carries a voltage. The result connecting tautologies and derivable propositions remains true.

3.2.4 Applications

Instead of thinking of a model as saying something about the formal system, one could think of the formal system making statements about the intended model. The fact that there are so many models available is one of the reasons why the language of logic is applicable in so many ways.

Those of you taking COMP12111 have met logic as a description of what happens in an electronic circuit. The conditional statements in Java use logic connectives, just with a different syntax so that they may be typed on a standard keyboard. You find more about these below.

There is also an obvious way of taking our 'key phrases' and turning them into logical connectives, and so we may think of a formal system like ours as talking about definitions and proofs from mathematics, or we can use it to talk about sets. But statements built from these phrases are used in many other areas as well, for example in database queries, or to describe the result such a query ought to have. The key phrases, or the corresponding logical connectives, allow us to make precise statements about a great variety of different topics. Because they appear in so many places it is important to understand these connectives and their intended meanings, as well as their properties.

In general, whenever people wish to formally establish something such as the correctness of a circuit or a program relative to a given specification they translate the specification into a formal system, and construct a formal model of the circuit or program within that system, and then try to establish a formal proof using a mechanism supplied by the system..

3.3 Properties of our system

We study the properties of our system with regard to the three models given. However, we restrict our arguments to Model 1 for the most part, since that is the one considered by most applications you will encounter in the future.

Redundancy of implication

In our system we have more connectives than we need if we are only concerned with the interpretations in Models1–3.

Assume we have a proposition of the form $A \to B$, where A and B may be complex proposition. Further assume that the propositional variables mentioned in A and B are given in $\{Z_1, Z_2, \ldots, Z_n\}$.

Then for every valuation $v \colon \{Z_1, Z_1, \ldots, Z_n\} \longrightarrow \{0,1\}$ it is the case that the interpretation of

$$A \to B$$

relative to that valuation is the same as that of

$$\neg A \vee B$$

in each of our three models.[19]

We can check this by filling in a table that calculates the values for all inputs for the following two functions from $\{0,1\} \times \{0,1\} \longrightarrow \{0,1\}$:

$$\{0,1\} \times \{0,1\} \xrightarrow{\neg \times \mathsf{id}_{\{0,1\}}} \{0,1\} \times \{0,1\} \xrightarrow{\vee} \{0,1\}$$

and

$$\{0,1\} \times \{0,1\} \xrightarrow{\to} \{0,1\}.$$

We give the inputs in the two leftmost columns, use the next column for an intermediate calculation, and give the outputs for both functions in the two rightmost columns.

x	y	$\neg x$	$\neg x \vee y$	$x \to y$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

Since the last two columns are equal we know that the two functions under consideration give the same output for every possible combination of input values. As a consequence, no matter which value is assigned to A and B in our model for the given valuation, the two propositions will be assigned the same value.

So when we are merely interested in the interpretation of propositions in our model then we may *remove* the connective $\to$ from consideration entirely, and instead use negation and conjunction.[20]

In a case like this, where the boolean interpretation of two terms is the same for all valuations, we write

$$A \to B \equiv \neg A \vee B.$$

[19] And indeed, it is also possible to establish that $\vdash A \to B$ is derivable precisely when the same is true for $\vdash \neg A \vee B$.

[20] But note that if we are interested in our system for other reasons then we should not just remove connectives.

Definition 27: semantic equivalence

Two propositions A and B are **semantically equivalent** if and only if it is the case that for every valuation[21] their boolean interpretations are the same. In this case we write $A \equiv B$.

In the literature you may find $A = B$, but this is imprecise because the two terms are not equal as propositions, or $A \equiv B$, but in computer science this is usually reserved for a syntactic equality. You may think of the two terms as equal in the semantics—that is, the model does not distinguish between them.

Note that many computer languages have propositional expressions: every language that has some kind of conditional 'if…then' instructions) uses them. The typical usage is

```
If <propositional expression> then ... else ...
```

The `propositional expression` is evaluated to true or false (our 1 or 0, respectively), and this evaluation is stable under $\equiv$, which means that if

`propositional expression 1` $\equiv$ `propositional expression 2`

then they evaluate to the same truth value. This means that you may use the rules given in the following sections to rewrite your boolean expressions into a more easily understandable form. This makes it much easier to check whether one has captured the various cases as intended.[22]

Note that in Model 2 we have used this idea as the *definition* of the interpretation of $\to$: Let the interpretation of A relative to valuation v be S, and that of B be T, both subsets of X. Then the interpretation of $A \to B$ relative to v is

$$(X \setminus S) \cup T$$

which is exactly the interpretation relative to v of $\neg A \vee B$.

Similarly one can argue that the two propositions will be assigned the same value in Model 3.

Exercise 57. In Chapter 2 we have notions of associativity and commutativity of binary operations. We can adopt these notions, but instead of asking certain expressions to evaluate to the *same* value we ask that they are semantically equivalent.

(a) Give an argument that $\wedge$ is commutative with respect to $\equiv$, that is $A \wedge B \equiv B \wedge A$. Now do the same for $\vee$. *Hint: Consider the argument we made for $A \to B \equiv \neg A \vee B$ on page 131.*

(b) Give an argument that $\wedge$ and $\vee$ are associative with respect to $\equiv$.

Double negation

Applying negation twice to the same proposition is semantically equivalent to not applying it at all.

[21] That is every valuation that assigns values to the variables mentioned in A and B combined.

[22] In *Java: Just in Time* in Chapter 8 you find truth tables as a way of evaluating propositional expressions—this is precisely the same idea as our boolean interpretation. Our 'semantically equivalent' means 'equivalent' in that book.

In other words, for every proposition A it is the case that under all valuations, it will be assigned the same value as $\neg\neg A$.[23]

We may summarise this[24] as

$$A \equiv \neg\neg A$$

for all propositions A.

Exercise 58. Prove the above claim by checking that the two functions

$$\{0,1\} \xrightarrow{\mathsf{id}_{\{0,1\}}} \{0,1\} \qquad\qquad \{0,1\} \xrightarrow{\neg} \{0,1\} \xrightarrow{\neg} \{0,1\}$$

give the same output for every possible input by constructing a table similar to the one given on page 131.

EExercise 59. Show that in Model 2 the interpretation of negation satisfies the same property, that is, applying it twice gives the identity function. *Hint: How is negation interpreted in the second model? What happens if you apply that operation twice to an arbitrary set?*

De Morgan's laws

There are further relations between the remaining connectives, $\neg$, $\vee$ and $\wedge$.

If A and B are arbitrary propositions, then relative to every valuation the values of

$$\neg(A \wedge B) \qquad\qquad \text{and} \qquad\qquad \neg A \vee \neg B$$

are the same.[25]

We write this as

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

and refer to it as **De Morgan's law**.

CExercise 60. Prove the claim that

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

by constructing a table for the corresponding functions as in the example establishing that $A \to B \equiv \neg A \vee B$ given on page 131.

Note that this means

$$\begin{aligned} A \wedge B &\equiv \neg\neg(A \wedge B) && A \equiv \neg\neg A \\ &\equiv \neg(\neg A \vee \neg B) && \text{De Morgan } \wedge \end{aligned}$$

Hence we may take a proposition and replace every occurrence of the $\wedge$ symbol using De Morgan's law, and we get a proposition which is semantically equivalent.

[23] And indeed, in our system A is derivable if and only if $\neg\neg A$ is derivable—in fact, something stronger is true.

[24] Compare this with Exercise 49 (a) and (b).

[25] Compare this with Exercise 49 (c) and (d).

Therefore we may eliminate $\wedge$ in the same way that we may eliminate $\to$ provided we are only interested in the values propositions take in the three models.

We could therefore use only the connectives $\neg$ and $\vee$, and still have the same 'expressive power' in the model as when using all connectives.

The above law is valid for all propositions, and if we use $\neg C$ in the place of A, and $\neg D$ in the place of B, we get

$$\neg(\neg C \wedge \neg D) \equiv \neg\neg C \vee \neg\neg B \equiv C \vee D,$$

giving us a second version of De Morgan's law, which by applying $\neg$ on both sides, turns into

$$\neg C \wedge \neg D \equiv \neg\neg(\neg C \wedge \neg D) \equiv \neg(C \vee D).$$

Hence we could use only the connectives $\neg$ and $\wedge$ instead of $\neg$ and $\vee$.

Exercise 61. Show that De Morgan's law also holds for Model 2, that is the interpretation in a powerset. *Hint: Use the instructions from Section 3.2.2 for replacing logical connections by set operations. Then prove that the two sets you get are equal.*

Idempotence

The two connectives conjunction and disjunction have an additional property in the boolean model that we have not looked at so far.

It is the case that in $\mathbb{B}$, $x \vee x = x$ as well as $x \wedge x = x$ for all x in $\{0, 1\}$. This can be easily checked by looking only at the diagonal of the tables defining these operations.

Conjunction:

$\wedge$	0	1
0	0	
1		1

Disjunction:

$\vee$	0	1
0	0	
1		1

This means that for every proposition A we have that the boolean interpretation of $A \vee A$, and that of $A \wedge A$, relative to any valuation, are both equal to the boolean interpretation of A relative to that valuation.

Hence we have for every proposition A that

$$A \vee A \equiv A \qquad \text{and} \qquad A \wedge A \equiv A.$$

We say that $\wedge$ and $\vee$ are **idempotent** for $\equiv$.

Exercise 62. Show that for Model 2 we also have that the interpretation of $A \vee A$ and $A \wedge A$ are equal to that of A, all relative to a given valuation.

Contradiction and excluded middle

There are some additional statements we can make regarding semantic equivalence of propositions.

First of all note that when we look at the boolean interpretations of A and $\neg A$ relative to the same valuation, then one of these is always 0 and one of these is always 1. This is because the interpretation of $\neg A$ arises from that of A by applying the negation function for $\mathbb{B}$.

This has the following consequences.

The boolean interpretation of a proposition of the form

$$A \wedge \neg A$$

is always 0.

The boolean interpretation of a proposition of the form

$$A \vee \neg A$$

is always 1.

The table below shows this:

boolean interpretation of			
A	$\neg A$	$A \wedge \neg A$	$A \vee \neg A$
0	1	0	1
1	0	0	1

We introduce new symbols for this case: We write

$$A \equiv \bot$$

in the case where the boolean interpretation of A relative to every valuation is 0, and we write

$$A \equiv \top$$

in the case where the boolean interpretation of A relative to every valuation is 1.

Here $\top$ is a symbol for 'truth' and $\bot$ is a symbol for 'falsehood'. These are quite widely used, but other symbols may also appear in the literature, for example 1 or **T** for truth, and 0 or **F** for false. In Java you have the reserved words `true` and `false`, which serve the same purpose.

It is possible to define $\top$ and $\bot$ as propositions in our system, but we do not do this here since it does not add much. However, when we make calculations to better understand a proposition we allow ourselves to use $\top$ and $\bot$ as part of the calculation in the sense introduced below.

Note that if A is a proposition then

- if $A \equiv \top$ then for every proposition B
  - we have $A \vee B \equiv \top \vee B \equiv \top$ and
  - we also have $A \wedge B \equiv \top \wedge B \equiv B$.
- if $A \equiv \bot$ then for every proposition B
  - we have $A \wedge B \equiv \bot \wedge B \equiv \bot$ and
  - we also have $A \vee B \equiv \bot \vee B \equiv B$.

Exercise 63. Explain why the above statements about semantic equivalence are valid.

There are two further useful principles for simplifying formula to ones that are semantically equivalent. The first of these is

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

which is semantically equivalent to

$$A.$$

Similarly we have

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

This requires checking.

Exercise 64. Show that for arbitrary propositions A and B we have that the boolean interpretation of $A \wedge (A \vee B)$, and that of $A \vee (A \wedge B)$, is the same as that of A.

We give a table to summarize our rules. We assume that A and B are arbitrary propositional formula.

name	**proposition**	**equivalent proposition**
implication	$A \to B$	$\neg A \vee B$
double negation	$\neg\neg A$	A
De Morgan $\vee$	$\neg(A \vee B)$	$\neg A \wedge \neg B$
De Morgan $\wedge$	$\neg(A \wedge B)$	$\neg A \vee \neg B$
idempotence $\vee$	$A \vee A$	A
idempotence $\wedge$	$A \wedge A$	A
excluded middle	$A \vee \neg A$	$\top$
contradiction	$A \wedge \neg A$	$\bot$
absorption	$A \wedge (A \vee B)$	A
absorption	$A \vee (A \wedge B)$	A

Additionally we have that

$$\bot \text{ behaves like a unit for } \vee$$

and that

$$\top \text{ behaves like a unit for } \wedge,$$

but these two are not part of our official formal system.

There is one important rule missing in the form of *distributivity laws*. We turn to those in the following section.

Normal forms

In particular in computer science it is customary to write propositions in a particular form.

Definition 28: conjunctive normal form

A proposition is in **conjunctive normal form, CNF**, if it is of the form

$$A_1 \wedge A_2 \wedge \cdots \wedge A_m,$$

where each A_i is of the form

$$B_1 \vee B_2 \vee \cdots \vee B_n,$$

where each B_j is of the form Z or $\neg Z$ for some propositional variable Z.

For every proposition we may find a semantically equivalent one that is in conjunctive normal form, and there is an algorithm for doing this.

Step 0 If the proposition contains any $\to$ symbols, replace them according to the redundancy of implication rule, so $A \to B$ becomes $\neg A \vee B$.

Step 1 For every negation symbol, use De Morgan's law to

- rewrite $\neg(A_1 \wedge \cdots \wedge A_n)$ to $(\neg A_1 \vee \cdots \vee \neg A_n)$ and
- rewrite $\neg(A_1 \vee \cdots \vee A_n)$ to $(\neg A_1 \wedge \cdots \wedge \neg A_n)$.

Repeat Step 1 until every negation symbol applies to a propositional variable only. Then erase every occurrence of $\neg\neg$.

Step 2 Rewrite every sub-term of the form

$$(A_1^1 \wedge \cdots \wedge A_{m_1}^1) \vee (A_1^2 \wedge \cdots \wedge A_{m_2}^2) \vee \cdots \vee (A_1^n \wedge \cdots \wedge A_{m_n}^n)$$

to one that is the conjunction of all possible combinations of terms of the form $A_{i_1}^1 \vee A_{i_2}^2 \vee \cdots \vee A_{i_n}^n$. Keep repeating this step until the term is in conjunctive normal form.[26]

Intuitively speaking it is clear that eventually we will no longer be able to repeat Step 2: What it does is to swap disjunction and conjunction symbols so that after the step, the disjunction is carried out before the conjunction when carrying out the calculation. Since there are only finitely many conjunction and disjunction symbols available the process has to stop eventually.

We have not yet justified Step 2 as providing a semantically equivalent proposition. This works due to the following distributivity laws:

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

which by commutativity of $\vee$ for $\equiv$ also gives

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

Exercise 65. Use this distributivity law to show that

$$(A \wedge B) \vee (C \wedge D) \equiv (A \vee C) \wedge (A \vee D) \wedge (B \vee C) \wedge (B \vee D).$$

[26] Note that this step increases the number of terms in the proposition. In the worst case this increase is exponential.

Give the appropriate law for the case where there are three propositions in each bracket.

What we do in Step 2 is to apply a more general version of the distributivity law, and the preceding exercise gives you an idea how that is derived.

Many tools expect propositions to be entered in CNF, so you need to be able to convert a formula into that form. There are also some problems from theoretical computer science which are concerned with propositions in this shape.

Example 3.5. We turn the proposition

$$(Z \wedge Z') \vee \neg(Z'' \wedge (Z \vee Z''))$$

into conjunctive normal form.

$$\begin{array}{ll}
(Z \wedge Z') \vee \neg(Z'' \wedge (Z \vee Z'')) & \\
\quad = (Z \wedge Z') \vee (\neg Z'' \vee \neg(Z \vee Z'')) & \text{Step 1, De Morgan} \\
\quad \equiv (Z \wedge Z') \vee (\neg Z'' \vee (\neg Z \wedge \neg Z'')) & \text{Step 1, De Morgan} \\
\quad \equiv (Z \wedge Z') \vee \neg Z'' \vee (\neg Z \wedge \neg Z'') & \text{associativity } \vee \\
\quad \equiv (Z \vee \neg Z'' \vee \neg Z) \wedge (Z \vee \neg Z'' \vee \neg Z'') & \\
\qquad \wedge (Z' \vee \neg Z'' \vee \neg Z) \wedge (Z' \vee \neg Z'' \vee \neg Z'') & \text{Step 2}
\end{array}$$

This is a conjunctive normal form: We have conjunction on the outside and only one level of disjunction inside that, and all negation symbols are applied to atoms.

Note that once one has reached a conjunctive normal form one can often apply further simplifications.

Example 3.6. We continue the previous example We know that

- $Z \vee \neg Z \equiv \top$ and so also $Z \vee Z'' \vee \neg Z \equiv \top$,
- $\neg Z'' \vee \neg Z'' \equiv \neg Z''$ and so also $Z \vee \neg Z'' \vee \neg Z'' \equiv Z \vee \neg Z''$ and $Z' \vee \neg Z'' \vee \neg Z'' \equiv Z' \vee \neg Z''$.

As a consequence we get that

$$\begin{array}{lll}
(Z \vee \neg Z'' \vee \neg Z) \wedge (Z \vee \neg Z'' \vee \neg Z'') & & \\
 & \wedge\, (Z' \vee \neg Z'' \vee \neg Z) & \wedge\, (Z' \vee \neg Z'' \vee \neg Z'') \\
\equiv & \top & \wedge\, (Z \vee \neg Z'') \\
 & \wedge\, (Z' \vee \neg Z'' \vee \neg Z) & \wedge\, (Z' \vee \neg Z'') \\
\equiv & (Z \vee \neg Z'') & \wedge\, (Z' \vee \neg Z''),
\end{array}$$

where for the last step we are using

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

Having a proposition in this form makes it much easier to understand what it is actually saying! Note that sometimes it is possible to apply simplifications before carrying out Step 2, and this is certainly allowed.

Example 3.7. Returning to the proposition from Example 3.6 once we have it in the form

$$(Z \vee \neg Z'') \wedge (Z' \vee \neg Z'')$$

we can see that this is not a tautology since it evaluates to 0 when we use the valuation

$$v\colon X \longmapsto \begin{cases} 0 & X = Z \text{ or } X = Z' \\ 1 & \text{else.} \end{cases}$$

X	vX
Z	0
Z'	0
Z''	1

CExercise 66. Give a conjunctive normal form for the following propositions. Then simplify them as far as you can.

(a) $(Z \wedge \neg Z') \to (Z'' \wedge \neg Z)$,

(b) $\neg(Z \wedge Z' \wedge \neg Z'') \vee (\neg Z'' \wedge Z)$,

(c) $\neg(Z \vee (Z' \vee Z) \wedge \neg(Z' \wedge Z))$,

(d) $(\neg Z \vee \neg Z') \wedge (\neg Z \vee (Z' \to Z''))$.

CExercise 67. For the following propositions, decide whether they are tautologies, and whether they are satisfiable. *Hint: find a conjunctive normal form first, and then simplify. Then calculate the boolean interpretation for the resulting proposition for all valuations, as in the example on page 127. Alternatively, calculate the table for the given proposition.*

(a) $(Z \wedge \neg Z') \vee Z' \vee \neg(Z \to Z')$,

(b) $(Z \wedge Z') \to ((Z \wedge \neg Z') \vee (Z \wedge Z'))$,

(c) $((Z \wedge Z') \to Z'') \to (Z \to (Z' \to Z''))$,

(d) $((Z \to Z') \wedge (Z' \to Z'')) \to (\neg Z \wedge Z'')$,

(e) $((Z \to Z') \wedge (Z' \to Z'')) \to (Z \to Z'')$,

(f) $\neg(Z \to \neg Z') \vee (\neg Z \wedge \neg Z')$;

(g) $\neg(Z \wedge Z') \to (\neg Z \vee \neg Z')$.

One can swap the role of conjunction and disjunction and obtain a different normal form.

Definition 29: disjunctive normal form

A proposition is in **disjunctive normal form, DNF**, if it is of the form

$$A_1 \vee A_2 \vee \cdots \vee A_m,$$

where each A_i is of the form

$$B_1 \wedge B_2 \wedge \cdots \wedge B_n,$$

where each B_j is of the form Z or $\neg Z$ for some propositional variable Z.

For every proposition we may find a semantically equivalent one that is in disjunctive normal form, and there is an algorithm for doing this. The algorithm is very similar to the one for CNF—we merely have to exchange the role of $\wedge$ and $\vee$.

Step 0 If the proposition contains any $\rightarrow$ symbols, replace them according to the redundancy of implication rule, so $A \rightarrow B$ becomes $\neg A \vee B$.

Step 1 For every negation symbol, use De Morgan's law to

- rewrite $\neg(A_1 \vee \cdots \vee A_n)$ to $\neg A_1 \wedge \cdots \wedge \neg A_n$ and
- rewrite $\neg(A_1 \wedge \cdots \wedge A_n)$ to $\neg A_1 \vee \cdots \vee \neg A_n$.

Repeat Step 1 until every negation symbol applies to a propositional variable only. Erase every occurrence of $\neg\neg$.

Step 2 Rewrite every sub-term of the form

$$(A_1^1 \vee \cdots \vee A_{m_1}^1) \wedge (A_1^2 \vee \cdots \vee A_{m_2}^2) \wedge \cdots \wedge (A_1^n \vee \cdots \vee A_{m_n}^n)$$

to one that is the disjunction of all possible combinations of terms of the form $A_{i_1}^1 \wedge A_{i_2}^2 \wedge \cdots \wedge A_{i_n}^n$. Keep repeating this step until the term is in disjunctive normal form.

The distributivity law used here is

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

Example 3.8. We consider the same proposition as in Example 3.5.

$$\begin{aligned}
&(Z \wedge Z') \vee \neg(Z'' \wedge (Z \vee Z'')) \\
&\equiv (Z \wedge Z') \vee (\neg Z'' \vee \neg(Z \vee Z'')) && \text{Step 1, De Morgan} \\
&\equiv (Z \wedge Z') \vee (\neg Z'' \vee (\neg Z \wedge \neg Z'')) && \text{Step 1, De Morgan} \\
&\equiv (Z \wedge Z') \vee \neg Z'' \vee (\neg Z \wedge \neg Z'') && \text{distributivity of } \vee
\end{aligned}$$

It is coincidence that needed fewer sooner here—this is just a function of the particular proposition we started with.

In general, converting to CNF and converting to DNF have the same difficulty level. Both make it easier to see whether some proposition is a tautology. For satisfiability the DNF can make things easier since one merely has to check whether there is one term that is interpreted by 1 relative to some valuation.

Example 3.9. If we look at the DNF from the previous example,

$$(Z \wedge Z') \vee \neg Z'' \vee (\neg Z \wedge \neg Z''),$$

we can immediately see that any valuation v with the property that

$$vZ'' = 0$$

will result in a boolean interpretation of 1.

Example 3.10. Note that again we may simplify this proposition further: The boolean interpretation of

$$\neg Z'' \vee (\neg Z \wedge \neg Z'')$$

is the same as that of

$$\neg Z''.$$

Hence we may simplify the DNF from the previous example as follows.

$$(Z \wedge Z') \vee \neg Z'' \vee (\neg Z \wedge \neg Z'') \equiv (Z \wedge Z') \vee \neg Z''.$$

CExercise 68. For the following propositions give a DNF. Then simplify them as far as you can. Which ones are satisfiable? Give a suitable valuation to support your claim.

(a) $(Z \vee \neg Z') \wedge \neg(Z \to \neg Z'')$,

(b) $\neg(Z \to Z') \wedge (Z \vee Z' \vee \neg Z'')$,

(c) $\neg(Z \to Z') \to (Z \wedge Z'' \wedge \neg Z')$,

(d) $(Z \vee \neg Z') \wedge \neg(Z'' \vee \neg Z)$.

Expressiveness

We briefly turn to the question which functions from (some power of) $\mathbb{B}$ to $\mathbb{B}$ can be seen as the interpretation of a proposition.

Let us first of all make sense of the question. Given a valuation the boolean interpretation of a proposition relative to that valuation is just a value in $\mathbb{B}$. We may, however, think of a proposition as a *function* which takes a valuation and returns a value in $\mathbb{B}$.

Assume the propositional variables $Z_1, Z_2, \dots, Z_n$ occur in our proposition. Then there are

$$2^n$$

possible valuations (see Exercise 52). We may think of these as the elements of $\mathbb{B}^n$:

An element of $\mathbb{B}^n$ is a vector with n elements each of which can be 0 or 1. We may therefore think of such a vector as giving a valuation.

Example 3.11. Consider the vector

$$(0, 0, 1, 0, 1).$$

We can think of it as giving us the valuation which assigns

Z_1	Z_2	Z_3	Z_4	Z_5
0	0	1	0	1

In that sense we may think of a proposition containing n propositional variables as giving us a function

$$\mathbb{B}^n \longrightarrow \mathbb{B}.$$

We first look at the question of which functions $\mathbb{B} \longrightarrow \mathbb{B}$ are captured in this way.

There are four such functions: These are captured respectively by the following propositions

$$Z \wedge \neg Z \qquad Z \qquad \neg Z \qquad Z \vee \neg Z.$$

We can think of what we have done here as thinking of

input 0 decodes $\neg Z$ and input 1 decodes Z

while

output 0 decodes switch off and output 1 decodes switch on.

So to encode for example the function

$$\begin{aligned} 0 &\longmapsto 1 \\ 1 &\longmapsto 0 \end{aligned}$$

we want $\neg Z$ to be switched on, and Z to be switched off, leading to $\neg Z$ as the proposition to be used. The only exception is the case where nothing is to be switched on, in which case we use $Z \wedge \neg Z$ to ensure the output is always 0.

This gives us the recipe for a function in n variables:

- If the output for every input is 0, use $Z_1 \wedge \neg Z_1$.
- Else: For every output that is equal to 1 do the following:
  - If the corresponding input at i is 0, choose $\neg Z_i$ and
  - if the corresponding input at i is 1, choose Z_i.
  - Now connect all the selected terms from this output by $\wedge$.

  Finally connect all the terms from this by $\vee$.

Note that the result is in DNF.

Example 3.12. Assume we want to find a proposition which, in this sense, encodes the function

$$\mathbb{B}^2 \to \mathbb{B}$$

given by the following table.

inputs		
x_1	x_2	output
0	0	1
0	1	0
1	0	1
1	1	0

The 'switched on' parts are in the first and third line. For the first of these the inputs are 0 for both variables, so we have to pick $\neg Z_1 \wedge \neg Z_2$ for the first line. For the third line the inputs are 1 and 0 respectively, so we want to pick $Z_1 \wedge \neg Z_2$ for that line, giving altogether

$$(\neg Z_1 \wedge \neg Z_2) \vee (Z_1 \wedge \neg Z_2).$$

This is a proposition in DNF.

We give an example of taking a proposition in DNF and turning it into CNF. This means that only Step 2 of the algorithm has to be applied.

Example 3.13. If we convert it to CNF we get

$$(\neg Z_1 \vee Z_1) \wedge (\neg Z_1 \vee \neg Z_2) \wedge (\neg Z_2 \vee Z_1) \wedge (\neg Z_2 \vee \neg Z_2).$$

We carry out some simplification steps:

$$\begin{array}{ll}
(\neg Z_1 \vee Z_1) \wedge (\neg Z_1 \vee \neg Z_2) \wedge (\neg Z_2 \vee Z_1) \wedge (\neg Z_2 \vee \neg Z_2) & \\
\equiv \top \wedge (\neg Z_1 \vee \neg Z_2) \wedge (\neg Z_2 \vee Z_1) \wedge \top & A \vee \neg A \equiv \top \\
\equiv (\neg Z_1 \vee \neg Z_2) \wedge (Z_1 \vee \neg Z_2) & \top \wedge A \equiv A, \text{ comm } \vee \\
\equiv (\neg Z_1 \wedge Z_1) \vee \neg Z_2 & \text{distr law} \\
\equiv \bot \vee \neg Z_2 & A \wedge \neg A \equiv \bot \\
\equiv \neg Z_2 & \bot \vee A \equiv A
\end{array}$$

And indeed, if we check the original table then we see that the output is 1 if and only if the second input is 0.

So given a table it is always worth checking whether there is a simple proposition to be read off. The advantage of learning the general method is that it applies to *every* problem you may be given to solve.

CExercise 69. For the following function give a proposition whose interpretation is the given function. Then give a CNF for your proposition and simplify the result as far as you can.

(a) For the function $\mathbb{B}^3 \longrightarrow \mathbb{B}$ given by the following table.

inputs			
x_1	x_2	x_3	output
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(b) For the function $\mathbb{B}^3 \longrightarrow \mathbb{B}$ given by the following table.

inputs			
x_1	x_2	x_3	output
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(c) For the function $\mathbb{B}^3 \longrightarrow \mathbb{B}$ given by the following table.

inputs			
x_1	x_2	x_3	output
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

(d) For the function $\mathbb{B}^3 \longrightarrow \mathbb{B}$ given by the following table.

inputs			
x_1	x_2	x_3	output
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

When programming

As already pointed out, many programming languages allow the user to build propositional expressions.

In Java, for example, this is done using:

- conjunction &&,
- disjunction ||,
- negation !
- relations
  - equal ==,

  - not equal `!=`,
  - less than `<`,
  - greater than `>`,
  - less than or equal `<=`,
  - greater than or equal `>=`,

  where the relations can be applied to those data types where there is a method implementing them.

When such an expression is evaluated at run-time each of the instances of the relations is evaluated to `true` or `false` (by calling the appropriate method), and then rules analogous to those given for the connectives in Model 1 are applied.[27]

This evaluation is stable under $\equiv$, that is, for example, two expressions that arise from each other by applying the rule for double negation or De Morgan's law, will always evaluate to the same value. This means that you can use these laws to rewrite the propositional expressions in your code to turn them into something that is more easily understandable (for example by putting them into CNF or DNF). This makes your code easier to read for others, but it also becomes easier to check whether what has been implemented is as intended. We know that implication can be expressed up to $\equiv$ by using the other connectives and this is the reason why many programming languages do not have an implication connective.

3.4 Predicate logic

In what is described above two of the key phrases from Section 2.2.1 do not occur, namely 'for all' and 'there exists'. The reason for this is that these are more complicated, and require more 'infrastructure' in our system. Even in Section 2.2.1 it was noted that we have to specify where the two phrases have to hold, and this is something we have to worry about in the formal system as well.

Building blocks

In order to build a formal system that allows the formation of propositions that might be interpreted by sentences such as 'for all $i \in \mathbb{N}$ there exists $j \in \mathbb{N}$ such that...' we need to include at least the following:

- variables (other than propositional ones) to which we may apply 'for all' and 'there exists' and
- predicates which allow us to talk about the properties that we may expect to hold

but possibly also

- symbols that denote functions and
- symbols that denote relations.[28]

[27] See page 128 of *Java: Just in Time.*

[28] See Section 0.4 for the mathematical meaning of this term, or read on to see examples.

Because all these new symbols have specific rules how they can be used the rules for stating which propositions may be built also become more complicated.

We here give an outline for such a system, but we study it in less detail.

We now assume that apart from propositional variables, we also have a supply of

- parameters a_1, a_2, …,
- variables x_1, x_2,…,
- function symbols f_1, f_2, …, each of which takes a known fixed number of arguments, and
- predicate symbols P_1, P_2, …, each of which takes a known fixed number of arguments.

We have to introduce the notion of a *term*, constructed in the following way:

- If a is a parameter then a is a term and
- if x is a variable then x is a term and
- if t_1, t_2,…, t_n are terms, and f is a function that takes n arguments, then $f(t_1, t_2, \ldots, t_n)$ is a term.

Propositions are then given by the following:

- If Z is a propositional variable then Z is a proposition,
- if P is a predicate that takes n arguments and t_1, t_2,…t_n are terms then $P(t_1, t_2, \ldots t_n)$ is a proposition;
- if A is a proposition then $\neg A$ is a proposition;
- if A and B are propositions then $A \wedge B$, $A \vee B$ and $A \to B$ are propositions.
- if A is a proposition and x is a variable then $\forall x.A$ and $\exists x.A$ are propositions.

Inference rules

We adopt all the inference rules we had above (but now using the new meaning of 'proposition'), and add rules for $\forall$ and $\exists$. In order to do so we need to introduce more notation.

Given a string A of symbols we write

$$[a/x]A$$

for the string that arises when every occurrence[29] of x has been replaced by a. For example,

$$[a/x](P(x, y) \wedge Z)$$

is

$$P(a, y) \wedge Z.$$

The inference rules for our new connectives are as follows.

[29] Strictly speaking one has to be a bit careful here but we do not have the time to go into the detail.

For all' introduction ∀I:

$$\frac{\Gamma \vdash [a/x]A}{\Gamma \vdash \forall x.A}\,\forall\mathrm{I}$$

'If given Γ we have A where x has been replaced by an arbitrary parameter a then given Γ we have that for all x, A.'

'For all' elimination ∀E:

$$\frac{\Gamma \vdash \forall x.A}{\Gamma \vdash [t/x]A}\,\forall\mathrm{E}$$

'If given Γ we have that for all x, A, then given Γ we have A where x has been replaced by t.'

This connective is also known as **universal quantification**.

'Exists' introduction ∃I:

$$\frac{\Gamma \vdash [t/x]A}{\Gamma \vdash \exists x.A}\,\exists\mathrm{I}$$

'If given Γ we have A, where x has been replaced by t then given Γ we have that there exists an x such that A.'

'Exists' elimination ∃E:

$$\frac{\Gamma \vdash \exists x.A \quad \Gamma, A \vdash B}{\Gamma \vdash B}\,\exists\mathrm{E}$$

'If given Γ we know that there exists x so that A and if given Γ and A, we have B then from Γ we have B provided that *x does not occur in B*.'

This is known as **existential quantification**.

We have a system for the classical predicate calculus.

3.4.1 Modelling the new system

It should not be difficult to imagine that a model of the new system is considerably more complicated than that of the previous one. We have to model all the new entities. We here give some basic ideas:

Everything takes place with respect to some domain of interpretation.

- Parameters are modelled by specific elements of the domain.
- Variables are modelled similarly to propositional variables, but with valuations in the domain, rather than in $\mathbb{B}$.
- Function symbols are modelled by specific functions of the appropriate arity within the domain.
- Predicate symbols are modelled by specific relations of the appropriate arity within the domain that give a value in $\mathbb{B}$ for each input.

This means that in the model

- a term is mapped to an element of the domain while
- a proposition is mapped to an element of $\mathbb{B}$ as before,

and that this is subject to two valuations,

- one mapping variables x_1, x_2, … to elements of the domain,
- and mapping propositional variables to $\mathbb{B}$ as before.

It then remains to have an interpretation for propositions that include the new connectives, $\forall$ and $\exists$. We only cover the boolean interpretation here. Assume for the given propositions we have

- a valuation v for propositional variables and
- a valuation w for variables.

The method for calculating boolean interpretations, relative to v and w, for $\forall x.A$ and $\exists x.A$ is as follows:

- In A, replace each variable y *other than* x by wy.
- Replace each propositional variable Z in A by vZ.
- Replace each predicate and function symbol by its interpretation.

This gives an expression that may almost be evaluated in $\mathbb{B}$ by using the previous model for the connective $\wedge$, $\vee$, $\rightarrow$ and $\neg$, but it my still contain x, $\forall$, or $\exists$.

To interpret

$$\forall x.A$$

do the following.

- For each element of the domain, replace x by that element and calculate a value in $\mathbb{B}$ for the expression obtained from A in the previous steps.
- If all the calculations from the previous step return 1, the interpretation of $\forall x.A$ is 1, else it is 0.

To interpret

$$\exists x.A$$

do the following.

- For each element of the domain, replace x by that value and calculate a value in $\mathbb{B}$ for the expression obtained from A in the previous steps.
- If at least one of the calculations from the previous step returns 1, the interpretation of $\exists x.A$ is 1, else it is 0.

Note that most programming languages do not support the $\forall$ and $\exists$ connectives. The exceptions are logic programming languages such as Prolog.

Examples of models

We give some examples of this.

Assume that we have the following specific symbols:

- A unary predicate M.

We model the resulting system by choosing the following:

- the domain is the set of first year students in the computer science department in Manchester;
- the interpretation of $M(x)$ is 1 if and only if the student x takes COMP11120.

Example 3.14. The proposition

$$\forall x. M(x)$$

is a valid proposition in our system (that is, it is syntactically formed correctly, we are not saying anything about its derivability here). The boolean interpretation of this proposition in our system does not require any valuations, and so we find out what it is by going through the first year students in the School one by one and checking whether they are enrolled on COMP11120.[30]

Example 3.15. The boolean interpretation of the proposition

$$\exists x. \neg M(x)$$

is very closely related to that of the previous one. It is 1 in our current model if we can find a first year student in the department for who does not take COMP11120.

It should be clear that a lot has been gained by all the complication added to the system in that we are now able to express quite complicated properties that allow us to reason about the real world, as well as about bits of mathematics.

We could change our interpretation of $M(x)$ to mean, for example 'the student x takes COMP16121'. Does that change the boolean interpretation of $\forall x. M(x)$?

Example 3.16. If we add another unary predicate symbol, say C, then we may form the proposition

$$\forall x. (C(x) \to M(x)).$$

If we add to our previous interpretation that $C(x)$ means 'student x takes COMP11212' then the boolean interpretation of the above proposition is 1 provided that every first year CS student who takes COMP11212 also takes COMP11120.

Since students on the Computer Science and Mathematics (CM) programme may take COMP11212 without taking COMP11120 this proposition is usually interpreted by 0.[31]

On the other hand, we could declare our domain of interpretation to be the set of first year single honours students in the School, in which case the boolean interpretation of this proposition is 1.

Every computer scientist has to be able to translate between statements made in English and propositions in predicate logic.

Exercise 70. Consider the system with the two unary predicate symbols H and O.

As the domain of definition we use the set of undergraduate students in the School of Computer Science. We use the following interpretations:

- $H(x)$ means that x has taken and passed the health-and-safety test.

[30] So is the boolean interpretation of $\forall x. M(x)$ in this model 0, or is it 1?

[31] There might be a year when no CM students takes COMP11212 but this has not happened so far.

- $O(x)$ means that x has an out-of-hours pass for the building.

What is the boolean interpretation of

(a) $\forall x.(O(x) \to H(x))$ and

(b) $\forall x.(H(x) \to O(x))$?

Answer to the best of your knowledge, but be prepared to be quizzed about your answer.

CExercise 71. Consider the system with the following predicate and function symbols.

- A binary predicate T,
- a binary predicate L,

Take as the domain of definition is that of all first year students in the School. Assume that $T(x, y)$ means that x and y are in the same tutorial group, and $L(x, y)$ means that x and y are in the same lab group.

What is the boolean interpretation of the following propositions? Answer to the best of your knowledge, but be prepared to be quizzed about your answer.

(a) $\forall x.\forall y.(T(x, y) \to L(x, y))$

(b) $\forall x.\forall y.(\neg T(x, y) \to \neg L(x, y))$

(c) $\forall x.\exists y.(T(x, y) \wedge L(x, y))$

(d) $\exists x.\exists y.(L(x, y) \wedge \neg T(x, y))$

(e) $\forall x.\exists y.(\neg L(x, y) \vee \neg T(x, y))$

(f) $\exists x.\forall y.\neg T(x, y)$

Translating into predicate logic

So far we have looked at propositions in the new system and how to find their boolean interpretation in a given model. Often it is necessary to start from an intended model and to find a proposition that expresses a particular statement.

CExercise 72. For the model from the previous exercise, give propositions which express the following. We use ‘student’ here as a shortcut for ‘every first year student in the School’.

(a) Every student is in the same tutorial group as him- or herself.

(b) For every student there is a student who is in a different tutorial group.

(c) There is a student who is in the same labgroup as all students.

(d) Every two students who are in the same tutorial group are in the same labgroup.

(e) No student is in the same labgroup as all the other students.

There are many examples given in Chapter 2 which you should be able to translate into propositions.

Example 3.17. Assume we wish to express the commutativity of an operation. What do we need in order to be able to state this? An operation is a binary function, so we need a binary function symbol, say m. Further we need to fix the set S on which the operation happens as our domain of interpretation, and we need to be able to say that two elements of that set are equal, which means having a binary predicate, say E. Then we can say

$$\forall x.\,\forall y.\, E(m(x,y), m(y,x)),$$

to demand that

applying the operation $m$to first argument x and second argument y
is the same as
applying the operation $m$to first argument y and second argument x.

Example 3.18. We give one example here in the form of primeness, Definition 17. We repeat the definition here.

Definition. An element $n \neq 1$ of $\mathbb{N}$ (or $n \neq \pm 1$ in $\mathbb{Z}$) is **prime** if and only if for all elements k and l of $\mathbb{N}$ (or $\mathbb{Z}$) it is the case that

n divides kl implies n divides k or n divides l.

This defines a unary predicate on the set of natural numbers, namely primeness, and it requires a binary predicate, namely 'x divides y', and one binary operation, namely multiplication (since it has 'n divides kl'). We therefore choose as our symbols

- a binary predicate D,
- a unary predicate P, and
- a binary function m.

We pick as the domain of interpretation the set of natural numbers $\mathbb{N}$, and as the interpretation of m multiplication. We set the interpretation of $D(x,y)$ to 1 if and only if x divides y, while the interpretation of $P(x)$ is 1 if and only if x is prime. Then the definition of primeness, Definition 17, is expressed in the following:

$$\begin{aligned}\forall x.((P(x) \to (\forall y \forall z. D(x, m(y,z)) \to (D(x,y) \vee D(x,z))))\\ \wedge(\forall y.\forall z. D(x, m(y,z)) \to (D(x,y) \wedge D(x,z)) \to P(x)))\end{aligned}$$

This is rather long because we have to express 'if and only if'. For this reason a new connective, $\leftrightarrow$, is often used, which is intended to mean that each side implies the other. If we adopt this then the definition become the more readable

$$\forall x.(P(x) \leftrightarrow (\forall y.\forall z.D(x, m(y, z)) \rightarrow (D(x, y) \vee D(x, z)))).$$

Note that *the same* proposition can be assumed to talk about primeness in $\mathbb{Z}$ merely by changing the domain of interpretation!

EExercise 73. Translate the following definitions from Chapter 2 into propositions. Make sure you state what your parameter, function and predicate symbols are and what your domain of interpretation is and how each symbol should be interpreted.

(a) Associativity of an operation, Definition 18.

(b) Existence of an inverse for every element with respect to an operation, Definition 21.

(c) Injectivity of a function with the same source and target, Definition 22.

(d) Surjectivity of a function with the same source and target, Definition 23.

(e) Bijectivity of a function with the same source and target, Definition 24.

(f) One function with the same source and target being the inverse of another, Definition 25.

(g) One function from $\mathbb{N}$ to $\mathbb{N}$ eventually dominating another, Definition 46.

You may want to think about why we demand that our functions have the same source an target.

3.4.2 De Morgan rules and normal forms

There are De Morgan rules for our new system. Before we describe these there is something else we need to pay attention to, and that is bracketing.

Strictly speaking when we are building up a proposition we need to put brackets around various parts, which so far we have taken for granted. Once we have predicate logic putting in all brackets that would be required makes such expressions very hard to read for people (as opposed to machines). There are some conventions that allow us to leave out some brackets. This is done in analogy with arithmetic, where you know that

$$a \cdot x + b$$

means you multiply a with x and add b to the result, whereas

$$a + x \cdot b$$

means you add a to the result of multiplying x by b. We have swapped the addition symbol with the multiplication symbol, but we cannot obtain the meaning by swapping 'add' and 'multiply' in the instructions in the text. The reason for this is what we call *operator precedence*.

We say that multiplication takes precedence over addition, which means that we carry out multiplication operations before we do so for addition operations.

If we had ot put brackets everywhere we would write

$$(a \cdot x) + b$$

and

$$a + (x \cdot b)$$

for the two expressions above. Knowing the multiplication *has precedence over addition* allows us to leave out brackets.

The precendence regarding evaluation in predicate logic is as follows:

- negation $\neg$ is evaluated first, followed by
- universal $\forall$ and existential $\exists$ quahtification, follwed by
- conjunction and disjunction, followed by
- implication.

So if we have the proposiiton

$$\forall x.\, \exists y.\, (x \wedge y) \to x$$

then the correct way of putting in brackets without changing the meaning is to write it as

$$(\forall x.\, (\exists y.\, (x \wedge y))) \to x.$$

You should bear that in mind when solving the next exercise.

The De Morgan rules given for proposiitonal logic rules remain valid, but we also obtain one each for the new connectives.

We have that

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

and

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

There is a notion of disjunctive and conjunctive normal forms for this kind of system, but because of the various extra symbols this becomes more difficult to describe.

However, if there are only predicates as extra symbols then we may still use the ideas we have to find a semantically equivalent expression that is easier to understand.

Example 3.19. In such a simplified case we show how to simplify the given proposition.

$$\begin{array}{ll}
\forall x.\neg(\forall y.(P(x) \to Q(x,y))) & \\
\quad \equiv \forall x.\neg(\forall y.(\neg P(x) \vee Q(x,y))) & A \to B \equiv \neg A \vee B \\
\quad \equiv \forall x.\exists y.\neg(\neg P(x) \vee Q(x,y)) & \text{De Morgan } \exists \\
\quad \equiv \forall x.\exists y.(\neg\neg P(x) \wedge Q(x,y)) & \text{De Morgan } \vee \\
\quad \equiv \forall x.\exists y.(P(x) \wedge \neg Q(x,y)) & \neg\neg A \equiv A
\end{array}$$

Performing these steps results in a proposition that is easier to understand.

CExercise 74. Simplify the following propositions as far as you can.

(a) $\neg(\forall x.(P(x) \to \exists y.Q(x,y)))$,

(b) $\exists x.\neg(\exists y.(Q(y) \wedge R(x)) \to \forall z.P(z,x,y))$,

(c) $\forall y.\neg(\exists z.(P(y,z) \to \neg Q(z)))$,

(d) $\exists z.\forall y.(\neg P(y) \wedge Q(z)) \to R(y,z)$.

3.4.3 Systems with several types

Note that in the formal system given above there is no way of distinguishing between different kinds of things. We have no way, for example, to talk about students and course units as separate entities. In order to do that one would have to create a **typed system**. Doing that as a formal inference system results in something quite complicated, but we here give an idea of what propositions might look like in such a system.

Take for example:

$$\forall x : S.\forall y : U.((E(x,y) \wedge C(x)) \to D(y)).$$

The colon is there to tell us that we wish to demand this only for those x which belong to S and for those y which belong to U. We pick the following interpretations:

- S is the set of first year students at the University of Manchester.
- U is the set of all course units offered by the University of Manchester.
- $E(x,y)$ is 1 if and only if[32] (student) x is enrolled in (course unit) y.
- $C(x)$ is 1 if and only if x is a SH computer science student.
- $D(y)$ is 1 if and only if y is a COMP unit.

The interpretation of the proposition is that every course unit which a first year SH computer science student is enrolled on must be a COMP unit.

The School has a piece of software that checks whether each student is enrolled in the course units required for the degree, and the above is one of the many checks it has to carry out.

This last section is to give you an idea how one might set up a system that allows us to talk about entities of different kinds.

In practice people often use some of the symbols of our formal system *without* defining a full formal system themselves—they rely on ideas discussed in this chapter being known well enough that their expressions can be assumed to have an intended meaning.

[32] Note that the first input to E has to be a student and the second a unit if the given proposition is legal in the system.

Chapter 4

Probability Theory

Probabilities play a significant role in computer science. Here are some examples:

- One mechanism in machine learning is to have *estimates* for the relative probabilities of something happening, and to adjust those probabilities as the system gets more data. The most popular way of doing this is *Bayesian updating*, see Section 4.3.4.

- If you are running a server of some kind you need to analyse what the average, and the worst case, load on that server might be to ensure that it can satisfy your requirements.[1] Calculating such averages is one of the techniques you learn in probability theory.

- When trying to analyse data you have to make some assumptions in order to calculate anything from the data. We look at the question of what assumptions have what consequences.

- In order to calculate the *average complexity* of a program you have to work out how to describe the relative frequency of the inputs, and then calculate the average number of steps taken relative to these frequencies. This means you are effectively calculating the expected value of a random variable (see Section 4.4.6).

- There are sophisticated algorithms that make use of random sampling, such as *Monte Carlo methods*. In order to understand how to employ these you have to understand probability theory.

4.1 Analysing probability questions

Before we look at what is required formally to place questions of probability on a sound mathematical footing we look at some examples of the kinds of issues that we would like to be able to analyse.

In computer science we are often faced with situations where probabilities play a role, and where we have to make the decision about how to model the situation.

Every time we are trying to judge the risk or potential benefits of a given decision we are using probabilistic reasoning, possibly without realizing it. We have to come up with a measure of how big the potential benefit, or the potential disadvantage is, and temper that judgement by the likelihood of it occurring.

[1] You wouldn't the student system to go down if all students are trying to access their exam timetable at the same time.

When somebody buys a lottery ticket, the potential disadvantage is losing their stake money, and the potential advantage is winning something. How many people know exactly what their chances are of doing the latter?

Many games include elements of chance, typically in the form of throwing dice, or dealing cards. When deciding how to play, how many people can realistically assess their chances of being successful?

In machine learning, one technique is to model a situation by assigning probabilities to various potential properties of the studied situation. As more information becomes available, these probabilities are updated (this constitutes 'learning' about the situation in question). How should that occur?

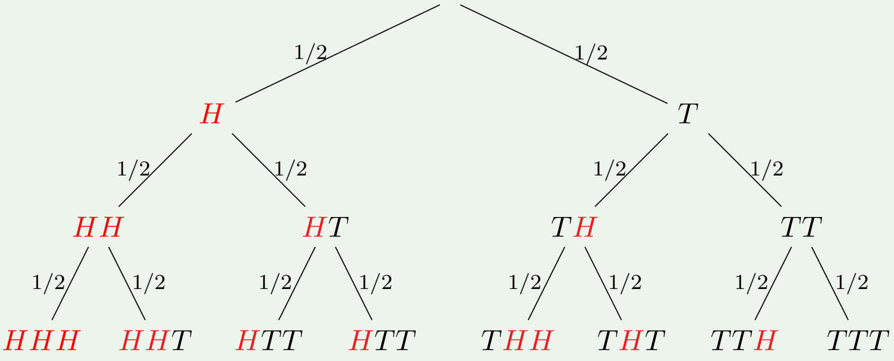
When looking at questions of the complexity of algorithms, one often applied measure is the 'average complexity', by which we mean the complexity of the 'average case' the program will be applied to. How does one form an 'average' in a situation like that?

All these questions are addressed in probability theory, but we have to restrict ourselves here to fairly basic situations to study the general principles. The first few problems we look at are particularly simple-minded.

4.1.1 Simple examples

Most people will have been confronted with issues like the following.

Example 4.1. An example much beloved by those teaching probabilities is that of a coin toss. When a fair coin is thrown we expect it to show 'heads' with the same probability as tails. For the chances to be even, we expect each to occur with the probability of $1/2$. What if we throw a coin more than once? We also expect that the outcome of any previous toss has no influence on the next one. This means we expect it to behave along the following lines.



In order to work out the probability of throwing, say, HTH, we follow down the unique path in the tree that leads us to that result, and we multiply the probabilities we encounter on the way down, so the probability in question is

$$\frac{1}{8}.$$

Note that because each probability that occurs in the tree is $1/2$, the effect will be that each outcome on the same level will have the same probability, which is as expected.

The tree also allows us to work out what the probability is of having the same symbol three times, that is having

$$HHH \qquad \text{or} \qquad TTT,$$

which means the event[2]

$$\{HHH, TTT\}$$

occurring. All we have to do is to add up the probabilities for each of the outcomes in the set, so the probability in question is

$$\frac{1}{8} + \frac{1}{8} = \frac{1}{4}.$$

See Section 4.1.4 for more examples where it is useful to draw trees.

Example 4.2. Whenever we throw a die, we expect each face to come up with equal probability, so that the chance of throwing, say, a 3 at any given time is $1/6$. It is quite easy to construct more complicated situations here. What if we throw two dice? What are the chances of throwing two 1s? What about throwing the dice such that the eyes shown add up to 7? See Exercise 80 and Example 4.22 for a detailed discussion of this particular question.

There are games where even more dice come into the action (for example Risk and Yahtzee), and while computing all probabilities that occur there while you're playing the game may not be feasible, it might be worth estimating whether you are about to bet on something very unlikely to occur.

Example 4.3. A typical source of examples for probability questions is as a measure of uncertainly of something happening. For example, a company might know that the chance of a randomly chosen motherboard failing within a year is some given probability. This allows both, the producing company and other manufacturers using the part, to make some calculations regarding how many cases of repairs under warranty they are likely to be faced with.

In particular, if you are a manufacturer seeking to buy 100,000 motherboards, then you have to factor in the costs of using a cheaper, less reliable part, compared with a more expensive and more reliable one. If you have a 10$ part which has a 5% chance to be faulty within the given period, you would expect to have around

$$100,000 \cdot .05 = 5000$$

cases. If on the other hand, you have a 12$ part that has a 3% chance of being faulty then you will have to pay 200,000$ more for the parts, and expect to have only

$$100,000 \cdot .03 = 3000$$

cases of failure under warranty. What is the better choice depends on how expensive it is to deal with each case, how many people you expect to make a claim, and whether you worry about the reputation of your company among consumers. Decisions, decisions…

[2] This is formally defined in Definition 32—for now just think of it as any set of outcomes.

Example 4.4. When you are writing software you may wonder how well your program performs on the 'average' case it will be given.

For a toy example, assume that your program takes in an input string, does some calculations, and returns a number. The number of calculation steps it has to carry out depends on the length of the input string. You would like to know how many calculation steps it will have to carry out on average so that you have an idea how long a typical call to that program will take.

Assume we have a string of length n. There is a function which assigns to each $n \in \mathbb{N}$ the number of calculation steps performed for a string of that length. It may not be easy to *calculate* that function, and you will learn more about how one might do that in both, COMP112 and COMP261. For the moment let's assume the function in question is given by the assignment

$$n \longmapsto n^2$$

from $\mathbb{N}$ to $\mathbb{N}$.

So now all we need is the average length of an input string to calculate the average number of calculations carried out. But what is that? This will depend on where the strings come from. Here are some possibilities:

- The strings describe the output of another program.
- The strings are addresses for customers.
- The strings encode DNA sequences.
- The strings describe the potential status of a robot (see Example 4.44).
- The strings are last names of customers.

In each situation the average length will be different. You need to know something about where they come from to even start thinking about an 'average' case.

If we have a probability for each length to occur then we can calculate an average, see Definition 42 for that.

Note that typically the number of instructions that has to be carried out in a typical computer program depends on more than just the size of the input. With many interesting algorithms (for example searching or sorting ones) what exactly has to be done depends on the precise nature of the input. See Examples 4.96 and 4.98 for a discussion of two such situations.

4.1.2 Counting

When modelling situations using probability we often have to count how many possibilities there are, and how many of those have particular properties.

We give some rules here that help with taking care of this.

Selection with return

Assume we are in a situation where there are n options to choose from, and that we may choose the same option as many times as we like. If we choose i many times and we record the choices in the order we made them, then there are

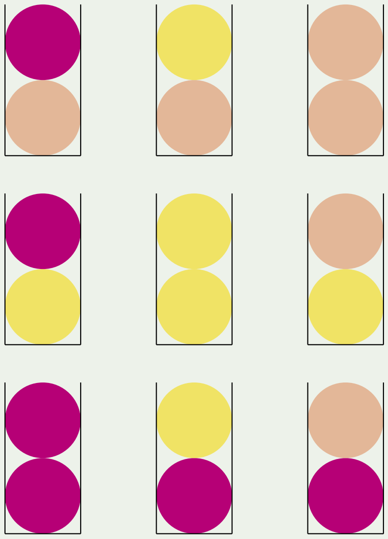
$$n^i$$

possible different possibilities.

Example 4.5. If we toss a coin then on each toss there are two options, heads and tails. If we toss a coin i times then there are 2^i many possible combinations.

Example 4.6. Let's assume we have various flavours of ice cream, and we put scoops into a tall glass so that they sit one above each other. If you may choose 3 scoops of ice cream from a total of n flavours then there are n^3 many combinations, assuming all flavours remain available.

Below we show all the combinations of picking two scoops from three flavours, say hazelnut, lemon, and raspberry.



There are $3^2 = 9$ possible combinations.

The reason this is known as 'selection with return' is that if we think of the choice being made by pulling different coloured balls from an urn (without being able to look into the urn), then one should picture this as drawing a ball, recording its colour before returning it to the urn, drawing a second ball, recording its colour before returning it, and so on.

Selection without return

If we have a choice of n possibilities, and we choose i times in a row, but we may not choose the same item twice, then there are

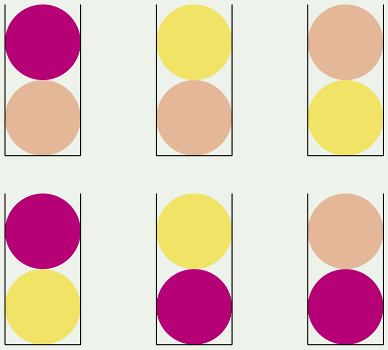
$$n(n-1)\dots(n-i+1) = \frac{n!}{(n-i)!}$$

different combinations, that is listings of choices in the order they were made.

Example 4.7. If you have to pick three out of fifteen possible runners to finish first, second and third in that order there are $15 \cdot 14 \cdot 13 = 2730$ possibilities.

Example 4.8. If you have a program that gives you a design for a webpage, where you have to pick three colours to play specific roles (for example, background, page banner, borders), and there are 10 colours overall, then you have $10 \cdot 9 \cdot 8 = 720$ combinations.

Example 4.9. Returning to the ice cream example, if children are given a tall glass in which they each are allowed two scoops from three flavours, but they may pick every flavour at most once (to make sure popular flavours don't run out) then they have the following choices.



There are now $3 \cdot 2 = 6$ possibilities.

This is known as selection without return because we can think of it as having an urn with n differently coloured balls, from which we choose one ball after the other, *without returning them to the urn* and recording the colours in the order they appear.

What happens if the balls don't each have a unique colour?

Ordering

If we have n different items then there are $n!$ many ways of ordering them, that is, of writing them one after the other. This is the same as choosing without return n times from n possible options. If the items are not all different then the number of visibly different possibilities is smaller.

Example 4.10. If we have a red, a blue, and three black mugs and we are lining them up in a row then the number of possibilities is

$$\frac{5!}{3!} = 20.$$

There would be $5!$ possibilities for lining up 5 different mugs, but in each one of those we wouldn't spot the difference if some of the black mugs were swapped. There are $3!$ ways of lining up the three black mugs (but if we assume that the mugs are indistinguishable then we cannot tell the difference between the different orderings).

In general, if we have n items and there are n_1 copies of the first design, n_2 copies of the second, and so on, to n_i items of the ith design then there are

$$\frac{(n_1 + n_2 + \cdots + n_i)!}{n_1! \cdot n_2! \cdot \cdots \cdot n_i!}$$

visibly different ways of lining up the items.

Selection without ordering

Sometimes we are confronted with the situation where we have to count how many different selections there are, but where we are not told the order in which this selection arises. A typical example is a lottery draw:

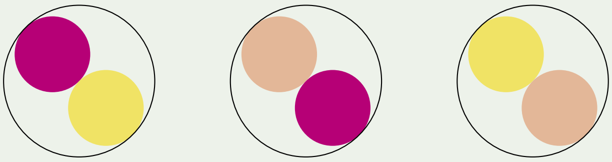
One way of counting these is to list all the options as we have done above, but that gets cumbersome if the numbers involved are bigger. An alternative way of counting is to count how many selections there are with ordering being taken into account, and then dividing by the number of different orderings there are for each choice.

Example 4.11. If we return to Example 4.9, then we can look at the situation where the children are given a shallow bowl rather than a tall glass with scoops of ice cream. Again they are allowed to choose two scoops from three flavours, and again they may pick every flavour at most once.

We know from Example 4.9 that there are 6 possible combinations when the order is taken into account. For each choice of two flavours there are two ways of ordering them, so we now have

$$\frac{3 \cdot 2}{2} = 3$$

combinations.



In general, when i items are picked from a choice of n different ones, there are

$$\frac{n(n-1)\ldots(n-i+1)}{i!} = \frac{n!}{(n-i)!i!}$$

different selections.

Summary

The formulae given above for the number of possibilities are summarized in the following table. Here n is the number of items available and i is the number of items that are selected. Note that the assumption is that in the unordered case, all items are different.

ordered		unordered
with return	without return	
n^i	$\frac{n!}{(n-i)!}$	$\frac{n!}{(n-i)!i!}$

Note that there is no simple formula for the number of possibilities there are when looking at unordered selections of items some of which may be identical. In this case the formula for the number of different orderings may be useful. This says that if there are n items, of which there are n_1 indistinguishable copies of a particular kind, n_2 copies (also indistinguishable among themselves) of a second kind, and so on, with i many kinds altogether, then there are

$$\frac{(n_1 + n_2 + \cdots + n_i)!}{n_1! \cdot n_2! \cdot \cdots \cdot n_i!}$$

many visibly different orderings.

Optional Exercise 8. Work out why there is no simple formula as discussed in the previous paragraph‘ by looking at some examples.

Exercise 75. Assume you have 3 red socks and 5 black ones. Answer the following questions

(a) Assume we put all the socks into a bag. Four times we draw a sock from the bag, putting it back each time. How many different draws are there?

(b) Make the same assumption as for the previous part, but now assume we don't put the drawn socks back into the bag. How many draws are there?

(c) Assume we put the socks onto a pile, close our eyes, mix them around, and pick four socks from the pile. How many different combinations do we get?

(d) Can you answer the same questions if you assume we have m red and n black socks? What if we pick k socks (for $k \leq m + n$) many socks on each occasion?

Exercise 76. A researcher in the rain forest has left his laptop unattended and a curious monkey has come to investigate. When the researcher looks up from the plant he is studying he sees the monkey at the keyboard. He makes threatening noises as he runs back. Assume that every time he shouts there's a 50% chance that he will manage to disrupt the monkey before it makes another key stroke, and that he will have reached the laptop before he has shouted six times. Draw a tree similar to that in Example 4.1 for the situation. What do you think is the average number of key strokes the monkey will manage in this situation?

4.1.3 Combinations

Sometimes we have to combine these ideas to correctly count something.

Example 4.12. If we throw a coin three times then there are 2^3 many possible outcomes. If we want to know how many of those contain at least two heads we have to think about how best to count the number of possibilities.

One possibility is to say that we are interested in

- the situation where there are three heads, of which there is one combination, and
- the situation where there are two heads and one tails. This asks for the number of different ways of ordering H, H, T and there are

$$\frac{3!}{2!} = 3$$

of those (or there are the positions where the unique T can go and then the two H take up the remaining positions).

But this way of thinking does not scale well. What if we want to know how many outcomes have at least 10 heads when we toss the coin 20 times? Following the above idea we have to add up the number of combinations with 20, 19, 18, and so on, down to 10 occurrences of H.

Or we can argue that there are 2^{20} possibilities overall, of these $20!/(10! \cdot 10!)$ contain exactly ten times heads and ten times tails and of the remaining combinations half will have a higher count of heads, and half will have a higher count of tails.

There are

$$\frac{20!}{10! \cdot 10!} = \frac{2 \cdot 19 \cdot 2 \cdot 17 \cdot 2 \cdot 15 \cdot 2 \cdot 13 \cdot 2 \cdot 11}{5!} = \frac{19 \cdot 17 \cdot 2 \cdot 13 \cdot 2 \cdot 11}{1} = 184756$$

ways of ordering ten heads and ten tails. The number of combinations of at least 10 heads is then

$$\frac{2^{20} - 184756}{2} + 184756 = \frac{2^{20}}{2} + \frac{184756}{2} = 616666.$$

By thinking about how to count in the right way calculations can be shortened significantly.

CExercise 77. Work out how many outcomes there are in the following cases. Please give an expression that explains the number you have calculated.

(a) Four digit personal identification numbers (PINs). How many times do you have to guess to have a 10% chance of finding the correct PIN?

(b) How many passwords are there using lower case letters? How many times do you have to guess now to have a 10% chance of being correct?

(c) What if upper case letters are included?

(d) How many possible lottery draws are there if six numbers are drawn from 49? How many bets do you have to make to have a 1% chance of having all numbers correct?

(e) Assume you have an array consisting of 10 different integers. What is the probability that the array is sorted? What happens if the integers are not all different?

(f) Assume you have an array consisting of 30,000 id numbers. What is the probability that you randomly pick the one you were looking for? What can you say about the case where the array is sorted?

(g) In an examples class there are 60 students and 6 TAs. Each TA marks 10 students. Assuming the students all have sat down in groups of ten, how many different combinations of TAs and groups are there? What is your chance of having a particular TA this week?

(h) Assume that there are 6 people who want to randomly split into three teams. For this purpose they put two red, two green and two yellow ribbons into a bag, and each person picks one of those out without looking into the bag.

What is the probability that Amy will be on the red team? What is the chance that she will be on the same team as Zenia? How many different ways of splitting the six members into teams are there?

(i) Students from CSSOC are wearing their hoodies. Four of them have a purple, two a green, and one a black one. They line up in a queue to leave the room they are in. What is the probability that all the people in the same colour hoodie are next to each other? What is the probability that no two people wearing a purple hoodie are next to each other?

Exercise 78. Work out how many outcomes there are in the following cases. Please give an expression that explains the number you have calculated.

(a) Assume you are at a party. Somebody asks each person when their birthday is. How many people have to be at the party for the probability that two of them share a birthday to be larger than 50%?[3]

(b) Assume you are composing a phrase of music over two four beat bars. You may use one octave, and any duration from a quaver (an eighth note) to a semibreve (a whole note). How many melodies are there?

4.1.4 Using trees

Sometimes we can picture what happens in a situation by using trees to provide structure.

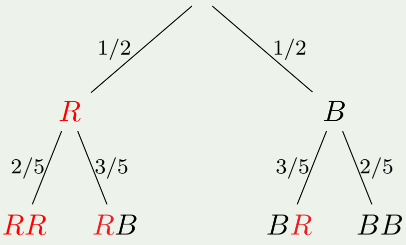
Example 4.13 (Drawing socks). We may use trees to gain a better understanding of a particular situation. The name 'decision tree' is slightly misleading here since we do not just model decisions that somebody might make but also random moves.

Assume you have a drawer with six individual socks, three red and three

[3] This is known as the *birthday paradox*, although it is not strictly a paradox, merely a question with a surprising answer. It is why computer scientist have to worry about *collisions* when designing hash tables.

black (let's not worry about how you ended up with odd number of socks in both colours). We may answer the question of how many socks we have to pick in order to be sure to get one matched pair—if we pick three socks then there will be at least two which are the same.

But what if we want to know how many socks we have to pick to have a chance of at least 50% of achieving this? We picture our first two draws as follows.



What is the chance of having two socks of the same colour after two attempts? Of the four possible outcomes two are of the kind we want, namely RR and BB. In order to find out the probability of these two events we *multiply* probabilities as we go down the tree.

- RR. The probability for this event is determined by multiplying the probabilities that appear along the path from the root of the tree to that outcome, so it is $1/2 \cdot 2/5 = 1/5$.

- BB The probability is determined in the same way, and also works out to be $1/2 \cdot 2/5 = 1/5$.

To calculate the probability of the event of having two socks of the same colour,

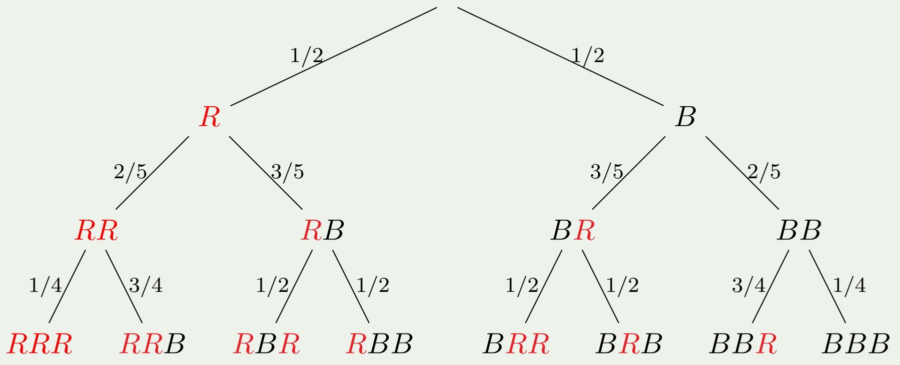
$$\{RR, BB\},$$

we *add* the probabilities of the two outcomes contained, and so we have

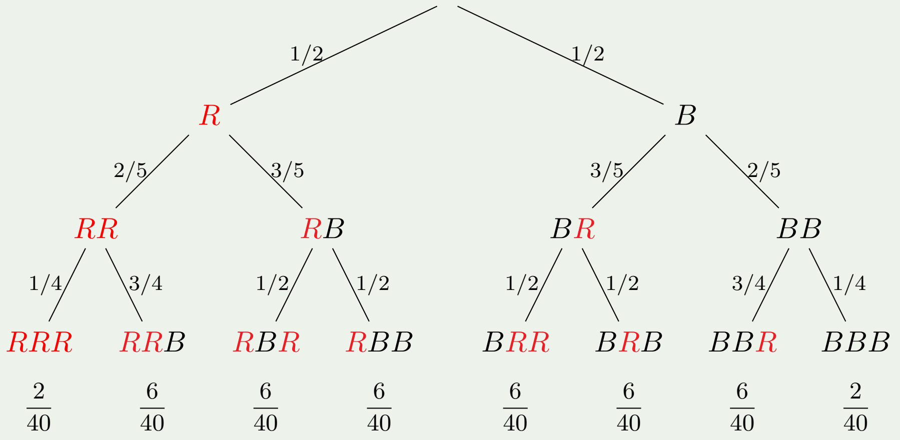
$$\frac{1}{2} \cdot \frac{2}{5} + \frac{1}{2} \cdot \frac{2}{5} = \frac{2}{5} = 40\%.$$

Hence in order to guarantee a success rate of at least 50% we have to have (at least) three draws, and in that case we know we will have a 100% success rate.

Let us look at the question of picking at least two black socks. With two draws the chance of succeeding is $2/10 = 1/5$. If we add a third draw we get the following.



We may now calculate the probability that any of these draws occurs by *multiplying* the probabilities that occur along the corresponding path; we give these probabilities below each leaf:



The outcomes where we have two black socks are

$$\{RBB, BRB, BBR, BBB\},$$

If we add up their probabilities we get

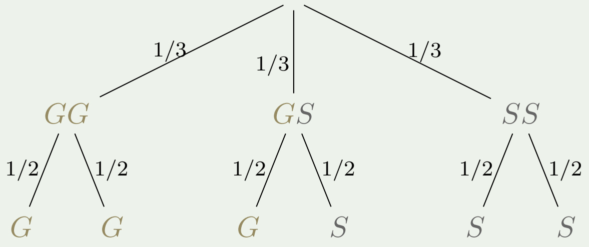
$$\frac{6}{40} + \frac{6}{40} + \frac{6}{40} + \frac{2}{40} = 20/40 = 50\%.$$

You might arrive at this result without drawing the tree, but it certainly clarifies matters to have it at hand, and if you have to answer more than one question about some situation you only have to draw it once.

Example 4.14 (Gold and Silver). Assume there are three bags, each with two coins. One has two coins of gold, another two coins of silver and a third one coin of each kind. Somebody randomly picks a bag, and then draws a coin from the bag without looking inside.

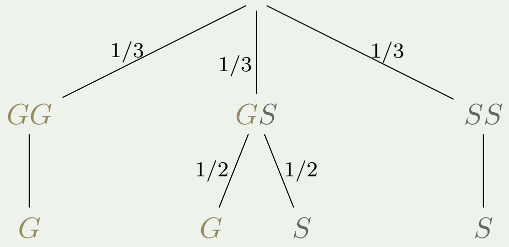
We are shown that the selected coin is gold. What is the chance that the remaining coin from that bag is also gold?

Again we use a tree to understand what is happening.



If we know that a gold coin has been drawn we must be seeing the first, second or third outcome from above. All these are equally likely, with a probability of $1/6$ each. Two out of the three have a second coin which is also gold, so the desired probability is $2/3$.

Instead of explicitly looking at both coins in the bag, as we did in the tree above, we could have a different event, namely the colour of the drawn coin. If those are our chosen outcomes then the corresponding tree looks like this.

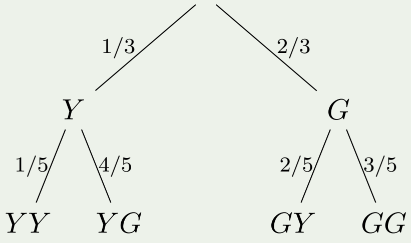


Now we argue that knowing the drawn coin is gold tells us that we have either the first or the second outcome. The former occurs with probability $1/3$, the second with probability $1/6$ overall, so the former is twice as likely as the latter, again giving a probability of $2/3$ that the second coin is also gold.

Example 4.15. Bonny[4] and Clyde are playing a game. They put two yellow and four green ribbons into a bag. Without looking inside, each of them reaches into the bag and draws a ribbon.

If the ribbons have the same colour Bonny wins and if they are different, then Clyde wins. We want to know whether the game fair, that is, if they both have an equal chance of winning.

This is question is much easier to answer if we draw a tree.



From the tree we can read off that the probability of drawing the same colour, the event $\{YY, GG\}$, has the probability

$$\frac{1}{3} \cdot \frac{1}{5} + \frac{2}{3} \cdot \frac{3}{5} = \frac{7}{15},$$

while the probability of drawing different colours, the event $\{YG, GY\}$, has the probability

$$\frac{1}{3} \cdot \frac{4}{5} + \frac{2}{3} \cdot \frac{2}{5} = \frac{8}{15}.$$

The two numbers are different and so the game is not fair. Clyde has a higher chance of winning.

Example 4.16 (**The Monty Hall problem**). A well-known problem that we may use for illustrative purposes is known as the *Monty Hall problem*.

Imagine you are in a game show. There are three closed doors labelled A, B and C, and you know that behind one of them is a valuable prize (in the original story a car) and behind two of them is something not worth having (in the original story a goat).

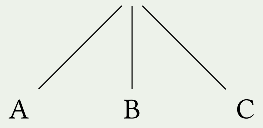
The way the game works is that you pick a door, and then the show master opens one of the remaining doors. You see the booby prize. You are now offered the chance to switch to the other closed doors. Should you switch, or stick with your original choice?

This situation has been endlessly discussed among various groups of people, often because somebody knows the solution and somebody else doesn't want to believe it.

So how does one model a situation like that reliably? Usually when there are steps in a situation it is worth modelling these steps one by one.

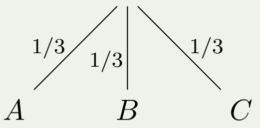
What do we know for sure? We know that at the beginning there are three doors, let's assume with two goats and a car. We assume that the probability of the car being behind any one of the doors is the same. From the point of view of the contestant this is like a random event. The production company picks an actual door, and there is no way of telling how they decide which one to hide the main prize behind, but one might hope that they really do pick any door with the probability of $1/3$, and that's the assumption the contestant should make. The action of the show master afterwards has to depend on the choice made by the contestant, and we make the additional assumption that if the show master has a choice of opening a door he will open them with equal probability.

We can model the choices step by step using a tree. In the first step we model the fact that the car might be behind any one of the doors.

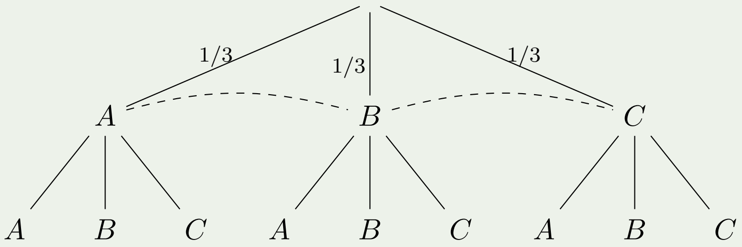


We put probabilities in the tree which indicate that the car can be behind each of them with equal probability.

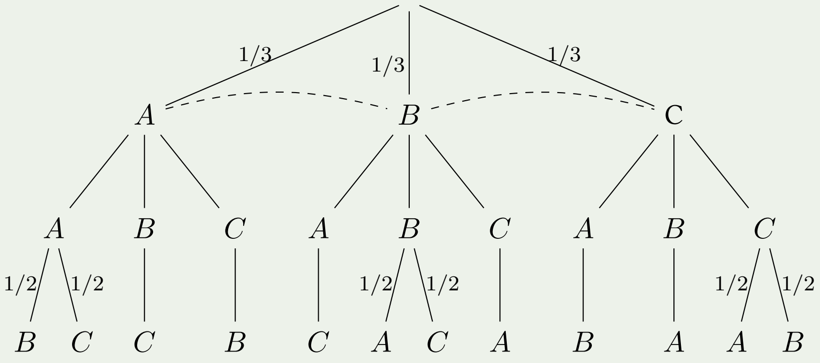
[4] This is a past exam question.



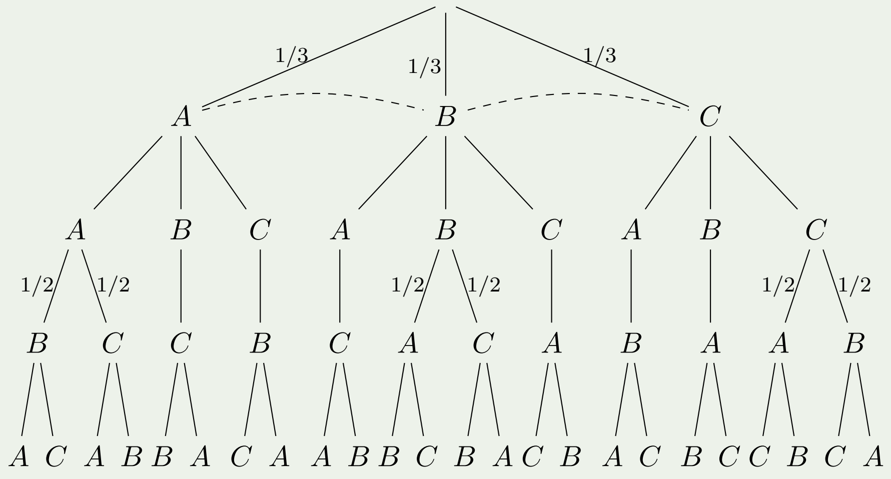
There are three possibilities for the player to choose a door. But note that the player does not know which of these three positions she is in. In game theory one says that the leaves of the tree are in the same *information set*. So from the player's point of view there are three choices (pick door A, B or C), and she cannot make that dependent on where the car is since she does not have that information. This is similar to the situation in many card games where the player has to choose what to play without knowing where all the cards are situated. Only in the course of further play does it become clear what situation the players were in. In the tree we denote this by a dashed line connecting the positions which the player cannot distinguish.



The next step is for the show master to open one of the doors showing the booby prize. In some cases there is only one possible door to open, in others there is a choice between two, and we assume that he picks either one of them with equal probability.

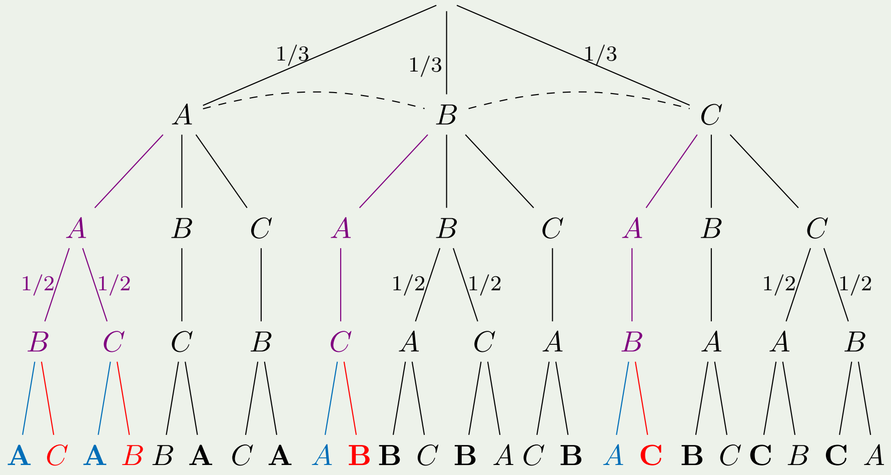


Now the player has to decide whether she wants to switch or not. Again we draw the possible options. We first give the door where the player has not switched, and then the one where she has.



We note that the three principal subtrees are the same up to renaming of nodes. This is because from the Player's point of view, there are only three options (which door to pick), and the first step drawn in the tree (the selection of the door which hides the prize) is completely hidden. The result of that step is not revealed to the player until the end of the game. We look at the question of what happens if the player switches or sticks with her first choice. In purple we highlight the case where the player's first choice is A. The remaining two possibilities give the same result. We also highlight those position where the player wins the main prize by giving it in bold. We now look at two strategies:

- Pick A on the first move and then stick with this choice, given in blue (this is the left choice in the fourth layer of the tree).
- Pick A on the first move and then switch when given the chance to do so, given in red (this is the right choice in the fourth layer of the tree).



If the player picks door A on the first move, and then sticks to that choice, there is a chance of $1/3$ that the original choice was correct, and then no matter which door is opened the main prize is achieved. So a player who does not switch will get the main prize with probability of $1/3$.

A player who picks door A on the first move and then switches, had the correct door with probability $1/3$ and then switches away, which means that he obtains the main prize with probability $2/3$.

For this reason a player who picks a door and then switches has a chance which is twice as high to get the main prize than the player who sticks.

Note that in particular we can see from this example that it may be that what looks like one choice to the player (for example 'pick door A') is effectively a choice taken in a number of different situations the player cannot distinguish between (here the player does not know behind which door the prize is hidden)—in that case the choice will be reflected in several subtrees.

Note that we can tell if a tree properly describes a probabilistic situation:

Proposition 4.1

A tree with probabilities along some of the edges describes a situation of choices and probabilistic moves if and only if for every node in the tree the probabilities of the edges going down from that node add up to 1.

Tip

Drawing a tree is often very useful when trying to understand probabilities. For a tree to work you have

- structure the process being described into distinct stages, each of which describes either a probabilistic process or a choice some agent may make (as in the Monty-Hall problem);
- for each such probabilistic process find some way of describing its possible outcomes (for example all the possible deals in a card game, or all the possible first cards you might receive in such a deal)—for example, the colour of the sock drawn in Example 4.13;
- annotate each branch that stands for a particular outcome of some probabilistic process with the probability that it occurs—in the same example the probability that we draw a red/black sock, given which socks have already been drawn;[5]
- the leaves of the tree should cover to all the overall outcomes you are interested in—for example, the various combinations of socks we may obtain having drawn three times in Example 4.13.

Note that can be several trees that describe the given situation, and which one suits you best will depend on what you are expected to calculate with that tree.

Tip

Once you have a tree it is easy to calculate probabilities of specific outcomes.

- The probability that a given leaf (which corresponds to an overall outcome of the situation we want to describe) can be computed by *mul-*

[5] Note that by the previous proposition, if we add up the probabilities annotating all the branches that start at one particular location, the result must be 1

tiplying all the probabilities that occur on the path from the root of the tree to that leaf.

- The probability that a particular set of outcomes occurs can be calculated by adding the probabilities of all the leaves of the tree which belong to that set.

CExercise 79. Suppose we have a deck of four cards,

$$\{Q\spadesuit, A\spadesuit, Q\heartsuit, A\heartsuit\}.$$

I draw two cards from this pack so that I can see their values, but you cannot. You tell me to drop one of my cards, and I do so. You ask me whether I have the ace of spades $A\spadesuit$ in my hand, and I answer yes.

What is the probability that the card I dropped is also an ace? *Hint: Draw a tree, but note that if you read the given information carefully you don't have to draw all possibilities. How many different draws are there? You'll make your life more complicated if your tree contains more nodes than needed.*

Exercise 80. Assume you are throwing two dice, a red and a blue one.

(a) What is the probability that the sum of the eyes is exactly 4?

(b) What is the probability that the sum of the eyes is at least seven?

(c) What is the probability that there is an even number of eyes visible?

(d) What is the probability that the number on the red die is higher than that on the blue?

4.1.5 Further examples

In the previous sections it was clear from the context which principles you had to apply to find a solution. The point of the following exercises is that you first have to think about what would make sense in the given situation.

EExercise 81. Assume two teams are playing a 'best out of five' series which means that the team that wins three matches is the winner of the series.[6] Note that once it is clear that one side has won, the remaining matches are no longer played. For example, if one team wins the first three matches the series is over.

(a) Assume that the two teams are equally matched. After what number of matches is the series most likely to end?

(b) How does the answer change if the probability of one team winning is 60%?

[6] Such series take part, for example, in men's matches in Grand Slam tennis tournaments, where the winner of each bout is determined in a 'best out of five' sets. In women's matches, and men's matches outside of Grand Slam tournaments, the winner is determined in a 'best out of three' series.

Exercise 82. Solve the same problem as for the previous exercise, but with a 'best out of seven' series.

CExercise 83. Imagine you have a die that is loaded in that even numbers are twice as likely to occur than odd numbers. Assume that all even numbers are equally likely, as are all odd numbers.

(a) What is the probability of throwing an even number?

(b) What is the probability that the thrown number is at most 4?

(c) With two dice of this kind what is the probability that the combined number of eyes shown is at most 5?

Exercise 84. Assume you have a coin that shows heads half the time and tails the other half, also known as a *fair coin*. Assume the coin is thrown 10 times in a row.

(a) What is the probability that no two successive throws show the same side?

(b) What is the probability that we have exactly half heads and half tails?

(c) What is the chance of having at least five subsequent throws showing the same symbol?

Exercise 85. Assume we toss a fair coin until we see the first heads. We want to record the number of tosses it takes. What is the probability that we require 10 tosses ore more?

4.2 Axioms for probability

In the examples above we have assumed that we know what we mean be 'probability', and that we have some rules for calculating with such numbers.

4.2.1 Overview

This section puts these intuitive ideas onto a firm mathematical footing. It does so in a very general way which you may find difficult to grasp. However by setting this up so generally we give rules that can be applied to *any* situation. Thinking about these rules also encourages you to think about how to model specific situations you are interested in, and to take care with how you do so.

The idea underlying probability theory is that we often find ourselves in a situation where we can work with

- a *sample space* S of all possible outcomes,
- a *set of events* $\mathcal{E}$ (which is a subset of the powerset of S) and
- a *probability distribution* which is given by a function

  $$P\colon \mathcal{E} \longrightarrow [0,1],$$

  where $[0,1]$ is the interval of real numbers from 0 to 1.

Example 4.17. The simplest kind of probability space is one where there are n options, say

$$S = \{s_1, s_2, \dots, s_n\},$$

and all these occur with equal probability. In this case the set of events is the set $\mathcal{P}S$ of all subsets of S, and the probability distribution

$$P\colon \mathcal{P}S \longrightarrow [0, 1]$$

is given by the assignment

$$S \longmapsto \frac{|S|}{n},$$

that is, every set is mapped to its number of elements divided by n.

We give precise definitions of what we mean with these notions below, but for the moment let's look at a slightly more complicated example.

Example 4.18. In a simple dice game the participants might have two dice which they throw together. If the aim of the game is to score the highest number when adding up the faces of the dice then it makes sense to have the possible outcomes

$$S = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}.$$

We call the set of possible outcomes the *sample space* S. We could now ask what the probability is of throwing at most 5, which is the event

$$\{2, 3, 4, 5\}.$$

This example is continued below.

Typically when we have a finite set of outcomes S we assume that the set of events $\mathcal{E}$ is the whole powerset $\mathcal{P}S$. When we have an infinite set of outcomes this is not always possible. There is a field in mathematics called *measure theory* which is concerned with which sets of events can be equipped with probability functions, but that goes beyond this course.

Often it is possible to make the sample space finite, and this frequently (but not always) happens for computer science applications.

Example 4.19. The following example is a toy version of a problem that was the basis of a lab in the introductory AI unit. It is concerned with a robot wanting to learn its location in a two-dimensional space. If we think of the location as being given by two coordinates, and the coordinates as real numbers, then there are uncountably many locations in the unit square

$$[0, 1] \times [0, 1].$$

But we cannot measure the location of the robot up to infinite precision (and indeed, we're not interested in the answer to that level of precision), and in the robot exercise a 100×100 grid is imposed on the space, and we are only interested in which one of the squares in the grid the robot inhabits. This means the sample space now has only $100 \cdot 100 = 10,000$ elements.[7]

Example 4.20. If you are interested in the price of a commodity, it typically makes sense to measure the price only up to a limited precision (typically a few post decimal digits), and again this has the effect of making the sample space finite.

We consider many finite sample spaces in this chapter, but we do have a look at infinite notions as well since that also occurs in some applications in computer science. For this reason we here give definitions which are general enough to apply to both cases.

Example 4.21. When we consider events that happen over a given time frame it is often more convenient to treat that time frame as a real interval, $[r, r']$, where r is the start time and r' is the end time. The reason for this is that the entities we would like to compute can typically be computed with the help of integrals. These ideas are pursued in Examples 4.27, 4.28 and 4.84 and Exercises 89 and 114.

Not every function satisfies the requirements of a probability function, and we look at what properties we expect below. In order to formulate what we expect from a probability function P we first have to look at what we expect from the set of events.

4.2.2 Events and probability distributions

The following two definitions are given here for completeness' sake.[8] Above we did not worry about the properties required of probability distributions, and we also did not wonder whether a given set of outcomes could be an event, or not.

When we wish to consider a sample space that is uncountable[9], for example the real interval $[0, 1]$, it is difficult to find a probability space for this set of outcomes. There are two difficulties:

- If we want to assign the same probability to each element of $[0, 1]$ then this probability *has* to be 0 (otherwise the probability for the whole interval would be infinite, compare Proposition 4.4). The only way of defining a probability function with this property is to define a function that takes as its input events (that is, sets of outcomes).

- It is not possible to give a probability distribution that assigns a probability to *every* subset of $[0, 1]$, see Proposition 4.5. How to define a probability space in this situation is sketched in Proposition 11. It has the property that the probability of any interval $[r, r']$ in $[0, 1]$ has a probability proportional $r' - r$, that is, its probability is determined by its length.

For this reason we describe here which collection of subsets of the sample space is suitable to form a probability space.

[7] See Example 4.44 for a simplified version of this scenario.

[8] In particular these two definitions are not part of the examinable material.

[9] See Definition 50—for now stay with the example of the unit interval.

Definition 30: σ-algebra

Let S be a set. A subset $\mathcal{E}$ of $\mathcal{P}S$ is a *σ-algebra* provided that

- the set S is in $\mathcal{E}$,
- if E is in $\mathcal{E}$ then so is its complement $S \setminus E$ and
- if E_i is in $\mathcal{E}$ for $i \in \mathbb{N}$ their union $\bigcup_{i\in\mathbb{N}} E_i$ is in $\mathcal{E}$.

We note some consequences of this definition. First of all, since S is in $\mathcal{E}$ we may form its complement to get another element of $\mathcal{E}$, and so

$$\emptyset = E \setminus E$$

is in $\mathcal{E}$.

Further note that the union of a finite number of events must also be an event: If we nave events E_0, E_1, ..., E_n then we can set $E_i = \emptyset$ for $i > n$, and then

$$\bigcup_{i\in\mathbb{N}} E_i = E_0 \cup E_1 \cup \cdots \cup E_n.$$

Note that for every set S the powerset $\mathcal{P}S$ is a σ-algebra.

Events which are disjoint play a particular role: If we have two sets of possible outcomes, say E and E', and these sets are disjoint, then we expect that the probability of $E \cup E'$ is the probability of E added to that of E'. But this is not a property of just two sets of outcomes—sometimes we need to apply it to to larger collections of sets. This means we have to worry about what the appropriate generalization of 'disjoint' is.

If we have three sets of outcomes, events E, E' and E'', then in order for

$$P(E \cup E' \cup E'')$$

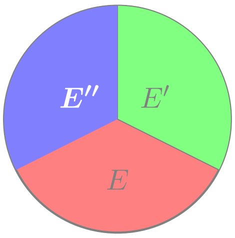
to be equal to

$$PE + PE' + PE''$$

to hold it must be the case that none of these sets 'overlap', in other words, we need that

$$E \cap E' = \emptyset, \qquad E \cap E'' = \emptyset, \qquad E' \cap E'' = \emptyset,$$

as for example in the following picture.



If we want to apply this idea to more than three sets we need to use a general definition.

Definition 31: pairwise disjoint

Let S be a set. Further assume that we have an arbitrary set I, and that for each element $i \in I$ we have picked a subset S_i of S. We say that the collection of the S_i, where $i \in I$, is **pairwise disjoint** if and only if

$$\text{for } i, j \in I \text{ we have} \qquad i \neq j \qquad \text{implies} \qquad S_i \cap S_j = \emptyset.$$

This means that the sets we have picked for different elements of I do not overlap.

EExercise 86. Assume that we have a set S. We are also given two disjoint subsets B_1 and B_2 of S and a collection E_i, for $i \in \mathbb{N}$, pairwise disjoint subsets of S.

(a) Show that for $A \subseteq S$ we have that $A \cap B_1$ and $A \cap B_2$ are disjoint. *If you can do the next part without doing this one you may skip it.*

(b) Show that for $A \subseteq S$ we have that $A \cap E_i$ is a collection of pairwise disjoint sets.

(c) Show that for $A \subseteq S$ we have that

$$A \cap (B_1 \cup B_2) = (A \cap B_1) \cup (A \cap B_2).$$

If you can do the next part without doing this one you may skip it.

(d) Show that for $A \subseteq S$ we have that

$$A \cap \bigcup_{i \in \mathbb{N}} E_i = \bigcup_{i \in \mathbb{N}} (A \cap E_i).$$

(e) Show that if $A \subseteq B_1 \cup B_2$ then A is the disjoint union of $A \cap B_1$ and $A \cap B_2$. *If you can do the next part without doing this one you may skip it.*

(f) Show that if $A \subseteq \bigcup_{i \in \mathbb{N}} E_i$ then A is the disjoint union of the $A \cap E_i$.

Definition 32: probability space

A **probability space** is given by

- a *sample set* S;
- a *set of events* $\mathcal{E} \subseteq \mathcal{P}S$ which is a σ-algebra and
- a *probability distribution*, that is a function

  $$P \colon \mathcal{E} \longrightarrow [0, 1],$$

  with the properties that

  - $PS = 1$ and
  - given E_i, for $i \in \mathbb{N}$, pairwise disjoint[10], then[11]

  $$P(\bigcup_{i \in \mathbb{N}} E_i) = \sum_{i \in \mathbb{N}} P(E_i).$$

[10] Note that some authors write a disjoint union using the addition symbol $+$, and $\sum$ for infinite such unions, but we do not adopt that practice here in case it causes confusion.

[11] Note that below appears a potentially infinite sum, that is, a sum which adds infinitely many

These axioms for probability go back to the Russian mathematician *Andrey Kolmogorov* who was trying to determine what the rules are that make probabilities work so well when describing phenomena from the real world. His rules date from 1933. What we have done here is translate them into a more modern setting.

These axioms may seem complicated, but they are quite short, and they have a lot of consequences which you may have learned about when studying probability previously. We look at these in the following section.

Tip

You are not expected to fully understand the definition of a probability space, in particular that of a σ-algebra, and in practice it is certainly sufficient to understand the examples given in the text. The formal definition is included to demonstrate that mathematics is built entirely using formal definitions.

Many students lose marks when asked to give a probability space, because they describe the outcomes, their probabilities but they neglect to mention the events. Study the examples in Section 4.2 until you are sure you can always identify the set of events.

The following optional exercises invite you to understand more about the formal definition of a probability space.

Optional Exercise 9. In the definition of a probability distribution we can see an infinite sum. Under which circumstances does it make sense to write something like that? Try to find a probability distribution for the natural numbers, with $P\mathbb{N}$ as the set of events. *Hint: It is sufficient to give probabilities for events of the form* $\{n\}$.

Optional Exercise 10. Assume you want to find a probability distribution for the sample space $[0, 1]$ with a σ-algebra which contains all sets of the form $\{r\}$ as events. What can you say about the probabilities of these sets?

Optional Exercise 11. Assume you are given the sample set $[0, 1]$ and you know that every interval in $[0, 1]$ is an element of the σ-algebra $\mathcal{E}$. Further assume that you are being given a probability distribution on $\mathcal{E}$ which maps every interval $[r, r']$ in $[0, 1]$ to $r' - r$. Convince yourself that these data satisfy the conditions for a probability space. What do you think should be the probability of the interval (r, r')?

Example 4.22. We continue Example 4.18.

- $S = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ and
- $\mathcal{E} = \mathcal{P}S$

but what is the probability distribution we should use here? Since every subset

numbers. We do not discuss these situations in general in this unit. We say a bit more about how to think of this rule in Definition 34 below.

of S can be written as a disjoint union of sets containing one element each the second condition for probability distributions tells us that it is sufficient to know the probability for each outcome since, for example

$$\begin{aligned} P\{2,3,4,5\} &= P(\{2\}\cup\{3\}\cup\{4\}\cup\{5\}) \\ &= P\{2\}+P\{3\}+P\{4\}+P\{5\}. \end{aligned}$$

This still leaves us with the question of what $P\{2\}$, $P\{3\}$, and so on, should be. If we look at our sample space more closely we find that it in itself can be viewed as a collection of simpler events.

If we look at the outcome 'the sum of the eyes shown by the two dice is 4' then we see that this is a complex event: Assume we have a red die and a blue die, then the following combinations will give the sum of four (giving the red die followed by the blue one):

⚂ ⚀ ⚁ ⚁ ⚀ ⚂.

So we might instead decide that our sample space should look different to make the outcomes as simple as possible to make it easier to determine their probabilities.

IF we record the result of throwing the two dice simultaneously as a pair

$$(i,j),$$

where the first component i tells us the value of the red, and j the value of the blue die. Then our new sample space becomes

$$\{(i,j) \mid 1 \le i,j \le 6\}.$$

If we assume that our two dice are both 'fair', that is, every number appears with equal probability then the event of throwing, say, a three with the red die will be $1/6$, as will be the probability for all the other possible outcomes from 1 to 6. The same is true for the blue die. If we now assume that throwing the red die has no effect on the blue die[12] then the probability of each possible outcome[13]

$$(i,j) \qquad \text{is} \qquad \frac{1}{6}\cdot\frac{1}{6}=\frac{1}{36}.$$

The outcomes in our previous sample space are now *events in the new space*, and the probability that the sum thrown is 4, for example, (the old event $\{4\}$) is given by the new event

$$\{(1,3),(2,2),(3,1)\},$$

and its probability is the sum of the probabilities for each singleton, that is

$$\begin{aligned} P\{(1,3),(2,2),(3,1)\} &= P\{(1,3)\}+P\{(2,2)\}+P\{(3,1)\} \\ &= \frac{1}{36}+\frac{1}{36}+\frac{1}{36} \\ &= \frac{3}{36} \\ &= \frac{1}{12}. \end{aligned}$$

For completeness' sake we give a full description of both probability spaces. Because the set of events is the powerset of the sample set it is sufficient to give the probability of each outcome. We begin by describing the second probability space in the somewhat boring table below, where the probability for the outcome (i, j) is the entry in the row labelled i and the column labelled j.

$i \backslash j$	1	2	3	4	5	6
1	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$
2	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$
3	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$
4	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$
5	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$
6	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$

The outcome k from the original space can be thought of as an event in the new space, namely that of

$$\{(i, j) \in \{1, 2, 3, 4, 5, 6\}^2 \mid i + j = k\},$$

and the probability of outcome k in the original space is equal to the probability of the corresponding event in the new space.

Below we give a table that translates the outcomes from our first sample space to events for the second sample space.

2	3	4	5	6	7	8	9	10	11	12
(1,1)	(1,2) (2,1)	(1,3) (2,2) (3,1)	(1,4) (2,3) (3,2) (4,1)	(1,5) (2,4) (3,3) (4,2) (5,1)	(1,6) (2,5) (3,4) (4,3) (5,2) (6,1)	(2,6) (3,5) (4,4) (5,3) (6,2)	(3,6) (4,5) (5,4) (6,3)	(4,6) (5,5)	(5,6)	(6,6)

Hence the original probability space has a probability distribution determined by the following table:

2	3	4	5	6	7	8	9	10	11	12
$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

There is a third sample space one could use here: As outcomes use an ordered list $[i, j]$ of numbers, to mean 'the die with the lower number shows i and the die with the higher number shows j'. The whole sample space is then

$$\{[i, j] \mid i, j \in \{1, 2, 3, 4, 5, 6\},\ i \leq j\},$$

and we give the probabilities for those outcomes below. We give the lower number in the set to determine the row and the higher number for the column.

$i\backslash j$	1	2	3	4	5	6
1	$1/36$	$2/36$	$2/36$	$2/36$	$2/36$	$2/36$
2		$1/36$	$2/36$	$2/36$	$2/36$	$2/36$
3			$1/36$	$2/36$	$2/36$	$2/36$
4				$1/36$	$2/36$	$2/36$
5					$1/36$	$2/36$
6						$1/36$

In summary, we have given here three probability spaces that describe the given situation, with different underlying sample spaces.

We learn from this example that there may be more than one suitable sample space, and that by making the possible outcomes as simple as possible we may find their probabilities easier to determine. If the sample space is finite then calculating the probability of any event amounts to adding up the probabilities for the individual outcomes.

Exercise 87. Assume you have two dice, one red, one blue, that show a number from 1 to 3 with equal probability. You wish to calculate the probabilities for the numbers that can occur when deducting the number shown by the blue die from that of the red one. For example, if the red die shows 2 and the blue die shows 3, the number to be calculated is -1.

Give two probability spaces that describe this situation and describe how to calculate the probabilities asked for in each case. *Hint: If you are finding this difficult then read on to the next section which contains more worked examples.*

So picking a suitable sample space for the problem that one tries to solve is important. It's not unusual to have a number of candidates, but some of them will be easier to describe correctly than others.

4.2.3 Discrete probability distributions

The above example suggests the idea of the following result. It tells you that if you have a finite sample space then describing a probability space for it can be quite easy.

Proposition 4.2

Let S be a finite set.

(i) If for each $s \in S$ we have the probability $ps \in [0, 1]$ that s occurs, and the sum of these probabilities is 1, then a probability space is given by

- the sample space is S,
- the set of events is the power set of S, $\mathcal{P}S$,
- the probability distribution P is given by

$$\{s_1,\, s_2,\, \dots,\, s_n\} \longmapsto ps_1 + ps_2 + \cdots ps_n,$$

[12] This property is known as *independence*, see Definition 34.

[13] Compare Example 4.77.

where $n \in \mathbb{N}$ and $s_1,\, s_2,\, \dots s_n \in S$, which means that for for every subset E of S, the probability of E is given by

$$PE = \sum_{s \in E} ps.$$

Moreover this is the only probability space where

- all sets of the form $\{s\}$ are events and
- the probability of the event $\{s\}$ occurring is ps.

(ii) If $(S, \mathcal{E}, P)$ is a probability space with the property that for $s \in S$, $\{s\} \in \mathcal{E}$ then

- $\mathcal{E} = \mathcal{P}S$ and
- we may read off the probability ps that any given outcome s occurs by considering $P\{s\}$.

Proof. (i) We have already stated that the powerset of any set is a σ-algebra, so it is sufficient to check that the probability distribution we selected satisfies the required properties.

- We note that the way we have defined the probability distribution, the probability of S is the sum of the probabilities for the outcomes, and the assumption explicitly stated is that this adds to 1, so $P(S) = 1$.
- If we have pairwise disjoint events E_i for $i \in \mathbb{N}$ then the probability of

  $$\bigcup_{i \in \mathbb{N}} E_i$$

  is the sum of all the probabilities of elements in this set. But if the E_i are pairwise disjoint then each element of $\bigcup_{i \in \mathbb{N}} E_i$ occurs in exactly one of the E_i, and so

  $$\begin{aligned} P(\textstyle\bigcup_{i \in \mathbb{N}} E_i) &= \sum_{s \in \bigcup_{i \in \mathbb{N}} E_i} Ps && \text{def } P \\ &= \sum_{i \in I} \sum_{s \in E_i} Ps && E_i \text{ pairwise disjoint} \\ &= \sum_{i \in \mathbb{N}} PE_i && \text{def } P.. \end{aligned}$$

(ii) The second statement really has only one property that we need to prove, namely that $\mathcal{P}S$ is the set of events for the given space.

But if S is finite, and all sets of the form $\{s\}$ are events, then for an arbitrary subset S' of S we can list the elements, for example

$$S' = \{s_1,\, s_2,\, \dots s_n\},$$

and by setting

$$E_i = \begin{cases} \{s_i\} & \text{for } 1 \le i \le n \\ \emptyset & \text{else} \end{cases}$$

we have events E_i for $i \in \mathbb{N}$ with the property that

$$S' = \bigcup_{i\in\mathbb{N}} E_i,$$

and since $\mathcal{E}$ is a σ-algebra we know that $S' \in \mathcal{E}$. Hence every subset of S is an event, and so $\mathcal{E} = \mathcal{P}S$.

Tip

This proposition says that in order to describe a probability space with a *finite* sample space S all we have to do is to

- describe the sample space S;
- say the σ-algebra $\mathcal{E}$ is $\mathcal{P}S$;
- give the probability for each outcome from S and state that the probability for each event is given by the sum of the probabilities of its elements.

Example 4.23. Throwing a single die can be described by the probability space given by

- $S = \{1, 2, 3, 4, 5, 6\}$;
- $\mathcal{E} = \mathcal{P}S$;
- the probability distribution assigns the probability of $1/6$ to each outcome; the probability of an event is given by the sum of the probabilities of its elements.

In practice we often leave out the last statement, or shorten it.

Example 4.24. Tossing a coin can be modelled by a probability space with

- sample set$S = \{H, T\}$,
- set of events $\mathcal{E} = \mathcal{P}S$ and
- a probability distribution determined by the fact that each outcome occurs with probability $1/2$.

Example 4.25. The probability space underlying Exercise 79 has as its underlying sample space the set

$$\{\{Q\heartsuit, A\heartsuit\}, \{Q\heartsuit, Q\spadesuit\}, \{Q\heartsuit, A\spadesuit\},$$
$$\{A\heartsuit, Q\spadesuit\}, \{A\heartsuit, A\spadesuit\}, \{Q\spadesuit, A\spadesuit\}\},$$

and the probability for each outcome is $1/6$. The probability distribution is derived from this in the usual way.

Proposition 4.3

If we have a sample set

$$S = \{s_i \mid i \in \mathbb{N}\},$$

then a probability space is uniquely determined by assigning to each element s of S a probability ps in $[0, 1]$ such that

$$\sum_{i \in \mathbb{N}} ps_i = 1.$$

Optional Exercise 12. Can you take the proof of Proposition 4.2 and turn it into one for Proposition 4.3?

Example 4.26. Assume we toss a coin until we see head for the first time, compare Exercise 85. To describe a probability space for this situation we pick the sample set

$$\{1, 2, 3, \ldots\} = \mathbb{N} \setminus \{0\},$$

which tells us how many times we tossed the coin until heads appeared. Again we may choose the powerset of this set as the set of events.

The probability for each of these outcomes is given in the following table.

1	2	3	4	5	6	…
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	…

which means that the probability of the outcome i is

$$pi = \frac{1}{2^i}.$$

It is the case (but a proof is beyond the scope of this unit) that

$$\sum_{i \in \mathbb{N}} pi = \sum_{i \in \mathbb{N}} \frac{1}{2^i} = 1,$$

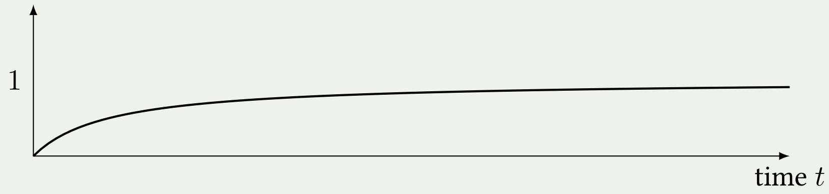
so this distribution satisfies the requirements from Proposition 85.

CExercise 88. Find probability spaces to describe the various situations from Exercise 77 (a)–(l) and Exercises 83 to 85. Note that your space should describe the *general situation* from the question, and the specific probabilities you were asked to calculate in those exercises do not matter now. It is fine to describe these in text where you find it difficult to use set and function notation.

4.2.4 Continuous probability distributions

Sometimes it is more appropriate to have a continuous description of a problem. This is often the case when we are plotting events over time. Note that we can only talk about 'continuous behaviour' if we may use a sub-interval of the real numbers to describe the outcome of our probability space. See Definition 38 for a formal definition of what we mean by the discrete versus continuous case here.

Example 4.27. The following curve of a function f, might describe[14] the probability that a piece of hardware will have failed by time t.

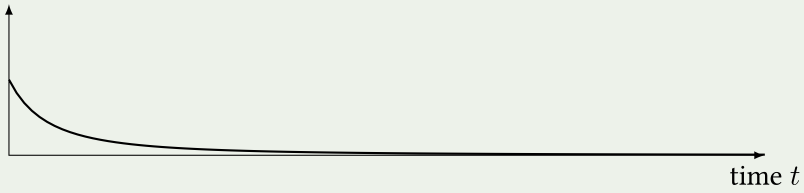


As time progresses the probability of the component having failed approaches 1. But how do we turn this kind of function into a probability space?

We need to identify a set of events, and we need to be able to derive the probability of that event. What we know is how to read off a probability the our device will have failed in the time interval from 0 to t: That probability is given by ft.

This is, in fact, known as a *cumulative probability distribution*: As time progresses the probability becomes higher and higher because the time interval covered becomes bigger and bigger.

In order to give a probability space we need the *probability density function*, which tells us the probability of the device failing at time t. For the function above this is given by the function g plotted below.



The relationship between the two functions is that for all t in $\mathbb{R}^+$ we have

$$ft = \int_{x=0}^{t} gxdx.$$

The reason for this becomes clear in Section 4.4.6.

It is possible to give a probability space based on the real numbers, but the precise description is quite complicated. For completeness' sake we note the following two facts.

Fact 11

There is a σ-algebra $\mathcal{E}_B$ on the set of real numbers $\mathbb{R}$ known as the *Borel σ-algebra* with the property that

- all intervals $[r, r']$, where $r, r' \in \mathbb{R}$, are elements of $\mathcal{E}_B$.

Let I be any interval in $\mathbb{R}$. Then we can restrict the Borel σ-algebra to this interval to obtain another σ-algebra $\mathcal{E}_B^I$ by setting

$$\mathcal{E}_B^I = \{E \cap I \mid E \in \mathcal{E}_B\}.$$

[14] For an actual piece of hardware one would prefer it if the probability were to rise more slowly at first!

Fact 12

Let $[s, s']$ be an interval in $\mathbb{R}$ with $s < s'$. There is a probability distribution[15] P to give a probability space $([s, s'], \mathcal{E}_B^{[s,s']}, P)$ with the property that for any interval $[r, r']$ in $[s, s']$ we have

$$P[r, r'] = \frac{r' - r}{s' - s}.$$

The probability space for Example 4.27 is then given by $(\mathbb{R}^+, \mathcal{E}_B^+, P_B)$ where

- $\mathcal{E}_B^{\mathbb{R}^+}$ is the restriction of the Borel σ-algebra from Fact 11 and
- the probability distribution is determined by the fact that it satisfies , for all $r \leq r'$ in $\mathbb{R}^+$,
$$P[r, r'] = \int_r^{r'} gxdx.$$

Whenever you are asked to define a continuous probability space you may assume that

- you may use the Borel σ-algebra adjusted as in the above example and
- we can calculate a probability distribution for this σ-algebra from any probability density function (see the Definition below).

So it is sufficient for you to give a probability density function in this case.

Definition 33: probability density function

Let I be a sub-interval of the real numbers. A **probability density function for** I is given by a function

$$g \colon I \longrightarrow \mathbb{R}^+$$

with the property that

$$\int_I gxdx = 1,$$

and such that

$$\int_r^{r'} gxdx$$

exists for all $r \leq r'$ in I.

Tip

It might seem odd that intervals play a role in calculating probabilities. Recall[16] that the integral from some t to some t' over a function g is the area under the curve given by g from t to t'. This is a generalization of adding up all the probabilities of outcomes, but this requires too much advanced maths to explain.[17] So my tip is to just treat the integrals as given, and not worry too much about why that makes sense.

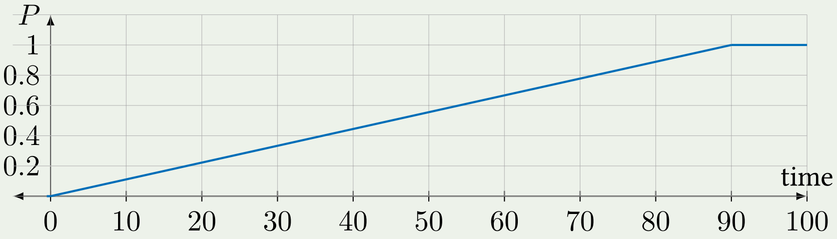
[15] This is based on the *Borel measure*.

[16] If you have not covered integrals in school then the following fact should get you through most of the two extended exercises that require integrals.

[17] But see Example 4.81!

Example 4.28. Assume you have travelled to Yellowstone National Park and want to see the famous geyser 'Old Regular' erupt. You know[18] that it does so every ninety minutes. You are pressed for time, and when you arrive you know you can only stay for twenty minutes. What is the probability that you will see the geyser erupt in that time?

We can describe this situation using the fact that we know that as time goes from 0 to 90 minutes the probability of seeing the geyser erupt rises steadily towards 1. The set of events is the Borel σ-algebra restricted to the interval from 0 to 90. The cumulative distribution function f looks as follows.



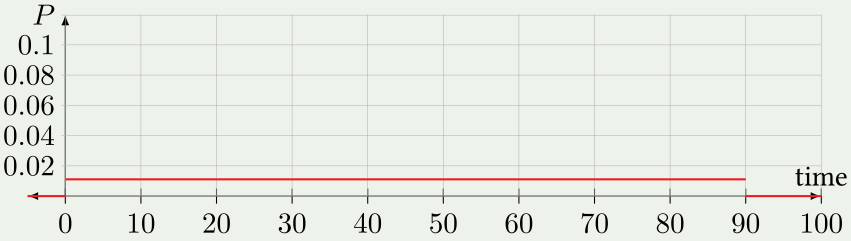
We are looking for a function g (the probability density function for the cumulative mass function f in the graph above) on the interval from 0 to 90 minutes with property that for all times t with $0 \leq t \leq 90$ we have

$$ft = \int_0^t gxdx.$$

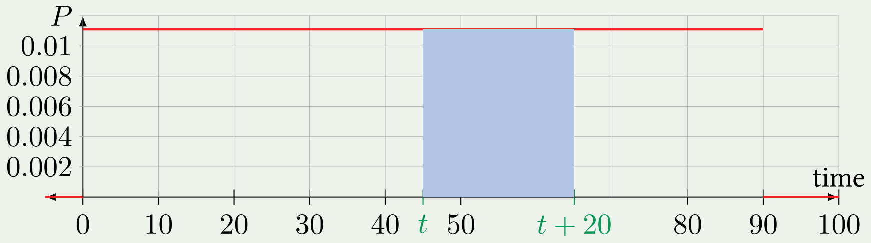
Solving this tells us that g is *constant*, and it only remains to calculate the constant which comes from the constraint that the integral over g from 0 to 90 must be 1. Assume that $gx = c$ for all $0 \leq x \leq 90$. Then we need

$$1 = \int_0^{90} cdx = c \cdot 90,$$

so we must have $c = 1/90$.



We can see that it does not matter when exactly you arrive - the distribution is uniform. So if you arrive at time t and stay for 20 minutes then the probability that you will see the geyser erupt is given by the integral from t to $t + 20$ over the function g, which is given by the shaded area.



This means that the desired probability, for the case $t = 45$, is

$$\int_{45}^{65} \frac{1}{90} dx = \frac{1}{90}(65 - 45) = \frac{20}{90} = 0.\overline{2},$$

so the probability is just over 20%.

Tip

Whenever you have to describe a probability space whose set of outcomes is an interval in $\mathbb{R}$ you should choose the Borel σ-algebra restricted to that interval as your set of events.

In the unit on data science COMP13212 you will see quite a few plots of either a probability density function or for a cumulative mass function (called *cumulative distribution function* there), for example in the lecture on hypotheses and how to test them.

EExercise 89. Describe probability density functions for the following situations:

(a) It is known that the probability of a component having failed rises from 0 to 1 over the time interval from 0 to 1 unit of time at a constant rate.

(b) A bacterium lives for two hours. It is known that its chance of dying in any 10 minute interval during those two hours is the same. What do you think the probability density function should be?

(c) Assume you have an animal which lives in a one dimensional space described by the real line $\mathbb{R}$. Assume that its den is at 0, and that the probability density function has the value r at that point and that it falls at a constant rate and reaches 0 when the animal is one unit away from its den. Give the probability density function for this situation. What does the corresponding cumulative probability distribution look like in this case? (If you think an animal liven in a one dimensional space is a bit limiting you can instead think of this as expressing the animal's east/west (or north/south) distance from its den in a space of two dimensions.)

(d) Try to extend the previous part to an animal that lives in a two dimensional space described by the real plane, $\mathbb{R} \times \mathbb{R}$.

[18] The most famous geyser that actually exists there, known as Old Faithful, does not erupt as regularly as my imaginary example.

4.2.5 Consequences from Kolmogorov's axioms

The axioms from Definition 32 have a number of consequences that are useful to know about.

We look at them one by one here and summarize them in a table at the end of the section.

The empty set

- The empty set $\emptyset$ is an event: Definition 32 says that if E is an event then so is $S \setminus E$. Since S is an event this means that $S \setminus S = \emptyset$ is an event.

- Now that we know that $\emptyset$ is an event we may calculate its probability as follows.

$$\begin{aligned} 1 &= PS && \text{Definition 32} \\ &= P(S \cup \emptyset) && S \cup \emptyset = S \\ &= PS + P\emptyset && S \text{ and } \emptyset disjoint, Def\ 32 \\ &= 1 + P\emptyset && PS = 1 \end{aligned}$$

  and so

$$P\emptyset = 1 - 1 = 0.$$

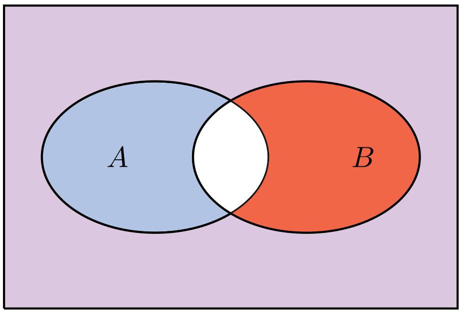
Intersection

If we know that A and B are events, what can we say about $A \cap B$? We note that there is nothing in the axioms that talks about intersections. But it turns out that we can use the axioms to argue that the intersection is an event.

We calculate[19]

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

In the following diagram $(S \setminus A) \cup (S \setminus B)$ is the coloured area, and the white part is its complement, that is the desired set.



$(S \setminus A) \cup (S \setminus B)$

Since the complement of an event is an event we know that $S \setminus A$ and $S \setminus B$ are events, and we have seen that the union of a finite number of events is another event.[20]

In general there is no way of calculating the probability of $A \cap B$ from the probabilities of A and B. When the two events are *independent* then this situation changes, see Definition 34.

We may summarize this as follows:

[19] See Exercise 7.

[20] Note that we can also show that the countable intersection of events is an event by generalizing this idea.

- If A and B are events then so is their intersection $A \cap B$.
- There is no general way of calculating the probability of $A \cap B$ from those of A and B.

Complement and relative complement

We begin by looking at the complement of a set.

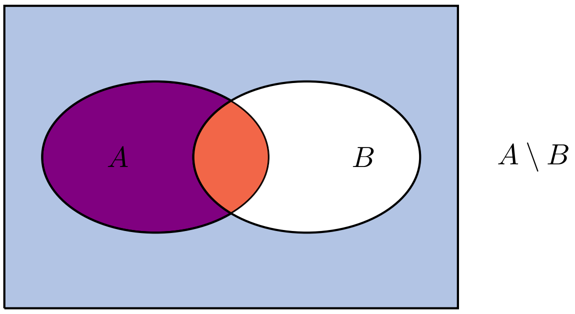
- If A is an event then we know that its complement $S \setminus A$ is also an event.
- We also know that a set and its complement are disjoint sets whose union is S. Hence we know that

$$\begin{aligned} 1 &= PS && \text{Definition 32} \\ &= P(A \cup (S \setminus A)) && S = A \cup (S \setminus A) \\ &= PA + P(S \setminus A) && A,\ S \setminus A \text{disjoint, Def 32} \end{aligned}$$

and so [21]

$$P(S \setminus A) = 1 - PA.$$

More generally, assume we have events A and B. The picture shows A in red and $S \setminus B$ in pale blue, with violet giving the overlap. The set whose probability we wish to compute is that overlap, the darkest set in the following picture.



We would like to argue that $A \setminus B$ is an event. We note that the definition of a σ-algebra tells us that since

$$S \setminus A \qquad\qquad \text{and} \qquad\qquad B$$

are events we may form another event in the form of

$$(S \setminus A) \cup B$$

and so we get an event when forming (compare Exercise 7 for the trick we employ here)

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

After all this preparation we may now split the event A into two disjoint events, namely

$$A = (A \setminus B) \cup (A \cap B),$$

[21] Some people write this as $P(\neg A) = 1 - PA$ or $P(A^C) = 1 - PA$, but we do not use that notation here.

and so (compare Exercise 86)

$$\begin{aligned} PA &= P((A\setminus B)\cap(A\cap B)) && A=(A\setminus B)\cup(A\cap B)\\ &= P(A\setminus B)+P(A\cap B) && (A\setminus B),\ A\cap B \text{ disjoint, Def 32}, \end{aligned}$$

which gives us

$$P(A\setminus B)=PA-P(A\cap B).$$

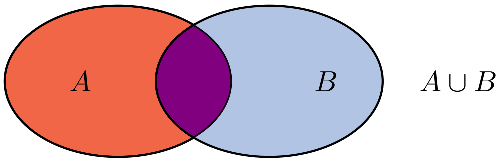
We may summarize this by saying the following.

- If A and B are events then so is $A\setminus B$.
- We have $P(A\setminus B)=PA-P(A\cap B)$.

Union

If we want to calculate the probability of the union of two events then in order to apply Kolmogorov's axiom we must write it as the union of disjoint events.

The Venn diagram for two non-disjoint set looks like this:



We can see that if we want to write $A\cup B$ as a disjoint union we have to pick for example the red and violet regions, which make up A, and the blue region, which is $B\setminus A$, and write

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

With the result for the relative complement we get

$$\begin{aligned} P(A\cup B) &= P(A\cup(B\setminus A)) && A\cup B=A\cup(B\setminus A)\\ &= PA+P(B\setminus A) && A,\ B\setminus A \text{ disjoint, Def 32}\\ &= PA+PB-P(A\cap B) && P(B\setminus A)=PB-P(A\cap B). \end{aligned}$$

In summary we can say that

- if A and B are events then so is $A\cup B$ and
- we have $P(A\cup B)=PA+PB-P(A\cap B)$.

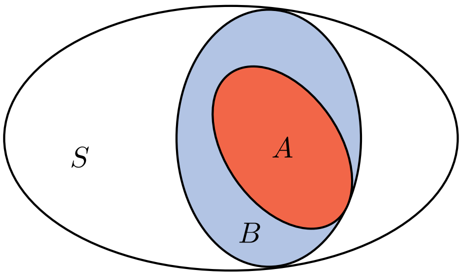
Note that if A and B do not overlap then

$$A\cap B=\emptyset \qquad\text{and}\qquad P(A\cup B)=PA+PB$$

as expected.

Order preservation

Assume we have two events A and B with the property that A is a subset of B.



What can we say about the probabilities of A and B? Certainly we can see that B is the disjoint union of A and $B \setminus A$, and so

$$PB = PA + P(B \setminus A).$$

Since the probability of $B \setminus A$ is greater than or equal to 0 we must have

$$PA \leq PB.$$

Summary

We give all the rules derived above. Let $(S, \mathcal{A}, P)$ be a probability space, and let A and B be events. Then the following rules hold.

$$\begin{aligned} PS &= 1 & P\emptyset &= 0 \\ P(S \setminus A) &= 1 - PA & P(A \setminus B) &= PA - P(A \cap B) \\ P(A \cup B) &= PA + PB - P(A \cap B) \\ A \subseteq B &\text{ implies } PA \leq PB. \end{aligned}$$

It may be worth pointing out that these conditions hold for all probability spaces, in particular they also hold for the case where we are given a probability density function. The first two conditions are trivially true, and the others are standard properties of integrals.

Optional Exercise 13. Convince yourself that the various equalities hold if the probability distribution is given by a probability density function. You may want to draw some pictures for this purpose.

4.2.6 Kolmogorov's axioms revisited

How should we think of the Kolmogorov axioms? The definition of a σ-algebra is something of a formality that ensures that the sets for which we have a probability (namely the *events*) allow us to carry out operations on them.

We may think of the probability distribution as a way of splitting the probability of 1 (which applies to the whole set S) into parts (namely those subsets of S which are events). If S is finite then we only have to know how the probability of 1 is split among the elements of S, and then we can assign a probability to each subset of S by adding up all the probabilities of its elements.

This becomes significantly more complicated if the set is infinite.

Proposition 4.4

If S is an infinite set then there is no probability distribution which assigns the same probability to each event $\{s\}$ of S.

The simplest infinite set we have met is the set of natural numbers $\mathbb{N}$. If we had a probability distribution on $\mathbb{N}$ which assigned a fixed probability $r \in [0, 1]$ to each element then it would have to be the case that the sum of all these probabilities is 1, that is

$$\sum_{i \in \mathbb{N}} r = \sum_{i \in \mathbb{N}} P\{i\} = 1$$

and there is no real number r with that property.

Note that the probability space defined in Fact 12 is uniform in that it assigns the same probability to intervals of the same length. So it is possible to distribute probability uniformly in two cases:

- the sample set S is finite, in which case we may assign the same probability, $1/|S|$, to each outcome or
- the sample set S is an interval in $\mathbb{R}$, in which case the probability of any one outcome is 0, but intervals can have non-0 probabilities which are determined by their length.

However, there is no way of taking *all* the subsets of $\mathbb{R}$ (or any interval I), and turning that into a probability space.

Proposition 4.5

Let I be an interval on the real line. There is no probability distribution P with the property that, $(I, \mathcal{P}I, P)$ is a probability space which maps intervals of the same size to the same probability.

This proposition explains why we cannot have a simpler definition of probability space, where the set of events is always the powerset of the sample space.

4.3 Conditional probabilities and independence

One of the questions that appears frequently in the context of probability theory is that of how information can be used. In other words, can we say something more specific if we already know something about the situation at hand. This section is concerned with describing how we may use the axioms of probability to make this work.

4.3.1 Independence of events

Kolmogorov's axioms are not strong enough to allow us to calculate the probability of

$$A \cap B$$

if we know the probabilities of A and B. This section sheds some light on the question why there cannot be a general formula that does this.

When we throw two dice, one after the other, or when we throw a coin repeatedly, we are used to a convenient way of calculating the corresponding probabilities for the outcomes.

Example 4.29. Assume we record the outcome of a coin toss with H for head and T for tails. We assume the coin is fair and so the probability for each is $1/2$. If we toss the coin twice then the possible outcomes are HH, HT, TH and TT and the probability for each is $1/4$. We may calculate the probability HT, that is the first coin toss $C1$ coming up H, and the second, $C2$, T as follows.

$$P((C1 = H) \cap (C2 = T)) = P(C1 = H) \cdot P(C2 = T) = 1/2 \cdot 1/2 = 1/4$$

But it is not safe to assume that for general events A and B we have that the probability of $A \cap B$ can be calculated by multiplying the probabilities of A and B, see Example 4.30.

Definition 34: independent events

Given a probability space $(S, \mathcal{E}, P)$ we say that two events A and B are **independent** if and only if

$$P(A \cap B) = PA \cdot PB.$$

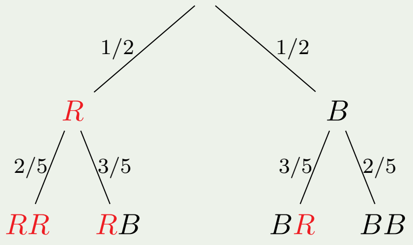
What we mean by 'independent' here is that neither event has an effect on the other. We assume that when we throw a coin multiple times then the outcome of one toss has no effect on the outcome of the next, and similar for dice. We look at this issue again in Section 4.4.5 when we have random processes which are more easily described. In particular we talk about independence for processes with a continuous probability distribution.

Example 4.30. Let us look at a situation where we have events which are not independent. In Example 4.13 we discussed pulling socks from a drawer. We assume that we have a drawer with three red and three black socks from which we draw one sock at a time without looking inside. If you pick a red sock on the first draw, then the probability of finding a red sock on the second draw is changed.

The probability of drawing a red sock on the first attempt is

$$P(D_1 = R) = 1/2,$$

but what about the probability of drawing a red sock on the second attempt? Again it is best if we look at the tree that shows us how the draw progresses.



We can see that the probability of drawing a red sock on the second attempt is

$$P(D_2 = R) = \frac{1}{2} \cdot \frac{2}{5} + \frac{1}{2} \cdot \frac{3}{5} = \frac{2+3}{10} = \frac{1}{2}.$$

But we can also see from the tree that

$$P((D_1 = R) \cap (D_2 R)) = \frac{1}{2} \cdot \frac{2}{5} = \frac{1}{5},$$

which is not equal to

$$P(D_1 = R) \cdot P(D_2 = R) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4},$$

so (very much expectedly) the two events are not independent.

Example 4.31. A more serious example is as follows. SIDS, or 'Sudden Infant Death Syndrome' refers to what is also known as 'cot death'—young children die for no reason that can be ascertained. In 1999 an 'expert witness' told the court that the approximate probability of a child of an affluent family dying that way is one in 8500. Since two children in the same family had died this way, the expert argued, the probability was one in 73 million that this would occur, and a jury convicted a young woman called Sally Clark of the murder of her two sons, based largely on this assessment.

The conviction was originally upheld on appeal, but overturned on a second appeal a few years later. While Clark was released after three years in prison she later suffered from depression and died from alcohol poisoning a few years after that.

What was wrong with the expert's opinion? The number of 1 in 73 million came from multiplying 8500 with itself (although 72 million would have been more accurate), that is, arguing that if the probability of one child dying in this way is

$$\frac{1}{8500},$$

then the probability of two children dying in this way is

$$\frac{1}{8500} \cdot \frac{1}{8500}.$$

But we may only multiply the two probabilities if the two events are independent, that is, if the death of a second child cannot possibly be related to the death of the first one. This explicitly assumes that there is no genetic or environmental component to SIDS, or that there may not be other circumstances which makes a second death in the same family more likely. Since then data have been studied that show that the assumption of the independence of two occurrences appears to be wrong.

While there were other issues with the original conviction it is shocking that such evidence could be given by a medical expert without anybody realizing there was a fallacy involved. I hope that this example illustrates why it is important to be clear of the assumptions one makes, and to check whether these can be justified.

Note that if we know that two events are independent then we may derive from that the independence of other events.

Example 4.32. If A and B are independent events in a probability space with sample set S then A and $S \setminus B$ are also independent.

To prove this we have work out the probability of the intersection of the two events. We calculate

$$\begin{aligned}
P(A\cap(S\setminus B)) &= P(A\setminus B) && A\setminus B = A\cap(S\setminus B)\\
&= PA - P(A\cap B) && \text{Summary of Section 4.2.5}\\
&= PA - PA\cdot PB && A \text{ and } B \text{ independent}\\
&= PA(1-PB) && \text{arithmetic}\\
&= PA\cdot P(S\setminus B) && P(S\setminus B) = 1-PB
\end{aligned}$$

which establishes that the two given events are indeed independent.

Exercise 90. Show that if A and B are independent then so are $S\setminus A$ and $S\setminus B$.

A common fallacy is to assume that two events being independent has something to do with them being disjoint, that is, there not being an outcome that belongs to both. The following exercise discusses why this is far from the truth.

Exercise 91. Assume that you have a probability space with two events A and B such that A and B are disjoint, that is $A\cap B=\emptyset$. What can you say about PA, PB and $P(A\cap B)$ under the circumstances? What can you say if you are told that A and B are independent?

Give a sufficient and necessary condition that two disjoint events are independent.

4.3.2 Conditional information

For example, if I have to guess the colour of somebody's eyes, but I already know something about the colour of their hair then I can use that information to guide my choice.

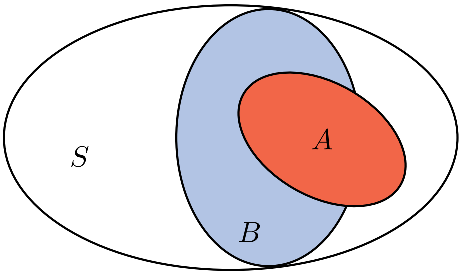
Example 4.33. Let us assume we have a particular part of the population where 56% have dark hair and brown eyes, 14% have dark hair and blue eyes, 3% have fair hair and brown eyes and 27% have fair hair and blue eyes.

If I know a person has been randomly picked from the population, and I have to guess the colour of their eyes, what should I say to have the best chance of being right?

	brown eyes	blue eyes
dark haired	56%	14%
fair haired	3%	27%

We can see from the numbers given that we are better off guessing brown (lacking additional information). But what if we can see that the person in question has fair hair? In that case we are better off guessing blue. What is the appropriate way of expressing these probabilities? This example is continued below.

What we are doing here can be pictured by assuming that in the sample space S we have two sets, A and B.



We are interested in the probability of A (say blue eye colour) already knowing that B holds (say fair hair). In the picture above this means the probability that we are in the red set A, provided we already know that we are in the blue set B.

What we are doing effectively is to change the sample space S to B, and we want to know the probability of $A \cap B$.

Proposition 4.6

If $(S, \mathcal{E}, P)$ is a probability space and A an event with non-zero probability then a probability space is given by the following data:

- sample set A,
- set of events
$$\{A \cap E \mid E \in \mathcal{E}\},$$
- probability distribution P' defined by
$$A \cap E \longmapsto \frac{P(A \cap E)}{PA} \quad .$$

We can think of the new space as a restriction of the old space with sample set S to a new space with sample set A, where we have redistributed the probability entirely to the set A, and adjusted all the other probabilities accordingly.

Optional Exercise 14. Define a probability space that is an alternative to the one given in Proposition 4.6. Again assume that you have a probability space $(S, \mathcal{E}, P)$ and a subset A of S with non-zero probability. Use

- sample set S,
- set of events: $\mathcal{E}$,
- a probability density function that assigns to every event of the form $A \cap E$, where $E \in \mathcal{E}$, the same probability as the function given in said proposition.

Optional Exercise 15. Show that the new set of events in Proposition 4.6 is a σ-algebra.

Exercise 92. For the probability distribution P' from Proposition 4.6 carry out the following:

(a) Calculate $P'A$.

(b) For $B \subseteq A$ calculate $P'B$.

(c) Show that P' is a probability distribution.

Definition 35: conditional probability

Let $(S, \mathcal{E}, P)$ be a probability space, and let A and B be events, where B has a non-zero probability. We say that the **conditional probability of A given B** is given as

$$P(A \mid B) = \frac{P(A \cap B)}{PB}.$$

It is the probability of the event $A \cap B$ in the probability space based on the restricted sample set B given by Proposition 4.6.

Note that if $PB = 0$ then $P(A \mid B)$ is not defined, no matter what A is.

Example 4.34. Continuing Example 4.33 we can see that the probability that a randomly selected person has blue eyes, given that he or she has fair hair, is

$$P(\text{blue eyes} \mid \text{fair hair}) = \frac{P(\text{blue eyes and fair hair})}{P(\text{fair hair})} = \frac{.27}{.3} = .9.$$

In other words, if I am presented with a randomly selected person whose hair I happen to know to be fair then by guessing their eye colour is blue I have a 90% chance of being correct.

On the other hand, if I can see the person has dark hair, then the chance that they have brown eyes is

$$P(\text{brown eyes} \mid \text{dark hair}) = \frac{P(\text{brown eyes and dark hair})}{P(\text{dark hair})} = \frac{.56}{.7} = .8.$$

Hence we can use conditional probabilities to take into account additional information we have been given before making a decision.

Example 4.35. If we revisit Example 4.14 we can see that what we calculated was the probability that we have the bag GG given that we have seen a gold coin. According to the above

$$P(GG \mid G) = \frac{P(GG \cap G)}{PG} = \frac{\frac{1}{3}}{\frac{1}{2}} = \frac{2}{3},$$

just as we concluded on our first encounter of this example.

Example 4.36. In the Monty Hall problem, Example 4.16, we can think of being shown that there is a booby prize behind one of the doors as adding information. Effectively the show master is asking us: What is the probability that you picked the correct door, knowing that the door I've just shown you is not the correct door? In other words we are interested in the event that

- the prize is behind the door the player chose under the condition that
- we were shown the booby prize behind another door.

The probability that the player has chosen the correct door on the first move is $1/3$. The probability that the player chose the incorrect door on the first move is $2/3$, and that is the probability that the prize is hidden behind the door to which the player can switch.

EExercise 93. Assume that $(S, \mathcal{E}, P)$ is a probability space with events A, A' and B. Further assume that $PB \neq 0$.

(a) If you know that $PA \leq PA'$ what can you say about $P(A \mid B)$ and $P(A' \mid B)$?

(b) If you know that $A \cap B = \emptyset$ what can you say about $P(A \mid B)$?

(c) If you know that A and B are independent what can you say about $(A \mid B)$?

(d) If you know that $A \subseteq B$ what can you say about $P(A \mid B)$?

(e) What is $P(B \mid B)$?

(f) How do $P(A \cap B)$ and $P(A \mid B)$ compare?

In each case justify your answer.

Exercise 94. Assume you know a family with two children.

(a) If you know the family has at least one girl what is the chance that both children are girls?

(b) If we know that the family's firstborn was a girl, what is the probability that both children are girls?

You may assume that every birth yields a girl and a boy with equal probability.

CExercise 95. Go back to the game described in Exercise 79 whose probability space is given in Example 4.25. For this exercise the game remains the same: I draw two cards from the six available ones, and then I randomly drop one of them. Below you are asked to answer a number of questions about the situation either directly after the draw, or after I have dropped a card.

(a) What is the probability that I have at least one ace after the draw?

(b) What is the probability that I have two aces after the draw?

(c) What is the probability that the dropped card is an ace?

(d) What is the probability that I have the ace of spades $A\spadesuit$ given that I dropped a queen?

(e) What is the probability that at the end of the game I have the ace of spades $A\spadesuit$ given that I dropped an ace?

(f) What is the probability that the dropped card was a queen given that at the end of the game I have the ace of spades $A\spadesuit$?

(g) In the original exercise you were asked to calculate the probability that the dropped card was an ace given that at the end the end of the game I have the ace of spades $A\spadesuit$. Express this using conditional probabilities and recalculate the answer.

(h) Consider the following narrative: After the deal you ask me whether one of my cards is an ace, and I answer in the affirmative. You then ask me to drop a card, and to make sure I keep an ace. What is the probability that the dropped card is an ace?

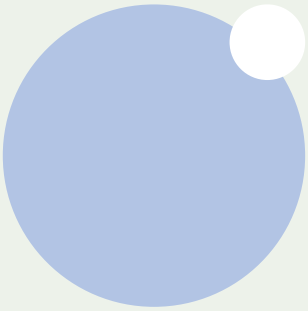
Exercise 96. Assume you have a probability space $(S, \mathcal{E}, P)$, A and B are events, and you know the following:

- $PA > 0$, $PB > 0$, $P(A \cap B) > 0$;
- $P(A \mid B) = P(B \mid A)$ and
- $P(A \cup B) = 1$.

Show that $PA > 1/2$. Why is the condition $P(A \cap B) > 0$ needed?

Note that it does make sense to apply the same ideas in the case where we have a probability density function.

Example 4.37. Assume that you have a probability density function describing an animal's location. Further assume that the space in question is centred on the animal's den. Let's assume the animal is a fox, and that we know that its presence is influenced by the presence of another animal, say a lynx. To make the situation simpler let's say that the fox avoids a circle around the lynx.



If we assume the fox avoids the lynx completely then the fox being in the white area of its range, given there is a lynx at the centre of the white circle, has a probability of 0. But that means the 'mass' of probability that resided in the white area has to go somewhere else (since the overall probability that the fox is somewhere in the area has to be equal to 1)! In the above example we haven't got enough information to decide where it goes.

Further if the situation is more interesting, and the lynx only inhibits, but does not prevent, the foxes presence, the analysis is more complicated. We return to this question in Section 4.4.5 where we restrict how we think of the events that occur, which makes it substantially easier to mathematically describe the situation.

4.3.3 Equalities for conditional probabilities

From the definition of the conditional probability we may derive some useful equalities.

Recall that the probability of an event conditional on another is defined only if the latter has a probability greater than 0, but the following equality is true even if the probability is 0:

$$P(A \mid B) \cdot PB = P(A \cap B),$$

which is also known as the **multiplication law**.

Note that the expression on the left hand side is symmetric in A and B since $A \cap B = B \cap A$ and so we have

$$P(A \mid B) \cdot PB = P(A \cap B) = P(B \mid A) \cdot PA.$$

If we like we can use this equality to determine $P(B \mid A)$ from $P(A \mid B)$, provided that $PA \neq 0$. The equality

$$P(B \mid A) = \frac{P(A \mid B) \cdot PB}{PA},$$

is known as **Bayes's Theorem**. It allows us to compute the probability of B given A, provided we have the probabilities for A given B, A and B.

Example 4.38. Revisiting Example 4.34 we have calculated the probability that a fair-haired person has blue eyes. What about the probability that a blue-eyed person has fair hair? Using Bayes's law we have

$$\begin{aligned} P(\text{fair hair} \mid \text{blue eyes}) &= \frac{P(\text{blue eyes} \mid \text{fair hair}) \cdot P(\text{fair hair})}{P(\text{blue eyes})} \\ &= \frac{.9 \cdot .3}{.41} \\ &\approx 65.9\%. \end{aligned}$$

On the other hand the probability that a brown-eyed person has dark hair is

$$\begin{aligned} P(\text{dark hair} \mid \text{brown eyes}) &= \frac{P(\text{brown eyes} \mid \text{dark hair}) \cdot P(\text{dark hair})}{P(\text{brown eyes})} \\ &= \frac{.8 \cdot .7}{.59} \\ &\approx 95\%. \end{aligned}$$

There are further equalities based around conditional probabilities that can be useful in practice. Sometimes the sample space can be split into disjoint events, where we know something about those.

In particular, given an event B we know that B and $S \setminus B$ cover the whole sample space S. This means we know that (see Exercise 86)

$$A = (A \cap B) \cup (A \cap (S \setminus B)),$$

and this is a disjoint union. By Kolmogorov's axioms given in Definition 32 this implies

$$\begin{aligned} PA &= P((A \cap B) \cup (A \cap (S \setminus B))) \\ &= P(A \cap B) + P(A \cap (S \setminus B)), \end{aligned}$$

and if we use the multiplication law twice, and the properties for probability distributions as needed, then we obtain the following rule.

$$\begin{aligned} PA &= P(A \mid B) \cdot PB + P(A \mid S \setminus B) \cdot P(S \setminus B) \\ &= P(A \mid B) \cdot PB + P(A \mid S \setminus B) \cdot (1 - PB). \qquad (*) \end{aligned}$$

This law is a special case of a more general one discussed below. But even this restricted version is useful, for example, when there is a given property and whether or not that property holds has an influence on whether a second property holds.

Note that if we pick B so that its probability is either 0 or 1 then the law does not help us in calculating the probability of A.

Example 4.39. Assume that motherboards from different suppliers have been stored in such a way that it is no longer possible to tell which motherboard came from which supplier.

Further assume that subsequently it has become clear that those from Supplier 1 ($S1$) have a 5% chance of being faulty, while that chance is 10% for ones from Supplier 2 ($S2$). It is known that 70% of supplies in the warehouse came from Supplier 1, and the remainder from Supplier 2. What is the probability that a randomly chosen motherboard is defective?

The rule $(*)$ from above tells us that

$$\begin{aligned} &P(\text{defect}) \\ &= P(\text{defect} \mid \text{from S1}) \cdot P(\text{from S1}) + P(\text{defect} \mid \text{from S2}) \cdot P(\text{from S2}) \\ &= .05 \cdot .7 + .1 \cdot .3 \\ &= .065. \end{aligned}$$

Example 4.40. The following is an important case that applies to diagnostic testing in those cases where there is some error (certainly medical tests fall into this category).

Assume a test is being carried out whether some test subject suffers from an undesirable condition. From previous experience it is known that

- if the subject suffers from the condition then with a probability of .99 the test will show this correctly and
- if the subject does not have the condition then with a probability of .95 the test will show this correctly.

We assume that for an arbitrary member of the test population the chance of suffering from the condition is $.00001$. If a subject tests positive for the condition, what is the probability that they have the condition?

We would like to calculate

$$P(\text{has condition} \mid \text{test positive}).$$

We do not have this data given, but we do have

$$P(\text{test pos} \mid \text{has cond}) \qquad \text{and} \qquad P(\text{has cond}).$$

If we apply Bayes's theorem we get

$$P(\text{has cond} \mid \text{test pos}) = \frac{P(\text{test pos} \mid \text{has cond}) \cdot P(\text{has cond})}{P(\text{test pos})}.$$

We miss

$$P(\text{test pos}),$$

but we may use rule $(*)$ above to calculate

$$\begin{aligned}
&P(\text{test pos})\\
&= P(\text{test pos} \mid \text{has cond}) \cdot P(\text{has cond})\\
&\qquad + P(\text{test pos} \mid \text{doesn't have cond}) \cdot P(\text{doesn't have cond})\\
&= .99 \cdot .00001 + .05 \cdot .99999\\
&\approx .05
\end{aligned}$$

So we may calculate the desired probability as

$$\begin{aligned}
P(\text{has cond} \mid \text{test pos}) &= \frac{P(\text{test pos} \mid \text{has cond}) \cdot P(\text{has cond})}{P(\text{test pos})}\\
&\approx \frac{.99 \cdot .00001}{.05}\\
&\approx .0002.
\end{aligned}$$

So if we test something, and in the event it tests positive, there's a $.02\%$ chance that the subject is ill, would we think this is a good test?

The issue in this example is the extremely low probability that anybody has the condition at all. If we change the numbers and instead assume that the chance that an arbitrary member of the test population has the condition is .1 then we get

$$\begin{aligned}
&P(\text{test pos})\\
&= P(\text{test pos} \mid \text{has cond}) \cdot P(\text{has cond})\\
&\qquad + P(\text{test pos} \mid \text{doesn't have cond}) \cdot P(\text{doesn't have cond})\\
&= .99 \cdot .1 + .05 \cdot .9\\
&= .144
\end{aligned}$$

and

$$\begin{aligned}
P(\text{has cond} \mid \text{test pos}) &= \frac{P(\text{test pos} \mid \text{has cond}) \cdot P(\text{has cond})}{P(\text{test pos})}\\
&= \frac{.99 \cdot .01}{.144}
\end{aligned}$$

$$= .06875.$$

So in this case the chance that a subject that tests positive has the condition is almost 69%.

In general when you are given the outcome of a test you should ideally also be given enough data to judge what that information means!

Our rule $(*)$from above is a special case of a more general law. Instead of splitting the sample space into two disjoint sets, B and $S \setminus B$, we split it into more parts. If B_1, B_2,…, B_n is a collection of pairwise disjoint events such that

$$A \subseteq B_1 \cup B_2 \cup \cdots \cup B_n$$

then it is the case (see Exercise 86) that

$$A = (A \cap B_1) \cup (A \cap B_2) \cup \cdots \cup (A \cap B_n),$$

and by Kolmogorov's axioms given in Definition 32 we may use the fact that the $A \cap B_i$, for $1 \leq i \leq n$, are pairwise disjoint (again see Exercise 86) to calculate the probability of A as

$$\begin{aligned} &PA \\ &\quad = P(A \cap B_1) + P(A \cap B_2) + \cdots + P(A \cap B_n) && \text{Def} \\ &\quad = P(A \mid B_1)PB_1 + P(A \mid B_2)PB_2 + \cdots + P(A \mid B_n)PB_n && \text{mult law} \\ &\quad = \sum_{i=1}^{n} P(A \mid B_i) \cdot PB_i. \end{aligned}$$

This is sometimes referred to as the **law of total probability**. The way to think about it is that if we split the event A into disjoint parts of the form

$$A \cap B_i,$$

then the probability of A can be recovered from the probabilities of the parts, and the probabilities of these parts can be calculated using the multiplication law. Splitting a set into pairwise disjoint parts is also known as *partitioning* the set, and so we can think of this law as telling us something that the probability of an event can be recovered from the probabilities of its parts, provided the probabilities for the parts can be calculated from the given data using the multiplication law.

The law of total probability is used for a procedure known as Bayesian updating which is discussed in the following section. Examples for the application of this rule can be found there.

Summary

For events A and B, with $PB \neq 0$, and pairwise disjoint collections of events (B_i), where $i \in \mathbb{N}$, we have the following laws concerning total probabilities:

$$P(A \mid B) \cdot PB = P(A \cap B)$$

$$P(B \mid A) = \frac{P(A \mid B) \cdot PB}{PA}$$

$$PA = P(A \mid B) \cdot PB + P(A \mid S \setminus B) \cdot (1 - PB).$$

$$PA = \sum_{i=1}^{n} P(A \mid B_i) \cdot PB_i \qquad\qquad \text{if } A \subseteq \bigcup_{i=1}^{n} B_i$$

In the course unit on data science you will use these laws in order to derive information from data, in particular in the part about *Bayesian statistics*.

Example 4.41. You have a friend who likes to occasionally bet on a horse, but no more than one bet on any given day. From talking to him about his bets, you have some statistical data. There's a five percent chance that he's won big and a twenty-five percent chance that he has won moderately, or else he has lost his stake.

If he has won a significant amount of money there's a seventy percent chance that he has gone to the pub to celebrate, and if he's lost there's an eighty percent chance that he has gone to drown his losses, whereas if he's won a small amount there's only a twenty percent chance that you'll find him in the pub.

If you know he has placed a bet today, and you go to the pub, what's the chance that you will find him there?

We can use the law of total probability to help with that. We partition the overall space into your friend having won big, moderately, or not at all. We know the probabilities for each of these events, and also the conditional probability that he is in the pub for each of those, so the overall probability is

$$\begin{aligned}\frac{70}{100} \cdot \frac{5}{100} + \frac{20}{100} \cdot \frac{25}{100} + \frac{80}{100} \cdot \frac{70}{100} &= \frac{7}{10} \cdot \frac{1}{20} + \frac{2}{10} \cdot \frac{5}{20} + \frac{8}{10} \cdot \frac{14}{20} \\ &= \frac{7 + 10 + 112}{200} = \frac{129}{200} = .645,\end{aligned}$$

so there's a 64.5% chance that you'll find him in the pub.

Exercise 97. Let $(S, \mathcal{E}, P)$ be a probability space, and assume that A, B and C are events. What might we mean when we refer to the probability of A, given B, given C? Can you find a way of expressing that probability? You may assume that B, C and $B \cap C$ all have non-zero probabilities.

Exercise 98. Prove that the law of total probability holds.

CExercise 99. Assume that you have found the following statistical facts about your favourite football team:

- If they score the first goal they win the game with a probability of .7.
- If they score, but the other team scores the first goal, your team has a probability of .25 of winning the game.
- If your team scores then the probability that the game is a draw is .1.

You have further worked out that in all the matches your team has played, in 55% of all games they have scored, and in 40% of those they have scored first. What is the probability that your team wins a randomly picked game?

After further analysis you have worked out that they lose 80% of all games in which they haven't scored. What is the probability that a randomly picked game your team is involved in is a draw?

Exercise 100. One of your friend claims she has an unfair coin that shows heads 75% of the time. She gives you a coin, but you can't tell whether it's that one or a fair version.

You toss the coin three times and get HHT. What is the probability that the coin you were given is the unfair one?

Exercise 101. Assume you have an unfair coin that shows heads with probability $p \in (0, 1]$. You toss the coin until heads appears for the first time. Show that the probability that this happens after an even number of tosses is

$$\frac{1-p}{2-p}.$$

This is a tricky exercise. It depends on cleverly choosing events, and using the law of total probability.

Exercise 102. Consider the following situation: Over a channel bits are transmitted. The chance that a bit is correctly received is p. From observing previous traffic it is known that the ratio of bits of value 1 to bits of value 0 is 4 to 3.

If the sequence 011 is observed what it the probability that this was transmitted?

4.3.4 Bayesian updating

In AI it is customary to model the uncertainty regarding a specific situation by keeping probabilities for each of the possible scenarios. As more information becomes available, for example through carrying out controlled experiments, those probabilities are updated to better reflect what is now known about the given situation. This is a way of implementing machine learning. It is also frequently used in spam detection software.

In this section we look at how probabilities should be updated.

Example 4.42. Assume you are given a bag with three socks in it. You are told that every sock in the bag are either red or black. You are asked to guess how many red socks are in the bag. There are four cases:

$$\{0, 1, 2, 3\}.$$

We model this situation by assigning probabilities to the four. At the beginning we know nothing, and so it makes sense to assign the same probability to every one of these. Our first attempt at modelling the situation is to set the following probabilities.

Original distribution

	0	1	2	3
P	$1/4$	$1/4$	$1/4$	$1/4$

This expresses the fact that nothing is known at this stage. Assume somebody reaches into the bag and draws a red sock which they hold up before returning it to the bag. No we have learned something we didn't know before: there is at least one red sock in the bag. This surely means that we should set $P0$ to 0, but is this all we can do?

The idea is that we should update *all* our probabilities based on this information. The probability $P(i)$ that we have i red socks in the bag should become

$$P(i \mid R),$$

that is, it should be the probability that there are i red socks *given that the drawn sock was red*. Bayes's Theorem helps us to calculate this number since it tells us that

$$P(i \mid R) = \frac{P(R \mid i) \cdot P(i)}{P(R)}.$$

Let us consider the various probabilities that occur in this expression:

- $P(R \mid i)$. This is the probability that a red sock is drawn, given the total number of red socks. This is known, and it is given by the following table:

i	0	1	2	3
$P(R \mid i)$	0	$1/3$	$2/3$	1

  So if the number of red socks is i then the probability of $P(R \mid i)$ is $i/3$.

- $P(i)$. We don't know how many red socks there are in the bag, but we are developing an estimated guess for the probability, and that is what we are going to use. So where this appears we use the probabilities provided by the first table, our original distribution.

- $P(R)$. This is the the probability that the first sock drawn is red, *independent from how many red socks there are*. It is not clear at first sight whether we can calculate that. The trick is to use the law of total probability, as described below.

We should pause for a moment to think about what the underlying probability space is here to make sensible use of the law of total probability.

In the table above we have assigned probabilities to the potentially possible numbers of red socks in the bag. But by drawing a sock from the bag we have expanded the possible outcomes:

These now have to be considered as combinations:: They consist of the number of red socks in the bag, plus the outcome of drawing a sock from the bag. We can think of these as being encoded by

- a number from 0 to 3 (the number of red socks in the bag) and
- a colour, R or B, denoting the outcome of the draw.

In other words, for the moment we should think of the sample space as

$$\{0R, 0B, 1R, 1B, 2R, 2B, 3R, 3B\}.$$

Note that our original outcome i now becomes a shortcut for the event

$$\{iR, iB\}.$$

If we draw further socks from the bag then each current outcome iC will become an event

$$\{iCR, iCB\}.$$

Returning to the probability that a red sock is drawn, $P(R)$, we can now see that this is the probability of the event

$$\{0R, 1R, 2R, 3R\}.$$

Since we can split this event into the disjoint union of

$$\{0R\} \cup \{1R\} \cup \{2R\} \cup \{3R\},$$

the law of total probability tells us that

$$\begin{aligned} &P(R) \\ &\quad = P(R \mid 0)P(0) + P(R \mid 1)P(1) + P(R \mid 2)P(2) + P(R \mid 3)P3 \\ &\quad = 0 \cdot 1/4 + 1/3 \cdot 1/4 + 2/3 \cdot 1/4 + 3/3 \cdot 1/4 \\ &\quad = 1/2. \end{aligned}$$

This should be no surprise: At the moment all the events 0 to 3 are considered to be equally likely, which gives us a symmetry that makes drawing a red and drawing a black sock equally likely, based on what we know so far.

We use this information to update our description of the situation.

First update

	0	1	2	3
P	0	$1/6$	$2/6 = 1/3$	$3/6 = 1/2$

Note that the probability that there is just one red sock has gone down, and that the sock are all red has gone up the most.

Assume another sock is drawn, and it is another red sock. This extends the sample space in that events are now of the form

$$iRR, iRB, iBR, iBB.$$

But the way most implementations of the algorithm work is not to look at it from that point of view. Instead of keeping track of the colour of the socks drawn so far the assumption is that everything we know about what happened so far is encoded in the probabilities that describe what we know about the current situation.

This has the advantage that what we have to do now looks very similar to what we did on the previous round of updates, and it means that one can write code that performs Bayesian updating which works for every round.

So again we are seeking to update $P(i)$ by setting it to

$$P(i \mid R) = \frac{P(R \mid i) \cdot P(i)}{P(R)},$$

where now the $P(i)$ are those calculated in the previous iteration, the first update to the distribution. Note that the value of $P(R)$ has changed. It is now

$$\begin{aligned} &P(R) \\ &\quad = P(R \mid 0)P(0) + P(R \mid 1)P(1) + P(R \mid 2)P(2) + P(R \mid 3)P3 \\ &\quad = 0 \cdot 0 + 1/3 \cdot 1/6 + 2/3 \cdot 1/3 + 3/3 \cdot 1/2 \\ &\quad = 7/9. \end{aligned}$$

The updated probabilities are

Second update

	0	1	2	3
P	0	1/14	4/14	9/14.

If instead the second drawn sock had been black then we would have to update $P(i)$ to

$$P(i \mid B) = \frac{P(B \mid i) \cdot P(i)}{P(B)},$$

where we can read off the probabilities of drawing a black sock given that there are a given number of red socks from the table

i	0	1	2	3
$P(B \mid i)$	1	2/3	1/3	0

which means that

$$P(B \mid i) = (3 - i)/3,$$

and based on the probabilities after the first update we have

$$\begin{aligned} &P(B) \\ &\quad = P(B \mid 0)P(0) + P(B \mid 1)P(1) + P(B \mid 2)P(2) + P(B \mid 3)P3 \\ &\quad = 3/3 \cdot 0 + 2/3 \cdot 1/6 + 1/3 \cdot 1/3 + 0 \cdot 1/2 \\ &\quad = 2/9. \end{aligned}$$

leading to updated probabilities of

Alternative second update

	0	1	2	3
P	0	1/2	1/2	0.

Note that in this case, the probabilities for both cases that have been ruled out, 0 and 3, have been set to 0. Based on what we have seen in this situation, that is a red sock being drawn followed by a black one, it seems reasonable to have the probabilities for the remaining options to be equal

We can see that Bayesian updating is a way of adjusting our model of the current situation by updating the probabilities we use to judge how likely we are to be in any of the given scenarios.

The preceding example is comparatively simple, but there are two issues worth looking at in the context of this example. The first of these is already hinted at in the example: What is the underlying probability space in a case like this?

The sample space changes with the number of socks drawn—one might think of it as evolving over time. At the stage when n socks have been drawn from the bag the outcomes are best described in the form of strings

$$iX_1X_2\cdots X_n,$$

where $i \in \{0,1,2,3\}$ and $X_i \in \{R,B\}$ for $1 \leq i \leq n$. In other words, each outcome consists of the number of red socks, and the result of the sock draws conducted.

As we move from one sample space to the next each outcome

$$iX_1X_2\cdots X_n$$

splits into two new outcomes,

$$iX_1X_2\cdots X_nR \qquad \text{and} \qquad iX_1X_2\cdots X_nB.$$

Note that what is happening here is that the number of red socks in the bag is *fixed* for the entirety of the experiment, and so the actual probability distribution for the first probability space (before the first sock is drawn) is one which

- assigns 1 to the actual number of socks and
- 0 to all the other potential numbers of red socks under consideration.

If, for example, the number of red socks in the bag is 1 then the actual probability distribution is

	0	1	2	3
P	0	1	0	$0.$

Under those circumstances, the actual probabilities for the probability space based on the set of outcomes

$$\{0R, 0B, 1R, 1B, 2R, 2B, 3R, 3B\}.$$

is

	$0R$	$0B$	$1R$	$1B$	$2R$	$2B$	$3R$	$3B$
P	0	0	$1/3$	$2/3$	0	0	0	$0.$

What Bayesian updating is trying to do is to *approximate* this actual probability distribution for the original set of outcomes in a number of steps.

Note that since we do not know what the actual distribution does, it is at first sight surprising that with what little information we have, we can write a procedure that will succeed in approximating the correct distribution. The probabilities used for Pi are quite different from the actual ones given above. But if we keep conducting our random experiments then our approximated distribution will almost certainly converge towards the actual distribution—see Fact 13 for a more precisely worded statement.

The underlying probability space is one where the set of events is the powerset of the sample space, but many events have the probability 0. We don't know what the distribution is, and so we cannot describe that space and use that description in our procedure.

Note that we are very careful about which events play a role in our calculation. These are of two kinds:

- The first kind consists of events whose probability we are trying to estimate. These are the outcomes from the original sample space which expand into events whose number of elements doubles each time we draw a sock.

- The second kind consists of events whose approximated probability is calculated by forming a 'weighted average' over all the events of the first kind. In other words, we are using all the data from our current approximation to give an approximated probability for those events. In the example, this is the probability of drawing a red/black sock. The aim here is to ensure that we do not introduce any additional uncertainty or bias into our calculations.

The reason Bayesian updating is so useful is that it allows us to approximate the unknown probability distribution by conducting experiments (or observing events), with very little information being required for the purpose. At each stage we treat the present approximating distribution as if it were the actual distribution, and we are relying on the idea that over time, the available information will tell us enough to ensure that our approximation gets better.

Note that it will not necessarily get better on every step—whenever a comparatively unlikely event (according to the actual distribution) occurs, our approximation is going to get worse on the next step! But there is the *Law of Large Numbers* Fact 13 which can be thought of as saying that if we keep repeating the same experiment (drawing a sock from the bag) often enough, then almost certainly we will see red socks appearing in the correct proportion.

It is worth pointing out that the idea in Bayesian updating relies on us being able to perform the same experiment more than once—if we don't put the drawn sock back into the bag the idea does not work.[22]

An interesting question is also what we can do if the number of socks in the bag is unknown. It is possible instead to consider the possible ratios between red and black socks. The most general case would require us to cope with infinite sums (since there are infinitely many possible ratios), and that is beyond the scope of this unit. Note also that there is no way of starting with a probability distribution on all possible ratios that assigns to each ratio the same probability in the way we did here, see Proposition 4.4.

If the number of possible ratios is restricted, however, then one may employ the same idea as in the example above, see Exercise 105.

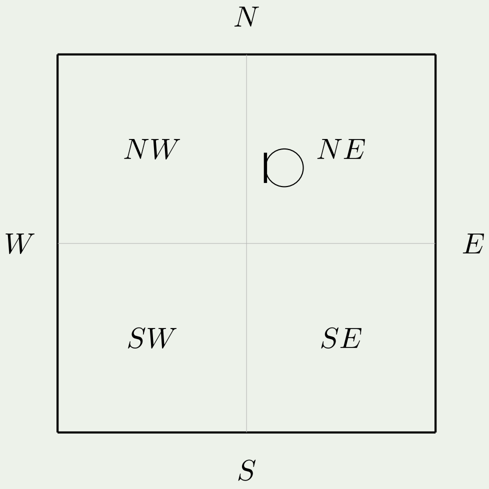
A fun example for Bayesian updating, created by my colleague Gavin Brown, can be found here: `http://www.cs.man.ac.uk/~gbrown/BTTF/`.

We present a toy[23] version of the following example to help you understand the more complex considerations in that example.

[22] Of course, if the drawn sock is not returned then after three draws how many red socks were in the bag originally.

[23] What do you call a toy version of a toy version of an example?

Example 4.43. Assume you have a robot that is in a room of size $l \times l$ metres that has been split into four quadrants. The robot has a sensor, indicated by the line, which is known to face west.



The robot would like to determine which quadrant it is in. To do so it can invoke its sensor, which will detect how far it is to the nearest wall. Depending on whether the measured distance is smaller than $l/2$ or larger than $l/2$ the robot can then deduce whether it is in the quarter adjacent to that wall or not. However, the sensor is inaccurate, and will report a wrong distance $1/4$ of the time.

Assume the robot has some information encapsulated in the following distribution:

Original distribution

	NW	NE	SW	SE
P	0	2/6	3/6	1/6.

Assume the robot conducts a sensor reading and finds that the distance to the wall is less than $l/2$. We use C for ‘close’ to record this outcome (one might use F for ‘far’ if the measured distance is greater than $l/2$). We can determine the probability that it gets this reading for each of the four possibilities:

Q	NW	NE	SW	SE
$P(C \mid Q)$	3/4	1/4	3/4	1/4.

It is close to the wall if it is in one of the two western quadrants, and it gets the correct reading with probability $3/4$. If it is in one of the eastern quadrants then it shouldn’t get this reading unless the measurement is inaccurate, which happens with probability $1/4$.

Using the law of total probability we may now calculate the probability that the robot gets this reading as

$$\begin{aligned} P(C) &= P(C \mid NW) \cdot P(NW) + P(C \mid NE) \cdot P(NE) \\ &\qquad + P(C \mid SW) \cdot P(SW) + P(C \mid SE) \cdot P(SE) \\ &= \frac{1}{4} \cdot \frac{1}{6}(3 \cdot 0 + 1 \cdot 2 + 3 \cdot 3 + 1 \cdot 1) \end{aligned}$$

$$= \frac{1}{4} \cdot \frac{1}{6} \cdot 12$$
$$= \frac{1}{2}.$$

We may now use Bayes' Theorem to find the formula for the updated probabilities of our distribution. Assume Q is one of the for quadrants. Then we want to update PQ to

$$\frac{P(C \mid Q) \cdot PQ}{PC},$$

leading to the following.

Updated distribution

	NW	NE	SW	SE
P	0	2/12	9/12	1/12.

We can see that the probability that the robot is in the SW quadrant has gone up substantially—and indeed, given the fact that this was already the most likely position, the sensor reading further confirmed that opinion.

The following example is a more complicated version of the previous one but considerably simpler than the one that used to appear in an AI lab.

Example 4.44. We extend the previous example as follows: It is not known which way the robot is facing.

The location and orientation of the robot can then be described by a string of length 3, made up from the symbols $\{N, E, S, W\}$: The first two of the symbols give the quadrant in which the robot is, and the third its orientation. In the picture in Example 4.43 you can see a robot in state NEW

The first symbol has to be N or S, and the second symbol has to be E or W, so altogether there are

$$2 \cdot 2 \cdot 4 = 16$$

possible states the robot could be in:

$$\begin{array}{cccc} NEN & NEE & NES & NEW \\ NWN & NWE & NWS & NWW \\ SEN & SEE & SES & SEW \\ SWN & SWE & SWS & SWW \end{array}$$

Again we assume that there is a probability distribution regarding which state the robot is in, either by assigning the same probability to each possible outcome, or by using partial information the robot has.

The robot is using a probability space where the outcomes are as in the table above. Since this is a finite set we can calculate the probability for each potential event, that is each subset of the sample space, by having a probability for each of the sixteen cases.

The robot can perform the same sensor readings as before. This means there is an event of taking a sensor reading, and the outcome can be that the nearest wall in the direction the robot is facing can be less than $l/2$ or more than $l/2$. Hence we should think of the sample space as being given by strings of length four, where the last symbol tells us whether the wall is close (C) or far (F). The robot, however, is only interested in the events consisting of the outcome where the last symbol has been ignored.

So where in the table above we wrote, for example, NEN, the underlying event is really $\{NENC, NENF\}$. We call these events 'status events' because they tell us the potential status of the robot.

Querying the sensor is another event, which we can think of as getting the reading C, or getting the reading F, where the former is given by the set

$$\begin{aligned}\{&NENC, NEEC, NESC, NEWC,\\ &NWNC, NWEC, NWSC, NWWC,\\ &SENC, SEEC, SESC, SEWC,\\ &SWNC, SWEC, SWSC, SWWC\}.\end{aligned}$$

It is convenient to abbreviate that event with C.

Based on what we've said above it should be clear that we know something about the conditional probabilities for sensor readings.

If the position of the robot is NEN then the nearest wall is close. The probability that the sensor reading will be C is therefore $3/4$ (because the sensor is correct 75% of the time), and $1/4$ that the reading will be F.

This means that we know that $P(C \mid NEN) = 3/4$, and similarly we can determine the conditional probabilities for C and F given the various other status events.

How should the robot update information about its status? It should apply Bayesian updating.

When the robot performs a sensor reading it should update the probability for all status events to reflect the result. If the sensor reading returns F then the probability that the robot is in, for example, square NEN should reduce, since if everything works properly the sensor should return C in that situation. The new value for the probability of NEN should be

the probability of NEN given the outcome F.

In other words, we would like to set $P(NEN)$ to

$$P(NEN \mid F).$$

To calculate that probability we can use Bayes's Theorem which tells us that

$$P(NEN \mid F) = \frac{P(F \mid NEN) \cdot P(NEN)}{PF}.$$

We know that $P(F \mid NEN)$ is $1/4$, and we know the current probability for NEN. Hence it only remains to calculate PF. For this remember that F is a shortcut for all events of the form $???F$, that is, the last symbol is F. We have a pairwise disjoint collection of events with the property that F is a subset of their union, since

$$\begin{aligned}F = &\{NENF\} \cup \{NEEF\} \cup \{NESF\} \cup \{NEWF\}\\ &\cup \{NWNF\} \cup \{NWEF\} \cup \{NWSF\} \cup \{NWWF\}\\ &\cup \{SENF\} \cup \{SEEF\} \cup \{SESF\} \cup \{SEWF\}\\ &\cup \{SWNF\} \cup \{SWEF\} \cup \{SWSF\} \cup \{SWWF\}\end{aligned}$$

$$= \bigcup_{X\in\{N,S\},Y\in\{E,W\},Z\in\{N,E,S,W\}} \{XYZF\}.$$

Hence we may use the law of total probability to deduce that

$$PF = \sum_{X\in\{N,S\},Y\in\{E,W\},Z\in\{N,E,S,W\}} P(F \mid XYZ) \cdot P(XYZ).$$

This means we now can calculate the updated probability for NEN.

In general, given a status event L (for location), the robot should update the probability for L to account for the outcome of querying the sensor, so if the outcome is C, it should set

$$P(L) \qquad \text{to} \qquad P(L \mid C).$$

More generally, if we use D (for distance) for an element of the set $\{C, F\}$ then the robot should set

$$P(L) \qquad \text{to} \qquad P(L \mid D),$$

after it has observed the event D. How do we calculate this? We are given

- the probabilities $P(L)$,
- the probabilities $P(D \mid L)$ for $D \in \{C, F\}$.

As discussed above Bayes's Theorem allows us to calculate the desired probability. It tells us that for each status event L we have

$$P(L \mid D) = \frac{P(D \mid L) \cdot PL}{PD}.$$

Looking at the probabilities that appear on the right hand side of this equality, we know $P(D \mid L)$ from the basic setup (information about the robot's sensor), and we have a value for PL since that is what the robot is keeping track of. What about PD?

Remember that this is a shortcut for all events of the form $???D$, so repeating what we have done above for the case where D is equal to F we can see that we have a pairwise disjoint collection of events with the property that D is a subset of their union, since

$$\begin{aligned} D &= \{NEND\} \cup \{NEED\} \cup \{NESD\} \cup \{NEWD\} \\ &\quad \cup \{NWND\} \cup \{NWED\} \cup \{NWSD\} \cup \{NWWD\} \\ &\quad \cup \{SEND\} \cup \{SEED\} \cup \{SESD\} \cup \{SEWD\} \\ &\quad \cup \{SWND\} \cup \{SWED\} \cup \{SWSD\} \cup \{SWWD\} \\ &= \bigcup_{X\in\{N,S\},Y\in\{E,W\},Z\in\{N,E,S,W\}} \{XYZD\}. \end{aligned}$$

Hence we may use the law of total probability to deduce that

$$PD = \sum_{X\in\{N,S\},Y\in\{E,W\},Z\in\{N,E,S,W\}} P(D \mid XYZ) \cdot P(XYZ).$$

The expressions we have found here get quite unwieldy. We show how to adapt that notation to our toy example, and give these equalities using that notation.

Instead of writing XYZ to describe the potential location and orientation of the robot, let's call the events in question

$$L_{i,j,k},$$

where

- $i \in \{0,1\}$, where 0 stands for N and 1 for S,
- $j \in \{0,1\}$, where 0 stands for E and 1 for W, and
- $k \in \{0,1,2,3\}$, where 0 stands for N, 1 for E, 2 for S and 3 for W.

Our encoding means that $L_{0,0,3}$ is equivalent to the status event NES. We can then write the update rule for the probabilities as follows: After a sensor reading resulting in D (where D is still in $\{C,F\}$), the probability

$$PL_{i,j,k}$$

should be set to

$$\begin{aligned} P(L_{i,j,k} \mid D) &= \frac{P(D \mid L_{i,j,k}) \cdot PL_{i,j,k}}{\sum_{i' \in \{0,1\}, j' \in \{0,1\}, k' \in \{0,1,2,3\}} P(D \mid L_{i',j',k'}) \cdot PL_{i',j',k'}} \\ &= \frac{P(D \mid L_{i,j,k}) \cdot PL_{i,j,k}}{\sum_{i',j',k'} P(D \mid L_{i',j',k'}) \cdot PL_{i',j',k'}}, \end{aligned}$$

where the last line is a short-cut for the case when it is understood what values the variables i', j' and k' are allowed to take.

In general, Bayesian updating is performed in the situation where we have the following.

- There are a number of possibilities that may apply, say Q_1, Q_2,...Q_n which are events in some probability space such that they are disjoint, and their union is the whole sample space. It is assumed that there are estimates $P(Q_i)$ for all $1 \leq i \leq n$.
- There is a way of collecting information about the situation, in such a way that there are a number of outcomes s_1, s_2, ..., s_m, and so that each event Q_i can be thought of as

  $$Q_i = \{Q_i s_1, Q_i s_2, \dots, Q_i s_m\}.$$

- When the outcome s_k is observed then for each Q_i its probability is updated to

  $$P(Q_i \mid s_k) = \frac{P(s_k \mid Q_i) \cdot P(Q_i)}{P(s_k)},$$

  where it is assumed that $P(s_k \mid Q_i)$ is known for all combinations, and where the calculation of $P(s_k)$ is performed as

  $$P(s_k) = \sum_{i=1}^{n} P(s_k \mid Q_i) \cdot P(Q_i),$$

giving an overall update of $P(Q_i)$ to

$$\frac{P(s_k \mid Q_i) \cdot P(Q_i)}{\sum_{i=1}^{n} P(s_k \mid Q_i) \cdot P(Q_i)}.$$

Tip

To perform Bayesian updating you need to perform the following steps:

- Determine the possibilities you want to distinguish between,say Q_1, Q_2, to Q_n. Initialize the probability distribution P by setting all probabilities to be equal, unless you have further information.

- Determine which random experiment you may conduct to find out more about the given situation. For each outcome s of this experiment, and for each possibility Q from the first step, determine

  $$P(s \mid Q).$$

  You must be able to find these numbers from the description of the situation. These numbers are used on every step of the calculation and they do not change.

- Assume you carry out the experiment once, and find the outcome s. Calculate

  $$Ps = P(s \mid Q_1) \cdot PQ_1 + P(s \mid Q_2) \cdot PQ_2 + \cdots + P(s \mid Q_n) \cdot PQ_n,$$

  where Q_1, Q_2, $\ldots Q_n$ are all the possibilities determined in step 1, the $P(Q_i)$ come from the current estimate of the probability distribution, and the $P(s|Q_i)$ were determined in step 2. This number has to be recalculated after each update to the distribution.

- Update the probability distribution P by setting

  $$P(Q_i) = \frac{P(s \mid Q_i) \cdot P(Q_i)}{Ps}.$$

  where these numbers were determined in the previous steps, and repeat from step 3.

Example 4.45. In Example 4.43 we have the following:

- The possibilities Q_i we are trying to distinguish between are the four quadrants.

- The outcomes of the experiment that can be conducted to collect further information are C and F.

- One may determine the probability of recording C given that the current position is Q (and similarly for F). The table for C is given in the example. We give the table for F here:

Q	NW	NE	SE	SW
$P(F \mid Q)$	$1/4$	$3/4$	$3/4$	$1/4.$

Note that if the robot turns to face in a different direction then these numbers change.

- One may now compute PC (or PF) using the law of total probability.
- It is now possible to update the distribution using Bayes's Theorem, and then one repeats from step 3.

Example 4.46. In Example 4.42 we have the following.

- The possibilities Q_i we are trying to distinguish between are the possible number of red socks, that is 0, 1, 2 or 3, At the start the distribution P assigns to each outcome the probability $1/4$.
- The outcomes of the experiment we can conduct repeatedly are the two possible colour of the sock drawn, R and B.
- The probability of drawing a red sock if the total number of red socks is i as $i/3$ (and the probability of drawing a black sock as $1 - i/3$).
- One may now calculate $P(R)$ using the law of total probability.
- One may now update the distribution P using Bayes's Theorem, and then repeat from step 3.

Example 4.47. In Example 4.44 we had the following.

- The possibilities Q_i we are trying to distinguish between are the various quadrants and the direction in which the robot's sensor is facing. We assume there is a given probability distribution P at the start.
- The outcomes of the experiment we could conduct repeatedly are the two possible outcomes of using the sensor, C and F.
- We determine the probability of getting C (or F) for each possibility Q_i based on the probability of the sensor working accurately, and the currently assumed situation Q_i using the law of total probability.
- Using Bayes's Theorem one may use this data to update the probability distribution.

Every time we conduct an experiment we update our estimate of the probability distribution underlying the situation. In Bayesian statistics the following terminology is used:

- Let ϕ describe *parameters* whose probability distribution we are aiming to approximate.
- Let α describe *evidence* that we may collect (for example by carrying out a random experiment).

- Let a describe a particular *outcome* of that random experiment.
- Let $P\phi$ be the probability distribution for ϕ, where we use the current best approximation.

$$\text{posterior} \longrightarrow P(\phi \mid a) = \frac{\overset{\text{likelihood}}{P(a \mid \phi)} \cdot \overset{\text{prior}}{P(\phi)}}{\underset{\text{evidence}}{Pa}}$$

The viewpoint there is that

- The *posterior* is the updated distribution based on what we know so far, or more generally in Bayesian statistics it describes what we want to know. It's called 'posterior' because it is what we know *after* we have collected (more) data.
- The *prior* describes our belief before we acquire more data/evidence.
- The *likelihood* is the probability that a happens given the parameters in ϕ.
- The *evidence*, also referred to as *normalization* can be hard to find—in Bayesian updating it's the best approximation to the probability that the observed event does happen.

You will meet these ideas once again in the data science unit in Semester 2.

CExercise 103. Imagine your friend claims to have an unfair coin, which they give to you. From the rather vague description they gave you you aren't sure whether the coin is fair, or whether it gives heads with probability $3/4$, or whether it gives tails with that probability. You want to conduct Bayesian updating to work out which it is.

You are going to mimic having the coin as follows: Take two coins. Every time you would toss our fictitious coin, toss both your coins. If at least one of them shows H, assume the result was H, else assume it was T.

The above procedure allows you to mimic an unfair coin using two fair ones. Follow the instructions to carry out three coin tosses and the corresponding Bayesian updating steps. *Hint: Read the text carefully: How many possibilities for the coin are there that you are trying to distinguish between?*

Note that I expect you to really use a random device, and therefore for different students to have *different* sequences of coin tosses!

Exercise 104. Assume a friend is trying to send you a message which consists of 'yes' or 'no'. He's a bit mischievous, and what he is actually going to do is tell three of your friends something which he claims you can decode into a 'yes' or a 'no' each time.

You are very sceptical about whether you will be able to extract the correct message from your friends, and you only give yourself a 60% chance to do so correctly in each case.

Carry out Bayesian updating to determine your friend's answer. Assume that the messages you extract from your three friends are 'yes', 'yes' and 'no' in that order.

What do you think of the final distribution? How confident are you that you have decoded the message correctly?

Exercise 105. Consider Example 4.42. Instead of knowing the total number of socks, all you know is that the ratio of red to black socks is an element of the following set:

$$\{1/4, 1/3, 1/2, 2/3\}.$$

What is the Bayesian update rule for this situation? Assume a black sock is drawn, followed by a red one. Starting from a probability distribution that assigns the value of $1/4$ to each ratio, give the updated probabilities for each of the given ratios after each draw.

Optional Exercise 16. Assume you are asked to perform Bayesian updating in a case where there are only two possible options, and where information is gained by performing an experiment which also has two possible outcomes.

The resulting case can be described using three parameters:

- The probability p that we have assigned to the first case'
- the probability q that tells us how likely Outcome 1 is if we are in Case 1 and
- the probability r that tells us how likely Outcome 1 is if we are in Case 2.

Write down the rule for a Bayesian update in this situation. Can you say anything about subsequent calculations?

4.4 Random variables

Often when we study situations involving probabilities we want to carry out further calculations. For example, in complexity theory (see COMP11212 and COMP26120) we are frequently looking for the 'average case'—that is, we would like to know what happens 'on average'. By this one typically means taking all the possible cases, each weighted by its relative frequency (not all cases my be equally frequent), and forming the average over all those. For examples of what is meant by an 'average case' for two search algorithms see Examples 4.95 to 4.98.

But in order to carry out these operations we have to be in a situation where we can *calculate* with the values that occur. If we look at some of the examples studied then we can see that some of them naturally lend themselves to calculating averages (it is possible, for example, to ask for the average number of eyes shown when throwing two dice), and some don't (there's no average colour of a sock drawn from one of our bags of socks).

This is why people often design questionnaires by giving their respondents a scale to choose from. The university does this as well: When you will be asked to fill in course unit questionnaires for all your units, then part of what you are asked to do is to assign numbers. 'On a scale of 1 to 5, how interesting did you find this unit.' This allows the university to form averages. But what does it mean that the average interest level of COMP11120 was 3.65 (value from 2014/15)?[24] Certainly every time you assign numbers so that you may form averages, you should think about what those numbers are supposed to mean, and whether people who are asked to give you numbers are likely to understand the same as you, and as each other, by those numbers.

Nonetheless, forming averages can be a very useful action to perform, and that is why there is a name given to functions that turn the outcomes from some probability spaces into numbers. We see below that this does not merely allow us to calculate averages but also to describe particular events without knowing anything about the events or outcomes from the underlying probability space.

4.4.1 Random variables defined

Random variables are functions that translate the elements of a sample space, that is the possible outcomes from a random experiment, to real numbers. But this translation has to happen in such a way that we know what the probabilities for the resulting numbers are, and that requires a technical definition. In order to formulate that we have to define an additional concept.

Definition 36: measurable function

Let $(S, \mathcal{E}, P)$ be a probability space. The function

$$f \colon S \longrightarrow \mathbb{R}$$

is *measurable* if and only if for all elements r of $\mathbb{R}$ the sets

- $\{s \in S \mid fs \leq r\}$ and
- $\{s \in S \mid r \leq fs\}$

are events, that is, elements of $\mathcal{E}$.

Note that in the case where $\mathcal{E} = \mathcal{P}S$, as is often the case for applications, every function from S to $\mathbb{R}$ is measurable.

Definition 37: random variable

Given a probability space $(S, \mathcal{E}, P)$ a **random variable** over that space is a measurable function from S to $\mathbb{R}$.

Example 4.48. When we toss a coin, but record the outcomes as numbers, say 0 for heads and 1 for tails, you have a random variable.

Example 4.49. If you have a population whose height distribution you know

[24] On these questionnaires they try to make the numbers slightly more meaningful by assigning 5 to 'agree' and 1 to 'disagree', but when does one move from one grade to another? Is it really meaningful to average those out?

(compare Example 4.59) you may think of randomly picking a person and recording their height as a random variable.

Example 4.50. When you're playing a game of chance, and you assign a value of -1 to losing, 0 to a draw and 1 to a win, you have a random variable. You could also give 3 for a win, 1 for a draw, and 0 for a loss, and that would also result in a random variable.

Note that it is often tempting to define a random variable as a function from a sample set S to a subset of $\mathbb{R}$. Strictly speaking this does not satisfy the above definition. One should instead make the target set of the random variable $\mathbb{R}$ and observe that its *range* is a proper subset of $\mathbb{R}$. If one changes the definition above then many of the results and definitions below become more complicated. Theorem 4.11 gives a technical result that argues that we could, instead of looking at all of $\mathbb{R}$, restrict ourselves to the range of the function from the start.

Note that whenever we assign numbers to outcomes, and then carry out calculations with those numbers, we have to worry about whether our interpretation of those numbers makes sense. In game theory it is customary to use any items (money or points) won or lost to encode the outcome of a game in a number, but that may not be a faithful description of what a win or loss means to the individual playing.

Whenever you have a probability space $(S, \mathcal{E}, P)$ such that the set of outcomes S is a subset of $\mathbb{R}$ then you have a probability variable, provided you can calculate the probabilities of all sets of the form

$$S \cap [r, \infty) \qquad \text{and} \qquad S \cap (-\infty, r],$$

where $r \in \mathbb{R}$.

For some random experiment one would naturally record the outcome as a number, and that gives a random variable, but in other cases one has to translate the outcome to a real number first. See the first example given above, but also more interestingly see the following example.

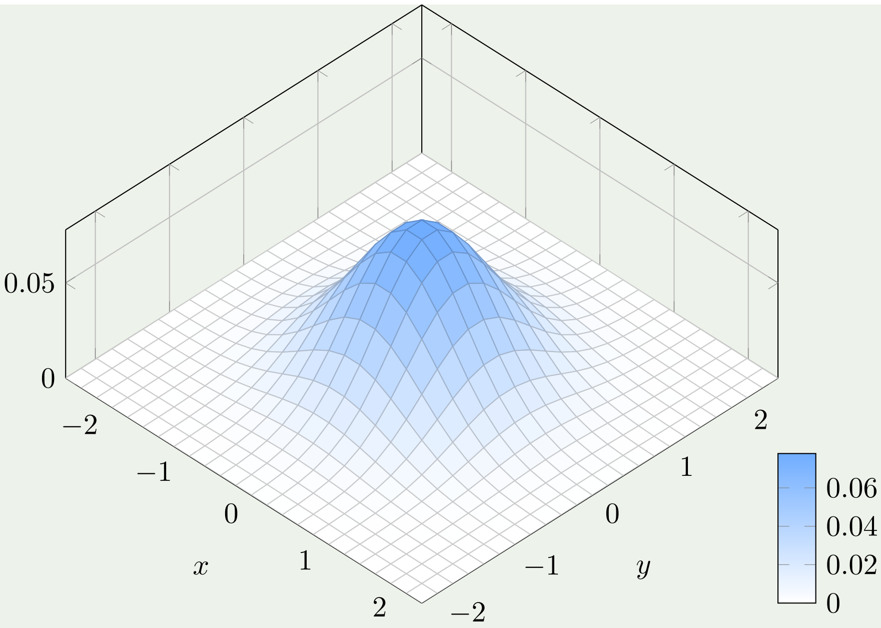
Example 4.51. If you are plotting the position of a butterfly in the form of two coordinates, (x, y), then to get a random variable you have to turn those two numbers into one. You could, for example, compute the distance of the butterfly from a fixed point, and that could be considered a random variable.

Technically this amounts to doing the following. We have a probability space with underlying sample set $\mathbb{R} \times \mathbb{R}$, and a set of events based on the Borel σ-algebra, where all sets of the form

$$[r, r'] \times [s, s'],$$

for $r, r', s, s' \in \mathbb{R}$ are events. We assume that there is a probability density function describing the probability that the butterfly is at point a given point (x, y). One suitable such function is

$$\begin{tikzcd}
\mathbb{R} \times \mathbb{R} \arrow[r] & \mathbb{R}^{+} \\
(x, y) \arrow[r, mapsto] & 222\dfrac{e^{-1/2(x^2+y^2)}}{2\pi}.
\end{tikzcd}$$



To create a random variable we would like to apply the function

$$\begin{aligned} \mathbb{R} \times \mathbb{R} &\longrightarrow \mathbb{R} \\ (x, y) &\longmapsto \sqrt{x^2 + y^2} \end{aligned}$$

to the location, which gives us the butterfly's distance from some chosen point that here is assumed to be $(0, 0)$. We have taken two-dimensional data and turned it into a random variable, which requires the restriction to just one dimension. However, calculating the probability density function of this random variable is non-trivial.

Alternatively you could measure the distance relative to a north/south (or other) axis. For example, you could project your position onto its x-axis, and then you could calculate the probability density function of the resulting random variable as

$$\begin{aligned} \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto \int_{-\infty}^{\infty} \frac{e^{-1/2(x^2+y^2)}}{2\pi} dy. \end{aligned}$$

Example 4.52. Consider Example 4.22 where we have given several probability spaces one might use to describe throwing two dice. If you pick as the space the one with outcomes

$$\{(i, j) \mid i, j \in \{1, 2, 3, 4, 5, 6\}\},$$

then the function which maps the pair (i, j) from that set to the sum of eyes shown

$$i + j,$$

(viewed as an element of $\mathbb{R}$) is a random variable[25]

$$\begin{aligned} X\colon \{1,2,3,4,5,6\}\times\{1,2,3,4,5,6\} &\longrightarrow \mathbb{R} \\ (i,j) &\longmapsto i+j. \end{aligned}$$

Whenever we have a random variable we get an induced probability distribution. In order to calculate the probability that X takes the value 4 we have to calculate[26]

$$\begin{aligned} &P(\{(i,j)\in\{1,2,3,4,5,6\}\times\{1,2,3,4,5,6\}\mid X(i,j)=4\}) \\ &\qquad = P(\{(1,3),(2,2),(3,1)\}) \\ &\qquad = P(\{(1,3)\})+P(\{(2,2)\})+P(\{(3,1)\}) \\ &\qquad = \frac{1}{36}+\frac{1}{36}+\frac{1}{36} \\ &\qquad = \frac{3}{36}=\frac{1}{12}. \end{aligned}$$

This is usually written in the shortcut notation of

$$P(X=4).$$

But note that since we have translated our outcomes into real numbers we may also ask, for example, what the following probabilities are:

$$\begin{gathered} P(X\le 4) \\ P(X\le -4) \\ P(X\le 5.5) \\ P(X\ge 10) \end{gathered}$$

The events described here do not look as if they have anything to do with the original experiment of rolling two dice, but since we have translated the outcome from that experiment into real numbers we may construct such events.

These probabilities can be calculated as follows:

- $P(X\le 4)$. This can be calculated by splitting it into the possible outcomes satisfying that property.

$$\begin{aligned} P(X\le 4) &= P((X=2)\cup(X=3)\cup(X=4)) \\ &= P(X=2)+P(X=3)+P(X=4) \\ &= \frac{1}{36}+\frac{2}{36}+\frac{3}{36}=\frac{1}{6}. \end{aligned}$$

- $P(X\le -4)$. Clearly there are no possible outcomes which satisfy this condition, so this probability is 0.

- $P(X\le 5.5)$. This works similar to the first calculation.

$$P(X\le 5.5)$$

$$\begin{aligned} &= P((X = 2) \cup (X = 3) \cup (X = 4) \cup (X = 5)) \\ &= P(X = 2) + P(X = 3) + P(X = 4) + P(X = 5) \\ &= \frac{1}{36} + \frac{2}{36} + \frac{3}{36} + \frac{4}{36} = \frac{10}{36} = \frac{5}{18}. \end{aligned}$$

- $P(X \geq 10)$. This is similar to the previous example.

$$\begin{aligned} P(X \geq 10) &= P(X = 10) + P(X = 11) + P(X = 12) \\ &= \frac{3+2+1}{36} = \frac{1}{6}. \end{aligned}$$

Below we describe how this works for arbitrary random variables.

In general given a random variable X on a sample space $(S, \mathcal{E}, P)$, and real numbers r and r', we define

- $P(r \leq X \leq r') = P\{s \in S \mid r \leq X(s) \leq r'\}$
- $P(r \leq X) = P\{s \in S \mid r \leq X(s)\}$,
  $P(r < X) = P\{s \in S \mid r < X(s)\}$,
- $P(X \leq r') = P\{s \in S \mid X(s) \leq r'\}$,
  $P(X < r) = P\{s \in S \mid X(s) < r\}$..

The general case is given by the following proposition.

Proposition 4.7

Let X be a random variable over the probability space $(S, \mathcal{E}, P)$. The probability distribution of X is determined by the fact that, for any real interval I we have

$$P(X \in I) = P\{s \in S \mid X(s) \in I\}.$$

In other words, if we are given an interval in $\mathbb{R}$ then in order to determine its probability we ask for the probability of the event given by all those elements of the original sample space which are mapped into that interval. Note that the sets that appear on the right hand side of the equal sign appear in the definition of measurability. This ensures that in the original probability space we have a probability for the set in question.

Definition 38: discrete/continuous random variable

A random variable X is **discrete** if and only if its range is a countable[27] subset of $\mathbb{R}$. A random variable which is not discrete is **continuous**.

While there is a mathematical theory that allows the discrete case to be treated at the same time as the continuous one, covering the mathematics that allows this is beyond the scope of this course unit. In what follows the discrete case is frequently treated separately. In the text, and in some of the results given, some guidance is given on how the discrete case may be seen as a special case of the continuous one.

[25] Random variables are typically named using capital letters from the end of the alphabet.

[26] You may want to return to Example 4.22 for an explanation.

[27] What this means formally is discussed in Section 5.2. Every finite set is countable, and you may think of countable sets as ones that can be described in the form $\{s_i \mid i \in \mathbb{N}\}$.

Note in particular that if a random variable X has a finite range. then Proposition 4.7 indicates that we can treat it in much the same way as we did a probability space with a finite sample set where every set of the form $\{s\}$, for $s \in S$, is an event.

Example 4.53. If we look at Example 4.48 it is clear that there are only two possible outcomes of the given random variable X, namely 0 and 1, and that each of those occurs with probability $1/2$.

This means that when we calculate the probability

$$P(X \leq r),$$

then this is completely determined by which of 0 and 1 is in the given interval. In particular we have

$$P(X \leq r) = \begin{cases} 0 & r < 0 \\ 1/2 & 0 \leq r < 1 \\ 1 & \text{else.} \end{cases}$$

Note that every discrete random variable has a range of the form

$$\{r_i \in \mathbb{R} \mid i \in \mathbb{N}\},$$

that is a subset of $\mathbb{R}$ that is indexed by the natural numbers. For such a random variable, say X, further note that

$$X = r_i, \qquad \text{for } i \in \mathbb{N}$$

gives us a collection of pairwise disjoint events which collectively have probability 1, as you are asked to show in the following exercise.

Exercise 106. Let X be a discrete random variable with range

$$\{r_i \in \mathbb{R} \mid i \in \mathbb{N}\}.$$

Show that

$$\sum_{i \in \mathbb{N}} P(X = r_i) = 1.$$

Example 4.54. Assume we are conducting a random experiment that consists of tossing a coin three times. In order to record the possible outcomes we can use strings of length three which give the outcomes for each toss, resulting in the sample space

$$S = \{HHH, HHT, HTH, HTT, THH, THT, TTH, TTT\}.$$

This means we can describe the elements of S via

$$S = \{s_1 s_2 s_3 \mid s_1, s_2, s_3 \in \{H, T\}.\}.$$

Each of these occurs with probability $1/8$. We can turn this experiment into a random variable by converting each outcome into into a number. Here we pick the number of heads shown.

The resulting random variable, say X, has the following range:

$$\{0, 1, 2, 3\}.$$

since the number of heads may range from 0 to 3. In order to find the probability of each of the possible results we have to work out which events are mapped to which number. Recall that a random variable is a *function*, and our function is given by

$$\begin{aligned} \{S\} &\longrightarrow \mathbb{R} \\ s_1s_2s_3 &\longmapsto \text{no of } H \text{ in } s_1s_2s_3. \end{aligned}$$

The probability for each possible value for X is given by adding all the probabilities of outcomes from the original space which are mapped to it. For example,

$$\begin{aligned} P(X = 2) &= P\{s_1s_2s_3 \in S \mid \text{ no of } H \text{ is } 2\} \\ &= P\{HHT, HTH, THH\} \\ &= \frac{1}{8} + \frac{1}{8} + \frac{1}{8} \\ &= \frac{3}{8}. \end{aligned}$$

We can conveniently give these probabilities in a table. To help illustrate how those probabilities come about we also give the outcomes from the original space in the column of the number they are mapped to by X.

TTT	HTT THT TTH	HHT HTH THH	HHH
0	1	2	3
$1/8$	$3/8$	$3/8$	$1/8$

This is the *probability mass function* for the random variable, which is formally defined below. We can now ask questions such as what is the probability that the number of heads is at most 2, or what is the average number of heads tossed.

Exercise 107. Consider the experiment that consists of tossing a fair coin four times. Consider the random variable X which records the number of heads thrown. Calculate the following probabilities.

(a) $P(X = 2)$,

(b) $P(X \leq 3)$,

(c) $P(X \leq \pi)$,

(d) $P(X \geq 3)$,

(e) $P(X \geq 10)$,

(f) $P(X < -1)$.

(g) $P((X = 1) \cup (X = 3))$.

(h) $P(X$ is even$)$. Note that this only makes sense because we know the range of X is a subset of $\mathbb{N}$—it does not make sense to talk of evenness for numbers in $\mathbb{R}$.

4.4.2 A technical discussion

What follows is a fairly technical discussion regarding why we can define probabilities in the way outlined above. The material from this subsection is not examinable, and you should feel free to skip it when reading the notes.

Optional Exercise 17. Show that if $(S, \mathcal{E}, P)$ is a probability space, and if we have a random variable $X : S \longrightarrow \mathbb{R}$ then given r and r' in $\mathbb{R}$ we have that

$$(r \leq X \leq r')$$

is an event. This means that $P((r \leq X \leq r')$ is always defined.

Proposition 4.8

Let $(S, \mathcal{E}, P)$ be a probability space, and let $f : S \longrightarrow \mathbb{R}$ be a function. If f is measurable (and so a random variable) then for every B in the Borel σ-algebra $\mathcal{E}_B$ on $\mathbb{R}$ we have that

$$\{s \in S \mid fs \in B\}$$

is an event, that is an element of $\mathcal{E}$.

Proposition 4.9

Let $(S, \mathcal{E}, P)$ be a probability space, and let be $f : S \longrightarrow \mathbb{R}$ a measurable function. For $i \in \mathbb{N}$ let I_i be an interval in $\mathbb{R}$ such that the the I_i are pairwise disjoint. If we define, for $i \in \mathbb{N}$,

$$E_i = \{s \in S \mid fs \in I_i\},$$

then the E_i are pairwise disjoint and so

$$\begin{aligned} P(\bigcup_{i\in\mathbb{N}} I_i) &= P(\bigcup_{i\in\mathbb{N}} E_i) \\ &= \sum_{i\in\mathbb{N}} PE_i \\ &= \sum_{i\in\mathbb{N}} PI_i. \end{aligned}$$

As a consequence we get the following result:

Theorem 4.10

Let $(S, \mathcal{E}, P)$ be a probability space, and $f : S \longrightarrow \mathbb{R}$ a measurable function. Then a probability space is given by $\mathbb{R}$, the Borel σ-algebra $\mathcal{E}_B$, and the probability distribution

$$\begin{aligned} \mathcal{E}_B &\longrightarrow [0,1] \\ E &\longmapsto P(\{s \in S \mid fs \in E\}). \end{aligned}$$

Example 4.55. Looking back at Example 4.53 we can see how we have effectively defined a probability distribution for $\mathbb{R}$. It can be described as follows: Given an element E of the Borel σ-algebra $\mathcal{E}_B$ the probability of E is given as

$$PE = \begin{cases} 0 & 0, 1 \notin E \\ 1/2 & \text{exactly one of } 0, 1 \text{ in } E \\ 1 & \text{else.} \end{cases}$$

In general, if a random variable X has a finite range,[28] say

$$\{r_1, r_2, \ldots, r_n\} \text{ in } \mathbb{R}$$

then given an interval I we have that

$$P(X \in I) = P(I \cap \{r_1, r_2, \ldots, r_n\}) = \sum_{i \in \{1,2,\ldots,n\}, r_i \in I} P(X = r_i).$$

In other words we add up all the probabilities for those elements r_i of the range of X which are elements of I.

Alternatively we may restrict ourselves to the range of the underlying measurable function to define a probability space—in this way we remove those parts of $\mathbb{R}$ which are assigned a probability of 0.

Theorem 4.11

Let $(S, \mathcal{E}, P)$ be a probability space, and $f \colon S \longrightarrow \mathbb{R}$ a measurable function. Then a probability space is given by the range T of f, the σ-algebra

$$\{E \cap T \mid E \in \mathcal{E}_B\},$$

and the probability distribution

$$\begin{aligned} \{E \cap T \mid E \in \mathcal{E}_B\} &\longrightarrow [0,1] \\ B &\longmapsto P(\{s \in S \mid fs \in B\}). \end{aligned}$$

Example 4.56. If we once again look at Example 4.53 then the range of the random variable is $\{0, 1\}$. The set of events given in the previous theorem is then merely the powerset of this set. The probability distribution is given by the following assignment:

$$P\emptyset = 0$$

[28] This result can be extended to the case where X has a range that can be expressed as $\{r_i \mid i \in \mathbb{N}\}$.

$$P\{0\} = P\{1\} = \frac{1}{2}$$
$$P\{0,1\} = 1.$$

Tip 1

Theorem 4.11 effectively tells us that it is okay to define a random variable as a function from some sample set S to *a subset of* $\mathbb{R}$. This can be useful when describing specific situations. Below we point out when we do this the first few times and then we do this tacitly.

4.4.3 Calculating probabilities for random variables

Above we define probabilities for random variables. You can think of them as translating the original outcomes into numbers in such a way that we can look at the probabilities of subsets of $\mathbb{R}$ instead of events from the original space. The previous section establishes that we can take the original probability distribution and transfer it to the random variable, which gives another probability distribution, this time over the real numbers, which means that all the usual results (see Sections 4.2.5 and 4.3.3) hold.

One of the advantages of considering random variables is that it allows us to compute probabilities with very little information, in particular without knowing too much about the original probability space.

Example 4.57. Assume that X is a random variable and that we know that

- $P(X = 1) = 1/2$,
- $P(X = 2) = 1/4$, and
- $P(X \geq 2) = 1/2$.

This is enough to allow us to calculate, for example

- $P(X = 0) = 0$,
- $P(X \leq .5) = 0$,
- $P(X > 2) = 1/4$.

We can see from the given information that the total probability of 1 is distributed in the following way:

- $1/2$ of the available 'probability mass' goes to 1;
- $1/4$ of it goes to 2;
- the remaining $1/4$ goes to the interval $(2, \infty)$, and we cannot tell more precisely where it goes from the given data.

In particular this means that none of the probability goes to 0, or to any number below 1, which explains the first two claims. We can also derive the final result more formally by noting that

$$(X \geq 2) = (X = 2) \cup (X > 2)$$

and that the two sets whose union we form are disjoint, which means we have

$$1/2 = P(X \geq 2) = P(X = 2) + P(X > 2) = 1/4 + P(X > 2),$$

from which we may deduce the last result.

Note in particular that we do not know whether X is a discrete or a continuous random variable! The fact that it has non-zero probability for being equal to 1 and 2 might suggest it is the former, but it could still be the case that the behaviour is continuous for values beyond 2.

See Example 4.71 for a way of picturing some of this information.

Example 4.58. Recall Example 4.54, where we look at a random variable X given by the number of heads recorded when tossing a coin three times.

For this random variable we can see, for example, that

$$P(X > 2) = P(X = 3) = \frac{1}{8},$$

of that

$$P(X \leq .5) = P(X = 0) = \frac{1}{8},$$

and that

$$P(X \in (-\infty, -2] \cup [5, \infty)) = 0.$$

Because we know that we may consider the outcomes as real numbers we can write down (and calculate) the probability for the random variable taking its value in any interval, for example. This is quite useful, and in Section 4.4.4 we demonstrate that we can also use this to graphically represent the given probability distribution.

Exercise 108. Assume you have a probability space with outcomes

$$\{s_1, s_2, s_3, s_4, s_5\},$$

and that the following hold:

- The outcomes s_1 and s_2 are equally likely.
- The outcomes s_3, s_4 and s_5 are equally likely.
- The outcomes of the first kind are three times as likely as the outcomes of the second kind.

A random variable is given by the function X defined by

$$\begin{aligned} X\colon \{s_1, s_2, s_3, s_4, s_5\} &\longrightarrow \mathbb{R} \\ x &\longmapsto \begin{cases} 1 & x = s_1 \text{ or } x = s_2 \\ 3 & x = s_3 \\ 5 & \text{else.} \end{cases} \end{aligned}$$

Compute the following:

(a) $P(X \leq 1.5)$,

(b) $P(X \geq 3)$,

(c) $P(2.5 \leq X \leq 3.2)$,

(d) $P(X \geq 6)$.

Sometimes we want to take a random variable, which is a function that maps outcomes to real numbers), and apply another function to it so as to translate the outcomes.

Example 4.59. Assume that I've been given the measures in height of a group of people, where the measures have been carried out with great precision. If I have the measures for everybody in the population I can give the probability distribution of the random experiment given by picking a person (randomly) from the group. One might want to treat this like a continuous random variable if a lot of people are involved.

But maybe for my purposes I only care about how many people I have in much loser categories. Assume that I'm only interested in the following categories:

- People who are at most than 140 cm tall or
- people who are from 140 to 160 cm tall or
- people who are from 160 to 180 cm tall and
- people who are taller than 180 cm.[29]

I would like to count how many people out of the group belong to each category to construct a probability space which allows me to work out the probabilities that a randomly chosen person from that group falls into a particular category.

I may create another random variable by composing X with the function $f\colon \mathbb{R} \longrightarrow \mathbb{R}$ given by the following assignment.

$$x \longmapsto \begin{cases} 1 & x \leq 140 \\ 2 & 140 < x \leq 160 \\ 3 & 160 < x \leq 180 \\ 4 & 180 < x. \end{cases}$$

I can then compute the probability for the new outcomes, given by the range of the composite $f \circ X$, by counting how many people fall into each category and dividing by the total population count—which gives the same result as taking the original probability distribution for X and using f to translate it to the new outcomes.

We can see from the preceding example that it can be useful to take a given random variable and use a function on its possible values (here mapping actual heights to representative of some height categories) to get a different (but related) random variable that better expresses whatever we are concerned with.

Example 4.60. In the robot Example 4.44) one might want to consider the orientation of the robot and view it as an angle from 0 to 360 degrees. The orientation is a continuously varying entity, but for the purpose of performing calculations one might split it into a finite number of parts of equal size, creating a discretely valued random variable, which makes it easier to carry out calculations (Bayesian updating in that case).

The following result tells us that composing with a function $\mathbb{R} \longrightarrow \mathbb{R}$ always gives us another random variable, provided that the function is well behaved.

Proposition 4.12

If X is a random variable and $f\colon \mathbb{R} \longrightarrow \mathbb{R}$ is a measurable function then

$$f \circ X$$

is a random variable.

For the random variable $f \circ X$ and an interval I in $\mathbb{R}$ we have

$$P(f \circ X \in I) = P\{r \in \mathbb{R} \mid fr \in I\}.$$

Optional Exercise 18. Show that if X is a measurable function from some probability space to $\mathbb{R}$, and if $f\colon \mathbb{R} \longrightarrow [r, r']$ is measurable for the Borel probability space on the interval $[r, r']$ then their composite $f \circ X$ is measurable.

Example 4.61. Recall Example 4.52 of adding the eyes shown by two dice, which we may consider a random variable X. We might instead only wish to record whether this number is even or odd. Theorem 4.11 tells us that it is okay to view X as a function with target the set of natural numbers from 2 to 12.

With that observation we may express our new object of interest by composing X with the following function.

$$\begin{array}{rcl} f\colon \{2,3,4,\dots,11,12\} & \longrightarrow & \{0,1\} \\ x & \longmapsto & x \bmod 2 \end{array}.$$

[29] Clearly one has to think about what should happen on the borderline—let's assume here this belongs to the lower height category.

We may now compute the probabilities for $f \circ X$ as follows.

$$\begin{aligned}
P(f \circ X = 0) &= P\{n \in \{2, 3, 4, \dots, 11, 12\} \mid n \bmod 2 = 0\} \\
&= P\{2, 4, 6, 8, 10, 12\} \\
&= \frac{1}{36} + \frac{3}{36} + \frac{5}{36} + \frac{5}{36} + \frac{3}{36} + \frac{1}{36} \\
&= \frac{18}{36} = \frac{1}{2}.
\end{aligned}$$

To calculate $P(f \circ X = 1)$ it is sufficient to note that the two probabilities have to add up to 1, and so this is also $1/2$.

Example 4.62. Recall Example 4.54 where we considered the random variable X given by counting the number of heads that appear when tossing a fair coin three times. We know that the range of X is $\{0, 1, 2, 3\}$, so we may think of X as a function from the original sample space to that set. The probabilities for the various outcomes are given in the following table.

0	1	2	3
$1/8$	$3/8$	$3/8$	$1/8$

Now assume we are interested only in whether the number of heads is more than one away of the number of tails, or not. The outcomes 1 and 2 satisfy that new property, and the outcomes 0 and 3 do not. Consider the following function.

$$\begin{array}{rcl}
f\colon \{0, 1, 2, 3\} & \longrightarrow & \{0, 1\} \\
x & \longmapsto & \begin{cases} 0 & x = 1 \text{ or } x = 2 \\ 1 & \text{else.} \end{cases}
\end{array}$$

Once again we use Theorem 4.11 to think of X as a function with target set $\{0, 1, 2, 3\}$. Then composing X with f gives another random variable, with range $\{0, 1\}$, where 0 means the number of heads is at most one different from the number of tails, and 1 means the difference is larger.

We can determine the probability for the new outcomes by adding the probabilities of the old outcomes which are mapped to it. Again we give a table that provides the probabilities for each outcome, and above each outcome of $f \circ X$ we give the outcomes from X that are mapped to it by f.

new outcomes	0	1
old outcomes	1, 2	0, 3
probabilities	$P(X = 1) + P(X = 2)$ $= \frac{3}{8} + \frac{3}{8} = \frac{3}{4}$	$P(X = 0) + P(X = 3)$ $= \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$

Example 4.63. Assume that we are again starting with the random variable X that turns tossing a coin three times into the number of heads that appear among the three tosses, see the previous example. This time we want to change the random variable by only recording whether the number of heads is even or

odd. This means we are composing the random variable X (viewed as having target set $\{0,1,2,3\}$ as before with the function

$$\begin{aligned} g\colon \{0,1,2,3\} &\longrightarrow \{0,1\} \\ x &\longmapsto x \bmod 2. \end{aligned}$$

Then the probabilities for the possible values of the random variable $g \circ X$ are given in the following table.

new outcomes	0	1
old outcomes	0, 2	1, 3
probabilities	$P(X=0)+P(X=2)$ $=\frac{1}{8}+\frac{3}{8}=\frac{1}{2}$	$P(X=1)+P(X=3)$ $=\frac{3}{8}+\frac{1}{8}=\frac{1}{2}$

Example 4.64. In Example 4.59 the random variable X had four possible values, namely

$$\{1,2,3,4\}.$$

Assume that the probabilities that a randomly chosen person from the monitor group fits into each category is given by the following table:

	1	2	3	4
P	$1/2$	$1/4$	$1/8$	$1/8$

We can now calculate with these probability much as if we had a discrete probability space at the start.

For example, if I want to know the probability that a member of my population is below 160cm, that is, belongs to categories 1 or 2,

$$P(X<160) = P(f\circ X=1)+P(f\circ X=2) = \frac{1}{2}+\frac{1}{4}=\frac{3}{4}.$$

Example 4.65. Assume that I am in the situation of Example 4.59, but now I am only interested whether somebody is below 160 cm or above.

Then I can take my previous random variable, which produced the possible values

$$\{1,2,3,4\},$$

and compose it with the function

$$\begin{aligned} \{1,2,3,4\} &\longrightarrow \{1,2\} \\ x &\longmapsto \begin{cases} 1 & x=1 \text{ or } x=2 \\ 2 & \text{else} \end{cases} \end{aligned}$$

to get a new random variable whose only values which only distinguishes between people with a height of less than or equal to 160cm, which are in category 1, and those who are taller than 160cm, which are in category 2.

Example 4.66. Assume we have a random variable X that has a range of values

$$\{-n, -(n-1), \dots, -2, -1, 0, 1, 2, \dots, n-1, n\}.$$

Maybe for some purposes we are not interested in the values as such, but only in how far distant they are from the mid-point, 0. This might be because we are only interested in the difference between some value and 0, but not whether that difference is positive or negative (compare also Definition 43).

By composing the random variable with the absolute function

$$\begin{aligned} |\cdot| \colon \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto |x|, \end{aligned}$$

we obtain a new random variable Y which takes its values in the set

$$\{0, 1, \dots, n\}.$$

To calculate probabilities for Y we have to know that

$$P(Y = i) = \begin{cases} P(X = i) + P(X = -i) & 0 \leq i \leq n \\ 0 & \text{else.} \end{cases}$$

CExercise 109. Recall the unfair die from Exercise 83. Take as a random variable X the number of eyes shown. Calculate the following.

(a) $P(X \leq 3)$,

(b) $P(X \geq 5)$,

(c) $P(4 \leq X < 6)$,

(d) $P(X \leq \pi)$,

(e) $P(X \geq 7)$.

Now assume that the random variable Y is given by the sum of the eyes shown by two such dice. Calculate the following.

(f) $P(Y \leq 4.5)$,

(g) $P(Y \geq 11.5)$.

Finally assume that we have the random variable Y and we compose it with the following function:

$$\begin{aligned} f \colon \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto (x-7)^2. \end{aligned}$$

Calculate the following.

(h) $P(f \circ Y \geq 6)$,

(i) $P(f \circ Y \leq .5)$.

4.4.4 Probability mass functions and cumulative distributions

There are many example of random variables where we do not need to worry about all real numbers but only those that appear in the range of the random variable. We can give a graphical presentation of how the probabilities is spread over that range. It is the equivalent to a probability density function for the case where we have discrete values.

Definition 39: probability mass function

Let X be a random variable with a countable range, say

$$\{r_i \mid i \in \mathbb{N}\}.$$

The **probability mass function (pmf) for** X is given by

$$\begin{aligned} \{r_i \mid i \in \mathbb{N}\} &\longrightarrow [0,1] \\ r_i &\longmapsto P(X = r_i). \end{aligned}$$

It is appropriate to think of a probability mass function as the discrete version of a probability density function.

Example 4.67. For the random variable that consists of assigning the total number of eyes to the throw of two dice, see Example 4.52, the pmf is given by

2	3	4	5	6	7	8	9	10	11	12
$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

This is of course the original probability distribution from Example 4.2 for one of the sample spaces discussed there—if the outcomes are already described as numbers then this is what happens.

Example 4.68. If we throw a coin three times, see Examples 4.54, and use the random variable that arises from assigning to each output the number of heads that appear then we get the pmf as described in that example,

0	1	2	3
$1/8$	$3/8$	$3/8$	$1/8.$

The following is a version of Proposition 4.2 for random variables with finite range. It says that if we have a pmf for a random variable then to know the probability distribution for that random variable we merely need to know the probabilities for each of the values in that range.

Corollary 4.13

Let X be a random variable with finite range, say T, and pmf p. Then there is a unique probability space $(T, \mathcal{P}T, P)$ with the property that for all elements $t \in T$ we have

$$P\{t\} = pt.$$

For this space we may calculate for all subsets E of T that

$$PE = \sum_{t \in E} pt.$$

Proof. This is an application of Proposition 4.2.

What this means is that if we have a probability mass function then we have a uniquely determined probability space, and so for a random variable with finite range all we need to understand the situation is the pmf. For this reason some people call a probability mass function a probability distribution.

In Section 4.2.4 the idea of a cumulative probability distribution is introduced. At this point we are ready to define that concept generally.

Definition 40: cumulative distribution function

Given a random variable X the **cumulative distribution function (cdf) for** X is the function

$$\mathbb{R} \longrightarrow [0,1]$$

which assigns, for $t \in \mathbb{R}$,

$$t \longmapsto P(X \leq t.)$$

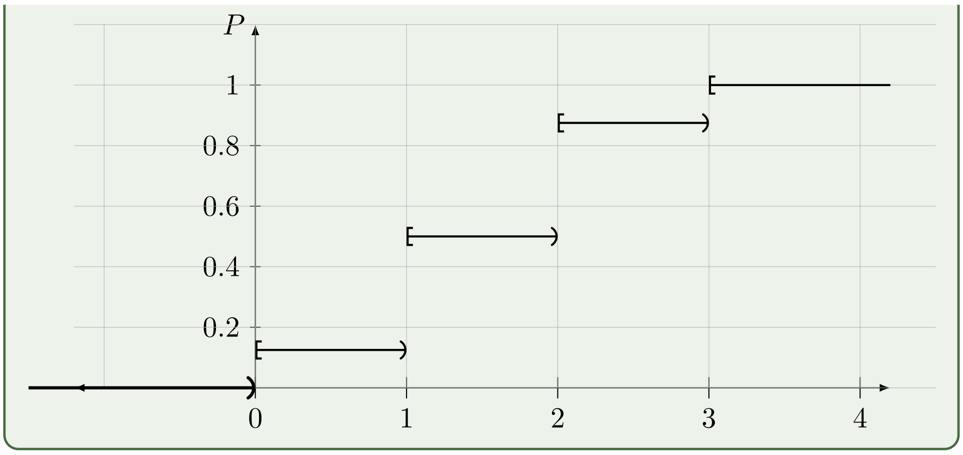
We are using here the fact that the real numbers are ordered and so it makes sense to ask for the probability that the random variable is at most some given number. In particular we can meaningfully draw the graph of this function and visualize the probability distribution in a way that we only do when the outcomes are given as numbers.

When we have a random variable which can take a finite number of values we have to draw a non-continuous function, and you may find this a bit odd at first. Look at the following example to see how that works.

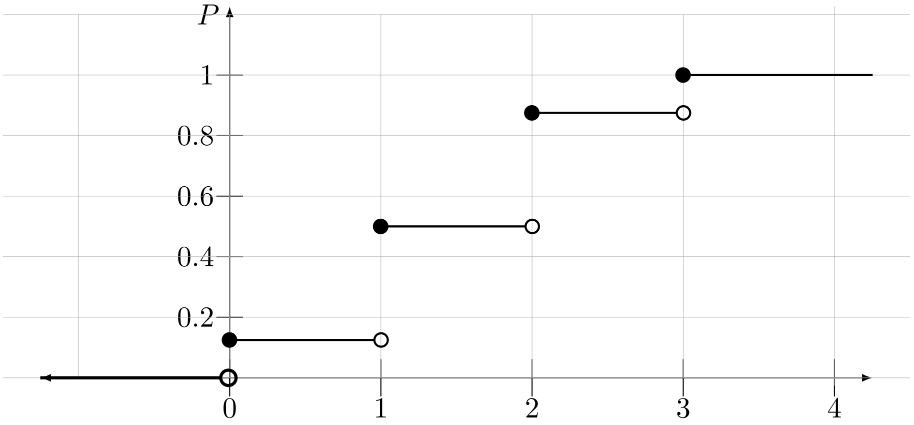
Example 4.69. If we look at the situation from Example 4.68 where the pmf is described in the table

0	1	2	3
1/8	3/8	3/8	1/8.

then the corresponding cdf can be drawn as follows.



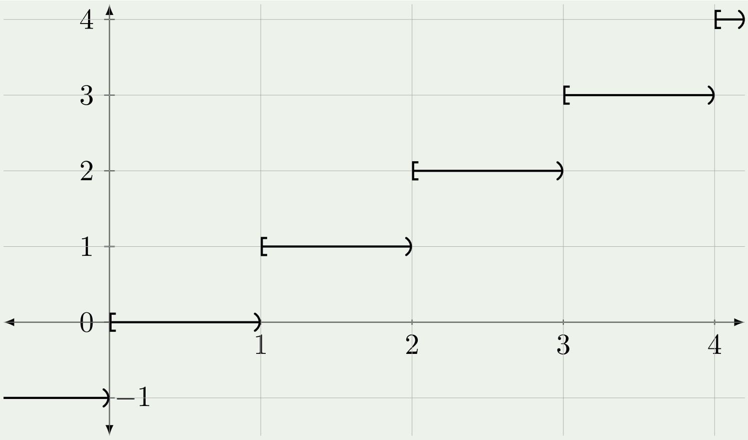
Note that when drawing discontinuous functions like the above we have to specify what the value at at a discontinuity is, the lower or the upper of the two lines. The convention used in the picture above is to use the interval notation, so that [and] mean that the point at the end of the line belongs, and (as well as) mean that it doesn't. An alternative way of drawing the same function is to use the following convention:



In the picture above the filled circle indicates that the endpoint of the line is included, and the unfilled circle that it is excluded.

In both pictures we can see that the functions jumps to a higher accumulated probability as the next possible value of the random variable is reached. The probability of fewer than 0 heads is 0, the probability of getting at least 0, but fewer than 1 heads is $1/8$, and so on.

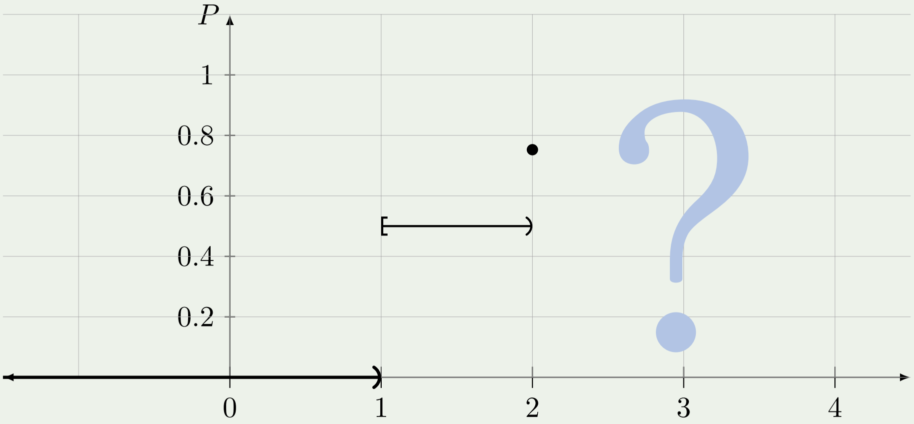
Example 4.70. If we want to draw the graph of the floor function $\lfloor\ \rfloor\colon \mathbb{R} \longrightarrow \mathbb{N}$, see page 42, we need this idea as well.



Example 4.71. We return to Example 4.57, where the information given is that X is a random variable and the following is known about its probability distribution is the following:

- $P(X = 1) = 1/2$,
- $P(X = 2) = 1/4$, and
- $P(X \geq 2) = 1/2$.

This is sufficient to be able to draw some of the cdf, but there is uncertainty:



We know that the probability is 0 until the value 1 is reached, and that it rises to .5 at that point, and rising further to .75 from 2. What we don't know is when it takes on the value 1 or which values it take between .75 and 1.

Example 4.72. An example for the continuous case is given in Section 4.2.4 in the form of Examples 4.27 and 4.28.

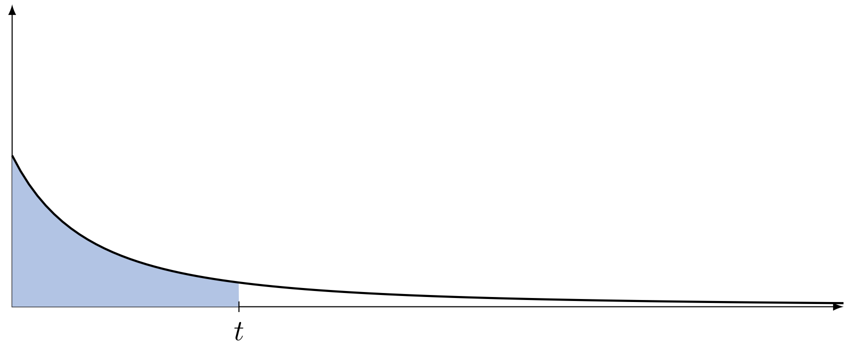
Recall that in the case of a continuous random variable X with range contained in an interval $I \subseteq \mathbb{R}$ the probability distribution is given in the form of a probability density function, say

$$g\colon I \longrightarrow \mathbb{R}^+.$$

There are two cases.

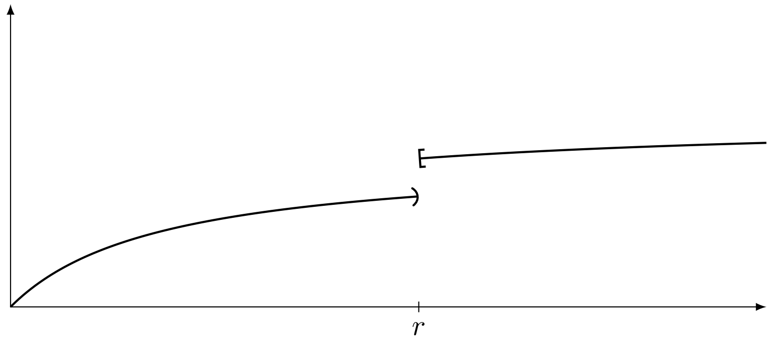
- If the interval is of the form $(-\infty, r')$, $(-\infty, r']$ or $\mathbb{R}$ then the cdf for X is given by
$$P(X \leq t) = \begin{cases} \displaystyle\int_{-\infty}^{t} gxdx & t \leq r' \\ 1 & \text{else.} \end{cases}$$
- If the interval is of the form (r, r'), $(r, r']$, (r, ∞) $[r, \infty)$ then the cdf for X is given by
$$P(X \leq t) = \begin{cases} 0 & t \leq r \\ \displaystyle\int_{r}^{t} gxdx & r \leq t \leq r' \\ 1 & \text{else.} \end{cases}$$

  In the case below I is $[0, \infty)$, and we calculate the probability that X is below t.



Note that the *derivative* of a cumulative distribution function is the corresponding probability density function (which in the discrete case is the corresponding probability mass function). We have no time to discuss here exactly how the derivative is formed in the discrete case.

However there is something that is easy to see. Assume you have a random variable whose cdf F makes a jump as in the following picture.



Then it has to be the case that the probability of the random variable at the point where the jump is concerned is the difference of the two values, that is
$$P(X = r) = F(r) - \lim_{n \to \infty} F(r - \frac{1}{n}).$$

Proposition 4.14

Let X be a random variable. If P is its cumulative distribution function then its derivative is the corresponding probability density (mass density) function.

CExercise 110. For Exercise 83 consider the random variable given by the number of eyes the die shows. Give its pmf, and draw a graph for its cdf.

Then do the same with the random variable $f \circ Y$ from Exercise 109.

EExercise 111. Assume that teams are regularly playing in a 'best out of five' series against each other, compare Exercise 81. We assume here that the winner is determined via a random process.

We are interested in the random variable given by the number of matches Team A wins in a given series.

(a) Describe a probability space that describes a 'best out of five' series. For the probabilities assume that the two teams have an equal probability of winning any match.

(b) Describe the function that underlies this random variable by writing down a mathematical function that carries out the required assignment.

(c) For the case where team A is equally matched by Team B, give the pmf and draw a graph for its cdf.

(d) Now assume that it is known that A wins the first match. We can now look at the random variable X conditional on this event. Describe the pmf and cdf for the resulting random variable. *Hint: Because the event A is part of the original probability space, but cannot be formulated for the outcomes of the random variable X, you cannot use the usual formula for conditional probabilities but have to analyse each case anew. If you are finding this part hard then looking ahead to Example 4.75 may help.*

Exercise 112. Carry out the same tasks as for the previous exercise for a 'best out of seven' series.

4.4.5 Conditional probabilities for random variables

Recall that there is no example for conditional probabilities in the continuous case in Section 4.3 above. The reason for this is that describing the probability density function for the general case, where we may make no assumptions about the possible outcomes, requires mathematical techniques beyond this course unit.

However, this is feasible once we restrict ourselves to random variables, where we know that the outcomes are elements of $\mathbb{R}$. We revisit the idea of conditional probabilities, now confined to random variables. You can see below that in that case the definition for the continuous case is the same as that for the discrete one.

The conditional probability density function

Recall that given two events A and B, where $PB \neq 0$, the conditional probability of A given B is defined as

$$P(A \mid B) = \frac{P(A \cap B)}{PB}.$$

If X is a random variable with probability distribution function F then we may define the conditional distribution of X given the event B (where we still assume $PB \neq 0$) as

$$P(X \leq r \mid B) = \frac{P((X \leq r) \cap B)}{PB}.$$

There is a conditional probability density function, which is once again the derivative of the corresponding distribution. The probability that the conditionally distributed random variable falls into a given interval is then the integral over that derivative over the given integral.

If X is a discrete random variable with pmf p then given an event B with $PB \neq 0$ we can calculate the pmf q of the random variable

$$(X \mid B), \qquad X \quad \text{given} \quad B,$$

by setting, for r in the range of X,

$$qr = \begin{cases} \frac{P(X=r)}{PB} & r \in B \\ 0 & \text{else.} \end{cases}$$

In other words, if we know that B happens, and r is a possible result of X not in B, then it has the probability 0, and otherwise the probability is adjusted by dividing through PB as expected.

Example 4.73. We return to Example 4.54. The pmf for the random variable X which gives the number of heads when a coin is tossed three times is as follows.

0	1	2	3
$1/8$	$3/8$	$3/8$	$1/8$.

Let the event A be that the result heads occurs at least once among the three tosses. The probability of A is $PA = 7/8$. We may calculate

$$P((X = i) \mid A) = \frac{P((X = i) \cap A)}{P(A)},$$

where

$$P((X = i) \cap A) = \begin{cases} P(X = i) & i \in A \\ 0 & \text{else.} \end{cases}$$

Hence the pmf of

$P(X \mid A)$ is

0	1	2	3
0	$3/7$	$3/7$	$1/7$.

If B is the event that the number of heads is even then $PB = 1/2$ and the pmf of

$P(X \mid B)$ is

0	1	2	3
1/4	0	3/4	0.

We look at the continuous case. Let X be a random variable with range $\mathbb{R}$ and probability distribution f, let r be in $\mathbb{R}$, and assume that B is the event

$$B = (X \leq r).$$

We may calculate

$$PB = P(X \leq r) = \int_{-\infty}^{r} fx dx.$$

We might then wonder how to calculate, for $s \in \mathbb{R}$,

$$P(X \leq s \mid B).$$

What we do know is that if we have a probability density function g for the resulting random variable we can calculate this probability as

$$\int_{-\infty}^{s} gx dx.$$

We can work out what the probability density function, say g, should do: If the argument is not in B it should return 0, and otherwise I should return the probability of x adjusted by the probability of B. Assuming that the probability of B is non-zero, g is given by

$$\begin{array}{rcl} g\colon \mathbb{R} & \longrightarrow & \mathbb{R}^+ \\ x & \longmapsto & \begin{cases} \dfrac{fx}{\int_{-\infty}^{r} fx dx} & x \leq r \\ 0 & \text{else.} \end{cases} \end{array}$$

In the general case, where we make no assumptions about the shape of B, we merely assume that the probability of B is not zero. The probability distribution g of the random variable

$$Y = (X \mid B)$$

is given by

$$\begin{array}{rcl} g\colon \mathbb{R} & \longrightarrow & \mathbb{R}^+ \\ x & \longmapsto & \begin{cases} \dfrac{fx}{PB} & x \in B \\ 0 & \text{else.} \end{cases} \end{array}$$

Note that the range of Y is included in B.

Example 4.74. If we return to Example 4.28 we have a random variable given by the time until the geyser next erupts. The probability density function is

$$\begin{array}{rcl} f\colon [0,90] & \longrightarrow & [0,1] \\ x & \longmapsto & \dfrac{1}{90}. \end{array}$$

Consider the event B that the geyser hasn't erupted in the 30 minutes we've already waited for it. We may calculate the probability of B occurring by calculating the probability that the geyser does erupt in the first 30 minutes, and deducting that from one. The probability that the geyser erupts between minute 0 and 30 is

$$\int_0^{30} \frac{1}{90} dx = \left[\frac{x}{90}\right]_0^{30} = \frac{30}{90} - 0 = \frac{1}{3},$$

so

$$PB = 1 - \frac{1}{3} = \frac{2}{3}.$$

The probability density function g of the random variable

$$(X \mid B)$$

is then given by

$$\begin{aligned} g\colon [0,90] &\longrightarrow [0,1] \\ x &\longmapsto \begin{cases} 0 & 0 \le x \le 30 \\ \frac{1}{60} & \text{else.} \end{cases} \end{aligned}$$

The examples we have considered here only work if the event on which we are conditioning can be expressed in terms of outcomes of the random variable in question. Sometimes we wish to condition on an event that can only be formulated in the original probability space, see Exercise 111 for an example. In that case the various conditional probabilities have to be calculated more painstakingly since we cannot apply the formulae derived above. We return to this idea in Section 4.4.6 after considering one more example.

Example 4.75. We return to the random variable X which counts the number of heads when tossing a coin three times, see Example 4.54, and contrast with Example 4.73. The pmf of X is given by the following table.

0	1	2	3
$1/8$	$3/8$	$3/8$	$1/8$.

Assume we wish to condition this random variable on the event C that the first toss is heads. The given pmf does not help us in calculating the pmf of $(X \mid C)$. Instead we have to start over from the original probability space. We analyse the possible values of X and the probabilities with which they occur. Assume the first toss is heads. The possible numbers of heads among the three tosses are as follows.

- 0. This cannot occur.
- 1. This means the toss must be HTT. This occurs with probability $1/4$.
- 2. This means the toss is HHT or HTH. This occurs with probability $1/2$.
- 3. This means the toss is HHH. This occurs with probability $1/4$.

Hence the pmf of

$P(X \mid C)$ is

0	1	2	3
0	$1/4$	$1/2$	$1/4.$

Note that it is also possible to perform Bayesian updating for random variables: In the case of a discrete random variable, the update procedure is just as described in Section 4.3.4. If the random variable is continuous then instead of updating the pmf by adjusting all the individual values we have to update the probability density function. Spelling out the resulting definition of the new probability density function goes beyond this course unit.

One random variable depending on another

The material in this subsection is not examinable. You may want to return to it if you ever have to cope with a situation where one random variable depends on another.

Recall Example 4.37, where we were wondering about how to describe the probability density function for the location or a fox whose behaviour is influenced by the location of a lynx (if the latter is close enough).

What we have there is one random process, describing the movements of the fox, conditional on another random process, namely the movement of the lynx.

We can only do this in the situation where we have a *joint distribution*, that is, a probability distribution, or a density function/pmf, that describes the combined probability.

It is then the case that if f is the joint density function for random variables X and Y, we can derive density functions for X and Y, namely

- The probability density function for X is[30]

$$\int_{-\infty}^{\infty} f(x,y)dy,$$

- while that for Y is

$$\int_{-\infty}^{\infty} f(x,y)dx.$$

In this situation we can look at the case of the density function g for X given $(Y = s)$, for some $s \in \mathbb{R}$. We get

$$gx = \frac{f(x,s)}{\int_{-\infty}^{\infty} f(x,s)dy}.$$

If instead we are interested in the probability distribution for X given

$$(s \leq Y \leq s'),$$

we have

$$P(X = r \mid s \leq Y \leq s') = \frac{\int_{-\infty}^{r} \left(\int_{s}^{s'} f(x,y)dy\right) dx}{\int_{s}^{s'} \left(\int_{-\infty}^{\infty} f(x,y)dx\right) dy}.$$

If X and Y are discrete random variables then we can look at their joint pmf. This is a function that, given

[30] You can calculate with these integrals by treating the other variable as if it were a parameter, that is, you integrate the first expression for y and treat x as if it was a number. You swap the treatment of the two variables for the second expression.

- a value r from the range of X and
- a value s from the range of Y,

returns the probability $P(X = r \text{ and } Y = s)$.

Example 4.76. We return to the example of tossing a coin three times, see Example 4.54. The pmf of the random variable X, which counts the number of heads, is

0	1	2	3
$1/8$	$3/8$	$3/8$	$1/8$.

The random variable Y, which records the absolute of the difference between the number of heads and tails, has a pmf given by the following table.

1	3
$6/8$	$2/8$.

The joint pmf of X and Y is given by the following table.

$Y\backslash X$	0	1	2	3
1	0	$3/8$	$3/8$	0
3	$1/8$	0	0	$1/8$

Independent random variables

When we have two random variables which are independent from each other it becomes easier to calculate with both.

Definition 41: independent random variables

Two random variables X and Y are *independent* if and only if it is the case that for all elements of the Borel σ algebra E and E' we have that

$$P(X \in E \text{ and } Y \in E') = P(X \in E) \cdot P(Y \in E').$$

In particular this means that

- if X is a random variable with density function f and
- Y is a random variable with density function g then

the joint density function for X and Y is given by

$$(x, y) \longmapsto fx \cdot gy.$$

We need this information when we wish to look at situation where we have several random variables, for example the failure of a number of pieces of equipment. This is easier if we assume that the failure of one is independent from the failure of the others but this assumption is only justified if we can exclude factors that would affect more than one piece of equipment, such as a power surge at some location.

Example 4.77. We have already seen an example of this. When we look at the random variable X which gives us the number of eyes shown by the red die, and the random variable Y which gives us the number of eyes shown by the blue die, then their joint pmf is given as follows.

$i\backslash j$	1	2	3	4	5	6
1	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$
2	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$
3	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$
4	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$
5	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$
6	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$

Exercise 113. Are the two random variables X and Y from Example 4.76 independent? Justify your answer.

EExercise 114. Assume that you are tasked by your boss with making sure that you have enough servers that the probability of no server being currently online is at most 1% for the entire year. Because you are able to place your servers at separate locations you are allowed to assume that one server failing will have no effect on the other servers.

(a) Assume that the chance of one of your servers failing in a given year is $.05$. How many servers do you need to comply with your boss's demand? How much safety do you get out of an extra server?

(b) Assume that the probability of one of your servers failing has the probability density function[31] f given by

$$\begin{array}{rcl} f\colon [0,365] & \longrightarrow & [0,1] \\ x & \longmapsto & \dfrac{x^2}{2\cdot 365^3}, \end{array}$$

which we need to consider from $x=0$ to $x=365$ to cover the year. In other words, the probability that the server will have failed by the end of the year is given by the integral, from 0 to 365, over the given density function. How many servers do you have to buy and install to comply with the specification you were given?

4.4.6 Expected value and standard deviation

One of the motivations for introducing the notion of random variables is the ability to form averages.

Expected value

Example 4.78. Returning to the example of the number of heads when tossing a coin three times, Example 4.68, you may wonder what the average number of heads might be. This case is so simple that you can probably guess the answer, but in more complicated situations you will want to carry out analogous calculations. If we weigh each possible outcome by its probability then this

[31] I'm not claiming this is a realistic density function, but hopefully it's not too bad to calculate with.

number is given by

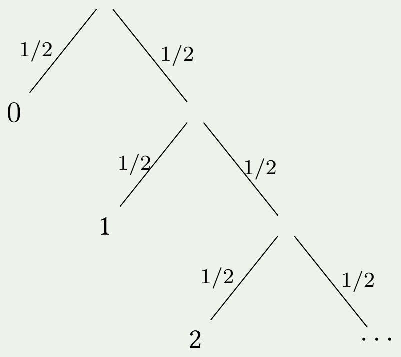
$$0 \cdot \frac{1}{8} + 1 \cdot \frac{3}{8} + 2 \cdot \frac{3}{8} + 3 \cdot \frac{1}{8} = \frac{0+3+6+3}{8} = \frac{12}{8} = \frac{3}{2},$$

so on average the number of heads is 1.5, which in this simple case you may have been able to guess. Note that if you wanted to bet on the outcome of this experiment then it does not make sense to bet the expected value since it cannot occur.

We look at more interesting examples. Note that solving the following two examples require knowledge beyond this course unit—it is included here to give you an idea of how powerful the idea is.

Example 4.79. Assume that we have strings which are generated in a random way, in that after each key stroke, with a probability of $1/2$, another symbol is added to the string. We would like to calculate the average length of the strings so created. Before we can do this we have to specify when the random decision starts: Are all strings non-empty, or is there a chance that no symbol is ever added? We go for the latter case, but the calculation for the former is very similar.

As is often the case when picturing a step-wise process we can draw a tree that describes the situation. At each stage there is the random decision whether another symbol should be added or not. We give the length of each generated string.



What this means is that we have a random variable, namely the length of the generated string, and we can see that its probability mass function has the first few values given by the following table.

0	1	2	3	4
$1/2$	$1/4$	$1/8$	$1/16$	$1/32$

More precisely the pmf is given by the function

$$\begin{aligned} \mathbb{N} &\longrightarrow \mathbb{R} \\ n &\longmapsto \frac{1}{2^{n+1}}. \end{aligned}$$

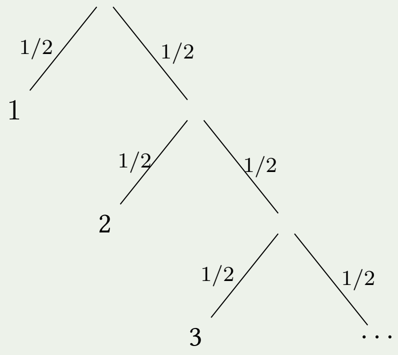
What is the average string length? The idea is that we should give each possible length the probability that it occurs. This means that we should calculate

$$0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{4} + 2 \cdot \frac{1}{8} + \cdots = \sum_{n \in \mathbb{N}} n \cdot \frac{1}{2^{n+1}}.$$

With a bit more mathematics than we can teach on this unit it may be calculated[32] that this required number is 1. So certainly when producing strings in this way we don't have to worry about there being a lot of long ones! But note that we have described a process for producing potentially infinite strings (with a probability of 0), and the power of the methods we use here is such that we can still calculate the average.

We say more about how to cope with situations where we have to compute an infinite sum in Section 4.4.6.

Example 4.80. Assume we are tossing a coin until we get heads for the first time, and then we stop (compare Exercise 85 and Example 4.26. We wonder what the average number of coin tosses is. Again it makes sense to draw a tree.



This is quite similar to the previous example! The pmf for this random variable is given by

$$\begin{array}{rcl} \mathbb{N} & \longrightarrow & \mathbb{R} \\ n & \longmapsto & \dfrac{1}{2^n}. \end{array}$$

The expected value is

$$\sum_{n \in \mathbb{N}} n \cdot \frac{1}{2^n} = 2.$$

Again calculating such expected values is not part of this unit, but it gives you one motivation why mathematicians care about what happens if infinitely many numbers are added up.

We say more about how to cope with situations where we have to compute an infinite sum in Section 4.4.6, in particular Example 4.89 is relevant.

What is it that we have calculated in these examples?

[32] In mathematical parlance, we have defined a series whose limit is 1.

Definition 42: expected value

Let X be a random variable with probability density function p. Then the **expected value of** X, $E(X)$, is given by

$$E(X) = \int_{-\infty}^{\infty} x \cdot p(x) dx.$$

Note that this definition does allow for the possibility that $E(X)$ is infinite. This can never occur if X is a discrete random variable with a finite range, but in the other cases this is a possibility. Calculating with infinities is beyond the scope of this unit, and all the examples we study give a finite result.

Note that if X is a discrete random variable with range

$$\{r_i \mid i \in \mathbb{N}\},$$

then its expected value is

$$E(X) = \sum_{i \in \mathbb{N}} r_i \cdot P(X = r_i).$$

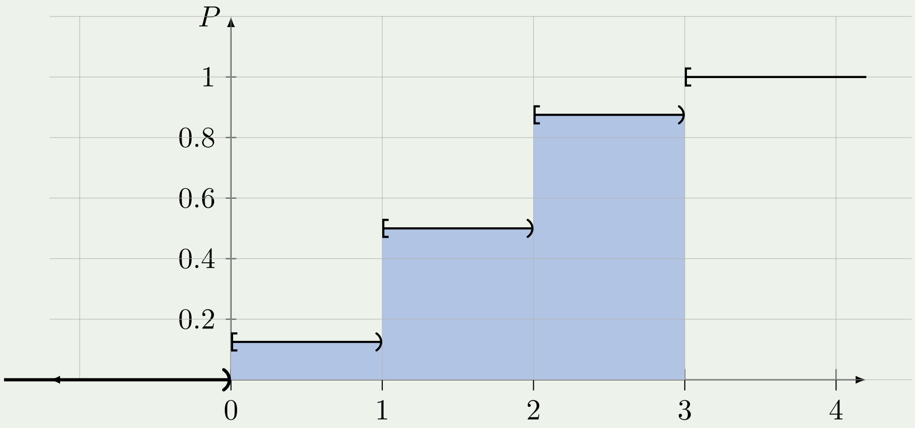
This means that if X is a discrete random variable with finite range

$$\{r_1, r_2, \ldots, r_n\},$$

then its expected value is

$$\begin{aligned} E(X) &= r_1 P(X = r_1) + r_2 P(X = r_2) + \cdots + r_n P(X = r_n) \\ &= \sum_{i=1}^{n} r_i P(X = r_i). \end{aligned}$$

Example 4.81. In Example 4.78 the expected number of heads when tossing a coin three times is calculated as being 1.5. In Example 4.69 the cumulative distribution for that random variable is drawn:



The area under the function from 0 to 3, shown in blue above, is given by

$$.125 \cdot 1 + .5 \cdot 1 + .875 \cdot 1 = 1.5,$$

which is the same as the expected value. In general this is always the connection between the expected value and the area under the cdf, and this is the best indication I can give that this area (and so an integral) has something to do with probabilities.

Note that in the discrete case, the expected value need not be in the range of X. In Example 4.78 the expected value is 1.5 heads in 3 tosses of a coin, which clearly is not a valid result of tossing a coin three times.

Further note that even if the expected value is a possible outcome it need not in itself be particularly likely.

Example 4.82. Assume we are playing a game with a deck consisting of four aces and the kings of spaces and hearts,

$$\{A\clubsuit, A\spadesuit, A\heartsuit, A\diamondsuit, K\spadesuit, K\heartsuit\}.$$

We each draw a card from the pack. If one of us has an ace and the other a king, the holder of the ace gets two pence from the other player. If we both have an ace, then if one of us has a black ace $A\clubsuit$ or $A\spadesuit$ then he gets three pence from the other player. If we have aces of the same colour neither of us gets anything. If both of us have a king then the holder of the black king gets one penny from the other player.

We look at the random variable formed by the number of pence gained or lost by one of the players (since the rules are symmetric it does not matter which player we pick). Its range and pmf are given in the following table.

-3	-2	-1	0	1	2	3
$2/15$	$4/15$	$1/30$	$2/15$	$1/30$	$4/15$	$2/15$

We calculate the expected pay-off. It is

$$\frac{1}{30}(-3\cdot 4+(-2)\cdot 8+(-1)\cdot 1+0\cdot 4+1\cdot 1+2\cdot 8+3\cdot 4)=0.$$

We could have saved us this calculation by making the following deductions: The game is completely symmetric, and wins for one player are paid for by the other.[33] So if one player were to expect a gain the other player would have to expect a loss to make up for that game, but the rules are exactly the same for both.

We note that the expected value 0 does not occur with a particularly high probability.

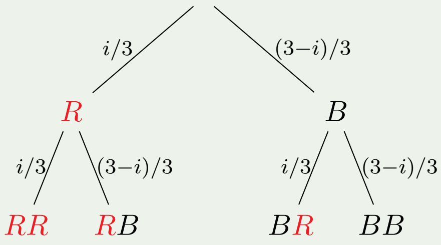
Also note that the expected value does not have to be halfway between the extremes of the possible outcomes. This is illustrated (among other things) in the following example, where we calculate the average of an average to show that it is possible to have several layers of random variables, which still allow us to calculate an overall expected value.

Example 4.83. For a more down to earth example let us revisit Example 4.42. There we are faced with 4 possibilities regarding which situation we are in (given by the number of red socks in the bag). This gives us an opportunity to look at an expected value for different probability distributions.

Here is a tree that describes the drawing of two socks (with replacement)

[33] This is a *zero-sum game* in the parlance of game theory.

from a bag that contains i red socks from a total of 3 socks.



We have here a random variable which maps the outcomes from this tree to the number of red socks drawn. Hence it maps RR to 2, the outcomes RB and BR to 1 and the outcome BB to 0. The pmf of this random variable is

2	1	0
$\frac{i^2}{9}$	$2\frac{i(3-i)}{9}$	$\frac{(3-i)^2}{9}$

Hence the expected value for the number of socks is

$$2\frac{i^2}{9} + 2\frac{i(3-i)}{9} = \frac{2i^2 + 6i - 2i^2}{9} = \frac{6i}{9}.$$

So the expected value in each case is

i	0	1	2	3
$E(X)$	0	$2/3$	$4/3$	2

Note how the expected value varies with the underlying situation, and note that in none of the cases we get as the expected value the halfway point between the two extremes 0 and 2.

We can use these expected values to calculate an *overall expected value* based on our current estimate for the true probability distribution.

At the beginning, the probability of the possible outcomes,

$$\{0, 1, 2, 3\}$$

is equal, $1/4$ for each. If we draw two socks (returning the sock to the bag after each draw) then we would expect to draw one red and one black sock on average.

After the first update the pmf is

0	1	2	3
0	$1/6$	$1/3$	$1/2$

If we want the expected value based on our current knowledge, which is given by the current distribution, then we should form an average where each of the previously calculated expected values is weighted by the probability that we think it's the correct one, giving an overall expected value of

$$0 \cdot 0 + \frac{2}{3} \cdot \frac{1}{6} + \frac{4}{3} \cdot \frac{1}{3} + 2 \cdot \frac{1}{2} = \frac{2+8+18}{18} = \frac{28}{18} = 1.\overline{5}.$$

Example 4.84. For a simple continuous example we return to the random variable that describes the amount of time until a geyser erupts from Examples 4.28 and 4.74. The expected time we have to wait until the geyser erupts is

$$\int_0^{90} \frac{x}{90} dx = \left[\frac{x^2}{2 \cdot 90} \right]_0^{90} = \frac{90^2}{2 \cdot 90} - 0 = 45,$$

which tells us that we have to wait 45 minutes on average as expected.

Whenever we calculate an expected value we calculate a *probability-weighted average*, that is, we try to give some kind of number that occurs 'on average'. We should be careful when we use such calculations to make decisions—for example, the expected pay-off of playing some game being positive is by itself not a good enough reason to play that game. We've assigned numbers to certain outcomes, but these numbers might not adequately reflect our valuation of the situation.

Example 4.85. Assume somebody offers you a game: You toss a coin. If it gives heads, you pay a million pounds, if it's tails, you get a million and one pounds. The expected value of this game is $50p$ for you, but can you afford to lose this game?

Whenever we use expected values to give an assessment of risk, apart from making sure we have our probabilities right we should carefully check whether the numbers of the given random variable truly reflect how we judge the relevant outcomes.

Exercise 115. For the expected value given at the end of the previous example, what is the underlying random variable? Give its range and its pmf.

CExercise 116. You are invited to play the following game: There are three cards:

- One is black on both sides,
- one is red on both sides,
- one is black on one side and red on the other.

You and another person pay one pound each into a kitty. The three cards are put into a bag and mixed together. Without looking into a bag you draw a card. You pull it out of the bag in a way that only the upper side can be seen, and you place it on the table. The card is red on the side you can see.

The other player bets that the card has the same colour on the hidden side as is showing. You're unsure whether you should bet on it having a different colour on the other side. The other player points out that it can't be the card that is black on both sides, so you have a 50-50 chance.

The winner of the bet is to get the two pounds put into the kitty at the start. Should you accept this as a fair game, or should you ask for your pound back? Answer this question by calculating the expected value of the amount you have to pay.

Using conditioning to calculate expected values

Recall Example 4.80 where we determined the expected number of coin tosses until we get heads for the first time. If we use the definition of the expected value then we have to calculate with an infinite sum to find that number.

We can use conditional probabilities to help with this situation, see Section 4.4.5 for a general account of the probability distribution of a random variable conditioned on an event. In this section we are concerned with how to calculate the expected value. of such a random variable.

We first look at the general case. Let X be a random variable with probability density function f and let B be an event with non-zero probability which is a subset of $\mathbb{R}$. Then

$$E(X \mid B) = \int_B \frac{1}{PB} \cdot x \cdot fx dx = \frac{1}{PB} \int_B x \cdot fx dx,$$

Note that the integral looks similar to the integral defining the expected value of X, but we cannot use one to calculate the other since the areas over which we integrate differ.

Example 4.86. In Example 4.84 we calculate the expected value of the time we have to wait when we visit the geyser from Example 4.28, which is 45 minutes. In Example 4.74 we give a probability distribution conditioned on the event B that the geyser has not erupted in the last thirty minutes. Applying the ideas above we calculate the expected value of

$$(X \mid B),$$

using the probability density function g calculated in Example 4.74, which is

$$\begin{aligned} g\colon [0,90] &\longrightarrow [0,1] \\ x &\longmapsto \begin{cases} 0 & 0 \leq x \leq 30 \\ \frac{1}{60} & \text{else.} \end{cases} \end{aligned}$$

We have

$$\begin{aligned} E(X \mid B) &= \int_{-\infty}^{\infty} gy dy = \int_{30}^{90} \frac{x}{60} dx = \left[\frac{x^2}{2 \cdot 60} \right]_{30}^{90} \\ &= \frac{1}{2 \cdot 60}(90^2 - 30^2) = \frac{1}{120} \cdot 7200 = 60. \end{aligned}$$

This may strike you as unexpected: When we arrived we thought we might have to wait 45 minutes, but knowing that the geyser has not erupted for 30 minutes so far means that if we look at the conditional random variable we have to adjust our expectations to be rather more pessimistic!

In the discrete case the expected value can be expressed as follows. Let the range of X be given by

$$\{r_i \mid i \in \mathbb{N}\},$$

and let B be an event with non-zero probability. Then

$$E(X \mid B) = \frac{1}{PB} \sum_{i \in \mathbb{N}, r_i \in B} r_i P(X = r_i).$$

If we further reduce this to the case where the range of X is finite, then we may calculate the elements of the finite set

$$\{r \in B \mid r \text{ is in the range of X}\}, \qquad \text{say} \qquad \{r_1, r_2, \dots, r_n\}$$

and then we have

$$E(X \mid B) = \frac{1}{PB}(r_1 P(X = r_1) + r_2 P(X = r_2) + \dots + r_n P(X = r_n)).$$

Example 4.87. Recall the random variable X from Example 4.54 of the number of heads when tossing a coin three times. We calculate its expected value as 1.5 in Example 4.87 and we calculate its conditional pmf for the event A that there is at least one head in Example 4.73. The expected value of

$$(X \mid A),$$

whose pmf is (as per Example 4.73)

0	1	2	3
0	3/7	3/7	1/7.

is given by

$$E(X \mid A) = \frac{1}{7}(1 \cdot 3 + 2 \cdot 3 + 3 \cdot 1) = \frac{12}{7}.$$

Alternatively we can use the formula from above to carry out this calculation based on the pmf of X, which is given by

0	1	2	3
1/8	3/8	3/8	1/8.

The calculation then is

$$E(X \mid A) = \frac{8}{7} \cdot \frac{1}{8}(1 \cdot 3 + 2 \cdot 3 + 3 \cdot 1) = \frac{12}{7}.$$

Exercise 117. For the random variable of tossing a coin three times, and the event B from Example 4.73 calculate the expected value of the random variable $Y = (X \mid B)$.

Note that above we assume that the event on which we are conditioning is an event that can be formulated regarding the outcomes of the given random variable X. What happens quite frequently is that one wishes to condition on an event that can only be formulated in the original probability space. In those cases there is no way of applying the formulae derived above.

Example 4.88. Consider the random variable from Example 4.54 where we toss a coin three times. We might wish to condition over the event C that the first toss is heads. This is not something we can formulate by only referring to outcomes of this random variable. The pmf for the random variable $(X \mid C)$ is carried out in Example 4.75, where it is given as follows.

0	1	2	3
0	1/4	1/2	1/4

With the help of that pmf we can calculate the expected value

$$E(X \mid C) = \frac{1}{4}(0 \cdot 0 + 1 \cdot 1 + 2 \cdot 2 + 3 \cdot 1) = \frac{8}{4} = 2,$$

but we cannot calculate this expected value from the expected value of X.

There is a useful technique for calculating expected values of random variables when it is easier to calculate expected values for conditioned versions of that random variable. Applications of the following result follow below.

Proposition 4.15

Let X be a random variable over the probability space $(S, \mathcal{E}.P)$ and assume that we have pairwise disjoint events B_1, B_2, $\ldots B_n$ such that

$$S \subseteq B_1 \cup B_2 \cup \cdots \cup B_n.$$

Then

$$EX = E(X \mid B_1) \cdot PB_1 + E(X \mid B_2) \cdot PB_2 + \cdots + E(X \mid B_n) \cdot PB_n.$$

Proof. Proving the general case goes beyond what we cover on this course unit. For the discrete case, let us assume that the range of X is given by

$$\{r_i \in \mathbb{R} \mid i \in \mathbb{N}\}.$$

Then

$$\begin{aligned}
&EX \\
&= \sum_{i\in\mathbb{N}} r_i P(X = r_i) && \text{def } EX \\
&= \sum_{i\in\mathbb{N}} r_i \left(P(X = r_1 \mid B_1)PB_1 + P(X = r_i \mid B_2)PB_2\right. \\
&\qquad \left. + \cdots + P(X = r_i \mid B_n)PB_n\right) && \text{tot prob} \\
&= \sum_{i\in\mathbb{N}} r_i P(X = r_1 \mid B_1)PB_1 + \sum_{i\in\mathbb{N}} r_i P(X = r_i \mid B_2)PB_2 \\
&\qquad + \cdots + \sum_{i\in\mathbb{N}} r_i P(X = r_i \mid B_n)PB_n \\
&= PB_1 \sum_{i\in\mathbb{N}} r_i P(X = r_1 \mid B_1) + PB_2 \sum_{i\in\mathbb{N}} r_i P(X = r_i \mid B_2) + \\
&\qquad \cdots + PB_n \sum_{i\in\mathbb{N}} r_i P(X = r_i \mid B_n) \\
&= E(X \mid B_1)PB_1 + E(X \mid B_2)PB_2 + \cdots + E(X \mid B_n)PB_n.
\end{aligned}$$

This completes the proof.

We show how to use this idea to calculate expected values of a random variable conditioned over an event of the original probability space.

Example 4.89. We are interested in the random variable X which gives the number of tosses of a coin until we see heads for the first time. We would like to calculate its expected value EX. Note that in Example 4.80 we required an infinite sum to find that value. Here we give an alternative method that does works without infinite sums.

Since the probability of the first toss being heads is $1/2$, and since this is also the probability of the first toss being tails, we may use the proposition above to write

$$EX = \frac{1}{2}E(X \mid \text{first toss } H) + \frac{1}{2}E(X \mid \text{first toss } T).$$

We look at the two expressions on the right hand side.

- If the first toss is heads then we may stop tossing our coin, and so then the expected value of X, conditional on the first toss being heads, is 1.
- If the first toss is tails then it is as if we had not started to toss at all, and the expected value of X, conditional on that event, is one more than the expected value of X.

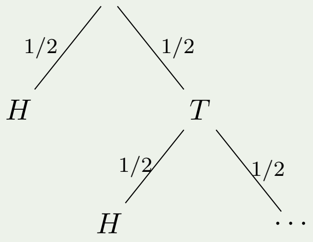
From these considerations we get

$$EX = \frac{1}{2}\cdot 1 + \frac{1}{2}(1 + E(X)) = 1 + \frac{1}{2}EX.$$

We can treat this as an equation in EX and solve it to give

$$EX = 2.$$

For an alternative way of looking at the situation we draw the appropriate tree.



we can see that below the node labelled T in the picture, we have another copy *of the same tree*. In other words, the tree branches to

- H, where it ends or
- T, below which another copy of the whole tree appears.[34]

Because a copy of the infinite tree appears within itself we can use the trick of establishing an equation for EX, where here we argue that the expected value, that is the 'average' number of tosses until the experiment ends, is given by

- with probability $1/2$, the first toss results in H and the experiment ends after 1 toss and

- with probability $1/2$, the first toss results in T, and then the expected number of additional tosses is the same as before, so the overall number of tosses is 1 added to the expected number of tosses.

This leads to the same equation as above, namely

$$EX = 1 + \frac{1}{2}EX.$$

In general we can often avoid having to calculate with infinite sums by using similar techniques. Assume we have a random experiment which has a particular result s with property p, and another result s' with property $1-p$ and that previous experiments have no effect on subsequent ones. We are interested in the expected value of the random variable X of how many times we have to repeat the experiment to get the second outcome we can see that we have

$$\begin{aligned}
EX &= (1-p)E(X \mid \text{first outcome } s') + p(E(X \mid \text{first outcome } s)) \\
&= (1-p) \cdot 1 + p(E(X \mid \text{first outcome } s)) \\
&= (1-p) + p(1 + EX) \\
&= 1 + pEX.
\end{aligned}$$

This means that in this situation we get that

$$EX = \frac{1}{1-p}.$$

Example 4.90. Assume we have a coin that shows head with probability p, and tails with probability $1-p$. Let X be the random variable of the number of coin tosses required until we see heads for the first time.

In Example 4.89 we calculate the expected value of X in the case of a fair coin. Here we want to establish that it is possible to condition on the two disjoint events, namely that the first toss gives heads, or that the first toss gives tails, and use those to express the expected value of X.

$$\begin{aligned}
EX &= \sum_{i \in \mathbb{N}} iP(X = i) && \text{def } EX \\
&= \sum_{i \in \mathbb{N}} i\,(P(X = i \mid \text{fst toss } H)P(\text{fst toss } H) \\
&\qquad + P(X = i \mid \text{fst toss } T)P(\text{fst toss } T)) && \text{law of tot prob} \\
&= \sum_{i \in \mathbb{N}} iP(X = i \mid \text{fst toss } H)P(\text{fst toss } H) \\
&\qquad + \sum_{i \in \mathbb{N}} iP(X = i \mid \text{fst toss } T)P(\text{fst toss } T) \\
&= E(X \mid \text{fst toss } H)P(\text{fst toss } H) \\
&\qquad + E(X \mid \text{fst toss } T)P(\text{fst toss } T) \\
&= E(X \mid \text{fst toss } H)p + E(X \mid \text{fst toss } T)(1-p).
\end{aligned}$$

[34] This can only work with infinite structures.

This idea generalizes to similar experiments with several outcomes. Assume there are n possible outcomes

$$s_1, s_2, \dots s_n$$

and that

- for $1 \leq i \leq n-1$ outcome s_i occurs with probability p_i and
- outcome s_n occurs with probability $1-(p_1+p_2+\cdots+p_{n-1})$.

Then the expected number of times we have to repeat the experiment to get outcome s_n has to satisfy the equation

$$\begin{aligned} EX &= 1-(p_1+p_2+\cdots+p_{n-1}) + p_1(1+E(X \mid \text{1st } s_1)) \\ &\quad + \cdots + p_{n-1}(1+E(X \mid \text{1st } s_n)) \\ &= 1+(p_1+p_2+\cdots+p_{n-1})EX \end{aligned}$$

and so we must have

$$EX = \frac{1}{1-(p_1+p_2+\cdots+p_{n-1})}.$$

EExercise 118. Assume you have a fair coin.

(a) What is the expected number of tosses until you have two heads in a row for the first time?

(b) What is the expected number of tosses until you have heads immediately followed by tails for the first time?

(c) Assume you are invited by one of your friends to play the following game: A coin is tossed unto either

- two heads occur in a row for the first time or
- we have heads immediately followed by tails for the first time.

In the first case you get 6 pounds and in the second case you have to pay the other player 5 pounds. Should you play this game?

Hint: Use the same idea as in Example 4.89. For the first part, check the situations you may find yourself in after two tosses.

Properties of expected values

We know from Proposition 4.12 that we may compose a random variable with a (measurable) function from its range to (a subset of) $\mathbb{R}$ and that gives another random variable. But in general there is no easy formula for the expected value in that situation:

Composing with a function will lead to a different probability density function, and forming the integral over that cannot in general be expressed in terms of the integral giving the expected value for the original random variable. Even if the given random variable is discrete we do not get a simple formula: Assume that f

is a measurable function from the range of a random variable X to a subset of $\mathbb{R}$. Then the new random variable has an expected value of

$$E(f \circ X) = \sum_{r \in \text{range}(f \circ X)} r \cdot P(f \circ X = i).$$

This indicates that there is no easy to calculate the expected value of $f \circ X$ from that of X. This situation only changes when f is a very simple function.

Exercise 119. Let X be a random variable; consider the following function:

$$\begin{aligned} f \colon \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto 1. \end{aligned}$$

Calculate the expected value of the random variable $f \circ X$.

If the function f is a linear function (compare Chapter 0) then we can compute the expected value of $f \circ X$ from that of X. Assume we have a discrete random variable X, with range

$$\{r_i \in \mathbb{R} \mid i \in \mathbb{N}`\}.$$

Let a and b be real numbers. We can compose X with the function

$$\begin{aligned} \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto ax + b. \end{aligned}$$

What is the expected value of the resulting random variable? We can calculate

$$\begin{aligned} E(aX + b) &= \sum_{i \in \mathbb{N}} (a \cdot r_i + b) \cdot P(aX + b = a \cdot r_i + b) \\ &= \sum_{i \in \mathbb{N}} (a \cdot r_i + b) \cdot P(X = r_i) \\ &= \sum_{i \in \mathbb{N}} a \cdot r_i \cdot P(X = r_i) + b \cdot P(X = r_i) \\ &= a \sum_{i \in \mathbb{N}} r_i \cdot P(X = r_i) + b \sum_{i \in \mathbb{N}} P(X = r_i) \\ &= a \cdot E(X) + b. \end{aligned}$$

See Exercise 106 for an explanation of the last step.

Proposition 4.16

Let X be a random variable, and let a and b be real numbers. Then the expected value of the random variable $aX + b$, which is formed by composing X with the function

$$\begin{aligned} \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto ax + b. \end{aligned}$$

has an expected value given by

$$E(aX + b) = aE(X) + b.$$

Proof. An argument for the discrete case is given above. The general argument proceeds as follows.

$$\begin{aligned} E(aX+b) &= \int_{-\infty}^{\infty} ax + b \cdot p(ax+b)dx \\ &= a\int_{-\infty}^{\infty} x \cdot p(x)dx + b\int_{-\infty}^{\infty} p(x)dx \\ &= aE(X) + b. \end{aligned}$$

If we have two random variables then we can say something about combining them.

Proposition 4.17

If X and Y are random variables then

$$E(X+Y) = EX + EY.$$

If X and Y are independent then we also have

$$E(X \cdot Y) = EX \cdot EY.$$

Example 4.91. This can be a very useful result when we want to calculate expected values. For example, if we want to calculate the expected number of heads when tossing a coin 20 times then carrying out a calculation where we look at all the possible permutations of results we might get is tough. So instead of doing that we can think of the random variable X thus created as being the sum

$$tsum_{1 \leq i \leq 20} X_i$$

where X_i is the random variable we get from the number of heads on the ith toss of the coin. For each X_i we have $EX_i = 12$, so

$$EX = E \sum_{1 \leq i \leq 20} X_i = \sum_{1 \leq i \leq 20} EX_i = \sum_{1 \leq i \leq 20} \frac{1}{2} = \frac{20}{2} = 10$$

and we have found an easy way to calculate this number. I assume many of you would have guessed this to be the expected value, but now we can be sure this answer is the correct one.

Exercise 120. For the situation where a 'best out of five' series is played carry out the following tasks.

(a) Calculate the expected value for the number of matches that occur in a 'best out of five' series, see Exercise 81.

(b) Calculate the number of matches A can expect to win, see 111.

(c) There is a connection between these two expected values. What is it, and can you explain why it has to be like that?

We are now able to paraphrase an important law that, for example, explains why Bayesian updating works. You will meet this idea again in COMP13212, Data Science, in one of the early lectures.

Fact 13: The Law of Large Numbers

Let X_i, for $i \in \mathbb{N}$, be pairwise independent random variables with the same distribution. Further assume that the expected value of the X_i is $v \in \mathbb{R}$, and that the random variables have a finite variance (see Definition 43). Then

$$\frac{X_1 + X_2 + \cdots + X_n}{n}$$

converges towards v with probability 1, as n tends towards infinity.

Example 4.92. Assume we are tossing a coin, and we use the random variable X_n (say 0 for heads and 1 for tails) to express the nth coin toss. The expected value of each of the random variables is $1/2$. Then if we keep tossing the coin we find that with probability 1 the average

$$\frac{X_1 + X_2 + \cdots + X_n}{n}$$

will move closer and closer to $1/2$—in other words, the more often we toss the coin, the closer to 1 will be the ratio of heads to tails observed.

A rather simplified way of paraphrasing this law is to say that the more often we carry out a random process the closer the average of all our observations is to the expected value.

4.4.7 Variance and standard deviation

The expected value of a random variable allows us to 'concentrate' it's behaviour into just one number. But as Examples 4.78 and 4.82 illustrate, the expected value can be misleading regarding which values are likely to occur. One way of measuring how far a random variable deviates from its expected value is to do the following:

Let X be a random variable.

- Calculate the
$$\text{expected value} \qquad e \qquad \text{of } X.$$
- Create a new random variable in two steps:
  - Subtract the expected value e from X to form the random variable
  $$X - e.$$
  - To ensure that positive and negative differences from the expected value cannot cancel each other out (and to amplify differences), form the square of the previous random variable to give
  $$(X - e)^2.$$

- Calculate the expected value of the new random variable.

Example 4.93. We return to Example 4.68 of tossing a coin three times, where we count the number of heads seen to get a random variable X. We recall from Example 4.78 that the expected value of X is 1.5.

If we form $X - 1.5$ we get a new random variable with range

$$\{-1.5, -.5, .5, 1.5\}$$

and pmf

-1.5	$-.5$	$.5$	1.5
$1/8$	$3/8$	$3/8$	$1/8.$

If we square the result we have the random variable $(X - 1.5)^2$ with range

$$\{.25, 2.25\}$$

and pmf

$.25$	2.25
$2/8 = 1/4$	$6/8 = 3/4.$

Its expected value is

$$.25 \cdot \frac{1}{4} + 2.25 \cdot \frac{3}{4} = \frac{0.25 + 6.75}{4} = \frac{7}{4} = 1.75.$$

Hence the variance (see definition below) of the random variable X is 1.75.

Definition 43: variance

If X be a random variable with expected value e its **variance** is given by

$$E((X - e)^2).$$

As pointed out above, the variance amplifies larger deviations from the expected value by squaring the difference, and it returns the square of the expected difference. For some considerations it is preferred not to carry the last step, leading to a slightly different way of measuring how far a random variable strays from its expected value.

Definition 44: standard deviation

If X is a random variable then its **standard deviation** is given by the square root of its variance.

The standard deviation gives an idea of what is ‘normal’ for a given distribution. If we only consider ‘normal’ those values which are equal to the expected value then this is too narrow for most purposes. If the average height in a given population is 167cm, then we don’t consider somebody who measures 168cm far from the norm.

Typically values which are within one standard deviation on either side of the ave considered ‘normal’. If the standard deviation is large that means that there are a lot of data points away from the expected value, and we should not have too narrow an idea of what is ‘normal’.

Example 4.94. In the Example 4.93 the standard deviation is $\sqrt{1.75} \approx 1.32$. This means that for the coin example, almost anything is normal. If we increase the number of coin tosses that changes.

The standard deviation can be thought of as giving us a measurement of the variability of the possible values of a random variable. For some purposes the variance (which is closely related) has nicer properties. These ideas will appear in the unit on data science in an early lecture. Related ideas are to use data gathered to calculate *sample variance*, also known as *empirical variance*.

Exercise 121. (a) Show that for a random variable X with expected value e the variance is $E(X^2) - e^2$.

(b) Show that if X and Y are independent random variables we have that the variance of $X + Y$ is the variance of X plus the variance of Y. *Hint: You may want to use part (i).*

4.5 Averages for algorithms

A very important application of expected values in computer science is that of the *average complexity* of an algorithm. You will meet this idea in COMP11212 and COMP26120 (and COMP36111. Mathematically it is quite tricky to make precise the average that is formed here. In subsequent course units you will not see formal derivations of the average complexity of an algorithm, and the examples we study below give you an idea why that would take up a great deal of time. The examples we do look at stand serve as case studies that illustrate the procedure.

4.5.1 Linear search

Assume you have an array of integers (for example of student id numbers, pointing to the student file). Assume you are trying to find a particular id number in that array.

A simple-minded algorithm for doing this will look at all the possible values in the array until the given number is found.

Code Example 4.1. Here's a code snippet that implements this search idea.

```
for (int index=1; index < max_index; index++)
   if (array[index]=given_number) ...
```

This algorithm is known as *linear search*. How many times is the algorithm going to perform look-up for the array on average? In other words, how often will array[index] be invoked? We begin by looking at an example.

Example 4.95. If the array has 8 entries then the chance that the entry we are looking for is any one of them is $1/8$. If we are lucky, and we find the entry on the first attempt[35] at array[1] then we have needed one look-up, whereas if we have to keep checking until we reach array[8] we need 8 look-ups. We have a random variable which takes its values in

$$\{1, 2, 3, 4, 5, 6, 7, 8\},$$

and each of these values occurs with the same probability, namely $1/8$. Hence the expected value for this random variable is

$$\begin{aligned} 1 \cdot \frac{1}{8} + 2 \cdot \frac{1}{8} + \cdots + 8\frac{1}{8} &= \sum_{i=1}^{8} i\frac{1}{8} \\ &= \frac{1}{8}\sum_{i=1}^{8} i \\ &= \frac{1}{8} \cdot \frac{8(8+1)}{2} \\ &= \frac{8+1}{2}. \end{aligned}$$

This means we have to expect 4.5 look-ups on average.

Of course most real-world applications have considerably larger arrays. For this reason it pays to think about the general case.

Example 4.96. We now assume that we have an array with n entries, and that the chance of the searched for entry being in any of the n positions is the same, namely

$$\frac{1}{n}.$$

We apply the same algorithm as before, namely looking at each entry until we find the one we are looking for. In particular note that we are implicitly assuming that not finding the entry in the first position does not tell us anything about the probability of it being in the second (or any other) position.

If the looked-for entry is the first entry of the array then we need one look-up operation, if it is the second entry we need two look-ups, and so on until the end of the array. So we have a random variable that can take values in the set

$$\{1, 2, \dots, n\},$$

and for which the probability that any one of them occurs is $1/n$. Hence the expected value for this random variable is

$$\begin{aligned} 1 \cdot \frac{1}{n} + 2 \cdot \frac{1}{n} + \cdots + n\frac{1}{n} &= \sum_{i=1}^{n} i\frac{1}{n} \\ &= \frac{1}{n}\sum_{i=1}^{n} i \\ &= \frac{1}{n}\frac{n(n+1)}{2} \\ &= \frac{n+1}{2}. \end{aligned}$$

In other words we have to look through roughly half the array on average before finding the looked-for entry. You might have been able to work this

[35] Typically arrays start at index 0 but for our example it makes life less complicated if we start at index 1.

out without any knowledge of random variables, but we can now put these ideas on a firm mathematical footing.

People who study algorithms are also interested in the *worst case* which in this example is that we have to perform n look-up operations until we finally find our number.

So the average case of the algorithm is that the number of look-ups required is roughly half the size of the input, whereas the worst case is that it is the size of the input.

4.5.2 Binary search

In the above example we were using an algorithm that is not particularly clever. If the entries appear in the array sorted by their size then we can do much better.

Assume we are trying to solve the same problem as in the previous example, but this time we have an array whose entries are sorted. In that case we can come up with a faster algorithm effectively by making use of this extra information.

Here's the idea:[36] The first index we try is the one halfway through the array, say the 4th entry. If the entry at that position is the one we were looking for then we are done. If not, then if the entry at that position is below the one we are looking for then we know that the looked-for entry has to be to the right of the current position at a higher index, else to the left at a lower index. Of course we might be really lucky and have found our entry already!

We now apply the same trick again: We find an entry roughly halfway through the appropriate half of the array. If the entry at the current position is below the one we are looking for. . . .

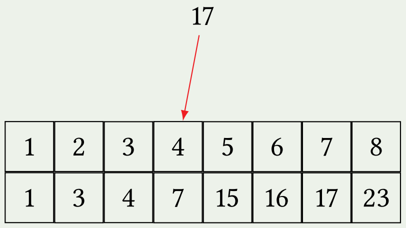
What's the expected number of look-ups required for this algorithm? What we do on each step is to look up one entry, and split the remaining array in two parts whose sizes differ by at most 1. We look at a concrete example to better understand the situation.

Example 4.97. Again we assume that we have an array of size 8. Say our array looks as follows:

1	2	3	4	5	6	7	8
1	3	4	7	15	16	17	23

If we look for the entry 17 we perform the following steps:

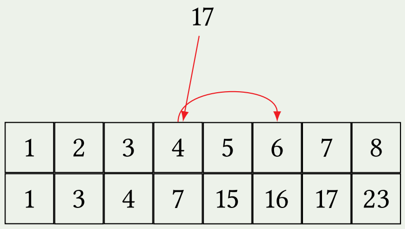
- We look at the entry at index 4, where we find the entry 7. This is smaller then the entry we are looking for.

1	2	3	4	5	6	7	8
1	3	4	7	15	16	17	23

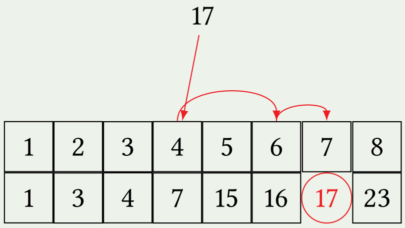
  We know that if our number is in the array it has to be to the right of the index 4.

- On the next step we look halfway through the indices 5, 6, 7 and 8. There are 4 entries, so (roughly) halfway along is at index 6. We find the entry 16, which is again smaller than the one we are looking for.



1	2	3	4	5	6	7	8
1	3	4	7	15	16	17	23

- We now have to look halfway along the indices 7 and 8. There are two entries, so halfway along is at index 7. We have found the number we were looking for,



1	2	3	4	5	6	7	8
1	3	4	7	15	16	17	23

Here is a description of the algorithm when looking for an arbitrary number in this array:

We assume that we cannot be sure that the entry is in the array at all (somebody might have given us an invalid id number). On the first step we look up the entry at index 4. If this doesn't give us the entry we were looking for then this leaves us with

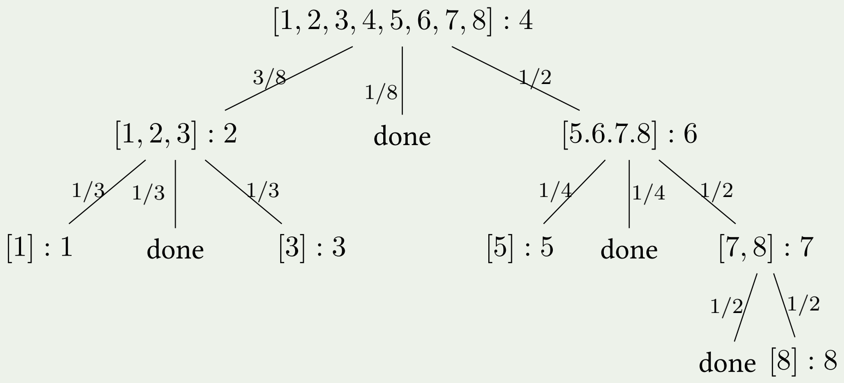
- either our entry is smaller than the one at index 4, so if it is there it must be at indices 1, 2 or 3, in which case
    - we look up the entry at index 2, and if we are not successful then
        - if our entry is below that at index 2 we look up index 1 or
        - if our entry is above that at index 2 we look up index 3,

  or

- our entry is greater than the one at index for, so if it is there at all it must be at indices 5, 6, 7 or 8, in which case we
    - look up the entry at index 6, and if that is not the correct one then
        - if our entry is smaller than that at index 6 we look at index 5
        - if our entry is greater than that at index 6 we look at index 7.
            - and if it is not at index 7 we look at index 8,

This information is more usefully collected in a tree. Here the nodes are given labels where

- the first part is a list of indices we still have to look at, then there is a colon and
- the second part is the index we are currently looking at.



We can see that in the worst case we have to look at indices 4, 6, 7 and 8, which makes four look-ups.

We can also calculate the expected value for this situation:

- The probability that we need only one look-up is $1/8$;
- we need two look-ups with probability
$$3/8 \cdot 1/3 + 1/2 \cdot 1/4 = 2/8;$$
- we need three look-ups with probability
$$3/8 \cdot (1/3 + 1/3) + 1/2 \cdot (1/4 + 1/2 \cdot 1/2) = 4/8;$$
- we need four look-ups with probability $1/2 \cdot 1/2 \cdot 1/2 = 1/8$.

Hence the expected value for the number of look-ups is

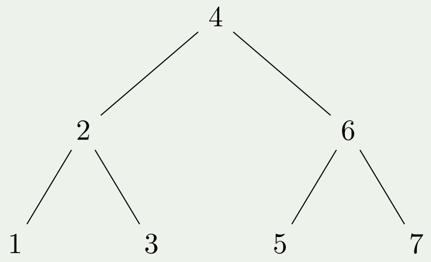
$$1 \cdot \frac{1}{8} + 2 \cdot \frac{2}{8} + 3 \cdot \frac{4}{8} + 4 \cdot \frac{1}{8} = \frac{21}{8} = 2.625.$$

Again we want to analyse the general case of this algorithm, which is known as *binary search.*

Example 4.98. From the example above we can see that some cases are easier to analyse than others: If the elements of the array exactly fit into a tree then the calculation becomes much easier.

If we look at the example of eight indices we can see that 7 indices would fit exactly into a tree with three levels of nodes. We can also see that we don't need to have separate nodes labelled 'done'; instead, we can just use the parent node to record that the search is over. In the case where there are seven entries in the array we could calculate the expected value using the following tree, where now we only list the index that is currently looked up for each node:

[36] I think you will have seen this if you have been at one of our Visit Days.



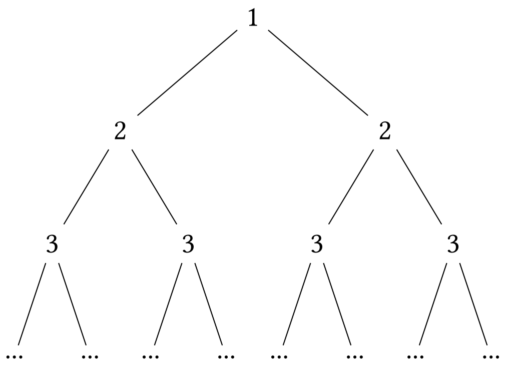
The node on the top level requires one look-up, the two nodes on the second level require two look-ups, and the four nodes on the third level require three look-ups. Each of those nodes will be equally likely to hold our number. Hence we can see that the average number of look-ups is

$$1 \cdot 1 \cdot \frac{1}{7} + 2 \cdot 2 \cdot \frac{1}{7} + 3 \cdot 4 \cdot \frac{1}{7} = \frac{17}{7} \approx 2.43.$$

We can generalize the idea from the preceding example provided that the number of indices is of the form

$$2^0 + 2^1 + \cdots + 2^{k-1} = \sum_{i=0}^{k-1} 2^i = 2^k - 1.$$

We can think of the situation as being given as in the following tree, where on each level we give the number of look-ups required.



The expected number of look-ups can be described by a function

$$f \colon \mathbb{N} \longrightarrow \mathbb{N}$$

which behaves as follows:

$$f(2^{k+1} - 1) = 1 + f(2^k - 1),$$

because if we have an array with $2^{k+1} - 1$ elements, which exactly fit into a tree with $k + 1$ layers, we need one look-up, and are then left with a tree with k layers, which requires $f(2^k - 1)$ look-ups. This kind of description of a function is known as a *recurrence relation*, and we look at simple cases for solving these in Section 6.4.5, and give a few further examples in Chapter 8. Here we can analyse the situation fairly easily:

We start counting the levels from the top, starting with level 0 and ending at $k - 1$. Then

level i	has 2^i nodes

each needing $i+1$ look-ups

each holding the right value with probability $\dfrac{1}{2^k-1}$.

Hence the expected value of the number of look-ups is

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

To check that we have derived the correct formula we can look at the case for $k=3$, that is seven entries in the array, and compare the result we get from the formula with the one calculated above. The formula gives approximate 2.43 look-ups which agrees with the result previously calculated.

We give a few (approximate) values of this sum:

k	3	4	5	6	7	8
n	7	15	31	63	127	255
exp no look-ups	2.43	3.27	4.16	5.1	6.06	7.03

As k grows large the sum given above approximates k, If we only have values for arrays of sizes of the form

$$2^k-1,$$

do we have to worry about the other cases? The answer is that one can show with a more complicated analysis that for an array with n entries, we require approximately $\log n$ look-ups, even if n is not of the shape s^k-1. This means that the average number of look-ups for an array with n entries is approximately $\log n$.

We note that the worst case for this algorithm is that we have to look up one node on each level in the tree, which means that in the worst case the number of look-ups is the height of the tree, which is $\log n$. So here we are in a situation where the average case is the same as the worst case!

Occasionally it is easier to analyse particular problem sizes, and as long as the values for other values deviate in only a minor way from the function so deduced, this is sufficient for most purposes in computer science. You will learn in COMP11212 that we are typically only interested in the 'rate of growth' of a function describing the number of instructions required for a given problem size, and that all other aspects of the function in question are dropped from consideration.

Often when looking at issues of complexity it is sufficient to have approximate counts, and more generally we only care about how quickly the number of instructions grows as n grows large. We look a little into how one can measure the 'growth' of a function in Section 5.1.

EExercise 122. Assume you have an array whose entries are natural numbers, and you are given a natural number k that occurs in the array. You want to change the order of the entries in the array in such a way that it satisfies the following two conditions:

- All numbers which are less than k occur to the left of k and
- all numbers which are larger than k occur to the right of k.

This is a part of an important sorting algorithm called *Quicksort*. In what follows we make the assumption that the number k occurs in the array exactly once.[37] The way this algorithm is implemented is as follows:

- There are two pointers, low and high.
- At the start the low pointer points to the lowest index and the highest pointer points to the highest index.
- You start a loop. This loop runs until the low pointer and the high pointer point at the same entry.
    - Look at the entry the low pointer points to.
        - If the entry is less than k then increase the low pointer by one, check that it has not reached the index of the high pointer, and repeat.
        - If the entry is greater than or equal to k then do the following.
            - Look at the entry the high pointer points to.
            - If the entry is greater than k then decrease the high pointer by one, check that it has not reached the low pointer, and repeat.
            - If the entry is less than or equal to k then swap the two entries.
    - Repeat, looking again at the low pointer.

(a) Carry out this algorithm for the following array and $k = 17$.

1	2	3	4	5	6	7	8
19	2	17	5	1	27	0	31

How many times does the algorithm ask you to swap elements?

Now look at carrying out the algorithm for an arbitrary array of size n.

(b) In the best case, how many times does the algorithm have to swap elements? Justify your answer.

(c) Assume you have an array with five elements. In the worst case, how many times does the algorithm have to swap elements? Try to generalize your idea to an array with n elements. Justify your answer by describing how to construct an array where the worst case will occur.

(d) Assume that the element k occurs in the middle of the array and that the array has an odd number of entries. What is the average number of swaps the algorithm has to perform if you may assume[38] that given an arbitrary element of the array,

- the probability that it is less than k is $1/2$ and
- the probability that it is greater than k is $1/2$?

Write one sentence about how this changes if the probability that an arbitrary element of the array is less than k is p, and the probability that it is greater than k is $1 - p$.

(e) On average, how many times does the algorithm have to swap elements if you may assume everything from the previous part, with the exception that the element k is located in the middle of the array?

(f) Can you say how many times the algorithm has to swap elements on average if you are not allowed to make this assumption, but if the element k still occurs in the middle of the array?

Note that this is a tricky exercise, and its main point is to show how difficult it is to properly calculate the average complexity of any algorithm.

Hint: For any of the parts from (b) onwards if you struggle to work out the general situation try some small arrays to see whether you can see what happens.

You can see from the examples given, however, that a proper analysis can be quite tricky (the cases discussed above are relatively simple ones), and that one often has to make decisions about using approximations. When people claim that an algorithm has, say *an average case quadratic complexity* then this has to be read as an approximate description of its behaviour as the input grows large. The above preceding four examples give you an idea of what is meant by 'average number of steps'. Note that the typical assumption is that every possible configuration is equally likely (that is in our example that the sought-for number is equally likely to occur at any given index in the array), and that these assumptions are not always justified.

4.6 Some selected well-studied distributions

In many situations it is hard to determine the probability distribution of a given random variable from the given data. In those cases it is standard to make the assumption that it behaves according to some well known distribution.

Clearly if this assumption is not justified then any calculations based on it are not going to be of much practical use. When you are asked to cope in such a situation you should, at the very least, think about what you know about the given situation and which well-known distribution this suits best.

We here give an overview of only a very small number of distributions. There is plenty of material available on this topic, and so there is no need to add to that.

4.6.1 Normal distributions

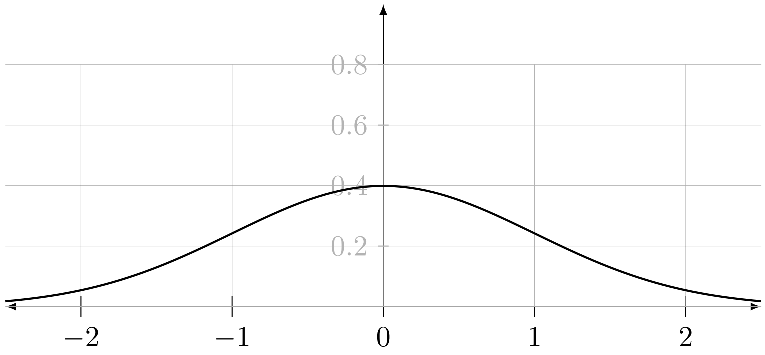
Normal distributions are used on many occasions. They are continuous probability distributions— although 'normal distribution' refers to a whole family people often use this term in the singular.

In its simplest form the probability density function of a normal distribution is given by

$$\begin{aligned}\mathbb{R} &\longrightarrow \mathbb{R}\\ x &\longmapsto \frac{1}{\sqrt{2\pi}}e^{-x^2/2}.\end{aligned}$$

[37] Although it also works if the number doesn't occur at all in that you get a block of numbers less than k followed by a block of numbers greater than k.

[38] The assumption is equivalent to assuming that there are as many numbers below k in the array as there are numbers greater than or equal to k.



The expected value of a random variable with this probability density function is 0, and the standard deviation is 1.

It is possible to create a normal distribution for a given expected value and a given standard deviation. Let v and s be real numbers, where $s > 0$. Then a random variable with probability density function

$$\begin{aligned} \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto \frac{1}{s\sqrt{2\pi}} e^{(x-v)^2/2s^2} \end{aligned}$$

has expected value v and standard deviation s.

One of the reasons that this is such a useful distribution is that, under fairly general assumptions, it is the case that the average of a (large) number of random variables which are independent and have independent distributions converges against having a normal distribution. For this reason random variables that are created from a number of independent processes obey a distribution which is close to a normal distribution. You will meet this idea once again in COMP13212, and we have just summarized the reason why the normal distribution often appears in applications.

Normal distributions are known to occur in the natural work, for example as the velocities of molecules in an ideal gas. There are many resources available to study phenomena which follow these distributions.

4.6.2 Bernoulli and binomial distributions

We have used Bernoulli distributions already without naming them. Given a random variable with two possible outcomes, say

$$r \qquad \text{and} \qquad r' \qquad \text{in } \mathbb{R},$$

to give a probability distribution of the random variable it is sufficient to determine

$$P(X = r),$$

and all other probabilities are then uniquely determined (compare Corollary 4.13). In particular we know that

$$P(X = r') = 1 - P(X = r),$$

since the probability of all possible outcomes have to add up to 1.

Typically for a Bernoulli distribution we assume the only possible values of the random variable are

$$0 \qquad \text{and} \qquad 1.$$

Example 4.99. Tossing a coin is an experiment that follows a Bernoulli distribution, where one of head or tails is assigned the value 1, and the other the value 0. You can think of this as the random variable that counts the number of heads (or tails) that appear in a single coin toss.

To make the notation less tedious, assume that

$$P(X = 1) = p.$$

The expected value of this distribution is given by

$$0 \cdot (1 - p) + 1 \cdot p = p,$$

and the variance is

$$\begin{aligned} E((X - p))^2) &= E(X^2 - 2pX + p^2) \\ &= (0^2 - 2p \cdot 0 + p^2)(1 - p) + (1^2 - 2p \cdot 1 + p^2)p \\ &= p^2(1 - p) + (1 - 2p + p^2)p \\ &= p^2 - p^3 + p - 2p^2 + p^3 \\ &= p - p^2 \\ &= p(1 - p). \end{aligned}$$

The binomial distributions arise from assuming an experiment with a Bernoulli distribution is carried out repeatedly, in a way where the previous incarnations have no influence on the following ones, such as tossing a coin a number of times, and adding up the results (for example the number of heads that appear). You can find the description of the pmf, expected value, and standard deviation for these distributions from many sources, including online.

4.6.3 The Poisson distribution

The Poisson distribution is a discrete distribution that applies to process of a particular kind, namely ones where

- we look at the probability of how many instances of a given event occur within a given time interval or a given space,
- we know the average rate for these events and
- the events occur independently from the time of the last event.

Typical examples are:the following.

- The number of births per hour on a given day.
- The number of mutations in a set region of a chromosome.
- The number of particles emitted by a radioactive source within a given time span.
- The number of sightings of pods of dolphins along a given path followed by an observing plane.

- Failures of machines or components in a given time period.
- The number of calls to a helpline in a given time period.

It is assumed that the expected number of occurrences (on average) of the event in the given time frame is known, so assume this is given by $v \in \mathbb{R}^+$. A random variable X obeying the Poisson distribution has the pmf

$$P(X = n) = \frac{v^n e^{-v}}{n!}.$$

Its expected value is v, which is also the variance.

Example 4.100. Assume we have motherboards for which it is known that on average, $.5\%$ are faulty. If we pick a sample of 200 motherboards, what is the probability that three of them are faulty?

From the given data we would expect $.005 \times 200 = 1$ to have one faulty board on average in such a sample, but this does not tell us how to answer the question about the probability that we have three of them. If we assume that this event follows the Poisson distribution then we get

$$P(X = 3) = \frac{1^3 e^{-1}}{3!} \approx .06,$$

so the probability is 6%.

4.6.4 Additional exercises

We look at situations here for which I don't want you to make any assumptions about which part of the notes you should use to solve them.

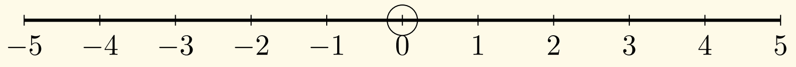
Exercise 123. Consider the following marking scheme for multiple choice questions: Each question has precisely one correct answer of four choices given, and students may pick as many of the available choices as they like. The marking scheme is as follows: For choosing the correct answer the student gets three marks, and for each chosen incorrect answer the student loses a mark.

Show that if a student randomly chooses how many alternatives to include, and which ones those should be, the number of marks they get is 0.

Exercise 124. A lecturer believes that students have a better chance of doing well on their unit if they also take another unit at the same time. He looks at the numbers from the past academic year to see whether he can find statistical evidence for his belief. In the past year he had 200 students on his course of which 40 got a very good mark. Of the student on his course 67 were enrolled on the other unit in question, and of these 27 receive a very good mark.

Do you think he is right in his belief?

Exercise 125. Assume you have a line with 11 points points from -5 to 5. There is an ant at point 0.



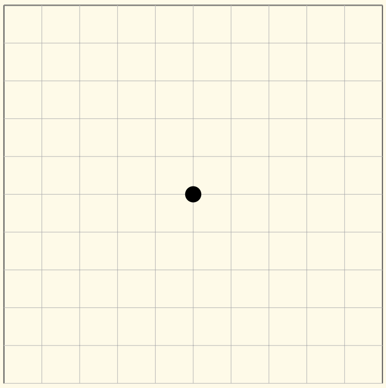
Assume that with probability $1/2$ the ant moves one point to the left, and with probability of $1/2$ it moves one step to the right. If it wants to make a step that causes it to leave the grid it stops. *This exercise requires a lot of calculations and is a bit fiddly in places.*

(a) What is the probability that the ant will have stopped after 10 steps?

(b) What is the expected position of the ant after 10 steps?

Exercise 126. This exercise is a generalization to 2 dimensions of the previous one, so you may want to solve that first.

Assume you have an eleven-by-eleven grid, which we may give coordinates from from $(-5, -5)$ to $(5, 5)$ as in the following picture. There is an ant on the grid, initially in position $(0, 0)$.



Assume that with the probability of $1/4$ the ant selects a direction from $\{N, E, S, W\}$ and takes one step in that direction. If it wants to move in a direction that would cause it to leave the grid it stops.

(a) What is the probability that the ant has stopped after ten steps?

(b) What is the expected position of the ant after ten steps?

Assume that the ant is not allowed to change direction by more than 90 degree on each step, and that each of the possible three directions is equally likely.

(c) What is the average distance that the ant will have from the starting point after five steps?

(d) What is the probability that the ant will have have stopped after eight steps?

Exercise 127. Assume you are looking after a cluster containing 50 machines. One of your machine has been affected by an odd virus. Its behaviour is as follows:

- It randomly picks one of the other 49 machines in the cluster. It copies itself to that machine. It then becomes inert.
- If a machine that was infected previously becomes infected again it

behaves as if it hadn't been infected before, that is, the virus is copied to one machine randomly picked from the other 49 machines in the cluster.

(a) What is the probability that after eight infection steps, the number of infected computers is 8? (In other words, no computer has been infected twice.)

(b) What is the expected number of infected computers after 5 infection steps? *Hint: draw a tree where each node is labelled by the number of machines currently infected. On the first step there is one such machine, on the second step there are two (you may want to think about why), and after the third step there can be two or three.*

(c) Picture the tree that would fully describe the possible numbers of infected machines after 50 steps. How many paths in that tree lead to exactly three machines being infected? What is the probability for each of those paths?

CExercise 128. Calculate the expected values asked for in the following situations. Make sure you give a full calculation, not just a number, and be prepared to explain your calculation.

(a) You are staying at a guest house with seven rooms. You know from chatting to the owner that three rooms have couples staying, two rooms have singles, and one room is empty. At the breakfast buffet you get to know one of the other guests. What is the expected number of occupants of their room?

(b) You have lined up 10 pound coins. You flip each one of them, and then move to one side the ones that show heads. You flip the remaining ones again, and once more move to one side the ones that show heads. You flip the remaining ones again and once more move those showing heads to one side. How many coins do you expect to have put aside altogether?

(c) Assume you are offered the following game: You roll a die. You can decide to stop here and get the number of points shown on the die, or you can roll it again. After the second roll you again have the choice to obtain the number of points shown on that roll, or to roll one final time.

Describe the strategy that maximises the expected number of points you win in this game, and give the number of points you may expect.

Exercise 129. Assume you have an animal that lives on the real interval from 0 to 1, and it is equally likely to be any of these locations. Now assume we have a second animal of this kind. What is the expected distance between the two?

Chapter 5

Comparing sets and functions

In computer science we are interested in comparing functions to each other because when we decide which algorithm to choose we want to pick the one that shows the better behaviour for the given range of inputs. By 'better behaviour' we mean an algorithm that performs faster for the given inputs. As you will see in COMP11212 when we do this we only compare such functions regarding how fast they grow, and one of the aims of this section is to introduce that idea.

We also have to be able to compare sets with each other. In Chapter 4 there is frequently a distinction between three cases regarding random processes into

- those with a finite number of outcomes and
- those with a countable[1] number of outcomes and
- those we consider continuous.

This chapter makes these ideas formal. There are other applications for these ideas, and we sketch one here:

- There are countably many Java programs.
- There are uncountably many functions from $\mathbb{N}$ to $\mathbb{N}$.

This mismatch tells us that there are some functions from the natural numbers to the natural numbers which cannot be implemented by a Python or Java program.

5.1 Comparing functions

In Section 4.4.6 we discuss how to calculate the number of instructions that a program has to carry out on average. It is a first step to analysing the efficiency of an algorithm.

Sometimes we have a choice of programs (or algorithms) to solve a particular problem. For small problem sizes it won't matter too much which one we pick, but as the size of our problem grows (for example, sorting millions of entries in some array as opposed to a few tens) we need to seriously think about what is the best choice. It might be the case that some programs take so long (or requires so many resources in the form of memory) that one cannot feasibly use them.

To measure the efficiency of programs it is standard to count the number of some instructions that measures how long the program is taking, depending on

[1] These are the ones where the set S of outcomes may be described in the form $S = \{s_i \mid i \in \mathbb{N}\}$.

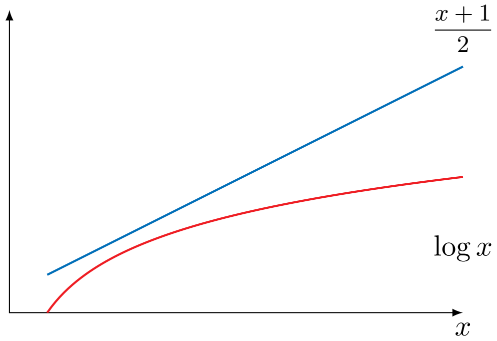
the size of the problem. The question then is how to compare such functions. Examples 4.96 and 4.98 in Chapter 4 give a measure of efficiency of two algorithm, the first one being known as *linear search* while the second is called *binary search*. For these algorithms we counted the number of look-up operations performed to measure their complexity.

For an array with n entries, the former has an average number of $(n+1)/2$ look-ups to perform, while the latter requires approximately $\log n$ look-ups.

We picture the corresponding functions by drawing their graph when viewing them as functions from $\mathbb{R}^+$ (or a subset thereof) to $\mathbb{R}^+$.

instead of functions from $\mathbb{N}$ to $\mathbb{N}$. In the following graph consider the two functions given

$$[1,\infty) \longrightarrow \mathbb{R}^+.$$



We can see that for every input value binary search requires fewer look-ups than linear search. In this case it looks like an easy choice to make between the two. However, we have to bear in mind that binary search requires the given array to be sorted, and that does require additional computation time and power.

The picture suggests a definition for comparing functions.

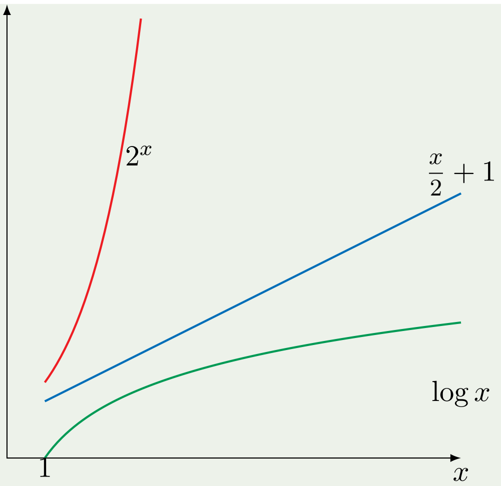
Definition 45: dominate

Let N be a set of numbers, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$, and let f and g be two functions from a set S to N. We say that f **dominates** g (or f **is above** g) if and only if

$$\text{for all } s \in S \text{ it is the case that } fs \geq gs.$$

When we draw the graphs of two functions where one dominates the other we can see that the graph of the first is entirely above the graph of the second (but the graphs are allowed to touch).

Example 5.1. Consider the following three functions from $[1,\infty)$ to $\mathbb{R}^+$.



The function

$$x \longmapsto 2^x$$

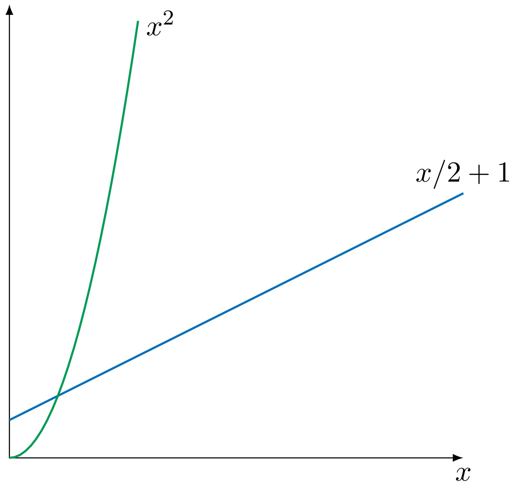
dominates the function

$$x \longmapsto x/2+1$$

which in turn dominates the function

$$x \longmapsto \log x \ .$$

But this notion is not sufficient for the intended application. If we want to establish whether one program outperforms another then using this idea for, say, the functions giving the number of instructions as a function of the size of the input for each program, may not give a useful result. Consider the functions below, going from $\mathbb{R}^+$ to $\mathbb{R}^+$.



Neither function dominates the other. But clearly if the problem size is large (that is, we move to the right in the graph) then the function

$$x \longmapsto x/2+1$$

offers a much preferable solution. This idea is encapsulated by the following definition.[2]

[2] You will meet the following definition again in COMP11212, and COMP21620.

Definition 46: eventually dominate

Let N and N' be sets of numbers from $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$, and let f and g be two functions from N to N'. We say that f **eventually dominates** g if and only if

there exists $k \in \mathbb{R}$
such that for all $x \in N$ with $x \geq k$ we have $fx \geq gx$.

We can think of this definition as saying that f dominates g if we restrict the source of f and g to

$$\{x \in N \mid x \geq k\},$$

or if we only look at the graphs of the two functions to the right of k.

Note that there is no need to find the *smallest* $k \in \mathbb{N}$ with this property—any such k will do!

Typically when we are interested in one function eventually dominating another in computer science, we are interested in functions from the natural numbers to some subset of the real numbers. When we try to draw the graph of such a function it is easier to draw it as a function from the *real numbers* to the real numbers. It is not a priori clear what happens when we change the source set of the function.

Note that if f is a function from some set S to a subset of the real numbers then there is a very closely related function whose target is $\mathbb{R}$, given by

$$\begin{array}{ccc} S & \longrightarrow & \mathbb{R} \\ x & \longmapsto & fx. \end{array}$$

The following result gives us information about extending the domain of definition of our function.

Proposition 5.1

Let N be a set of numbers from $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$, and let f and g be functions from N to $\mathbb{R}$. Assume that f' and g' are functions from $\mathbb{R}$ to $\mathbb{R}$ such that

- f' restricted to inputs from N is f and
- g' restricted to inputs from N is g. If f' eventually dominates g' then f eventually dominates g.

Proof. If f' eventually dominates g' then we can find $k \in \mathbb{R}$ such that for all $x \in \mathbb{R}$ with $x \geq k$ we have that $f'x \geq g'x$.

To show that f eventually dominates g, assume we have $y \in N$ with $y \geq k$. We know the following:

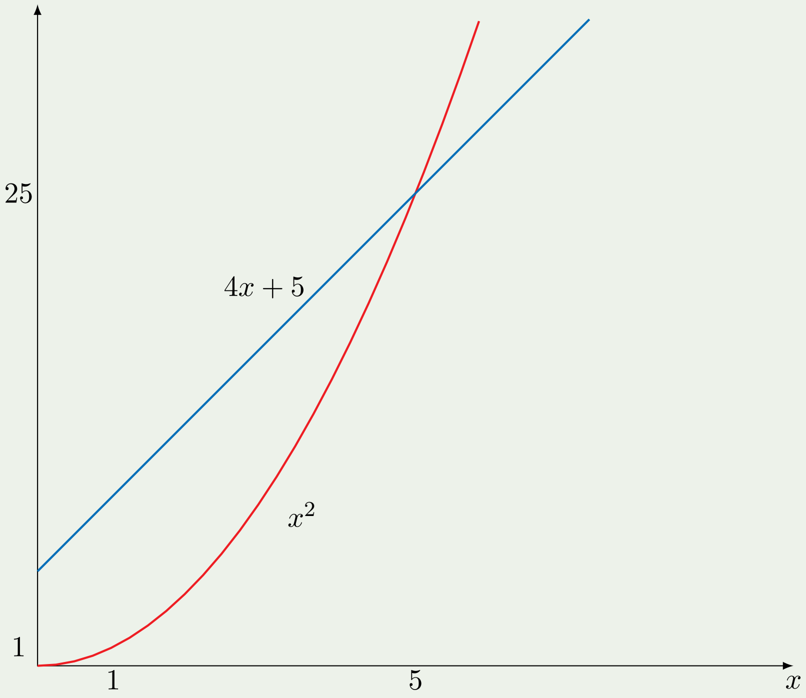
$$\begin{aligned} fx &= f'x && \text{assumption about } f' \\ &\geq g'x && x \geq k \\ &= gx && \text{assumption about } g'. \end{aligned}$$

Hence we may argue with suitable functions from the real numbers to the real numbers.

Example 5.2. Consider the two functions

$$\begin{aligned} f\colon \mathbb{N} &\longrightarrow \mathbb{N} \\ n &\longmapsto n^2 \end{aligned} \qquad\qquad \begin{aligned} g\colon \mathbb{N} &\longrightarrow \mathbb{N} \\ n &\longmapsto 4n+5. \end{aligned}$$

Again we use graphs to picture the situation,[3] where we treat both expressions as functions from the non-negative reals to the reals. The preceding proposition tells us that considering the graphs tells us something about the functions given.



Once the two lines have crossed (at $x = 5$) the graph of f stays above that of g. This suggests that we should try to find a proof that f eventually dominates g.

- First of all, we have to give a witness for the 'exists' part of the statement. The graph helps us to choose $k = 5$, but note that every natural number larger than 5 would also work.
- Now that we have k we have to show that for all $n \in \mathbb{N}$, with $n \geq k$, we have $gn \leq fn$. So let us assume that $n \in \mathbb{N}$, and that $n \geq 5$. Then

$$\begin{aligned} gn = 4n+5 &\leq 4n+n && 5 \leq n, \text{ Fact } 7 \\ &= 5n && \\ &\leq n \cdot n && 5 \leq n, \text{ Fact } 7 \end{aligned}$$

$$= n^2 = fn$$

as required.

Example 5.3. Here's an alternative way of proving the same statement.

- Again, we have to give a k, but assume this time we have not drawn the graph. We have to guess a k such that for all $n \geq k$ we have
$$4n + 5 \leq n^2.$$
We can see that we require a number k such that multiplying with k is at least as large as multiplying with 4 and adding 5. Say we're a bit unsure, and we are going to try to use $k = 10$ to be on the safe side.

- We have to show that for all $n \in \mathbb{N}$
$$\text{if } n \geq 10 \qquad \text{then} \qquad fn = n^2 \geq 4n + 5 = gn.$$
So assume $n \geq 10$. We work out that
$$\begin{aligned} gn = 4n + 5 &\leq 4n + 10 && 5 < 10, \text{ Fact 7} \\ &\leq 4n + n && 10 \leq n, \text{ Fact 7} \\ &= 5n \\ &\leq 10n && 5 < 10, \text{ Fact 7} \\ &\leq n^2 = fn && 10 \leq n, \text{ Fact 7}. \end{aligned}$$
Note that the shape of the proof has not changed much at all.

Example 5.4. We give another variation on this proof.

- Assume we use $k = 10$ again, but this time we produce a proof where we start by looking at the larger function.

- Let $n \geq 10$. Then
$$\begin{aligned} fn = n^2 &= n \cdot n \\ &\geq n \cdot 10 && n \geq 10, \text{ Fact 7} \\ &= 4n + 6n \\ &\geq 4n + 60 && n \geq 10, \text{ Fact 7} \\ &\geq 4n + 5 = gn && 60 > 5, \text{ Fact 7}. \end{aligned}$$

Example 5.5. Another variant of a proof of the same statement: Instead of using the assumption $n \geq k$ for whichever k we pick we express this as writing $n = k + i$, where i is an element of $\mathbb{N}$.

[3] But note that drawing graphs by hand can be time-consuming, and that in order to help with answering the question whether one function is eventually dominated by another a quick imprecise sketch can be sufficient.

For $k = 5$ the proof could then go like this:

$$\begin{aligned} gn = g(5+i) = 4(5+i)+5 & \qquad \text{def } g \\ = 25 + 4i & \qquad \text{calculations in } \mathbb{N} \\ \leq 25 + 10i & \qquad 4 \leq 10,\ \text{Fact } 7 \\ \leq 25 + 10i + i^2 & \qquad 0 \leq i^2,\ \text{Fact } 7 \\ = (5+i)^2 & \qquad \text{calculations in } \mathbb{N} \\ = f(5+i) = fn & \qquad \text{def } f. \end{aligned}$$

You can see from these examples that there are typically *many* ways of proving the desired statement. Different strategies are outlined in those examples, and you can pick whichever one you prefer in order to solve these kinds of questions.

CExercise 130. Determine whether one of the two functions given eventually dominates the other. Give a justification for your answer. You should not use advanced concepts such as limits or derivatives, just basic facts about numbers.

(a) $x \longmapsto \log(x+1)$ and $x \longmapsto x$ as functions from $\mathbb{R}$ to $\mathbb{R}$.

(b) $x \longmapsto x\log(x+1)$ and $x \longmapsto x^2$ as functions from $\mathbb{R}^+$ to $\mathbb{R}^+$.

(c) $x \longmapsto 2^x$ and $x \longmapsto 1{,}000{,}000x$ as functions from $\mathbb{Z}$ to $\mathbb{Z}$.

(d) $\sin$ and $\cos$ as functions from $\mathbb{R}$ to $\mathbb{R}$.

5.2 Comparing sets

In the introduction to this chapter we have argued that it is important to be able to compare the sizes of different sets. It turns out that the notions of injective and surjective functions from Section 2.6 is useful for this purpose.

In particular, if there is an injective function from a set S to a set T, then for every element of S there is an element of T, and all these elements are different. Hence we know that all elements of S ‘fit into’ T, and T must be as least as big as S.

Definition 47: comparison of set size

Let S and T be sets. We say that the size **of S is smaller than or equal to that of T** if and only if there is an injection from S to T.

Example 5.6. We have an injection

$$\{0,1,2,3,4\} \longrightarrow \{0,1,2,3,4,5\},$$

which is given by the assignment

$$x \longmapsto x \;,$$

which is clearly an injection. So the size of the set

$$\{0, 1, 2, 3, 4\}$$

less than or equal to the size of the set

$$\{0, 1, 2, 3, 4, 5\}.$$

This may seem like a trivial observation. Our definition really only comes into its own once we consider infinite sets.

Lemma 5.2

If S and T are sets with finitely many elements then the size of S is less than or equal to the size of T if and only if the number of elements of S is less than or equal to the number of elements of T.

Proof. Note that in Exercise 39 it is shown that the number of elements in the image of a set S under an injection is the same as the number of elements of S.

We show both implications separately.

- Assume that the size of S is less than or equal to the size of T. Then there is an injection, say f, from S to T. By Exercise 39 we know that the number of elements of the image $f[S]$ of S under f is the same as the number of elements of S. Since $f[S]$ is a subset of T we know that T has at least as many elements as S.

- Assume that the number of elements of S is less than or equal to the number of elements of T. This means that if we name the elements of S, say s_1, s_2, …, s_m, and those of T, say t_1, t_2, …, t_n then $n \geq m$. If we now define the function

$$\begin{aligned} \{s_1, s_2, \ldots, s_m\} &\longrightarrow \{t_1, t_2, \ldots, t_n\} \\ s_i &\longmapsto t_i \end{aligned}$$

from S to T it is an injection.

Here is an example with infinite sets.

Example 5.7. The natural numbers $\mathbb{N}$ can be mapped via an injection into the integers $\mathbb{Z}$ by defining

$$n \longmapsto n.$$

This is clearly an injection. Hence the size of $\mathbb{N}$ is less than or equal to the size of $\mathbb{Z}$.

If I had asked in the lecture whether the size of $\mathbb{Z}$ is at least that of $\mathbb{N}$ I am sure everybody would have told me that this is true. You may find the following example less intuitive. It shows that once we have sets with infinitely many elements our intuitions about their sizes become suspect.

Example 5.8. What you might find more surprising is that $\mathbb{Z}$ also has a size smaller than or equal to that of $\mathbb{N}$. We give an injection $f\colon \mathbb{Z} \longrightarrow \mathbb{N}$ by setting[4]

$$n \longmapsto \begin{cases} 2n & \text{if } n \geq 0 \\ -(2n+1) & \text{else.} \end{cases}$$

This is an injection for the following reason. Let m and n in $\mathbb{Z}$. We have to show that $fm = fn$ implies $m = n$. Since the definition of f is by cases we have to distinguish several cases in this proof.

- $m \geq 0$ and $n \geq 0$. If $2m = fm = fn = 2n$ we may conclude $m = n$.
- $m < 0$ and $n < 0$. If $-(2m+1) = fm = fn = -(2n+1)$ we may conclude that $2m+1 = 2n+1$ and so $m = n$ as required.
- $m \geq 0$ and $n < 0$. If $2m = fm = fn = -(2n+1)$ we get $2m = 2n+1$ which can never hold for m, n in $\mathbb{Z}$.
- $m < 0$ and $n \geq 0$. This case is identical to the previous one where n and m have been swapped.

Exercise 131. Show the following statements.

(a) The size of every set is less than or equal to itself.

(b) If the size of the set S is less than or equal to the size of the set T, and if the size of the set T is less than or equal to the size of the set U then the size of S is less than or equal to the size of U.

This means that we have defined a *reflexive* and *transitive* binary relation, which means we can think of it as a kind of order. This idea is looked at in more detail in Chapter 7.4.

What does it mean that $\mathbb{N}$ is at least as big as $\mathbb{Z}$, and $\mathbb{Z}$ is at least a big as $\mathbb{N}$? It means that they can be thought of as having the same size.

Definition 48: same set size

We say that two sets S and T **have the same size** if and only if

- the size of S is less than or equal to the size of T and
- the size of T is less than or equal to the size of S.

The previous two examples show that $\mathbb{N}$ and $\mathbb{Z}$ have the same size.

EExercise 132. Show that the following sets have the same size.

(a) $\mathbb{N}$ and $\mathbb{N} \times \mathbb{N}$;

(b) $\mathbb{N}$ and $\mathbb{N}^k$ where k is a finite number.

[4] Compare this to the function from the mid-term test in 2015/16,

(c) $\mathbb{N}$ and $\mathbb{Q}$.

(d) the set of functions from some set S to the two element set $\{0, 1\}$ and the powerset $\mathcal{P}S$ of S.

Only use facts from Chapter 0.

Exercise 133. Show that if there is a bijection from S to T then S and T have the same size.

You may have wondered why we used injections to determine the size of a set, and whether we could not have done this using surjections. The following exercise answers that question.

Exercise 134. Show that given a function $f\colon S \longrightarrow T$ the following are equivalent:

(i) f is a surjection and

(ii) the size of S is at least the size of T.

Optional Exercise 19. Show that if S and T have the same size then there is a bijection between them. This is known as the *Cantor-Bernstein-Schröder Theorem.*

We give a formal definition of infinity based on a notion known as *Dedekind infinite.*

Definition 49: infinite set

A set S is **infinite** if and only if there is an injection from S to a proper subset of S.

Proposition 5.3

A set S is infinite if and only if there is an injective function from S to itself which is not surjective.

Proof. We show the statement in two parts.

Assume that the set is infinite. Then there is an injective function

$$f\colon S \longrightarrow S'$$

where S' is a proper subset of S. We can define a function

$$\begin{aligned} g\colon S &\longrightarrow S \\ s &\longmapsto fs \end{aligned}$$

which is obviously also injective, but it is not surjective since we know there is an element of S which is not in S', and so cannot be in the image of g.

Assume that we have an injective function

$$g\colon S \longrightarrow S$$

which is injective but not surjective. Then there is an element s of S which is not in the image of g, that is, there is no $s' \in S$ with $gs' = s$. We define a new function

$$\begin{array}{rcl} f\colon S & \longrightarrow & S \setminus \{s\} \\ s & \longmapsto & gs \end{array}.$$

We note that f is injective since g is, and we note that its image is a proper subset of S.

Example 5.9. We show that there are infinitely many Java programs. We have to give an injective function from the set of Java programs to itself whose range does not include all Java programs.

We do this as follows: Given a Java' program we map it to the same Java program to which the line

```
System.out.println("Hello world!");
```

has been added.

This function is injective: If we have two Java programs that are mapped to the same program then they must be the same program once that new last line has been removed.

This function is not surjective since there are many programs which do not contain that line and so are not in the image of the function.

Hence this assignment from the set of all Java programs to itself is injective but not surjective, and so this set is infinite by Proposition 5.3.

Example 5.10. We show that the set $\mathcal{P}_f\mathbb{N}$ of finite subsets of $\mathbb{N}$ is infinite. We give an injective function from the $\mathcal{P}S$ to itself and show that it is not surjective.

Given a finite non-empty subset

$$\{s_1, s_2, \ldots, s_n\}$$

of $\mathbb{N}$ we map it to the set

$$\{s_1, s_2. \ldots s_n, (s_1 + s_2 + \cdots + s_n)\},$$

and we map the empty set to itself. In other words we have

$$\begin{array}{rcl} \mathcal{P}_f\mathbb{N} & \longrightarrow & \mathcal{P}_f\mathbb{N} \\ \{s_1, s_2, \ldots, s_n\} & \longmapsto & \begin{cases} \{s_1, s_2, \ldots, s_n, (s_1 + s_2 + \cdots + s_n)\} & n > 0 \\ \emptyset & \text{else.} \end{cases} \end{array}$$

This assignment maps a given non-empty set to the set where the sum of all the elements of that set has been added as an extra element. We observe that the extra element is always the largest element of the resulting set. Note that if the set we start with has only one element then it is mapped to itself by this function since no extra element is added.

This function is injective. If two sets are mapped to the same set then in particular their greatest elements must be equal, so the original sets must have had elements which add up to the same number. Moreover, all elements (if any) of the set which are below the largest element must also correspond to each other, so the sets must have been equal and our function is injective.

This function is not surjective since the set $\{1, 2\}$ is not in the image of this function. By Proposition 5.3 we know that the given set is infinite.

We show that all infinite sets are at least as big as the set of natural numbers $\mathbb{N}$.

Proposition 5.4

If S is an infinite set then there is an injection from $\mathbb{N}$ to S.

Proof. Let

$$f\colon S \longrightarrow S$$

be the function that shows that S is infinite, that is, we assume that f is injective but not surjective. Pick an element s of S which is not in the image of S.

We define a function

$$g\colon \mathbb{N} \longrightarrow S$$

as follows: We set

$$g0 = fs,\ g1 = fg0 = ffs,\ g2 = fg1 = fffs,\ g3 = fg2 = ffffs, \dots$$

More generally, we set

$$gn = f^{n+1}s,$$

where the power indicates applying f the given number of times. We have to show that the resulting function is injective. If we have m and n in $\mathbb{N}$ with

$$gm = gn,$$

then this means

$$f^{m+1}s = gm = gn = f^{n+1}s,$$

since f is injective we can conclude from this that

$$f^m s = f^n s,$$

and we can continue removing f on both sides until we have deleted all fs on one side of the equality. This means we have

$$s = f^l s$$

for some $l \in \mathbb{N}$. But s is not from the image of f, so we must have that $l = 0$, and so removing m many fs on one side is the same as removing n many fs on the other side, which means we must have $m = n$.

This means that the size of $\mathbb{N}$ is less than or equal to that of every infinite set. Since $\mathbb{N}$ itself is infinite (see Exercise 136) in this sense $\mathbb{N}$ is the smallest infinite set.

To ensure that our notion of infinity fits well with our notion of comparing the sizes of sets we establish the following proposition.

Proposition 5.5

If the size of $\mathbb{N}$ is less than or equal to the size of a set S then S is infinite.

Proof. Let $f \colon \mathbb{N} \longrightarrow S$ be the injective function which establishes that the size of $\mathbb{N}$ is less than or equal to the size of S.

We split S into two parts as follows. Let

$$S_1 = \{s \in S \mid \text{ there is } n \in \mathbb{N} \text{ such that } fn = s\}$$

and

$$S_2 = \{s \in S \mid \text{ for all } n \in \mathbb{N}\ fn \neq s.\}.$$

Then

$$S = S_1 \cup S_2,$$

since every element is either in the range of f, and so in S_1, or it is not in the range of f and so in S_2. Note that S_1 and S_2 are disjoint, so every element of S is either in S_1 or in S_2, but no element can be in both sets. Note that since f is injective, for every s in S_1 there is a unique $n \in \mathbb{N}$ with $fn = s$. Based on this we define a function g from S to itself as follows.

$$gs = \begin{cases} f(n+1) & s \in S_1,\ fn = s \\ s & else. \end{cases}$$

We claim that this function is injective, but not surjective. To show injectivity we have to consider four cases, similar to Example 5.8. Let s and s' be elements of S.

- $s \in S_1$ and $s' \in S_1$. There there exist unique elements n and n' in $\mathbb{N}$ with $s = fn$ and $s' = fn'$ and if $f(n+1) = gs = gs' = f(n'+1)$ then by injectivity of g we have $n = n'$, and so $s = fn = fn' = s'$.
- $s \in S_2$ and $s' \in S_2$. If $s = gs = gs' = s'$ we immediately have $s = s'$.
- $s \in S_1$ and $s' \in S_2$. We know that there exists a unique $n \in \mathbb{N}$ with $s = fn$. But now $gs = f(n+1)$ is an element of S_1, while $gs' = s'$ is an element of S_2 and so the two cannot be equal.
- $s \in S_2$ and $s' \in S_1$. This case is identical to the previous one where s and s' have been swapped.

We can see that the function g maps the set S_1 to itself, while it maps S_2 to itself as well, leaving every element as it is. The function g is not surjective since no element is mapped to $f0$:

Clearly $f0$ is in S_1, so by the previous observation if it were in the image of g there would have to be an element $s \in S_1$ with $gs = f0$. But for any such s we know that there exists a unique $n \in \mathbb{N}$ with $s = fn$, and so we would have

$$gs = f(n+1) = f0,$$

which by injectivity of f would imply $n + 1 = 0$, but no such number n exists in $\mathbb{N}$.

Exercise 135. Show that if a set has a finite number of elements then it is not infinite.

CExercise 136. Show that the following sets are infinite by proving that they satisfy Definition 49.

(a) $\mathbb{N}$,

(b) $\mathbb{R}$,

(c) the set of functions from $\mathbb{N}$ to the two element set $\{0, 1\}$ or the powerset $\mathcal{P}\mathbb{N}$ (you choose),

(d) every superset of an infinite set.

(e) Any set which is the target of an injective function whose source is infinite.

Optional Exercise 20. Show that if a set is not infinite then it has a finite number of elements.

Exercise 137. Show that if S is a set with a finite number of elements then so is its powerset $\mathcal{P}S$. Do so by determining the number of elements of $\mathcal{P}S$.

In computer science we particularly care about sets whose size is at most as big as that of the natural numbers. This is because given a finite number of symbols there are only countably many strings (and so programs) that can expressed using those symbols.

Definition 50: countable/uncountable

A set is **countable** if and only if there is an injection from it to the natural numbers. A set is **uncountable** if and only if there is no injection from it to the natural numbers. A set is **countably infinite** if it is both, countable and infinite.

Note that every finite set is countable.

Examples of countably infinite sets are:

- The set of natural numbers $\mathbb{N}$.

- The set of integers $\mathbb{Z}$.
- The set of rational numbers $\mathbb{Q}$.
- The set of finite subsets of $\mathbb{N}$, $\mathcal{P}_f\mathbb{N}$.
- The set of all programs in your favourite programming language.
- The set of all strings over a finite alphabet.

Examples of uncountable sets are:

- The set of real numbers $\mathbb{R}$,
- the set of complex numbers $\mathbb{C}$,
- the set of all subsets of $\mathbb{N}$, $\mathcal{P}\mathbb{N}$,
- the set of all functions from $\mathbb{N}$ to $\mathbb{N}$.

Note that the last example, together with the following exercise, illustrates that there are functions from $\mathbb{N}$ to $\mathbb{N}$ for which we cannot write a computer program!

Optional Exercise 21. Assume we have a finite set of symbols, say A.

(a) Show that A^k is finite for every $k \in \mathbb{N}$.

(b) Show that

$$\bigcup_{k\in\mathbb{N}} A^k$$

is countable.

(c) Show that there is a bijection between the set of finite strings built with symbols from A and the set $\bigcup_{k\in\mathbb{N}} A^k$,

(d) Conclude that there are countably many strings over the alphabet A.

(e) Put together a set of symbols such that every Python program can be built from those symbols.

(f) Prove that there is an injection from the set of Python programs to the set of strings over this set of symbols.

(g) Conclude that the set of Python programs is countable.

Proposition 5.6

A set is countable if and only if its size is at most that of $\mathbb{N}$.

Proof. If a set S is countable then by definition of that notion there exists an injection from S to $\mathbb{N}$, and by the definition of the size of a set this means that S is less than or equal to that of $\mathbb{N}$.

Assume that S is at most as big as $\mathbb{N}$. Then there is an injection from S to $\mathbb{N}$ and so S is countable.

In Chapter 4 the notion of a countable set appears. Indeed, in general, the definition of σ-algebra should refer to countable sets instead of talking about sets indexed by the natural numbers. In that chapter the notion is avoided as far as possible since the formal definition does not appear until a later chapter. We use this opportunity to connect the two ideas.

Proposition 5.7

If S is countable then there is a way of listing all its elements, that is, there is a surjective function g from $\mathbb{N}$ to S, allowing us to list all the elements of S as

$$g0,\ g1,\ g2,\ g3, \ldots$$

and we may think of them as

$$s_0,\ s_1,\ s_2, \ldots$$

where we delete any repeated elements from the list.

If S is a set such that there is a surjective function from $\mathbb{N}$ to S then S is countable.

Proof. We prove the first statement. Since S is countable there is an injective function

$$f\colon S \longrightarrow \mathbb{N}.$$

We use this function as follows: By Proposition 2.2 there is an injective function

$$g\colon \mathbb{N} \longrightarrow S$$

with the property that

$$g \circ f = \mathsf{id}_S.$$

This function is surjective, since given $s \in S$ we know that

$$s = \mathsf{id}_S s = gfs,$$

so we have found $fs \in \mathbb{N}$ which is mapped by g to s. This completes the proof.

To prove the second statement assume we have a surjective function

$$g\colon \mathbb{N} \longrightarrow S.$$

By Exercise 134 this means that the size of S is at most the size of $\mathbb{N}$, and by Proposition 5.6 we have completed the proof.

Optional Exercise 22. Show that every uncountable set is at least as big as any countable set.

Exercise 138. Show that every subset of a countable set is countable. Conclude that every superset of an uncountable set is uncountable.

Optional Exercise 23. Show that the following sets do not have the same size.

(a) Any set and its powerset;

(b) $\mathbb{N}$ and $\mathbb{R}$. Conclude that $\mathbb{R}$ is not countable.

As a consequence of Exercises 21 and 23 we can see, for example, that there are more real numbers than there are Python programs. This means that if we cannot hope to write a Python program that outputs the digits of a given real number, one at a time, for every real number.

Optional Exercise 24. Show that any two countably infinite sets have the same size.

Exercise 139. Give the sizes of the following sets:

(a) $\{a, b, c\}$,

(b) $\{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}\}$,

(c) the set of regular expressions over the alphabet $\{0, 1\}$,

(d) the set of finite state machines over the alphabet $\{0, 1\}$,

(e) the set of regular languages over the alphabet $\{0, 1\}$,

(f) the set of real numbers in the interval $[0, 1]$.

(g) the set of subsets of the real interval, $\mathcal{P}[0, 1]$.

COMP11120, Semester 1

Exercise Sheet 0 (for feedback only)

For examples classes in Week 1

Core Exercises for this week

CExercise 8 on page 29.

CExercise 9 on page 35.

CExercise 10 on page 45.

Extensional Exercises for this week

EExercise 7 on page 24.

EExercise 11 on page 46.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

You should make sure this week that you understand the content and in particular the notation used in Chapter 0

COMP11120, Semester 1

Exercise Sheet 1

For examples classes in Week 2

Core Exercises for this week

CExercise 13 on page 54.

CExercise 17 on page 57.

CExercise 22 on page 59. Carry out your proof in the style of that given on page 53 as far as you can.

Extensional Exercises for this week

EExercise 19 on page 58. Carry out your proof in the style of that given on page 53 as far as you can.

EExercise 20 on page 59.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

Exercises you could potentially do this week are all those in Chapter 1.

COMP11120, Semester 1

Exercise Sheet 2

For examples classes in Week 3

Core Exercises for this week

CExercise 25 on page 80. Do three of the parts, one from (a)–(c) and two from (d)–(g).

CExercise 27 on page 84. Do three of the parts, one from (a)–(d), one from (e)–(f) and one from (g)–(i).

CExercise 28 on page 87. Do two of the parts, one from (a)–(d) and one from (e)–(g).

Extensional Exercises for this week

EExercise 29 on page 90. Do two of the parts, one from (a)–(b) and one from (c)–(e).

EExercise 34 on page 92.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

Exercises you could do this week or those in Sections 2.1 to 2.5.

COMP11120, Semester 1

Exercise Sheet 3

For examples classes in Week 4

Core Exercises for this week

CExercise 37 on page 98. Do three of the parts, one from (a)–(c), one from (d)–(f). and one from (g)–(i). *Hint: If you find this hard then try to do the previous exercise first, where you know what the answer is in each case.*

CExercise 41 on page 105. Do three of the parts, one from (a)–(d), one from (e)–(f), and one from (g)–(h). *Hint: If you find this hard then try to do the previous exercise first, where yu know what the answer is in each case.*

CExercise 43 on page 106. Do two of the parts, one from (a)–(c) and one from (d)–(f).

Extensional Exercises for this week

EExercise 38 on page 99. Do any three parts.

EExercise 47 on page 115.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

Exercises you could do this week are those in Section 2.6.

COMP11120, Semester 1

Exercise Sheet 7

For examples classes in Week 8

Core Exercises for this week

Where the answers are probabilities don't just give a number, give an expression that explains how you got to that number!

CExercise 77 on page 163. Do three of the parts, one from (a)–(d), one from (e)–(f) and one from (g)–(i).

CExercise 79 on page 172.

CExercise 83 on page 173.

Extensional Exercises for this week

EExercise 81 on page 172.

EExercise 86 on page 177. *This is ahead of the lecture material but only requires calculating with sets. It covers important ideas for material to come.*

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

Exercises you could do this week are those in Section 4.1.

COMP11120, Semester 1

Exercise Sheet 8

For examples classes in Week 9

Core Exercises for this week

Where the answers are probabilities don't just give a number, give an expression that explains how you got to that number!

CExercise 88 on page 184. Do one from Exercise 77 (a)–(l) and two from Exercises 83 to 85.

CExercise 95 on page 199.

CExercise 99 on page 205.

Extensional Exercises for this week

EExercise 89 on page 188. Do any two parts.

EExercise 93 on page 199.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

Exercises you could do this week are those in Sections 4.2 to Section 4.3.3.

COMP11120, Semester 1

Exercise Sheet 9

For examples classes in Week 10

Core Exercises for this week

Where the answers are probabilities don't just give a number, give an expression that explains how you got to that number!

CExercise 103 on page 219.

CExercise 109 on page 236.

CExercise 110 on page 242.

Extensional Exercises for this week

EExercise 111 on page 242.

EExercise 114 on page 248.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

Exercises you could do this week are those in Section 2.6.

COMP11120, Semester 1

Exercise Sheet 10

For examples classes in Week 11

Core Exercises for this week

Where the answers are probabilities don't just give a number, give an expression that explains how you got to that number!

CExercise 116 on page 254.

CExercise 128 on page 278.

CExercise 130 on page 285. Do one from (a)–(b) and one from (c)–(d).

Extensional Exercises for this week

EExercise 118 on page 260.

EExercise 122 on page 271. Carry out parts (a)–(d).

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

Exercises you could do this week are those in Section 2.6.

Chapter 6

Recursion and Induction

Recursion is a powerful principle. It can be used to

- define sets,
- define data types in some programming language such as Java, C, and Haskell,
- define operations on recursively defined sets,
- define operations on recursively defined datatypes and
- design recursive algorithms.

Recursion as a tool for defining data types and operations on these is covered in COMP11212 and as a tool in designing algorithms. Recursion goes hand in hand with a proof principle which is known as **induction**.
There is a general principle at work in all these.

- In order to recursively define an entity we need
  - a number of base cases—these create particularly simple instances of our set or data type and
  - a number of step cases—these create more complicated instances from simpler ones already available.
- In order to define an operation on such a recursively defined entity all one has to do is
  - define what the operation should do for each of the base cases and
  - define what the operation should do for each step case, assuming that it has already been defined for the simpler entities.
- In order to prove that a recursively defined entity or operation has a particular property all one has to do is
  - check each base case and
  - assuming that the property holds for simpler constructs (known as the *induction hypothesis*) show that it holds after applying each step case.

Some of you will have come across *natural induction* as a proof principle. This is a special case of induction which relies on the idea that the natural numbers can be constructed via recursion:

- the *base case* is given by the number 0 and
- the *step case* is given by the fact that every number n has a unique successor, the number $n+1$.

We illustrate the general idea with a number of examples. We begin with the idea of recursive data types such as lists and trees. Many programming languages allow us to define recursive data types, typically using pointers or references. You will meet these ideas other course units.

When computer scientists define syntax they often use recursive definitions. For example, regular expressions (Example 6.24) are defined recursively, and in any programming language the definition of what constitutes a valid program provides another example of a recursive definition (see Example 6.30), typically using something called a *grammar*, see Examples 6.27 and 6.28. These topics are taught in COMP11212, but the recursive nature of the definitions is not the main focus, so we briefly discuss that here.

We finally look at more mathematical examples, based on the natural numbers. A recurring theme on this course unit has been the fact that mathematics is all about rigour. There are a number of formal definitions in the notes in preceding chapters, while we have taken for granted certain facts about sets of numbers, and familiar operations on these. In this and the following chapter we give an idea of how these sets of numbers, and their operations, can be formally defined. This allows us to formally prove properties for these operations, such as commutativity or associativity, that we have taken for granted. This illustrates that rigour in mathematics is not something built on top of systems of numbers, but instead these can be formally defined, and we don't have to rely on 'common sense' or 'experience' to justify properties of their operations.

6.1 Lists

We begin our study of recursion by looking at lists. Lists are a standard recursive data type. The idea is quite simple: For a set S we can have a list of elements from S, say

$$[s_n, s_{n-1}, \dots, s_2, s_1].$$

For example, this might be a list of pages relevant to a particular topic, or a list of heights of people that somebody has measured, or a list of students on a particular course unit.

6.1.1 Lists defined

We can think of a list as satisfying exactly one of the following conditions:

- The list is empty, that is, there are no elements in the list.[1]
- The list consists of at least one element, and we can think of it as being a smaller list, say l, together with an added element, say s.

The list

$$[4, 3, 2, 1]$$

[1] For example, if we have a program that calculates which elements should be added to the list we would like it to start from an empty list.

is the result of appending[2] the number

$$4 \qquad \text{to the list} \qquad [3,2,1],$$

which in turn is the result of appending the number

$$3 \qquad \text{to the list} \qquad [2,1],$$

which in turn is the result of appending[3] the number

$$2 \qquad \text{to the list} \qquad [1],$$

which in turn is the result of appending the number

$$1 \qquad \text{to the list} \qquad [\,],$$

We can use this idea to give a formal definition of lists over a set S, and we can use the same idea to define operations on such lists.

Definition 51: list

A list over a set S is recursively defined as follows.

Base case list. There is an *empty list* $[\,]$.

Step case list. Given a list l over S and an element s of S there is a list $s: \ l$ where s has been appended to l.

We use

$$\mathsf{Lists}_S$$

for the set of all lists over a given set S.

Example 6.1. What is written above regarding the list

$$[4,3,2,1]$$

can now be written in the notation that is introduced in the formal definition of a list to read

$$\begin{aligned}[4,3,2,1] &= 4: \ [3,2,1]\\ &= 4: \ 3: \ [2,1]\\ &= 4: \ 3: \ 2: \ [1]\\ &= 4: \ 3: \ 2: \ 1: \ [\,].\end{aligned}$$

Always having to deal with expressions like the one in the last row would be quite painful, but that is how a computer thinks of a list which is given by a list element and a pointer to the remainder of the list. Human readability is improved, however, by using expressions like that in the top row.

These kinds of recursive definitions are very typical for functional programming languages such as Haskell or ML, but when programming with lists in C you will find a similar idea: An object of class List

[2] We use terminology from the language Python for our lists. In MLthe *append* operation concatenates two lists, and the *cons* operation adds an element to the list.

[3] We have made a somewhat arbitrary decision here to append elements to the left of the list. Instead we could have used $[1,2,3]$ to mean the list which arises from appending 1, 2 and 3 (in that order) to the empty list.

- is empty or
- it consists of
  - an element of the list and
  - a pointer to the remainder of the list (if there is one).

Code Example 6.1. Here's a class that implements this kind of list in Java, where the elements of the list are integers.

```
public class List {
    public int value;
    public List next;

    public List (int s, List l)
    {value = s; next = l;}
   }
```

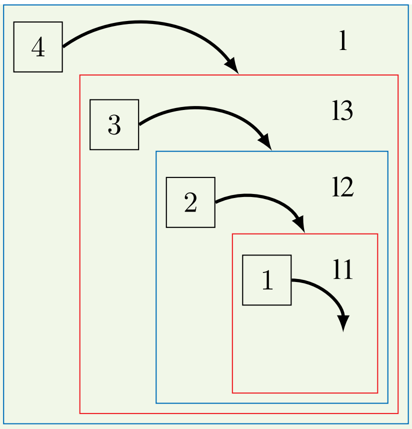
We have to cheat slightly to deal with empty lists: We do this by using the value **null** to indicate that an empty list is being referenced.
So an object of the class List is

- the empty list if we get the value **null** when referring to it or
- it consists of an element value and a List object next.

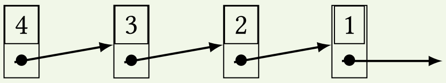
The fact that the class List is closely related to the lists defined in Definition 51 is not immediately obvious, but hopefully this explanation shows why they correspond to each other.
We give an idea of how to picture an object in this class. Assume that we have

- a List object l with l.value=4 and l.next=l3 and
- a List object l3 with l3.value=3 and l3.next=l2 and
- a List object l2 with l2.value=2 and l3.next=l1 and
- a List object l1 with l1.value=1 and l1.next=**null**.

You can picture these objects as follows:



Another picture that is sometimes used in this situation is the following.



6.1.2 Recursive definitions and proof by induction

Operations on such recursive datatypes are usually defined recursively. We give a number of examples for this particular construct.

Example 6.2. We define a very simple-minded function from Lists_S to Lists_S, which maps a given list l over S to the empty list. Because this function is very simple we do not need recursion to define it, we could merely set

$$l \longmapsto [\,].$$

But the point of this example is to introduce recursive definitions, and so we show here how to define the same function recursively. For this purpose we have to give it a name, say $k_{[\,]}$ since it is the constant function which maps everything to the empty list. Note that the following definition is inefficient, and one would not use it to program this function, but it is useful as a first simple example for how recursive functions work.

Base case $k_{[\,]}$. $\quad k_{[\,]}[\,] = [\,]$ and

Step case $k_{[\,]}$. $\quad k_{[\,]}(s : l) = k_{[\,]}l,$

Note that this definition matches the definition of a list: We have to say what the function does if its argument is the base case list, that is, the empty list $[\,]$, and we have to say what it does if its argument is a list built using the step case, so it is of the form $s : l$ for a list l.

The way this function works is to map

- the empty list to the empty list,
- and a non-empty list, which has an element s added to some list l, to the result of applying the function to l.

This is the typical shape of a recursive function on a recursive data type:

- it specifies what to do for the base case(s) of the data type and
- it specifies what to do for the step case(s).

Example 6.3. We continue Example 6.2 by carrying out a sample calculation to see how this definition allows us to compute the value of the function for a specific list, say $[i, 1+2i, 4]$ (a list over $\mathbb{C}$). We justify each step by referring to the definition of a list, Definition 51, and the definition of $k_{[\,]}$.

$$\begin{aligned}
k_{[\,]}[i, 1+2i, 4] &= k_{[\,]}(i : [1+2i, 4]) && \text{step case list} \\
&= k_{[\,]}[1+2i, 4] && \text{step case } k_{[\,]} \\
&= k_{[\,]}(1+2i : [4]) && \text{step case list} \\
&= k_{[\,]}[4] && \text{step case } k_{[\,]} \\
&= k_{[\,]}(4 : [\,]) && \text{step case list} \\
&= k_{[\,]}[\,] && \text{step case } k_{[\,]} \\
&= [\,] && \text{base case } k_{[\,]}.
\end{aligned}$$

Note that in a typical implementation it would not be necessary to invoke *step case list*—it is our notation for lists which requires this.

Code Example 6.2. A code snippet that implements this function as a method knull for an object l of class List looks as follows:

```
public static List knull (List l)
{
if (l == null)
    return null;
else
    return knull(l.next);
}
```

The way a computer carries out the corresponding calculation looks a bit different to the sample calculation given above: Instead of manipulating an expression that describes the output a computer stores each call to the recursively defined method, and that requires it to also store all the local variables.

Code Example 6.3. We look at the function calls and returns in an example. Assume we are calling knull(l), where l is the list from Code Example 6.1. The calls carried out by the program are as follows:

```
knull(l)
    knull(l3)
        knull(l2)
            knull(l1)
            knull(null)
            return null
          return null
        return null
     return null
   return null
```

This looks a bit boring but becomes more interesting if the function that is being implemented is more interesting, see Example 6.6.

If we put the mathematical definition of the function next to the implementation the similarities are very clear:

Base case $k_{[\,]}$. $k_{[\,]}[\,] = [\,]$ and

Step case $k_{[\,]}$. $k_{[\,]}(s : l) = k_{[\,]}l$,

```
public static List knull (List l)
{
if (l == null)
   return null;
else
   return knull(l.next);
}
```

It is not completely obvious from the definition that the function $k_{[\,]}$ as given there does indeed map every list to the empty list. We prove this formally as our first example of a proof by *induction*.

We want to show that for all lists l in Lists_S it is the case that

$$k_{[\,]}l = [\,].$$

Such a proof also follows the formal definition of the underlying data type. The pattern consists of a proof for the base case, a proof of the step case, and (optionally here) in between the statement of the *induction hypothesis*.

Base case list. We have to show that the statement holds for the empty list, that is $k_{[\,]}[\,] = [\,]$.

Induction hypothesis. We assume the statement holds for the list[4] l, that is we have

$$k_{[\,]}l = [\,].$$

Step case list. We have to show that given the induction hypothesis the statement holds for lists of the form $s : l$, that is we have

$$k_{[\,]}(s : l) = [\,].$$

[4] Note that sometimes we have to assume that the statement holds for all lists of a given length, or some other statement applying to more than one list.

Example 6.4. We continue with Example 6.2 and illustrate the formal proof of the statement from above.

Base case list. $k_{[\,]}[\,] = [\,]$. This is a simple application of the base case of the definition of the function $k_{[\,]}$.

Induction hypothesis. For the list l we have

$$k_{[\,]}l = [\,].$$

Step case list. We check that

$$\begin{aligned} k_{[\,]}(s:\ l) &= k_{[\,]}l && \text{step case}k_{[\,]} \\ &= [\,] && \text{induction hypothesis.} \end{aligned}$$

As is typical the base case is obvious, and it typically does not require many steps, while the step case has a little more substance.

Note that we were able to define this function without having to specify from which set S our lists take their elements.

Let's pause a moment to think about how this proof works. The base case is fairly easy to understand.

It seems as if in the middle of the proof, where the induction hypothesis is stated, we are assuming the very same thing we aim to prove.

This is not so, however. An analogy that is often invoked is that of a line of dominoes. Assume that you have got a line of dominoes, standing on their short side, starting in front of you, and extending to the right (you may imagine infinitely many dominoes).

Base case. The first domino falls over, to the right.

Induction hypothesis The nth domino falls over to the right.

Step case. If the nth domino falls over to the right then the $(n+1)$th domino falls over to the right.

In this way of writing the proof the induction hypothesis is often skipped because it appears in the step case (*'If the nth domino falls over …'*). If we do not state it explicitly we do not lose any information. After the first few examples I do not note the induction hypothesis explicitly if it is a precise copy of the statement we are proving; See Section 6.4 for examples where more sophisticated induction hypotheses appear.

The base and the step case together are sufficient to guarantee that if the first domino falls over to the right (the base case) then *all* dominoes fall over.

Another analogy you may find helpful is that of climbing a ladder.

Base case. I can climb the first rung of the ladder.

Step case. If I can get to the nth rung of the ladder then I can climb to the $(n+1)$th one.

If we have both these properties, then we can get onto the first, and all subsequent, rungs of the ladder.

Going back to the case of lists, the base case ensures that we know the desired result for the empty list, and the step case ensures that if we know the result for the list l then we can prove it for the list where some element s had been added to l to form $s\,:\,l$.

Example 6.5. We look at the proof that $k_{[\,]}l = [\,]$ for a specific list l. Let

$$l = [s_3, s_2, s_1] = s_3\,:\,(s_2\,:\,(s_1\,:\,[\,])).$$

We can see that in order to show that

$$k_{[\,]}(s_3\,:\,(s_2\,:\,(s_1\,:\,[\,]))) = [\,]$$

we need the step case three times:

$$\begin{aligned} k_{[\,]}(s_3\,:\,(s_2\,:\,(s_1\,:\,[\,]))) &= k_{[\,]}(s_2\,:\,(s_1\,:\,[\,])) && \text{step case } k_{[\,]} \\ &= k_{[\,]}(s_1\,:\,[\,]) && \text{step case } k_{[\,]} \\ &= k_{[\,]}[\,] && \text{step case } k_{[\,]} \\ &= [\,] && \text{base case } k_{[\,]} \end{aligned}$$

We can also see that if we have a list with five elements then we need the step case five times. A proof by induction takes a shortcut: by proving the base case, and that given a list with n elements which has the desired property we can show that adding another element to the list results in a list that also has the property in question, we can establish the property for all finite lists.

Note that sometimes in order to establish the step case (here $s\,:\,l$) we have to assume that the induction hypothesis for *all* entities which are somehow smaller than the current one. In the case of lists it might be that we assume the induction hypothesis for all lists we have built on the way to reach the list l, or possibly even for all lists which have at most as many elements as l.

A proof by induction for lists always takes the following shape:

Base case list. The property is established for the base case for lists, that is: We show it works for $[\,]$. If the property we want to show is an equality then we get the statement we want to prove by inserting the empty list $[\,]$ for every occurrence of l in the equality, so for example

$$k_{[\,]}l = [\,]$$

becomes

$$k_{[\,]}[\,] = [\,].,$$

and[5]

$$\text{sum}\,\text{rev}\,l = \text{sum}\,l$$

becomes

$$\text{sum}\,\text{rev}\,[\,] = \text{sum}\,[\,].$$

[5] Read on to find out how the operations used here are defined, but that is not required for understanding the point that is made here.

Ind hypothesis. We assume that the statement holds for the list[6] l.

Step case list. Assuming the induction hypothesis we show that the property holds for the list $s : l$. We obtain the statement we have to show by replacing every occurrence of l by $s : l$, , so

$$k_{[\,]} l = [\,]$$

becomes

$$k_{[\,]}(s : l) = [\,].,$$

and

$$\text{sum}\,\text{rev}\,l = \text{sum}\,l$$

becomes

$$\text{sum}\,\text{rev}(s : l) = \text{sum}(s : l).$$

We summarize these ideas as follows.

Tip

A proof by induction for lists over a given set S always has the following shape. Assume we are trying to show a statement given for all $l \in \mathsf{Lists}_S$.

Base case list. Prove the given statement for the case where all occurrences of l have been replaced by $[\,]$.

Ind hyp Assume the given statement holds for the list l.[7]

Step case list. Prove the statement where all occurrences of l have been replaced by $s : l$, where s is an arbitrary element of S. The induction hypothesis is used as part of the proof.

Note that when proving something by induction for a recursively defined structure other than the natural numbers, people often speak of *structural induction*, because the structure of the data type gives the shape of the induction argument.

6.1.3 Operations on lists

What more sophisticated operations are available on lists? Well, for example one might add up all the numbers in a list.

Example 6.6. The following is part of a previous exam question. Assume that N is a set of numbers $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ or $\mathbb{C}$. We want to define a function

$$\text{sum}\colon \mathsf{Lists}_N \longrightarrow N$$

which adds up all the members of a given list. In other words we would like to have

$$\text{sum}[4, 3, 2, 1] = 10.$$

[6] Note that for advanced properties one may have to extend this, and, for example, assume the statement for all lists built on the way to reaching l, or that it holds for all lists that have at most the same number of elements as l, but on this course we have no examples like that.

[7] It may be necessary to assume it for l and all sublists of l.

Tip

Before defining an operation that is described verbally it can be a good idea to think about what that operation should do to a concrete example to make sure that you understand the instructions given.

Example 6.7. Continuing our example we ask ourselves how do we define such an operation recursively? We have to ask ourselves: What do we want to happen in the base case? So what should it mean to add up all the numbers that appear in the empty list? The answer is that we should get the unit[8] for addition, 0.

We then have to think about what we want to happen in the step case, so *knowing the result of* $\mathrm{sum}\, l$, what should the result of $\mathrm{sum}(s : l)$ be? It can sometimes help to write it like that, namely

$$\mathrm{sum}(s : l) = ??? \; \mathrm{sum}\, l.$$

Well, if we already know what happens if we add up all the numbers of the list l, then to add up all the numbers in the list $s : l$, we just have to add s to the result. These considerations lead to the following definition.

Base case sum. $\quad \mathrm{sum}\,[\,] = 0.$

Step case sum. $\quad \mathrm{sum}(s : l) = s + \mathrm{sum}\, l.$

Note how this definition very closely follows the shape of the definition of a list. This is one of the hallmarks of definition by recursion. If you understand the definition of the underlying entity you know what shape an operation for the type will have. A sample calculation (this time without using our notation for lists to stay closer to the programming example below) looks as follows:

$$\begin{aligned}
&\mathrm{sum}(4 : (3 : (2 : (1 : [\,])))) && \\
&= 4 + \mathrm{sum}(3 : (2 : (1 : [\,]))) && \text{step case sum} \\
&= 4 + 3 + \mathrm{sum}(2 : (1 : [\,])) && \text{step case sum} \\
&= 4 + 3 + 2 + \mathrm{sum}(1 : [\,]) && \text{step case sum} \\
&= 4 + 3 + 2 + 1 + \mathrm{sum}\,[\,] && \text{step case sum} \\
&= 4 + 3 + 2 + 1 + 0. &&
\end{aligned}$$

See Example 6.11 for an inductive proof of a property of this function.

Tip

When you have defined an operation recursively it is always a good idea to apply it to an example to make sure that your definition makes sense. Note how this is done in the preceding example.

[8] If you were wondering how to deal with the empty list then note that adding no elements at all is usually taken to describe the number 0. More generally, for an operation with unit e, applying the operation to 0 many elements should return e.

Code Example 6.4. For our class List this method could be implemented by the following method add.

```
public static int sum (List l)
{
if (l == null)
    return 0;
else
    return l.value + sum(l.next);
}
```

We can work out the function calls for the calculation for the lists l, l3, l2, l1 as in Code Example 6.1, as follows:

```
sum(l)
  sum(l3)
    sum(l2)
      sum(l1)
        sum(null)
        return 0
      return 1 + 0
    return 2 + 1
  return 3 + 3
return 4 + 6
```

You can see why the computer has to create a stack which contains all the information needed to know what to do with the various return values and where to continue the computation.
Again note the similarities between the mathematical function definition and the code implementing it.

Base case $\mathrm{sum}\,[\,] = 0.$

Step case $\mathrm{sum}(s : l) = s + \mathrm{sum}\, l.$

```
public static int sum (List l)
{
if (l == null)
    return 0;
else
    return l.value + sum(l.next);
}
```

In general, defining a new operation for lists always takes the following shape:

Base case We have to say what the operation should do when given the base case of a list, that is, $[\,]$.

Step case. Assuming that we already know what the operation does for the list l we define what it does when the argument is $s : l$.

We define an operation

$$\mathsf{Lists}_S \times \mathsf{Lists}_S \longrightarrow \mathsf{Lists}_S$$

for a more interesting example.

Example 6.8. We want to define an operation that takes as input two lists, and returns one list where the two have been stuck together, that is, *concatenated*, so

$$[1,2,3] +\!\!+ [4,3,2,1] = [1,2,3,4,3,2,1].$$

We would like to have a formal definition of this operation. This formal definition can then be turned into code that implements this operation.
Now that we have two arguments we need to think about how to work that into the definition. For simple operations such as this one it is possible to consider one argument as a *parameter* and give a recursive definition in terms of the other argument. In this case we can leave the right argument alone an only have to look inside the left one, which is why we recurse over the *left* argument.
We define the operation

$$+\!\!+ : \mathsf{Lists}_S \times \mathsf{Lists}_S \longrightarrow \mathsf{Lists}_S$$

by setting

Base case $+\!\!+$. $[\,] +\!\!+ l' = l'$ and

Step case $+\!\!+$. $(s : l) +\!\!+ l' = s : (l +\!\!+ l')$,

where it is understood that l and l' are elements of Lists_S.

Note that the only way to define this is by recursion over the left argument. If we try to do it the other way round we have no way of defining the list we want from

$$l +\!\!+ (s : l').$$

In Exercise 143 we look at a way of doing this recursing over the right argument, but that requires another operation and is considerably more complicated.

Example 6.9. We continue Example 6.8 by illustrating how this operation works in practice. We show how two lists are concatenated step by step by following the process defined by this definition.

$$\begin{aligned}
[4,3] +\!\!+ [2,1] &= (4 : [3]) +\!\!+ [2,1] && \text{step case list}\\
&= 4 : ([3] +\!\!+ [2,1]) && \text{step case } +\!\!+\\
&= 4 : ((3 : [\,]) +\!\!+ [2,1]) && \text{step case list}\\
&= 4 : 3 : ([\,] +\!\!+ [2,1]) && \text{step case } +\!\!+\\
&= 4 : 3 : [2,1] && \text{base case } +\!\!+
\end{aligned}$$

$$\begin{aligned} &= 4 :\ [3,2,1] \\ &= [4,3,2,1]. \end{aligned}$$

Code Example 6.5. We give the code that corresponds to the mathematical definition.

```
public static List concat (List l, List l2)
{
if (l == null)
    return l2;
else
    return new List (l.value, concat(l.next,l2));
}
```

The code also illustrates why we have to use recursion over the left argument (try writing code which does it over the right argument if you can't see why—you just don't have access to the relevant part of the list).

From the point of view of programming we usually stop at defining operations, but when writing a compiler we have to worry about the properties of such operations. We illustrate here how the proof principle of induction allows us to establish properties for operations that have been defined using the principle of recursion. Such proofs once again follow the shape of the original definition of the operation, which in turn follows the shape of the definition of the original entity.

Example 6.10. We show that the empty list works like a unit on the right, that for all lists l over a set S, and all $s \in S$, we have

$$l \mathbin{+\!\!+} [\,] = l.$$

We follow the shape of the definition of the concatenation operation, by first considering the base case.

Base case list. We note that

$$[\,] \mathbin{+\!\!+} [\,] = [\,] \qquad \text{base case } \mathbin{+\!\!+}\ .$$

ind hyp We assume that the statement holds for the list l, that is

$$l \mathbin{+\!\!+} [\,] = l.$$

We turn to the step case.

Step case list. We calculate

$$\begin{aligned} (s :\ l) \mathbin{+\!\!+} [\,] &= s :\ (l \mathbin{+\!\!+} [\,]) && \text{step case} \mathbin{+\!\!+} \\ &= s :\ l && \text{induction hypothesis.} \end{aligned}$$

Taken together these two cases cover all possibilities, and they form a proof by *induction*.

Note that we have immediately from the base case of the definition of $\mathbin{+\!\!+}$ that $[\,]$ acts like a unit on the left, so overall we can see that it is the unit for this operation.

Below on we *do not write the induction hypothesis explicitly* if it consists of a statement that is identical to the one we are proving. In other words in examples of induction proofs which do not have an induction hypothesis spelled out you may assume that it looks like the statement we are proving.

CExercise 140. Answer the following questions. Give an argument for your answer, either by providing a counter example or by giving a proof by induction. *Hint: If you want to see more examples for proofs by induction then read on to Examples 6.11, 6.15 and 6.16.*

Compare this question with the one about concatenating strings, as part of Exercises 27–29.

Do not forget to justify each step of your proofs.

(a) Is the $\mathbin{+\!\!+}$ operation commutative?

(b) Is the $\mathbin{+\!\!+}$ operation associative?

(c) Does the $\mathbin{+\!\!+}$ operation have a unit? If yes, what is it? Justify your answer.

Example 6.11. The operations sum (see Examples 6.6 and 6.7) and $\mathbin{+\!\!+}$ (see Example 6.8) are defined above. We show by induction[9] that for lists l and l' in Lists_N it is the case that

$$\mathrm{sum}(l \mathbin{+\!\!+} l') = \mathrm{sum}\, l + \mathrm{sum}\, l'.$$

Base case list.

$$\begin{aligned} \mathrm{sum}([\,] \mathbin{+\!\!+} l') &= \mathrm{sum}\, l' && \text{base case } \mathbin{+\!\!+} \\ &= 0 + \mathrm{sum}\, l' && 0 \text{ unit for } + \\ &= \mathrm{sum}\, [\,] \mathbin{+\!\!+} \mathrm{sum}\, l' && \text{base case } \mathrm{sum}\,. \end{aligned}$$

Step case list.

$$\begin{aligned} &\mathrm{sum}((s : \ l) \mathbin{+\!\!+} l') \\ &\qquad = \mathrm{sum}(s : \ (l \mathbin{+\!\!+} l')) && \text{step case } \mathbin{+\!\!+} \\ &\qquad = s + \mathrm{sum}(l \mathbin{+\!\!+} l') && \text{step case } \mathrm{sum} \\ &\qquad = s + (\mathrm{sum}\, l + \mathrm{sum}\, l') && \text{ind hyp} \\ &\qquad = (s + \mathrm{sum}\, l) + \mathrm{sum}\, l' && + \text{ associative} \\ &\qquad = \mathrm{sum}(s : \ l) + \mathrm{sum}\, l' && \text{step case } \mathrm{sum}\,. \end{aligned}$$

[9] As previously discussed we do not state the induction hypothesis explicitly since its statement is identical to the statement we are proving.

Tip

Defining a function recursively can be quite tricky when you are not used to it. To get started you might as well make use of the fact that you know what such a definition needs to look like. Assume we intend to define a function

$$f\colon \mathsf{Lists}_{\mathbb{N}} \longrightarrow S$$

that behaves in a particular way. Then we know we need to have two cases:

Base case f. $\qquad f[\,] = ?$

Step case f. $\qquad f(s : l) = ???\ fl.$

It should be easy to read off from your description of f how to define it in the base case. For the step case you may require a bit of creativity. How does knowing fl help us to calculate $f(s : l)$? Note that you are allowed to use operations from the target set of f, here S. If you can answer that question you should be able to give a correct definition. Then check your definition using an example.

Another useful operation is determining the length of a list. You are asked to define this operator yourself in the following exercise, but assume for the remainder of this section that there is a function

$$\mathrm{len}\colon \mathsf{Lists}_S \longrightarrow \mathbb{N}$$

which, given a list, returns the number of elements in the list.

CExercise 141. This exercise concerns the length function described in the preceding paragraph.

(a) Give a definition of this len function.

(b) Use your definition to calculate $\mathrm{len}[3, 2, 1]$ step by step.

(c) Give the code for the corresponding function for objects of type List.

(d) Show that for all lists l and l' over a set S we have

$$\mathrm{len}(l +\!\!+\, l') = \mathrm{len}\, l + \mathrm{len}\, l'.$$

Justify each step.

Exercise 142. For lists over $\mathbb{N}$ give a recursive definition of a list being ordered,[10] that is, for example for the list

$$[k, l, m, n]$$

we demand $k \geq l \geq m \geq n$. *Hint: You want two base cases.*

We define additional operations for lists.

[10] We have chosen here to have the elements to get larger as they are added to the list—the opposite choice would also make sense.

Example 6.12. We can also reverse a list, that is turn it back to front.

Base case rev. $\quad \mathrm{rev}\,[\,] = [\,]$

Step case rev. $\quad \mathrm{rev}(s : l) = \mathrm{rev}\, l +\!\!+ [s].$

You are asked to work out how this definition works in the following exercise.

Code Example 6.6. We give the code that corresponds to the mathematical definition, using the concat method defined in Code Example 6.5.

```
public static List reverse (List l)
{
if (l == null)
    return l;
else
    return concat(reverse(l.next), new List (l.value, null));
}
```

EExercise 143. Carry out the following for the rev operator.

(a) Calculate $\mathrm{rev}[1, 0]$ step by step.

(b) Show that for every list l over an arbitrary set $\mathrm{len}\,\mathrm{rev}\, l = \mathrm{len}\, l$. *Hint: You may want to use a statement from a previous assessed exercise to help with this.*

(c) Show that $\mathrm{rev}(l +\!\!+ l') = \mathrm{rev}\, l' +\!\!+ \mathrm{rev}\, l$.

(d) Show that $\mathrm{rev}\,\mathrm{rev}\, l = l$ for all lists l over S. *Hint: You may want to show* $\mathrm{rev}[s] = [s]$ *separately.*

(e) Use the rev operator to give an alternative definition of the concatenation operator which uses recursion over the *second* argument. Give an argument that your definition agrees with the original.

Justify each step in your proofs.

Example 6.13. We give one more example where we look at code. When we have two lists it seems easy to decide whether they are 'the same'. By that we mean they consist of the same elements in the same order. A human being can check this for two elements of the set Lists_S by inspection, and below we give a recursive procedure that carries out this check.
When we have two List objects, however, our idea of having 'the same list' may not agree with that of the Java programming language. The boolean value

```
(l1 == l2)
```

evaluates to true precisely when l1 and l2 refer to the same object. If we want to check whether two such objects consist of the same elements in the same order we have to write code of the following kind.

```
public static boolean equal (List l1, List l2)
{
  if (l1 == null)
       return (l2 == null);
  else {
       if (l2 == null)
            return false;
       else
            return (l1.value == l2.value && equal(l1.next, l2.next));
    }
}
```

In our heads we are effectively defining a function that takes a List object and turns it into an element of $\mathsf{Lists}_{\mathbb{Z}}$, and we consider two lists equal if this function maps them to the same list.
The code above can be turned into a mathematical definition quite easily. We want to define a function, say

$$f_{=}\colon \mathsf{Lists}_S \times \mathsf{Lists}_S \longrightarrow \{0,1\}\ .$$

Base case $f_{=}$: $[\,],[\,]$. $\qquad f_{=}([\,],[\,]) = 1.$

Base case $f_{=}$: $[\,], s' : l'$. $\qquad f_{=}([\,], s' : l') = 0.$

Base case $f_{=}$: $s : l, [\,]$. $\qquad f(s : l, [\,]) = 0.$

Step case $f_{=}$.

$$f_{=}(s : l, s' : l') = \begin{cases} f_{=}(l, l') & s = s' \\ 0 & \text{else.} \end{cases}$$

where it is understood that l and l' are elements of Lists_S.
Why is the mathematical definition so much longer? Because in the code we make use of the equality function for integers and references.
Assume we have two functions

$$\begin{aligned} f_{=[\,]}\colon \mathsf{Lists}_S &\longrightarrow \{0,1\} \\ l &\longmapsto \begin{cases} 1 & l = [\,] \\ 0 & \text{else} \end{cases} \end{aligned}$$

and

$$\begin{aligned} f_{=_S}\colon S\times S &\longrightarrow \{0,1\} \\ (s,s') &\longmapsto \begin{cases} 1 & s=s' \\ 0 & \text{else.} \end{cases} \end{aligned}$$

then we can define instead

Base case $f_=$ $\qquad f_=([\,],l') = f_{=[\,]}l'$.

Step case $f_=\ :[\,]$. $\qquad f_=(s\ :\ l,[\,]) = 0$.

Step case $f_=\ :s'\ :\ l'$. $\qquad f_=(s\ :\ l,s'\ :\ l') = f_{=_S}(s,s') \wedge f_=(l,l')$,

which is very close to the code given above.

Exercise 144. The aim of this exercise is to define a function

$$\begin{aligned} \text{search}\colon \mathsf{Lists}_S\times S &\longrightarrow \{0,1\} \\ (l,s) &\longmapsto \begin{cases} 1 & s \text{ occurs in } l \\ 0 & \text{else.} \end{cases} \end{aligned}$$

(a) Recursively define this function.

(b) If your definition is a definition by cases then change that, using the $f_{=_S}$ function from Example 6.13. *Hint: You may use boolean operations on the set* $\{0,1\}$.

(c) Write code that implements this function for objects of the class List.

Example 6.14. If we have a list l over a set S, and a way of translating elements of S to elements of some set T via a function $f\colon S\longrightarrow T$ we may transform our list to one over the set T by using the function map f defined below. The type of this function is

$$\text{map}\, f\colon \mathsf{Lists}_S \longrightarrow \mathsf{Lists}_T.$$

It is given by the following recursive definition.

Base case map. $\qquad (\text{map}\, f)[\,] = [\,]$

Step case map. $\qquad (\text{map}\, f)(s\ :\ l) = fs\ :\ (\text{map}\, f)l.$

Assume we have the function $f\colon \mathbb{N}\longrightarrow\mathbb{Z}$ given by

$$n \longmapsto -n.$$

Applying the map function to f and the list $[3,2,1]$ we get

$$(\text{map}\, f)[3,2,1] = (\text{map}\, f)(3\ :\ [2,1]) \qquad \text{Step case list}$$

$$\begin{aligned}
&= f3 :\ (\operatorname{map} f)[2,1] && \text{step case map}\\
&= -3 :\ (\operatorname{map} f)(2 :\ [1]) && \text{def } f \text{ and step case list}\\
&= -3 :\ f2 :\ (\operatorname{map} f)[1] && \text{step case map}\\
&= -3 :\ -2 :\ (\operatorname{map} f)(1 :\ [\,]) && \text{def } f \text{ and step case list}\\
&= -3 :\ -2 :\ (f1 :\ \operatorname{map} f[\,]) && \text{step case map}\\
&= -3 :\ -2 :\ (-1 :\ [\,]) && \text{def } f\text{, base case map}\\
&= -3 :\ -2 :\ [-1]\\
&= -3 :\ [-2,-1]\\
&= [-3,-2,-1].
\end{aligned}$$

We use the map operator to give additional examples for proofs by induction.

Example 6.15. We show that for the identity function id_S on a set S we have for all lists l over the set S that

$$(\operatorname{map} \mathsf{id}_S)l = l.$$

This is a proof by induction.

Base case list. We have $(\operatorname{map} \mathsf{id}_S)[\,] = [\,]$ by base case map.

Step case list. We note the following.[11]

$$\begin{aligned}
(\operatorname{map} \mathsf{id}_S)(s :\ l) &= \mathsf{id}_S s :\ (\operatorname{map} \mathsf{id}_S)l && \text{step case map}\\
&= s :\ l && \mathsf{id}_S s = s;\ \text{ind hyp.}
\end{aligned}$$

Example 6.16. We show another property for the map operator. For two functions $f\colon S \longrightarrow T$ and $g\colon T \longrightarrow U$ we have for all lists l over S that

$$(\operatorname{map}(g \circ f))l = (\operatorname{map} g)((\operatorname{map} f)l).$$

We carry out the proof by induction.

Base case list.

$$\begin{aligned}
(\operatorname{map}(g \circ f))[\,] &= [\,] && \text{base case map}\\
&= (\operatorname{map} g)[\,] && \text{base case map}\\
&= (\operatorname{map} g)((\operatorname{map} f[\,])) && \text{base case map}\,.
\end{aligned}$$

Step case list.

$$\begin{aligned}
&(\operatorname{map}(g \circ f))(s :\ l)\\
&\quad= (g \circ f)s :\ (\operatorname{map}(g \circ f))l && \text{step case map}\\
&\quad= g(fs) :\ (\operatorname{map} g)((\operatorname{map} f)l) && (g \circ f)s = g(fs),\ \text{ind hyp}\\
&\quad= (\operatorname{map} g)(fs :\ (\operatorname{map} f)l) && \text{step case map}\\
&\quad= (\operatorname{map} g)((\operatorname{map} f)(s :\ l)) && \text{step case map}\,.
\end{aligned}$$

[11] As discussed above we do not spell out the induction hypothesis explicitly since its statement is identical to the statement we are proving.

Exercise 145. Use the map operator to solve the following problems. For each, define the function that you would like to apply the map operator to, and give an example of a list with at least three elements and show step-by-step how the required transformation is carried out.

(a) Turn a list of lengths in metres to one with the same lengths in kilometres.

(b) Turn a list containing numbers from 1 to 7 into a list of the corresponding days of the week.

(c) Turn a list of UK cities into a list of distances from the corresponding cities to Manchester (you don't have to describe the function for this case formally—just describe the idea and give some of the values).

(d) Turn a list of course units in the School into a list of academic staff who are the leaders for the corresponding course units (you don't have to describe the function for this case formally—just describe the idea and give some of the values).

CExercise 146. Show the following by induction, justifying each step.

(a) If l is a list over an arbitrary set S and f a function from S to some set T then $\operatorname{len}((\operatorname{map} f)l) = \operatorname{len} l$.

(b) If l and l' are elements of Lists_S and $f\colon S \longrightarrow T$ then

$$(\operatorname{map} f)(l +\!\!+ l') = (\operatorname{map} f)l +\!\!+ (\operatorname{map} f)l'.$$

(c) If l is a list over an arbitrary set S and f a function from S to some set T then $(\operatorname{map} f)(\operatorname{rev} l) = \operatorname{rev}((\operatorname{map} f)l)$.

(d) Let k_0 be the function from $\mathbb{N}$ to $\mathbb{N}$ which maps every element to 0, that is

$$\begin{aligned} k_0\colon \mathbb{N} &\longrightarrow \mathbb{N} \\ n &\longmapsto 0. \end{aligned}$$

Assume that l is an element of $\mathsf{Lists}_{\mathbb{N}}$. Show by induction that for the list

$$(\operatorname{map} k_0)l$$

every member is equal to 0.

You may invoke previous parts even if you have not proved them.

Optional Exercise 25. This exercise explores further the connection between sets, lists over that set, and the map operator.

(a) Define a function i_S from a set S to lists over the set by mapping each element of S to the one-element list containing just the element s, that is

$$i_S\colon s \longmapsto [s].$$

Show that for all functions $f\colon S \longrightarrow T$, and all $s \in S$ we have

$$(\mathrm{map}\, f)(i_S s) = i_T(f s).$$

(b) Note that given a set S there is nothing stopping us from building lists over the set of lists over S, that is, lists whose elements are lists (over S).

So given a function $f\colon S \longrightarrow T$ we can use $\mathrm{map}(\mathrm{map}\, f)$ to map lists of lists over S to lists of lists over T. Give some examples of how this operation works.

(c) Given a list of lists over S we can 'flatten' this list into a list over S, for example by turning

$$[[2, 3, 4], [1, 5], [\,], [6]]$$

into

$$[2, 3, 4, 1, 5, 6].$$

Give a recursive definition of this 'flattening' operator, and let's call this j_S.

(d) Show that for all lists l over the set S it is the case that

$$j_S((\mathrm{map}\, i_S)l) = l.$$

(e) Show that for all functions $f\colon S \longrightarrow T$, and all lists L of lists over S it is the case that

$$(\mathrm{map}\, f)(j_S L) = j_T((\mathrm{map}(\mathrm{map}\, f))L).$$

We have sketched here some advanced structure from an area called category theory which tells us something about how lists over a set relate to the original set.

6.2 Trees

A datatype that appears in many programming languages is that of a binary tree. Binary trees do feature in the programming in Java course unit COMP16412, where you will also see them used to introduce notions of writing recursive code in that language.

6.2.1 Binary trees defined

We give a formal definition for trees where every node has 0 or two children.

Definition 52: tree

A **full binary tree with labels from a set S** is given by

Base case tree. For every element s of S there is a tree, `tree` s, consisting of just one node labelled with the element s.

Step case tree. Given two trees t and t' labelled over S, and an element s of S, there is a tree $\texttt{tree}_s(t, t')$.

We use

$$\mathsf{FBTrees}_S$$

for the set of all binary trees with labels from S.

Code Example 6.7. Again we show how to define a corresponding class[12] in Java, again for integers as the entries. To emphasize that this is a bit different from the mathematical notion defined above we use a slightly different name for this class.

```
public class BTree {
      public int value;
      public BTree left, right;

      public BTree (int s, BTree t1, BTree t2)
      {value = s; left = t1; right = t2;}
}
```

Compare this to the following definition from a Java textbook::

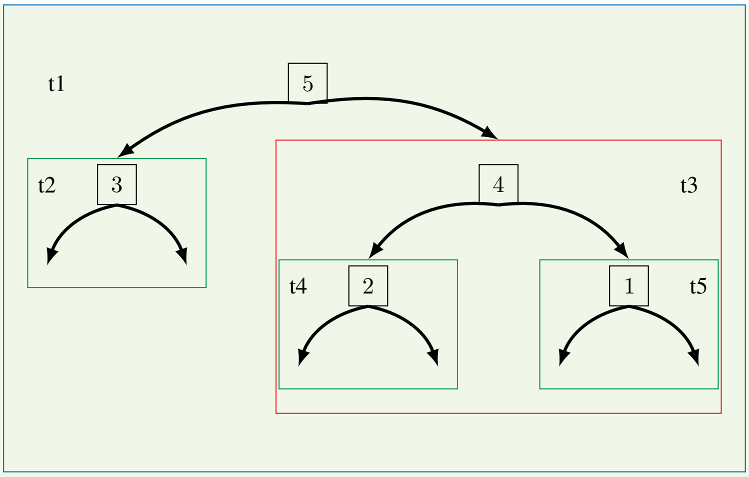
```
public class BinTreeNode
{
   private BinTreeNode left, right;
   private SomeClass nodeData;


   ...
}
```

Both these definitions of a class say that every binary tree node has two binary tree nodes, the left and the right one, and that it has a variable which contains data of some kind. In our case, the class SomeClass happens to be **int**, and with that instantiation we get our class BTree.
How can we picture an object of this class? We use the same idea as the visualization of an object of class List. Assume we have

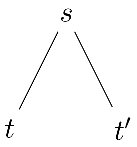
- a BTree object t with t.value=5 and t.left=t2 and t.right=t3 and
- a BTree object t2 with t2.value=3 and t2.left=**null** and t2.right=**null** and
- a BTree object t3 with t3.value=4 and t3.left=t4 and t3.right=t5 and
- a BTree object t4 with t4.value=2 and t4.left=**null** and t4.right=**null** and
- a BTree object t5 with t5.value=1 and t5.left=**null** and t5.right=**null**.

These objects may be pictured as follows,giving the whole tree in blue, the left subtree in red, and the right subtree (and its subtrees) in green.



See the following example for a less space-consuming way of drawing the corresponding full binary tree.

This kind of picture is a bit elaborate to draw, and in order to visualize instances of Definition 6.2 most people draw pictures such as the following for a tree with label s and left and right subtrees t and t' respectively.

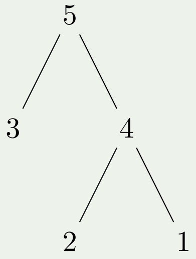


Note that t and t' are trees themselves, so the above does not give the full shape of the tree. To find that one has to look into how t and t' are defined.

Example 6.17. We look at the same tree as in Code Example 6.7, but this time we use the mathematical description.
If we take the complete description of a tree, such as[13]

$$\texttt{tree}_5(\texttt{tree}\,3, \texttt{tree}_4(\texttt{tree}\,2, \texttt{tree}\,1))$$

over $\mathbb{N}$, we may draw the full shape.

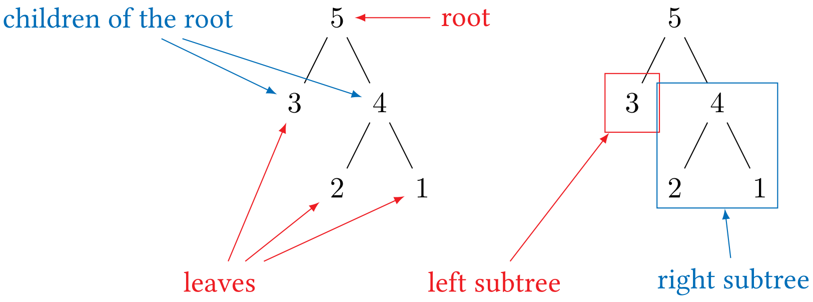


Note that in our definition there is no such thing as an empty tree. Every tree has at least one node.

[12] There is one difference between the mathematical definition and the code below. Can you see what it is? Think about **null** references.

[13] Note that elements of S may occur more than once in the tree—we give an example where this does not happen to make the connection with the picture clearer.

We introduce some nomenclature for trees. For the tree $\texttt{tree}_s(t, t')$ the node labelled with the s that appears in the description is the **root** of the resulting tree. In the above example that root has the label 5. The nodes labelled 3 and 4 are the **children** or the root node. Nodes that do not have any children are the **leaves** of the tree. Note that when you define a tree recursively then the leaves of the final tree are built by invoking the base case at the beginning of the building process. The inner nodes (that is, non-leaves) of the tree are built by invoking the step case. In our definition the two maximal subtrees of a tree are given one after the other (in an **ordered pair**), and so it makes sense to speak of the first of these as the left (rooted at the node labelled 3) and the second as the right (rooted at the node labelled 4) subtree.
We illustrate these definition with the following illustrations.



For completeness' sake we add two more definitions regarding trees. A binary tree is **perfect** if every leaf has the same distance to the root, namely the height of the tree. This is equivalent to demanding that on level i, counting down, with the root giving level 0, there are 2^i many nodes.
A binary tree is **complete** if and only if there are 2^i nodes on level i, with the possible exception of the final level, which must be 'filled' from left to right.

6.2.2 Operations on trees

Again we give examples for recursive definitions of operations for these recursively defined entities and give proofs by induction.

Code Example 6.8. For our Java class BTree we may want to know whether two objects describe the same tree structure. The following code does this for us.

```
public static boolean equal (BTree t1, BTree t2)
{
 if (t1 == null)
     return (t2 == null);
 else {
     if (t2 == null)
         return false;
     else
         return ((t1.value == t2.value)
         && equal (t1.left, t2.left)
```

```
            && equal (t1.right, t2.right));
        }
    }
```

This method returns true precisely when the two trees providing its arguments have the same structure as trees.

The **height of a tree** gives a (crude) notion of the size of the tree.[14] Compare the following definition with that of the length of a list, Exercise 141.

Example 6.18. We define the height function[15] recursively.

Base case hght. $\quad$ $\text{hght}\,\texttt{tree}\,s = 0.$

Step case hght. $\quad$ $\text{hght}\,\texttt{tree}_s(t, t') = \max\{\text{hght}\,t, \text{hght}\,t'\} + 1.$

We work out how this definition works for the sample tree from Example 6.17.

$$\begin{aligned}
&\text{hght}\,\texttt{tree}_5(\texttt{tree}\,3, \texttt{tree}_4(\texttt{tree}\,2, \texttt{tree}\,1)) && \\
&= \max\{\text{hght}\,\texttt{tree}\,3, \text{hght}\,\texttt{tree}_4(\texttt{tree}\,2, \texttt{tree}\,1)\} + 1 && \text{step case hght} \\
&= \max\{0, \max\{\text{hght}\,\texttt{tree}\,2, \text{hght}\,\texttt{tree}\,1\} + 1\} + 1 && \text{def hght} \\
&= \max\{0, \max\{0, 0\} + 1\} + 1 && \text{base case hght} \\
&= \max\{0, 1\} + 1 && \text{def max} \\
&= 1 + 1 && \text{def max} \\
&= 2 &&
\end{aligned}$$

Note that for this definition we have that the height of the tree is the integer part of the value of the logarithm (to base 2) of the number of nodes in the tree.

Code Example 6.9. We give the code that calculates the height of an object of class BTree.

```
public static int height (BTree t)
{
if (t.left == null && t.right == null)
    return 0;
else
    return 1 + vMath.max(height(t.left), height(t.right));
}
```

What happens if this program is called for a tree that consists of a **null** reference? This is where the distinction between our mathematical binary trees and the BTree class becomes significant.

[14] Some people call this the *depth* of the tree.

[15] Note that when defining the height of a tree it makes sense to either count the number of layers in a tree, or to count the number of connections between them. If we want to former we set the height of a one-node tree as 1, and the latter case is given here.

Example 6.19. We give a recursive definition of the function

$$\mathrm{no}\colon \mathsf{FBTrees}_S \longrightarrow \mathbb{N}$$

that counts the number of nodes in a tree in with labels from S.

Base case no. We set $\mathrm{no}(\mathtt{tree}\, s) = 1$.

Step case no. We set $\mathrm{no}(\mathtt{tree}_s(t,t')) = 1 + \mathrm{no}\, t + \mathrm{no}\, t'$.

Compare this definition with that of the length of a list, Exercise 141. We can show by induction that the number of nodes in a tree in the set $\mathsf{FBTrees}_N$ is odd.

Base case tree. We calculate $\mathrm{no}(\mathtt{tree}\, s) = 1$, which is odd.

Ind hyp. We assume that t and t' are binary trees each with an odd number of nodes.[16]

Step case tree. We check this case.

$$\mathrm{no}\,\mathtt{tree}_s(t,t') = 1 + \mathrm{no}\, t + \mathrm{no}\, t' \qquad \text{step case } \mathrm{no}\,.$$

By the induction hypothesis we can find k and k' in $\mathbb{N}$ such that

$$\mathrm{no}\, t = 2k+1 \qquad \text{and} \qquad \mathrm{no}\, t' = 2k'+1.$$

Hence we may continue the above calculation as follows.

$$\begin{aligned} \mathrm{no}\,\mathtt{tree}_s(t,t') &= 1 + \mathrm{no}\, t + \mathrm{no}\, t' && \text{step case } \mathrm{no} \\ &= 1 + 2k + 1 + 2k' + 1 && \text{ind hyp} \\ &= 2(k + k' + 1) + 1 && \text{calcs in } \mathbb{N} \end{aligned}$$

This is an odd number which completes the proof.

Note that this is an example where proving the desired property requires us to do a bit more than write a sequence of equalities.

Example 6.20. We show by induction that for every tree t in the set $\mathsf{FBTrees}_N$ it is the case that $\mathrm{hght}\, t \leq \mathrm{no}\, t$.

Base case tree.

$$\begin{aligned} \mathrm{hght}(\mathtt{tree}\, s) &= 0 && \text{base case } \mathrm{hght} \\ &\leq 1 \\ &= \mathrm{no}(\mathtt{tree}\, s) && \text{base case } \mathrm{no}\,. \end{aligned}$$

[16] This is the standard induction hypothesis for a data type with one step case which has two previously defined entities as inputs—we give it here because it's the first example of this kind.

Ind hyp. For the trees t and t' we have

$$\operatorname{hght} t \leq \operatorname{no} t \qquad \text{and} \qquad \operatorname{hght} t' \leq \operatorname{no} t'.$$

We observe that for all natural numbers m and n we have that

$$m \leq m + n \qquad\qquad \text{and} \qquad\qquad n \leq m + n,$$

and so

$$\max\{m, n\} \leq m + n.$$

For this reason the induction hypothesis implies

$$\max\{\operatorname{hght} t,\ \operatorname{hght}\} \leq \operatorname{hght} t + \operatorname{hght} t' \leq \operatorname{no} t + \operatorname{no} t',$$

which we use below.

Step case tree.

$$\begin{aligned}
&\operatorname{hght}(\texttt{tree}_s(t, t')) \\
&\qquad = \max\{\operatorname{hght} t, \operatorname{hght} t'\} + 1 && \text{step case } \operatorname{hght} \\
&\qquad \leq \max\{\operatorname{no} t,\ \operatorname{no} t'\} + 1 && \text{ind hyp} \\
&\qquad \leq \operatorname{no} t + \operatorname{no} t' + 1 && \max\{n, n'\} \leq n + n' \text{ in } \mathbb{N} \\
&\qquad = \operatorname{no} \texttt{tree}_s(t, t') && \text{step case } \operatorname{no}\,.
\end{aligned}$$

Tip

A proof by induction for trees in the set $\mathsf{FBTrees}_S$ always has the following shape. We are trying to prove a statement formulated in terms of the variable t which is an element of $\mathsf{FBTrees}_S$.

Base case tree. Prove the given statement for the case where all occurrences of t have been replaced by $\texttt{tree}\ s$, where s is an arbitrary element of S.

Ind hyp Assume the given statement holds for the trees[17] t and t'.

Step case tree. Prove the statement where all occurrences of t have been replaced by $\texttt{tree}_s(t, t')$, where s is an arbitrary element of S. The induction hypothesis is used as part of the proof.

Note how the general scheme derives from the shape of the definition of our trees.

Code Example 6.10. We give one last version of code for a recursive function. Note that once again the difference between our mathematical binary trees, which cannot be empty, and the trees of class BTree, which can, makes a difference.

[17] It may be necessary to assume it for t, t' and all their subtrees.

```
public static int no (BTree t)
{
if (t == null)
    return 0;
else
    return 1 + no(t.left) + no(t.right);
}
```

Example 6.21. We give a recursive definition of the function

$$\mathrm{lvs}\colon \mathsf{FBTrees}_S \longrightarrow \mathbb{N}$$

that counts the number of leaves in a tree in $\mathsf{FBTrees}_S$.

Base case lvs. $\quad \mathrm{lvs}\,\texttt{tree}\,s = 1.$

Step case lvs. $\quad \mathrm{lvs}\,\texttt{tree}_s(t, t') = \mathrm{lvs}\,t + \mathrm{lvs}\,t'.$

Note how little this definition differs from that of the function no. A small change in a recursive definition can lead to a very different effect.

Tip

To get started on defining a recursive function for trees you might as well make use of the fact that you know what such a definition needs to look like. Assume we intend to define a function

$$f\colon \mathsf{FBTrees}_{\mathbb{N}} \longrightarrow S$$

that behaves in a particular way. Then we know we need to have two cases:

Base case f. $\quad f\,\texttt{tree}\,n = ?$

Step case f. $\quad f(\texttt{tree}_n(t, t')) = ???\ ft\ ?\ ft'.$

It should be easy to read off from your description of f how to define it in the base case. For the step case you may require a bit of creativity. How does knowing ft and ft' help us to calculate $f(\texttt{tree}_n(t, t'))$? Note that you are allowed to use operations from the target set of f, here S. When you have a definition check whether it does the right thing using an example.

CExercise 147. Assume that N is a set of numbers, namely $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$.

(a) Draw the following tree: $\texttt{tree}_3(\texttt{tree}\,2, \texttt{tree}_{-5}(\texttt{tree}\,2, \texttt{tree}\,{-1}))$

(b) Define a function

$$\mathrm{sum}\colon \mathsf{FBTrees}_N \longrightarrow N$$

which sums up all the labels that occur in the tree (compare Example 6.6 where the analogous operation for lists is defined).

(c) Apply your function to the tree from part (a).

(d) Give the code for a function that performs the same job for an object of class BTree.

(e) Justifying each step show that if the set of labels is $\mathbb{N} \setminus \{0\}$ then for all binary trees t we have

$$\text{no}\, t \leq \text{sum}\, t.$$

Exercise 148. Consider the function

$$\text{search}\colon \mathsf{FBTrees}_S \times S \longrightarrow \{0, 1\}$$

which, for inputs t and s, returns 1 precisely when s occurs as a label in the tree t.

(a) Give a recursive definition for this function, You may use the function $f_{=_S}$ from Example 6.13 as well as the boolean operations on $\{0, 1\}$.

(b) Give code that implements this function for objects of class BTree.

CExercise 149. Assume we have a function $f\colon S \longrightarrow T$.

(a) Recursively define a function map f that takes a binary tree with labels from S to a binary tree with labels from T, that is

$$\text{map}\, f\colon \mathsf{FBTrees}_S \longrightarrow \mathsf{FBTrees}_T\ .$$

Hint: Consider the map *function for lists as an example.*

(b) For the function

$$\begin{aligned} f\colon \mathbb{N} &\longrightarrow \mathbb{N} \\ n &\longmapsto 2n \end{aligned}$$

apply the function map f you defined in part (a) step by step to the following tree.

$$\texttt{tree}_{17}(\texttt{tree}_5(\texttt{tree}\,3, \texttt{tree}\,19), \texttt{tree}_{25}(\texttt{tree}\,19, \texttt{tree}\,27))$$

(c) Show that for every such function f, and every tree t in the set $\mathsf{FBTrees}_S$, we have $\text{hght}((\text{map}\, f)t) = \text{hght}\, t$.

(d) Show that for every such function f and every tree t in the set $\mathsf{FBTrees}_S$ we have $\text{no}((\text{map}\, f)t) = \text{no}\, t$.

(e) Assume that N is a set of numbers between $\mathbb{N}$ and $\mathbb{R}$. Show that for the function

$$\begin{aligned} k_1\colon N &\longrightarrow N \\ n &\longmapsto 1 \end{aligned}$$

we have for all binary trees t that $\text{no}\, t = \text{sum}((\text{map}\, k_1)t)$.

Justify each step in your proofs.

EExercise 150. This exercise is concerned with defining a mathematical entity of binary trees that corresponds to the BTree class. We refer to such a tree as a **binary tree with labels from a set S**, and use BTrees_S for the set of all these trees.

(a) Give a definition, similar to Definition 52, of a mathematical entity of *binary trees* which corresponds to the BTree class.

(b) Recursively define an operation that takes two binary trees and returns a binary tree where the second argument has been added below the right-most leaf of the tree.

(c) Write code which implements the operation from the previous part.

Exercise 151. Give the following recursive definitions.

(a) Trees with labels from S such that every node has at most two children.*Hint: You may want to allow an empty tree. You may also want to draw some examples to give you an idea what you are looking for.*

(b) Trees with labels from S such that every node can have an arbitrary finite number of children.

Exercise 152. This exercise is concerned with *perfect binary trees.*

(a) Recursively define the set of all those full binary trees with labels from a set S that are perfect. Do so by describing, for each height, those elements of $\mathsf{FBTrees}_S$ which are perfect. *Hint: Have a look ahead to Example 6.43, and for each $n \in \mathbb{N}$ define a set PTrees_S^n of those elements of $\mathsf{FBTrees}_S$ which are perfect and have height n.*

(b) Show that for such a tree t we have

$$\operatorname{lvs} t = 2^{\operatorname{hght} t}.$$

(c) Show that it is indeed the case that, for $t \in \mathsf{PTrees}_S^n$, we have $\operatorname{hght} t = n$.

(d) Show that for a perfect tree t we have $\operatorname{lvs} t = 2^{\operatorname{hght} t}$.

(e) Show that for a perfect tree t we have

$$\operatorname{no} t = 2^{1+\operatorname{hght} t} - 1,$$

Conclude that

$$\operatorname{hght} t = \log(1 + \operatorname{no} t) - 1.$$

6.2.3 Ordered binary trees

Trees are useful structures when it comes to keeping data in a way that makes it easy to search. In Example 4.98 we looked at searching through an array, here we see in Exercise 154 how one can search through entries that are kept in a tree in an ordered way, rather than in an array. Note that the ideas that follow are usually employed for trees in binary trees rather than full binary trees.

Example 6.22. Sometimes we require the set of all the labels that occur in a tree, for example in order to define *ordered binary trees*, see below. The **set of labels** $\mathrm{lab}\, t$ **that occur in a tree** t with labels from S is given as follows:

Base case lab. We have $\mathrm{lab}(\texttt{tree}\, s) = \{s\}$.

Step case lab. We have $\mathrm{lab}(\texttt{tree}_s(t, t')) = \{s\} \cup \mathrm{lab}\, t \cup \mathrm{lab}\, t'$.

Note that this defines a function

$$\mathrm{lab}\colon \mathsf{FBTrees}_S \longrightarrow \mathcal{P}S,$$

since it maps a binary tree with labels from S to a subset of S.

If we have trees over a set of numbers N with $\mathbb{N} \subseteq N \subseteq \mathbb{R}$ then that set comes with an *order*[18] then we can recursively define what we mean by a binary tree over N being ordered.

Definition 53: ordered binary tree

A binary tree with labels from N is **ordered** under the following conditions.

Base case tree. A tree of the form $\texttt{tree}\, n$ is ordered.

Step case tree. A tree of the form $\texttt{tree}_n(t, t')$ is ordered if and only if

- For every $m \in \mathrm{lab}\, t$ we have $m \leq n$,
- for every $m' \in \mathrm{lab}\, t'$ we have $n \leq m'$ and
- t and t' are ordered.

Ordered binary trees are sometimes called *binary search trees.*

We use $\mathsf{OFBTrees}_S$ for the set of ordered full binary trees with labels from the set S.

In Java there the TreeSet class is used to store values in trees in ascending order, so you don't have to program your own data structure for this kind of thing. Note that there is a difference between that class and our mathematical definition.

Ordered binary trees are used to keep an unknown number of data in a structure that allows for the binary search algorithm to be defined. Typically one would like to be able to insert new nodes, or to delete them. For this application one cannot use ordered *full* binary trees because after an insertion or deletion the tree will no longer be full, and one really should stick to ordered binary trees. Binary trees are mathematically defined in Exercise 150.

Exercise 153. Which of the following trees are ordered over the obvious set of numbers? *Hint: You may find it easier to answer this question if you draw the tree first.*

[18] See Section 7.4.1 for a formal definition of this concept.

(a) $\mathtt{tree}_3(\mathtt{tree}\,2, \mathtt{tree}_{17}(\mathtt{tree}\,5, \mathtt{tree}\,19))$

(b) $\mathtt{tree}_5(\mathtt{tree}_2(\mathtt{tree}\,0, \mathtt{tree}\,1), \mathtt{tree}\,149))$.

(c) $\mathtt{tree}_{17}(\mathtt{tree}_5(\mathtt{tree}\,3, \mathtt{tree}\,19), \mathtt{tree}_{25}(\mathtt{tree}\,19, \mathtt{tree}\,27))$.

(d) $\mathtt{tree}_{25}(\mathtt{tree}_9(\mathtt{tree}\,4, \mathtt{tree}\,1), \mathtt{tree}_{49}(\mathtt{tree}\,36, \mathtt{tree}\,64))$.

(e) $\mathtt{tree}_{-3}(\mathtt{tree}_{-5}(\mathtt{tree}\,-7. \mathtt{tree}\,-4), \mathtt{tree}_5(\mathtt{tree}\,4, \mathtt{tree}\,7))$.

EExercise 154. Let N be a set of numbers between $\mathbb{N}$ and $\mathbb{R}$. Consider the function

$$\text{search}\colon \mathsf{OFBTrees}_N \times N \longrightarrow \{0, 1\}$$

that for an input consisting of a full[19] ordered binary tree t with labels from N and an element n of N gives 1 if n occurs as a label in t, and 0 otherwise.

(a) Give a recursive definition of search. We may think of this function as performing a binary search.

(b) Write code that implements this function, assuming we have a class OBTree which restricts BTree to ordered binary trees only.

(c) Draw a picture of the ordered binary tree

$$\mathtt{tree}_{-3}(\mathtt{tree}_{-5}(\mathtt{tree}\,-7, \mathtt{tree}\,-4), \mathtt{tree}_5(\mathtt{tree}\,3, \mathtt{tree}\,7)).$$

Imagine you want to add a node with label 4 to the tree. Where should that go? Draw your solution.[20]

(d) Write code that takes as inputs an object of the class OBTree and an integer and returns an object of class OBTree in which the second argument has been inserted if it is not already present. Use the previous part to guide you.

Hint: Your definition should take advantage of the fact that the tree is ordered and not look at every label.

6.3 Syntax

Many formal languages are recursively defined. Examples of these are computer languages, or the language of logical propositions, or the language of regular expressions that appears in COMP11212.
This allows us to recursively define operations as well as to give inductive proofs of properties of such operations. This is one of the reasons why understanding recursion and induction is very useful in computer science.
We describe some examples of this kind in this section.

[19] If you have solved EExercise 150 you may want to do the whole exercise for ordered binary trees without requiring fullness.

[20] Note that the result is not a full binary tree, but a binary tree, compare Exercise 150.

6.3.1 Strings

The simplest example of this kind is that of a string over a set S. We also speak of a **word over the alphabet** S.

Base case string. The empty string[21] ϵ is a string.

Step case string. If $s \in S$ and w is a string then ws is a string.

One can now give a recursive definition of concatenation for strings, and one can define the length of a string recursively, just as these operations are defined for lists. Indeed, lists and strings are closely related, see the following exercise for the precise connection.

Exercise 155. Carry out the following tasks. *This is a really nice exercise to work out whether you've understood all the concepts from this chapter.*

(a) Give a recursive definition for the concatenation $+$ of strings over a set S. Ditto reusing len and in particular $+$

(b) Give a recursive definition of the length len of a string over a set S.

(c) Show that if w and w' are strings over a set S then $\operatorname{len} w + \operatorname{len} w' = \operatorname{len}(w + w')$.

(d) Recursively define a function from lists over S to strings over S.

(e) Recursively define a function from strings over S to lists over S.

(f) Show by induction that your two functions are mutual inverses of each other.

(g) What can you say about your two functions regarding how they behave with respect to the len functions for strings and lists? What about these functions and concatenating strings, versus concatenating lists?

Justify each step in your proofs.

6.3.2 Logical propositions

We now turn to the language of logical propositions that is studied in Chapter 3. We recall the definition of the boolean interpretation of a propositional formula[22] from Section 3.2.1.

Base case prop. For every propositional variable[23] Z, Z is a proposition.

Step cases prop. Assume that A and B are propositions. Then the following are propositions.

Step case $\neg$. $(\neg A)$,

Step case $\wedge$. $(A \wedge B)$,

[21] Since it's a bit difficult to indicate something empty we use a symbol here. This is standard practice.

[22] Below we use the shorter 'proposition' instead of 'propositional formula'.

[23] In this section we use this letter for propositional variables.

Step case $\vee$. $(A \vee B)$ and

Step case $\rightarrow$. $(A \rightarrow B)$.

Example 6.23. The definition of an interpretation relative to a valuation of a proposition is also via recursion. Assume that we have a valuation v that gives a value to each propositional variable in the boolean algebra $\{0, 1\}$. We give the formal definition of the boolean interpretation of propositions. This is given by a function I_v which maps a proposition to its boolean interpretation relative to v. The source of this function is the set of all propositions, and its target is $\{0, 1\}$.

Base cases I_v. For a propositional variable Z we define $I_v Z = vZ$.

Step cases I_v. We define:

Step case $\neg$. $I_v(\neg A) = \neg I_v A$,

Step case $\wedge$. $I_v(A \wedge B) = I_v A \wedge I_v B$,

Step case $\vee$. $I_v(A \vee B) = I_v A \vee I_v B$ and

Step case $\rightarrow$. $I_v(A \rightarrow B) = I_v A \rightarrow I_v B$.

Note that to argue about the properties of semantic equivalence studied in the material on logic we did not use induction. To show, for example, that the boolean interpretation of $\neg\neg A$ is the same as that of A for all valuations is, when fully spelled out, an induction proof but it can be made plausible without that. See the following exercise for an idea of how this works formally.

Tip

A proof by induction for propositions usually takes the following shape:

Base cases prop. We show that the statement holds for every proposition of the form Z, where Z is a propositional variable.

Ind hyp. We assume that the statement holds for formulae A and B.

Step cases prop. We show that, given the induction hypothesis, the statement holds for formulae of the form

- $\neg A$,
- $A \wedge B$,
- $A \vee B$,
- $A \rightarrow B$.

CExercise 156. Give the formal definition of the powerset interpretation for the set of all propositions (see Section 3.2.2) relative to a valuation.

EExercise 157. Solve the following problems for propositional logic.

(a) Give a recursive definition of the function var which takes as its argument a propositional formula and returns the set of propositional variables that occur in that formula.

(b) Give a recursive definition of the subformula construction as a function subf which takes as its argument a propositional formula and returns the set of all its subformulae,[24] see Section 3.1.

(c) Justifying each step, show by induction that for all propositional formulae A we have

$$\text{var}\, A \subseteq \text{subf}\, A.$$

Note that the definition of a formal derivation in a natural deduction system is also recursive: The base cases are given by the axiom rules and the step cases by the other derivation rules.

6.3.3 Formal languages

We look at elements of formal languages as found in the course unit COMP11212.

Regular expressions

The following explains a recursive definition from Part 1 of COMP11212.

Example 6.24. Let Σ be a set of symbols. A **pattern** or **regular expression over** Σ is generated by the following recursive definition.

Base case regexp $\boldsymbol{\emptyset}$. The character $\boldsymbol{\emptyset}$ is a pattern;

Base case regexp $\boldsymbol{\epsilon}$. the character $\boldsymbol{\epsilon}$ is a pattern;

Base case regexp Σ. every symbol from Σ is a pattern;

Step case regexp $+$. if p_1 and p_2 are patterns then so is[25] (p_1p_2);

Step case regexp $|$. if p_1 and p_2 are patterns then so is $(p_1|p_2)$;

Step case regexp $*$. if p is a pattern then so is $(p*)$.

It is typical for formal languages that there are several base cases as well as several step cases. This is due to the fact that we have several notions of a 'primitive' term in our language (that is one that is not built from other terms), and we have several ways of putting terms together to get new terms.

Example 6.25. There is a related recursive definition which tells us when a string made up of symbols from Σ *matches* a given regular expression.
Let p be a pattern over a set of symbols Σ and let s be a string consisting of symbols from Σ. We say that s **matches** p if one of the following cases holds:

Base case regexp ϵ. The empty word ϵ matches the pattern $\boldsymbol{\epsilon}$.

Base case regexp Σ. A character x from Σ matches the pattern $p = x$.

Step case regexp $+$. The pattern p is a concatenation $p = (p_1p_2)$ and there are words s_1 and s_2 such that s_1 matches p_1, s_2 matches

[24] You will note that some people use formulas, and some people formulae, for the plural. The latter is the original Latin form, but in modern English some prefer to form an English plural.

[25] The name of this step case mentions an operator that does not appear in the term constructed, but it's not clear what better name there is for concatenation.

p_2 and s is the concatenation of s_1 and s_2.

Step case regexp $|$. The pattern p is an alternative $p = (p_1|p_2)$ and s matches p_1 or p_2 (it is allowed to match both).

Step case regexp $*$. The pattern p is of the form $p = (q*)$ and s can be written as a finite concatenation $s = s_1 s_2 \cdots s_n$ such that s_1, s_2, …, s_n all match q; this includes the case where s is empty (and thus an empty concatenation, with $n = 0$).

Noticeably there is no entry for Base case regexp $\boldsymbol{\emptyset}$, and the reason for this is that no string matches the pattern $\boldsymbol{\emptyset}$. Consequently there is no need to include that base case.

Example 6.26. A result that is given in the notes for COMP112, but not proved, is covered in the following (somewhat extended) example.
Given a regular expression p, we recursively define

$$\mathcal{L}(p)$$

as follows:

Base case regexp $\boldsymbol{\emptyset}$. $\mathcal{L}(\boldsymbol{\emptyset}) = \emptyset$.

Base case regexp ϵ. $\mathcal{L}(\epsilon) = \{\epsilon\}$.

Base case regexp Σ. $\mathcal{L}(x) = \{x\}$ for all $x \in \Sigma$.

Step case regexp $+$. $\mathcal{L}(p_1 p_2) = \mathcal{L}(p_1) \cdot \mathcal{L}(p_2)$.

Step case regexp $|$. $\mathcal{L}(p_1|p_2) = \mathcal{L}(p_1) \cup \mathcal{L}(p_2)$.

Step case regexp $*$. $\mathcal{L}(p*) = (\mathcal{L}(p))^*$.

The operations $\cdot$ and $(-)^{|}*$ for sets of strings are defined in the COMP112 notes.

Note that we can think of $\mathcal{L}$ as a function from the set of all regular expressions over the alphabet Σ to the powerset of the set of strings over Σ, $\mathcal{P}\Sigma^*$.
The claim made, but not proved, in the COMP112 notes is

Proposition

For all regular expressions p over the alphabet Σ we have that

$$\mathcal{L}(p) = \{s \in \Sigma^* \mid s \text{ matches } p\}.$$

Proof. One may prove this by induction. Note that the induction proof has the number of base and step cases indicated by the recursive definition of a regular expression.

Base case regexp $\boldsymbol{\emptyset}$. $\mathcal{L}(\boldsymbol{\emptyset}) = \emptyset$. Since no string matches the pattern $\boldsymbol{\emptyset}$ the set of strings that match this pattern is indeed the empty set.

Base case regexp ϵ. $\mathcal{L}(\epsilon) = \{\epsilon\}$. By definition, ϵ is the only string that matches the pattern $\emptyset$.

Base case regexp Σ. $\mathcal{L}(x) = \{x\}$ for all $x \in \Sigma$. By definition the string x is the only string that matches the pattern x.

Step case reg-ex $+$. $\mathcal{L}(p_1 p_2) = \mathcal{L}(p_1) \cdot \mathcal{L}(p_2)$. By definition of $\cdot$ for sets of words we have that,

$$\mathcal{L}(p_1) \cdot \mathcal{L}(p_2) = \{s_1 s_2 \mid s_1 \in \mathcal{L}(p_1), s_2 \in \mathcal{L}(p_2),$$

By the induction hypothesis, this latter set is the set of all concatenations of strings $s_1 s_2$ such that

$$s_1 \text{ matches } p_1 \qquad \text{and} \qquad s_2 \text{ matches } p_2,$$

and by definition, a string s matches the pattern $p_1 p_2$ if and only if there exist strings s_1 and s_2 such that

$$s_1 \text{ matches } p_1 \qquad \text{and} \qquad s_2 \text{ matches } s_2$$
$$\text{and} \qquad s = s_1 s_2,$$

which gives the claim.

Step case regexp $|$. $\mathcal{L}(p_1|p_2) = \mathcal{L}(p_1) \cup \mathcal{L}(p_2)$. By definition a string s matches $p_1|p_2$ if and only if at least one of

$$s \text{ matches } p_1 \qquad \text{or} \qquad s \text{ matches } p_2,$$

which by the induction hypothesis means that this is the case if and only if

$$s \in \mathcal{L}(p_1) \cup \mathcal{L}(p_2).$$

Step case regexp $*$. $\mathcal{L}(p*) = (\mathcal{L}(p))^*$. By definition the pattern $p*$ is matched by the string s matches if and only if there is

$$n \in \mathbb{N} \text{ and } s_1, s_2, \ldots s_n \in \Sigma^*$$

with

$$s = s_1 s_2 \cdots s_n$$
$$\text{and} \qquad s_i \text{ matches } p \ (0 \leq i \leq n).$$

But by the induction hypothesis, this means that

$$s_i \in \mathcal{L}(p) \text{ for } 0 \leq i \leq n,$$

and this is equivalent to

$$s = s_1 s_2 \cdots s_n \in (\mathcal{L}(p))^*$$

as required.

In the notes for COMP11212 no formal proof is given that for every regular expression there is a finite state automaton defining the same language—what is described there is effectively a sketch of such an argument. Such a proof would proceed by induction over the above definition, including all the given cases. If you look at the text that explains how to turn a regular expression into a finite state automaton you should be able to see how each base case, and each step case, is treated there. As a consequence the essence of the proof is given in those notes, and all you would have to do to complete it is to introduce the formal induction structure.

Context-free grammars

There are a number of recursive definitions connected with context-free grammars. If we look at the definition of a string being generated by a grammar we can see that these are instructions that follow recursive rules, using the following pattern:

Base case Γ. This is provided by the start symbol S.

Step cases Γ. We have a step case for each production rule.

If we generalize the definition of a string being generated by a grammar to strings consisting of both, terminal and non-terminal, symbols the recursive nature of the construction becomes obvious.

Example 6.27. The definition of when a string is *generated by a grammar* is one example:

Base case gen. by Γ. The string S is generated by Γ.

Step cases gen. by Γ. If $R \to Y$ is a rule of the grammar and the string X is generated by Γ then the string that results from replacing one occurrence of R in X by Y is a string generated by Γ.

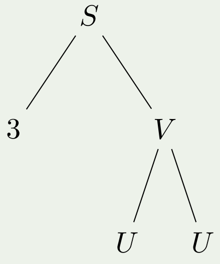
Example 6.28. Similarly the notion of a *parse tree*[26] for a grammar has an underlying recursive definition:

Base case parse tree for Γ. The tree consisting of only one node, which is labelled S, is a parse tree for Γ.

Step cases parse tree for Γ. If $R \to Y$ is a rule of the grammar and the tree t is generated by Γ then the tree that results from the following process is a parse tree for Γ:

- Identify a leaf labelled R.
- Add new children to that leaf, one for every symbol that occurs in Y in that order.

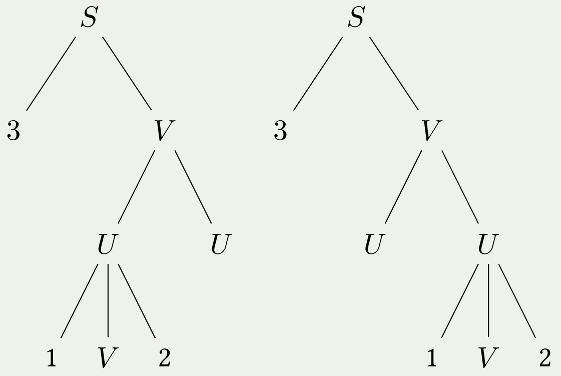
For example assume we have a grammar with non-terminal symbols S, U and V, and terminal symbols digits from 0 to 9, if we have a parse tree as follows.



and a rule

$$U \to 1V2$$

then both of the following trees are parse trees for Γ:



6.3.4 Elements of programming languages

Example 6.29. To give another example, assume that we have a set of numbers N, on which we have various operations and properties, see below for a concrete example. For such a set of numbers we may express calculations by forming terms that consist of various elements of N connected by the available operations. This gives us a language of such expressions. Many programming languages have such a notion of *arithmetic expressions* built in.
One might try to specify an arithmetic expression over the set N by the following recursive definition:

Base cases arex. For every $n \in N$ we have that n is an arithmetic expression.

Step cases arex. Assume that a and a' are arithmetic expressions. Then the following are arithmetic expressions:

Step case $()$. (a),

Step case $-$. $-a$,

Step case $^{-1}$. a^{-1},

Step case $+$. $a + a'$ and

Step case $\cdot$. $a \cdot a'$.

This is the kind of definition (see the While language below) that appears as part of

[26] In the COMP11212 notes the only parse trees drawn are those whose leaves are labelled with terminal symbols, but for a formal definition one needs to first allow a more general version.

a larger definition of a programming language.[27] If the set of numbers in question is some approximation to the real numbers then one might also have built-in operations such as the trigonometric functions, and there is probably support for exponentiation of numbers.

Typically the idea is that the programmer, as part of the code, uses arithmetic expressions to specify calculations to be carried out by the computer at run-time.

There's one problem with the definition from Example 6.29: Not all arithmetic expressions defined in this way work as intended: This is because we have done nothing to ensure that the multiplicative inverse is only applied to expressions which are different from 0. Some languages (such as Java) have mechanisms to deal with situations like this, for example by raising an exception, while in other cases one might actually get a result, but there are no guarantees that this result makes any sense.

This means that at run-time (or in some cases at compile-time) the machine will find itself being asked to carry out a multiplication by an inverse that does not exist, such as a division by 0 in the integers.

One would like to change the above definition by specifying a changed step case:

- If a is an arithmetic expression with $a \neq 0$ then a^{-1} is an arithmetic expression.

But in order to carry out the test to check whether an arithmetic expression is equal to 0, either the programmer, or the machine, would effectively have to do all the required calculations *before* the program is compiled. Clearly it would be pointless to expect the programmer to carry out this check when writing the program. After all, the point of writing code is to have the computer deal with such calculations. But it turns out that one can not ask the computer to do these calculations prior to run-time either. This is an issue related to the *Halting Problem*, which is explained in more detail in COMP11212.

Every programming language that supports arithmetic expressions has a *recursively defined evaluation procedure* for these expressions, which tell the machine how to calculate the *value* of such an arithmetic expression (or throw up an error if a division by 0 occurs).

For most programming languages the valid programs are defined recursively, typically using grammars. For most languages this definition is very large. Typically it involves defining more than one kind of entity. Below we give an example of a language that has a short definition to given an idea of how this might work. Note that this language is no less powerful (in a formally definable way) than, say, Java.

Example 6.30. We give the definition[28] of the While programming language as it appears in the notes for COMP11212.

First it is necessary to define *arithmetic expressions* for While. This assumes that there is a notion of number predefined, and that we have a notion of variable. To keep this definition short we do not cover those here.[29]

Base case AExp n. For every number n we have that n is a While arithmetic expression.

Base case AExp var. For every variable x we have that x is a While arithmetic

[27] Typically there are more expressions, for example the programmer is allowed to write $a - a'$ as a shortcut for $a + (-a')$.

expression.

Step case AExp $+$. If a_1 and a_2 are While arithmetic expressions then so is $a_1 + a_2$.

Step case AExp $-$. If a_1 and a_2 are While arithmetic expressions then so is $a_1 - a_2$.

Step case AExp $\times$. If a_1 and a_2 are While arithmetic expressions then so is $a_1 \times a_2$.

These arithmetic expressions are used in the following definition of While boolean expression:

Base case BExp `True`. There is a While boolean expression `True`.

Base case BExp `False`. There is a While boolean expression `False`.

Step case BExp $=$. If a_1 and a_2 are While arithmetic expressions then $a_1 = a_2$ is a While boolean expression.

Step case BExp $\leq$. If a_1 and a_2 are While arithmetic expressions then $a_1 \leq a_2$ is a While boolean expression.

Step case BExp $\neg$. If b is While boolean expressions then so is $\neg b$.

Step case BExp $\wedge$. If b_1 and b_2 are While boolean expressions then so is $b_1 \wedge b_2$.

Finally we can define While statements.

Base case Stm $:=$. If x is a variable and a is a While arithmetic expression then $x := a$ is a statement.

Base case Stm `skip`. There is a statement `skip`.

Step case Stm ;. If S_1 and S_2 are statements then so is $S_1; S_2$.

Step case Stm `if`. If b is a While boolean expression and S_1 and S_2 are statements then `if` b `then` S_1 `else` S_2 is a statement.

Step case Stm `while`. If b is a While boolean expression and S is a statement then `while` b `do` S is a statement.

This is not how a computer scientist would write down the definition. You can see that it is very verbose, and that many of the words are not really required to understand what is being defined. For this reason computer scientists have created their own notations for making such definitions more compact, and you will see some of those in COMP11212.

But it is only the notation that differs—mathematically speaking, the corresponding definitions are still examples of recursive definitions.

When we have a formal definition of a programming language we can then reason about all programs for that language. One area of computer science where this happens is the *semantics of programming languages*.

Also note that the definition from Example 6.30 only tells us what the valid While programs are—it does not tell us anything about how those should be executed, and what the result of any computation should be. You will see how one can do that, again making use of a recursive definition, in the second part of COMP11212.

6.4 The natural numbers

The classic example of a recursively defined set is that of the natural numbers, and it is the only way of formally defining what this set should be. Very little has to be assumed for this definition.

6.4.1 A formal definition

Definition 54: natural numbers

The elements of the **set of natural numbers** $\mathbb{N}$ are given by the following recursive definition.

Base case $\mathbb{N}$. There is a natural number 0.

Step case $\mathbb{N}$. For each natural number n there is a natural number Sn, the *successor of n*.

It is more customary to write $n+1$ for Sn, but strictly speaking this has to be justified (see Exercise 159 (b)), and we use Sn for the time being. We introduce

$$1 \text{ as a shortcut for } S0.$$

Code Example 6.11. We give a Java class that corresponds to this definition.

```
public class Nat {

    public Nat next;

    public Nat (Nat n);
    {next=n;}
}
```

This is a very odd class that one wouldn't want to use in practice, but the example here is to show you that this can be done. Note that this class does not have a built-in upper bound for the numbers so defined (unlike the Java class List). In practice the size of the numbers you can use in this way is limited by the machine's memory. Note also the way the class is constructed the number you get from using the argument encoding n is the number encoding $n+1$. You can picture an object of this class as shown below.

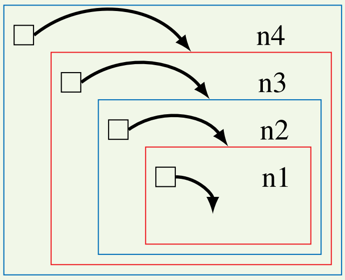
[28] Note that the definition given does not involve brackets but these are introduced by the backdoor in the paragraph following the definition, and you might argue that it would be cleaner to make them explicit from the start.

[29] Also, symbols used in such definitions are usually restricted to those that are readily available from a keyboard. In order to make our definition human readable we stick to symbols used in this course unit.

Assume that we have

- a Nat object n4 with n.next=n3 and
- a Nat object n3 with n3.next=n2 and
- a Nat object n2 with n2.next=n1 and
- a Nat object n1 with n1.next=**null**.

You may picture these objects as follows:



Just as we need to count the number of symbols S to know which number is given by

$$SSSSSSSSSSSSSSSSSSSSSSSSSSS0,$$

so we have to check how many references we have to follow until we reach the **null** reference to know which number is given by some object of class Nat.

Previously most of you will have assumed that the natural numbers exist, and that you know what they are, and how their operations work, without having seen a formal definition. For now you are expected to work with this formal definition.

6.4.2 Formal definitions for basic operations on the natural numbers

In what follows we illustrate how the familiar operations may be formally defined. You could also think of the following as something that could be implemented on a fairly primitive computer.

Example 6.31. An example of a recursively defined operation[30] is addition. The **sum** $m + n$ of two natural numbers n and m is defined as follows. For all m in $\mathbb{N}$:

Base case +. $\quad m + 0 = m$ and

Step case +. $\quad m + Sn = S(m + n)$.

We have defined $m + n$ *by recursion over* the argument n, covering the base case and the step case for natural numbers. Note that in the step case we are using the fact that $m + n$ may be assumed to be already defined when giving a definition for $m + Sn$. Further note that nothing is assumed about the nature of $+$, but, of course, our definition here gives the common notion of addition.

[30] Compare this with the definition of concatenating two lists, Example 6.8.

Example 6.32. We could just as well have used [31]

Base case +. $0 + n = n$ and

Step case +. $Sm + n = S(m + n)$

as our definition.[32] Since we are aiming to define the usual addition operator, we expect the result to be commutative, and so it should not matter which argument we recurse over. We show below that this definition does indeed give a commutative operator.
Note that the official definition of addition is the first one given—this is the one you must use in the exercises.

Code Example 6.12. We give the code for this operation for the class defined above.[33]

```
public static Nat plus (Nat n, Nat n2)
{
    if (n2 == null)
        return n;
    else
        return new Nat (plus(n,n2.next));
}
```

Now assume that we had a class nat in Java like **int**, but restricted to the natural numbers. Then you could write the following code to implement addition.

```
public static nat plus (nat n, nat n2)
{
    if (n == 0)
        return n2;
    else
        return 1 + (plus(n−1,n2));
}
```

Obviously one wouldn't write this in practice, but it does work and implements addition.

Exercise 158. Assume you would prefer your natural numbers to start with 1 rather than 0.

(a) Give a definition for the natural numbers starting with 1.

(b) Give code for a corresponding class.

[31] This is where the situation here differs from the $+\!\!+$ operation for lists.

[32] You may want to compare this definition with that of concatenating two lists, Example 6.8.

[33] Compare this with the code for concatenating two lists, Code Example 6.5.

(c) Give a definition for addition.

(d) Give the code for an addition operation for your class from part (b).

Example 6.33. Once again we may use *induction* to prove properties of the operation just defined. For a very simple example let us prove that for all natural numbers n we have

$$0 + n = n.$$

Here the *base case* is for $n = 0$, and the *step case* is for Sn, and we may use the *induction hypothesis* $0 + n = n$ for the latter. This is what the formal proof looks like.

Base case $\mathbb{N}$. We note $0 + 0 = 0$ by base case +.

Ind hyp. We assume that we have $0 + n = n$.

Step case $\mathbb{N}$. We calculate

$$\begin{aligned} 0 + Sn &= S(0 + n) && \text{step case } + \\ &= Sn && \text{induction hypothesis.} \end{aligned}$$

Tip

The standard structure for an inductive proof over the natural numbers is the following. Assume we have a statement given in terms of the variable n denoting an element of $\mathbb{N}$.

Base case. We show the statement for some number b. The base case is the case of the smallest number b for which the given statement holds, and it is obtained by replacing every occurrence of n by that smallest number.

Ind hyp We assume that the statement holds for n (in which case it coincides with the given statement), or possibly all numbers that are less than or equal to n, and greater than or equal to the number b from the base case.

Step case. We show the statement for Sn by proving the statement where every occurrence of n has been replaced by Sn.

Below we do not give the induction hypothesis explicitly if it coincides with the statement we are proving.

Example 6.34. As another example we show that

$$m + Sn = Sm + n. \qquad (*)$$

Often in a statement with two variables where induction is required it is sufficient to carry out the induction over *one* of the variables. Usually the better choice is to *follow the pattern given by the recursive definition*, here the definition of $+$, and so we carry out an induction over the variable n.
Note that for more complicated statements it may be necessary to carry out a

double induction proof, that is, a proof where induction has to be carried out over both arguments.
We begin the proof of statement $(*)$ with the base case, $n = 0$.

Base case $\mathbb{N}$. We have

$$\begin{aligned} m + S0 &= S(m+0) && \text{step case +} \\ &= Sm && \text{base case +} \\ &= Sm + 0 && \text{base case } + . \end{aligned}$$

Next one has to establish the step case, where Sn appears in the place of n, and where one may use the *induction hypothesis*, that is the claim $Sm+n = m+Sn$.

Step case $\mathbb{N}$. We have

$$\begin{aligned} m + SSn &= S(m+Sn) && \text{step case } + \\ &= S(Sm+n) && \text{induction hypothesis} \\ &= Sm + Sn && \text{step case } + . \end{aligned}$$

What is the point of property $(*)$? It turns out to be very useful, see below, and it will be helpful for you in Exercise 159. It takes a statement where we cannot apply the step case of the definition of $+$ and turns it into a statement where we can. This is how it is used below.

Example 6.35. As a more complicated example we show that addition is commutative, that is

$$\text{for all } m,\ n \in \mathbb{N} \qquad \text{we have} \qquad m + n = n + m.$$

Base case $\mathbb{N}$. This is established as part of Exercise 159 (a).

Step case $\mathbb{N}$. We have

$$\begin{aligned} m + Sn &= S(m+n) && \text{step case +} \\ &= S(n+m) && \text{induction hypothesis} \\ &= n + Sm && \text{step case +} \\ &= Sn + m && \text{property } (*). \end{aligned}$$

CExercise 159. This exercise is concerned with the properties of addition. For this exercise use the definition from Example 6.31—do not switch to the alternative mentioned in Example 6.32. Justify each step.

(a) Show that 0 is a unit for addition on the natural numbers.

(b) Show that $m + 1 = Sm = 1 + m$ for all $m \in \mathbb{N}$, where 1 is a shortcut for $S0$.

(c) Show that addition on the natural numbers is associative.

Defining further operations for the natural numbers, for example multiplication, is now possible. The more complicated the operation, the more complicated is the formal definition likely to be, and the same goes for proving properties of such operations. However, it is possible to use operations previously defined, and so one can build up complicated operations step by step.

Example 6.36. The **product** $m \cdot n$ of two natural numbers m and n is defined as follows. For all $m \in \mathbb{N}$ we have

Base case $\cdot$. $\quad m \cdot 0 = 0$

Step case $\cdot$. $\quad m \cdot Sn = m \cdot n + m.$

Again we could have instead recursed over the first variable instead of doing so for the second.

Exercise 160. This exercise is concerned with properties of multiplication. Justify each step.

(a) Show that 1 is a unit for multiplication on the natural numbers.

(b) Show that for all $m \in \mathbb{N}$ we have

$$m \cdot 0 = 0 = 0 \cdot m.$$

(c) Show that for natural numbers m and n we have $Sm \cdot n = m \cdot n + n$.

(d) Show that multiplication for natural numbers is a commutative operation.

(e) Show that we have the following distributivity law for natural numbers:

$$k \cdot (m + n) = k \cdot m + k \cdot n.$$

(f) Show that multiplication for natural numbers is associative.

CExercise 161. Carry out the following tasks, justifying each step in your proofs.

(a) Define a function d from $\mathbb{N}$ to $\mathbb{N}$ that doubles its argument, without referring to the operations $+$ or $\cdot$.

(b) Give code that implements this operation for the class nat from Code Example 6.11.

(c) Show that for all $n \in N$ we have $dn = SS0 \cdot n$. *Hint: You may find property* $(*)$ *useful, and you will definitely require addition in this proof.*

(d) Consider the function mod_2 defined as follows:

Base case $\mathrm{mod}_2 : 0$. $\quad \mathrm{mod}_2 0 = 0.$

Base case $\mathrm{mod}_2 : S0$. $\quad \mathrm{mod}_2 S0 = S0.$

Step case mod_2. $\quad \mathrm{mod}_2(SSn) = \mathrm{mod}_2 n.$

How would you describe the action of this function? Why do we need two base cases to define this function? Show that

$$\text{for all } n \in \mathbb{N} \text{ we have} \qquad \text{mod }_2 dn = 0.$$

Exercise 162. For natural numbers m and n define m^n. You are allowed to use operations defined above in this chapter. Then prove that for all n, k and l in $\mathbb{N}$ we have $n^{k+l} = n^k \cdot n^l$, justifying each step. You may use results from previous exercises as needed.

Example 6.37. We can define the *predecessor* Pn of a natural number n as follows.

Base case P. $\qquad P0 = 0$

Step case P. $\qquad P(Sn) = n.$

Code Example 6.13. Again we give code that implements this operation for the class Nat.

```
public static Nat pred (Nat n)
{
if (n == null)
    return n;
else
    return n.next;
}
```

Example 6.38. The notion of a predecessor allows us to define an operation related to subtraction for the natural numbers, which satisfies

$$m \dot{-} n = 0 \qquad \text{if} \qquad m < n,$$

that is we use the value 0 wherever subtraction in the integers would lead to a negative number. In other words we give a recursive definition of the function

$$m \dot{-} n = \begin{cases} m - n & m \geq n \\ 0 & \text{else} \end{cases}$$

Base case $\dot{-}$. $\qquad m \dot{-} 0 = m$

Step case $\dot{-}$. $\qquad m \dot{-} Sn = P(m \dot{-} n).$

Note that there is no sensible way of defining this operation by recursing over the first argument: The reason our definition works is that the result of $m \dot{-} Sn$ is related to that of $m \dot{-} n$: The result of $m \dot{-} Sn$ should be one below that of $m \dot{-} n$, unless the result of $m \dot{-} n$ is 0 already, in which case $m \dot{-} Sn$ must be 0 as well. This is exactly how the predecessor function P works. There is no such simple relation between $Sm \dot{-} n$ and $m \dot{-} n$.

Optional Exercise 26. Think about what would be required to define the $\dot{-}$ operator by recursing over the first argument. You may find that you need to define $\leq$ for natural numbers, see Exercise 163.

Example 6.39. Recall integer division from page 5. This is more complicated to define formally since we require case distinctions. For natural numbers m and n, where $m \neq 0$, we make the following definition.[34]

Base case mod. $0 \bmod m = 0$ and

Step case mod.

$$Sn \bmod m = \begin{cases} 0 & \text{if } S(n \bmod m) = m \\ S(n \bmod m) & \text{else} \end{cases}$$

Base case div. $0 \operatorname{div} m = 0.$

Step case div.

$$Sn \operatorname{div} m = \begin{cases} S(n \operatorname{div} m) & \text{if } (Sn \bmod m) = 0 \\ n \operatorname{div} m & \text{else} \end{cases}$$

Again there is a reason we recurse over the first argument: The result of dividing Sn by m is closely related to that of dividing n by m, but the result of dividing n by Sm is not close to that of dividing n by m. Working with these definitions is quite delicate and proofs of simple statements are quite involved. Note that giving a definition by cases may not be what one would like, and you are asked to find an alternative in Exercise 164.

Example 6.40. Note that in the preceding example we are assuming that we know when two elements of $\mathbb{N}$ are equal. Strictly speaking that is something we have not yet defined. We do this by thinking of it as function which takes two arguments from $\mathbb{N}$ and returns 0 or $S0 = 1$. We want to give a recursive definition of a function which for input m and n gives 1 if and only if $m = n$, and 0 else. This can be done as follows (compare Example 6.13).

Base case $f_=$. $f_=(0, 0) = S0.$

Step case $f_=$:$m, 0$. $f_=(Sm, 0) = 0.$

Step case $f_=$:$0, n$. $f_=(0, Sn) = 0.$

Step case $f_=$:m, n. $f_=(Sm, Sn) = f_=(m, n).$

[34] Note that a special case of this operation appears in Exercise 161.

Here you can see how recursion over two arguments works, but note that this is a particularly simple case. The code from Code Example 6.13 can be adjusted to implement this function for the class Nat.
Note that we have not said which source and target is intended for this function. Clearly the source is meant to be $\mathbb{N} \times \mathbb{N}$, but there are at least two sensible target sets: One might pick $\{0, 1\}$, or one might pick $\mathbb{N}$ again. It is the latter option that you are expected to use in Exercise 164.

Exercise 163. Similar to the definition of $f_=$ above give a definition of $f_\leq$, with $f_\leq(m, n) = 1$ if and only if $m \leq n$.

It is possible to combine $f_=$ with the previously defined operations $\dot{-}$ and $\cdot$ to give a definition of mod which does not require the use of a definition by cases, see the following exercise.

EExercise 164. Justifying each step show the following statements by induction.

(a) $0 \dot{-} n = 0$ for all $n \in \mathbb{N}$.

(b) $P(Sn) = n$ for all $n \in \mathbb{N}$.

(c) $P(Sn \dot{-} m) = n \dot{-} m$ for all $m, n \in \mathbb{N}$.

(d) $n \dot{-} n = 0$ for all $n \in \mathbb{N}$.

(e) Give a definition of mod which does not use a case distinction, but which instead uses the function $f_=$ from Example 6.40, viewed as a function from $\mathbb{N}$ to $\mathbb{N}$. Argue that your definition agrees with the one from the notes.

(f) Show that $n \bmod S0 = 0$ for all $n \in \mathbb{N}$. *Hint: Use the definition of* mod *from the previous part.*

Hint: You may use statements from previous parts even if you have not proved them.

Optional Exercise 27. If you are looking for a more challenging proof, try to show that $n \bmod n = 0$ for all $n \in \mathbb{N}$, or that $n \operatorname{div} 1 = n$ for all $n \in \mathbb{N}$.

The material in this section to this point gives us a rigorous definition of the natural numbers, as well as definitions of commonly used operations on them. This may appear somewhat tedious, covering only things you already knew. But mathematics is all about building everything from first principles, and doing so in a rigorous way. One might continue the development sketched above until one has assembled all the information about the operations on the natural numbers that is summarized in Chapter 0, but we don't have the time for this. Hopefully the above gives you a flavour of how this might work.

6.4.3 Advanced operations for natural numbers

From here on we go back to using the usual names for natural numbers, so we write 1 instead of $S0$, $n + 1$ instead of Sn, and so on. The step case in an induction proof is now the more familiar step from n to $n + 1$. Note that in some of the examples below we use induction hypotheses which are more complicated than assuming that the given statement holds for some number n and then proving that this implies it holds for $n + 1$. In these cases we make the induction hypothesis explicit.

From here on we also use the usual operations for natural numbers without referring to their formal recursive definitions. But you should still think about the properties you use, and make them explicit in your proofs.

Example 6.41. A standard example given to introduce the idea of a recursively defined function is that of the *factorial function*. We define this operation as follows.

Base case !. $\quad 0! = 1.$

Step case !. $\quad (n + 1)! = (n + 1) \cdot n!.$

Turned into Java code (for integers) this looks something like this:

```
public static int factorial (int n);
{
    if (n==0)
        return 1;
    else
        return n * factorial(n-1);
}
```

Note that we are explicitly carrying out a test:

- If we are in the base case then we return the appropriate number,
- else we know that we are in a case were n is of the form $m + 1$, and so it is safe to subtract one from n, and stay in $\mathbb{N}$.

Also note that this code makes no attempt to determine whether the argument is a negative number. You may want to think about what happens if it is called for a negative number n.

But apart from these minor differences the code is a very straightforward translation of the mathematical definition. Understanding how the mathematics works therefore can help you write recursive programs.

You might note that $1! = 1$, and so if $n = 1$ there's really no need to do a recursion step. You can therefore make the code more efficient (but maybe less clear) by writing instead:

```
public static int factorial (int n);
{
    if (n <= 1)
```

```
        return 1;
    else
        return n * factorial(n−1);
}
```

When it comes to calculating the result of reasonably complex operations on natural numbers recursion can be a useful tool.

Euclid's algorithm

Given two natural numbers m and n we can find the largest number dividing both of them, their *greatest common divisor*.

Example 6.42. The following is known as *Euclid's algorithm*. It appears in COMP26120 as an example algorithm.
Assume that $a \leq b$ are natural numbers. We set

$$r_0 = b \qquad\qquad \text{and} \qquad\qquad r_1 = a.$$

By Fact 2 from Chapter 0 we can find natural numbers k_1 and $r_2 < r_1$ with the property that

$$b = r_0 = k_1 r_1 + r_2 = k_1 a + r_2.$$

We keep applying this idea and, invoking Fact 2, we define

$$r_{i+2} \text{ with } 0 \leq r_{i+2} < r_{i+1} \qquad\qquad \text{and} \qquad\qquad k_{i+1}$$

to be the unique numbers with the property that

$$r_i = k_{i+1} r_{i+1} + r_{i+2}.$$

In other words, we have

$$k_{i+1} = r_i \operatorname{div} r_{i+1} \qquad \text{and} \qquad r_{i+2} = r_i \bmod r_{i+1}.$$

Note that we have that

$$r_0 > r_1 > \cdots$$

Any strictly descending sequence of natural numbers must be finite. We may apply Fact 2 and construct new elements for the sequence until we get the number 0, let's say when we reach r_{n+1}, so that

$$r_{n-1} = k_n r_n + 0.$$

The number r_n plays a particular role for a and b.
We claim that r_n is the *greatest common divisor* of a and b. More specifically we show that

- the number r_n divides both, a and b and
- if c divides both, a and b then c divides r_n.

These proofs are carried out by induction. To prove the first claim we show that

$$\text{for all } 0 \leq i \leq n \qquad r_n \qquad \text{divides} \qquad r_i.$$

This means that in particular, r_n divides both, $r_1 = a$ and $r_0 = b$.
Note that this proof is a bit different from other induction proofs we have seen in that

- It proceeds backwards from n down to 0.
- It requires two base cases since the step case requires that the claim holds for the two previously considered numbers.

For this reason we state the induction hypothesis explicitly.

Base case n. Obviously r_n divides r_n.

Base case $n-1$. Since $r_{n-1} = k_n r_n$ we have that r_n divides r_{n-1}.

Ind hyp. We know that r_n divides r_{i+1} and r_{i+2}.

Step case. We show that r_n divides r_i. By construction we have

$$r_i = k_{i+1} r_{i+1} + r_{i+2}.$$

Since by induction hypothesis we know that

$$r_n \text{ divides } r_{i+1} \qquad \text{and} \qquad r_n \text{ divides } r_{i+2}$$

we can find natural numbers l_{i+1} and l_{i+2} with the property that

$$r_{i+1} = l_{i+1} r_n \qquad \text{and} \qquad r_{i+2} = l_{i+2} r_n,$$

and so

$$\begin{aligned} r_i &= k_{i+1} r_{i+1} + r_{i+2} \\ &= k_{i+1} l_{i+1} r_n + l_{i+2} r_n \\ &= (k_{i+1} l_{i+1} + l_{i+2}) r_n, \end{aligned}$$

which means that

$$r_n \text{ divides } r_i.$$

We leave the proof of the second claim as an exercise.
In COMP26120 you will think about the complexity of such algorithms, that is, you will worry about how many r_i have to be calculated until the greatest common divisor has been reached.

Exercise 165. Show the second claim from the preceding example.

Recursion can be used beyond defining operations on natural numbers.

Example 6.43. We can also use recursion to define subsets of sets already defined. For example, the set of even numbers (compare Examples 0.9 and 0.17) M_2 can be defined to be the smallest set satisfying the following conditions.

Base case M_2. $0 \in M_2$.

Step case M_2. $n \in M_2$ implies $2 + n \in M_2$.

The base case tells us that $2 \cdot 0 = 0$ is in S, and using the step case we get

$$0 + 2 = 2 = 2 \cdot 1 \in S,$$

and applying the step case again we get

$$2 + 2 = 4 = 2 \cdot 2 \in S,$$

and so on. Note that we do not have to refer to multiplication in this definition—all we have to do is keep adding 2.

Example 6.44. For another example, to define a set $P_2 \subseteq \mathbb{N}$ containing exactly the powers of 2 we can use the following.

Base case P_2. $1 \in P_2$.

Step case P_2. $n \in P_2$ implies $2 \cdot n \in P_2$.

The base case tells us that $2^0 = 1$ is in S, and using the step case we get

$$1 \cdot 2 = 2 = 2^1 \in S,$$

and applying the step case again we get

$$2 \cdot 2 = 4 = 2^2 \in S,$$

and so on. Note that we do not have to refer to exponentiation in this definition—all we have to do is keep multiplying with 2.

Exercise 166. Give a recursive definition for each of the following sets.

(a) The set of odd natural numbers as a subset of the natural numbers.

(b) The set of all non-empty lists over $\mathbb{N}$ for which the nth element (from the right), for $n \geq 2$, is equal to twice the previous element.

(c) The set of all non-empty lists over $\mathbb{N}$ for which the nth symbol (from the right), for $n \geq 3$, is equal to the sum of the previous symbols.

(d) The set of full binary trees such that each label of a node that is not a leaf is the sum of the labels of its children.

(e) The set of full binary trees such that each label of a node that is not a leaf is the sum of the leaves below it.

Hint: You may use previously defined operations. If you aren't sure what the step case might look like, construct some lists or trees that satisfy the given criteria.

Example 6.45. We give a definition of the the $\sum$ operator which makes an appearance in Chapter 4. Assume that N is a set of numbers. Assume that n is a natural number and that $a_i \in N$ for $1 \leq i \leq n$, The operation

$$\sum_{i=1}^{n} a_i$$

is recursively defined as follows.

Base case $\sum$. $\quad \sum_{i=1}^{0} a_i = 0$

Step case $\sum$. $\quad \sum_{i=1}^{n+1} a_i = (\sum_{i=1}^{n} a_i) + a_{n+1}$.

For some sums it is possible to find a simplification, and in those cases a proof that this works is usually by induction.

Example 6.46. We show that for all $n \in \mathbb{N}$,

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

Base case $\mathbb{N}$. We have

$$\begin{aligned} \sum_{i=1}^{0} i &= 0 && \text{base case } \textstyle\sum \\ &= \frac{0 \cdot 1}{2} && 0 \cdot n = 0 \text{ in } \mathbb{N}. \end{aligned}$$

Step case $\mathbb{N}$. For this case we have

$$\begin{aligned} \sum_{i=1}^{n+1} i &= \sum_{i=1}^{n} i + (n+1) && \text{step case } \textstyle\sum \\ &= \frac{n(n+1)}{2} + (n+1) && \text{ind hyp} \\ &= \frac{n(n+1)}{2} + \frac{(n+1)2}{2} && 2/2 = 1, 1 \text{ unit for mult} \\ &= \frac{(n+1)n + (n+1)2}{2} && \text{distr and comm of mult} \\ &= \frac{(n+1)(n+2)}{2} && \text{distr for mult.} \end{aligned}$$

Note that sometimes sums start at 0 instead of at 1, such as in the exercise below. In that case you have

$$\sum_{i=0}^{n} a_i = \sum_{i=1}^{n+1} a_{i-1},$$

which is equivalent to the definition below:

Base case $\sum$. $\sum_{i=0}^{0} a_i = a_0$

Step case $\sum$. $\sum_{i=0}^{n+1} a_i = (\sum_{i=0}^{n} a_i) + a_{n+1}$.

CExercise 167. Show the following statements by induction.

(a) For $r \in \mathbb{R} \setminus \{1\}$ and $n \in \mathbb{N}$ we have $\sum_{i=0}^{n} r^i = \frac{r^{n+1} - 1}{r - 1}$.

(b) For $n \in \mathbb{N}$ we have $\sum_{i=0}^{n} i(i+1) = \frac{n(n+1)(n+2)}{3}$.

(c) For $n \in \mathbb{N}$ we have $\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$.

(d) For $n \in \mathbb{N}$ we have $\sum_{i=0}^{n} \frac{1}{2^i} = \frac{2^{n+1} - 1}{2^n}$.

(e) For $n \in \mathbb{N}$ we have $\sum_{i=0}^{n} i!i = (n+1)! - 1$.

(f) For $n \in \mathbb{N}$ we have $1 + \sum_{i=0}^{n} 2^i = 2^{n+1}$.

6.4.4 Combinatorial rules

Looking back to the formulae for counting combinations that appear in Section 4.1.2 we can see that these can be shown to do what they are supposed to do by using inductive arguments.

Selection with return.

Consider a situation where we have n items to randomly pick from, returning them each time. The claim we have from Section 4.1.2 is that there are

$$n^i$$

combinations when drawing i times. Can we use induction to create a formal argument that this is indeed so?

Base case N. If there is no draw there is one possible outcome.[35]

Step case N. If there are $i + 1$ draws then by the induction hypothesis there are n^i many outcomes from the first i draws. Each of those has to be combined with the n possible outcomes of the $(i+1)$th draw, which means there are

$$n^i \cdot n = n^{i+1}$$

many outcomes for $i + 1$ draws.

Selection without return

If we have n items to pick from then according to the formula from Section 4.1.2 for i draws without return there are

$$\frac{n!}{(n-i)!}$$

many combinations. Again we want to give a proof.

[35] If you prefer you can start with ‘If there is one draw there are n possible outcomes by assumption’.

Base case N. If there is no draw there is one possible outcome.[34]

Step case N. If there are $i+1$ draws then by the induction hypothesis there are

$$\frac{n!}{(n-i)!}$$

many outcomes from the first i draws. Since i items have been removed there are $n-i$ possibilities for the $(i+1)$th draw, each of which has to be combined with every outcome from the previous draws, giving

$$\frac{n!}{(n-i)!} \cdot (n-i) = \frac{n!}{(n-i-1)!} = \frac{n!}{(n-(i+1))!}$$

many outcomes

Unordered selection without return

We leave the remaining case as an exercise.

Exercise 168. For unordered selection without return prove by induction that the number of possible outcomes is

$$\frac{n!}{(n-i)!i!}.$$

6.4.5 Functions given via recursive specifications

When trying to compute the complexity of a recursive algorithm, that is trying to see how the number of steps changes as the problem size grows, one often ends up having to solve a *recurrence relation*. This means that one has a recursive specification of a function, and one would like to find a simpler description of that function (using known functions).

Example 6.47. Assume we have a program whose complexity we are trying to work out as a function which takes as its input the problem size n, and gives as its output the number of steps (or an approximation thereof) the program will take when given an input of size n. Further assume that by studying the program we have worked out that

Base case f. $f0 = 1$, that is, if the problem size is 0 the program takes one step and

Step case f. $f(n+1) = kfn$, for some $k \in \mathbb{N}$, that is, if the problem size grows by one the program needs k times the number of steps from before.

Working with this specification is unwieldy. In this case it's quite easy to give an alternative description of f, but let's see how one might arrive there. It's probably easiest to first of all compute the first few values.

n	0	1	2	3	4
fn	1	k	k^2	k^3	k^4

We might now guess that $fn = k^n$, but we really should verify that this fits with the original description. As with most recursive statements the obvious way of proving this is by induction.

Base case f. $f0 = 1 = k^0$, so we have a match.

Step case f.

$$\begin{aligned} f(n+1) &= kfn && \text{definition } f \\ &= k \cdot k^n && \text{induction hypothesis} \\ &= k^{n+1} && \text{calcs in } \mathbb{N}, \end{aligned}$$

and this gives another match.

Example 6.48. Let's do a more complicated example.

Base case g:0. $g0 = 0$.

Base case g:1. $g1 = 1$.

Step case g. $g(n+2) = 2g(n+1) - gn + 2$.

Note that we have two base cases here.They are required because the step case relies on *two* previously calculated values. Again we work out the first few values.

n	0	1	2	3	4
gn	0	1	4	9	16

Once more there's an obvious guess: We guess that $gn = n^2$. Again we have to provide a proof. Note that we have to cover two base cases now, and that the induction hypothesis is now: $gk = k^2$ for all $k < n+2$. In particular we use below the induction hypothesis that

$$g(n+1) = (n+1)^2 \qquad \text{and} \qquad gn = n^2.$$

Base case g:0. $g0 = 0 = 0^2$.

Base case g:1. $g1 = 1 = 1^2$.

Ind hyp g. We have

$$gn = n^2 \qquad \text{and} \qquad g(n+1) = (n+1)^2.$$

Step case g.

$$\begin{aligned} g(n+2) &= 2g(n+1) - gn + 2 && \text{def } g \\ &= 2(n+1)^2 - n^2 + 2 && \text{ind hyp} \\ &= 2n^2 + 4n + 2 - n^2 + 2 && \text{calcs in } \mathbb{N} \\ &= n^2 + 4n + 4 && \text{calcs in } \mathbb{N} \\ &= (n+2)^2 && \text{calcs in } \mathbb{N}. \end{aligned}$$

CExercise 169. For the following recursive specifications, work out an alternative *non-recursive* representation of the given function and prove by induction that it satisfies the given specification.

(a) **Base case** f. $\quad f0 = 0.$

Step case f. $\quad f(n+1) = fn + 2.$

(b) **Base case** f. $\quad f0 = 1.$

Step case f. $\quad f(n+1) = (n+1)fn.$

(c) **Base case** f. $\quad f0 = 2.$

Step case f. $\quad f(n+1) = (fn)^2.$

(d) **Base case** f:0. $\quad f0 = 0.$

Base case f:1. $\quad f1 = 3.$

Step case f. $\quad f(n+2) = 2f(n+1) - fn.$

(e) **Base case** f:0. $\quad f0 = 1.$

Base case f:1. $\quad f1 = 1.$

Step case f. $\quad f(n+2) = 2f(n+1) - fn.$

(f) **Base case** f:0. $\quad f0 = 1.$

Base case f:1. $\quad f1 = 2.$

Step case f. $\quad f(n+2) = f(n+1) + 2fn.$

(g) **Base case** f:0. $\quad f0 = 1.$

Base case f:1. $\quad f1 = 3.$

Step case f. $\quad f(n+2) = 2f(n+1) + 3fn.$

Note that the examples given in these notes have been carefully chosen so that it is possible to guess a *closed form* for the underlying function. Not all recurrence equations that appear in real-world programs are as easy as this. An example is provided by the following optional exercise. Typically recursive specifications arise when one wants to compute the complexity of a recursive program, and often those occurring in practice are much harder to solve than our examples.

Optional Exercise 28. A popular exercise in writing recursive programs is to compute the Fibonacci sequence.[36] Consider the following recursive specification for a function.

Base case f:0. $\quad f0 = 0.$

Base case f:1. $\quad f1 = 1.$

Step case f. $\quad f(n+2) = f(n+1) + fn.$

(a) Calculate the first few values of f.

(b) Show that for

$$\alpha = \frac{1+\sqrt{5}}{2} \qquad \text{and} \qquad \beta = \frac{1-\sqrt{5}}{2}$$

we have

$$\alpha^{n+2} = \alpha^{n+1} + \alpha^n \qquad \text{and} \qquad \beta^{n+2} = \beta^{n+1} + \beta^n,$$

so these numbers satisfy the equations defining f.

(c) Show that the function

$$\begin{aligned} fn &= \frac{\alpha^n - \beta^n}{\alpha - \beta} \\ &= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}} \\ &= \frac{(1+\sqrt{5})^n - (1-\sqrt{5})^n}{2^n\sqrt{5}} \end{aligned}$$

satisfies the original condition for f.

Would you have guessed this definition for f?

Optional Exercise 29. Note that it is possible to have recurrence relations with more than one variable. A typical example are the *binomial coefficient*, which can be thought of as given by the recurrence relation

Base case binom. $\binom{n}{0} = 1$.

Step case binom. $\binom{n+1}{i+1} = \binom{n}{i} + \binom{n}{i+1}$.

Show that this gives the same definition as the more common

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

Note that the recursive definition does not require multiplication, just addition. You may have seen this definition at work in *Pascal's triangle*.

6.5 Further properties with inductive proofs

Many properties involving natural numbers can be shown by induction. The following example and exercises gives a taste of those.

Example 6.49. Here is an inductive proof of a property of complex numbers,

[36] Check out `https://en.m.wikipedia.org/wiki/Fibonacci_number` to read up on where this sequence originates and some of its interesting properties.

but invoking natural numbers where it counts. We show that

$$|z| = 1 \qquad \text{implies} \qquad \text{for all } n \in \mathbb{N},\ |z^n| = 1.$$

Base case $\mathbb{N}$. We have $|z^0| = |1| = 1$.

Step case $\mathbb{N}$. For this case we have

$$\begin{aligned} |z^{n+1}| &= |z^n z| && \text{def } (-)^{n+1} \\ &= |z^n||z| && |zz'| = |z||z'|,\ \text{Ex } 16 \\ &= 1 \cdot 1 && \text{ind hyp, assumptn} \\ &= 1 && \text{1 unit for mult} \end{aligned}$$

Exercise 170. Show the statements from Fact 8 on page 42.

EExercise 171. The following statements can be shown by induction but some creativity is required to see how the proof might work.

(a) Show that every natural number greater than or equal to 12 is the sum of multiples of the numbers 4 and 5. This means that if we have 4p and 5p stamps then they can be combined to get any amount from 12p up. *Hint: Work out solutions for smaller numbers, up to 24 or so. A pattern should emerge. You will require a number of case distinctions.*

(b) Show that for all natural numbers $n \neq 0$ we have that $7^n - 1$ is divisible by 6, as is $n^3 - n$.

(c) Assume that $n \in \mathbb{N}$ and $n \geq 2$. Show that a set of size n has $n(n-1)/2$ two-element subsets. *Hint: This is an exercise that wants to be solved by reasoning in English, rather than manipulating some formula.*

After all this you may wonder what '*recursion theory*' is all about. It is the study of those functions $\mathbb{N}^n \longrightarrow \mathbb{N}$ which can be defined using recursion. This is more involved than you may think—because one may use previously defined functions to define more sophisticated ones the functions that can arise in this way are considerably more complicated than those that appear in this section.

6.6 More on induction

The examples given above are comparatively simple cases of proofs by induction. We briefly discuss situations where a proof becomes more complicated.

- The induction hypothesis in most of our examples is very straightforward in the sense that for each step case, we merely require the statement for each 'ingredient'. For example, in the case of a propositional formula, assuming the statement holds for the formula A and the formula B we show that it holds for $A \wedge B$. Sometimes one has to require the statement for 'all ingredients built so far'. In the above example this would mean assuming the statement for A, B, and all their subformulae.

- In some cases it is not sufficient to assume the given claim for all entities built so far'. Instead one finds that when trying to prove the step case that one needs a *stronger* statement than the one that is at issue. This often gives some idea of an alternative, stronger statement that can be shown by induction. This is sometimes called *strengthening the induction hypothesis*. We have not seen any examples of this.

- It is perfectly possible to *nest* induction proofs, that is, inside a proof by induction one requires another proof, also by induction, of an unrelated statement. We have not seen examples of this.

- If one tries to show a statement which contains more than one variable one of the following cases will apply if the statement is provable by induction at all:

  - We may prove the statement by induction over one of the variables, treating the others as parameters. Examples of this are Examples 6.11 and 6.34.
  - We may prove the statement by giving an induction proof for one of the variables, which inside contains induction proofs for the other variables (where one has to be careful to state the various induction hypotheses carefully to ensure that they do not assume anything unwarranted). In the case where there are two variables this is known as *double induction*. We have not seen examples of this.

Chapter 7

Relations

So far when trying to connect two sets we have only looked at functions. This assumes that we have a mechanism for turning elements from the source set into elements of the target set. However, there are other ways of making connections.

Example 7.1. In a database we typically want to think of tables, and indeed people often talk about 'relational databases'. These can be viewed as relations of a general kind. For a simple example, think of a library which has a table of members (uniquely determined by their membership numbers), a table of books (uniquely determined by catalogue numbers), and a table which keeps track of which book is currently on loan to which member. We can think of this as connecting the set of all members M (represented by the set of all valid membership numbers) to the set of all books B (represented by the set of all current catalogue numbers).
This connection cannot be thought of as a function: A given member may have no, or several, books on loan, so we cannot produce a unique output for each input. Instead we can think of this table as a subset L of the product $M \times B$ with the property that

$$(m, b) \in L \qquad \text{if and only if} \qquad \text{member } m \text{ currently has book } b \text{ on loan.}$$

This is an example of a relation from M to B.
In terms of a database one would describe this as a *relation schema*, and would give a type for it along the following lines:

OnLoan(member:**int**,book:**int**),

assuming that membership and catalogue numbers are implemented as integers. The entries in the database are exactly the members of the relation.

In these notes we largely restrict ourselves to relations connecting two sets, which means relation schema with two entries. Databases often have relation schemas with more entries, for example a database that keeps track of members of the university might have a relation schema including

(title,name,building,office number,phone number).

The general ideas regarding relations that are introduced below apply to this kind of situation as well.

7.1 General relations

We use relations all the time, even though you may not have been aware of that. It is often convenient to think of relations as generalizations of functions, and sometimes a similar notation is used.

A relation R from a set S to a set T is given by a subset of $S \times T$. This is sometimes written as

$$R \colon S \longrightarrow\!\!\!\!\!\!\!\!\mapsto T,$$

but this is not universal.

A function is a special kind of relation: Given a function $f \colon S \longrightarrow T$ we have its graph

$$\{(s, fs) \in S \times T \mid s \in S\},$$

which is a subset of $S \times T$. It is standard to identify f with its graph to view it as a relation. Proposition 2.1 tells us which relations are the graphs of functions, namely those relations $R \subseteq S \times T$ where for every $s \in S$ there is a unique $t \in T$ such that $(s, t) \in R$.

Examples of relations are abundant.

Example 7.2. We give a few examples of relations from a number of areas.

(a) The relation from the set of students in the School to the set of academics where a student is related to an academic if during the current academic year the student is enrolled on a course unit on which the academic teaches.

(b) The relation from the set of students in the School to the set of COMP course units offered which relates a student to all the course units he or she is enrolled on.

(c) The relation from the set of real numbers to the set of real numbers where x is related to y if $x = y^2$.

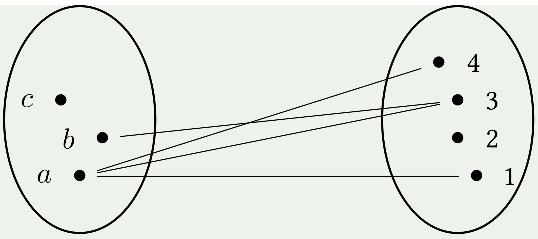
(d) The relation between Java programs where one program is related to another if they can be viewed as computing the same thing.[1]

(e) The relation of equality for a number of entities, for example, equality of numbers, equality of fractions, and more generally the equality of 'arithmetic expressions', that is, expressions written using numbers and operations such as addition, multiplication and inverses with respect to these. We usually consider two such expressions to be equivalent (or equal) if they evaluate to the same number.

We can picture relations between small finite sets in a picture, similar to how we draw functions between such sets.

Example 7.3. We show how to picture a small relation between two different sets. For small relations on the same set see Section 7.2.2.

[1] Defining this in general in a rigorous way is non-trivial, but it is fairly easy if one only looks at Java programs which can be thought of as having a natural number as input and a natural number as output.



This relation goes from the set $\{a, b, c\}$ to the set $\{1, 2, 3, 4\}$. It relates

- a to 1, 3 and 4,
- b to 3, and
- c to no element.

We typically write this relation as the collection of pairs from the set

$$\{a, b, c\} \times \{1, 2, 3, 4\}$$

which it contains, which in the case of the example above is

$$\{(a, 1), (a, 3), (a, 4), (b, 3)\}.$$

7.1.1 Important notions

There are two common notations for relations. One is to rely on the idea that a relation R from S to T is a subset of $S \times T$ and so to denote the fact that R relates an element s of S to an element t of T by

$$(s, t) \in R.$$

Sometimes infix notation is preferred, and instead of $(s, t) \in R$ one might write

$$s \; R \; t.$$

Where infix notation is used it is not unusual to see symbols, rather than letters, to denote a relation; for example you may find a relation $\sim$ from S to T where the same fact is written as

$$s \sim t.$$

Example 7.4. An example of a relation that is typically written in this manner this is *equality of arithmetic expressions.* To denote that the two expressions $2/4$ and $1/2$ denote the same number in $\mathbb{Q}$ or $\mathbb{R}$ we write

$$2/4 = 1/2;$$

the equal symbol $=$ being written in infix notation.

Example 7.5. Another example is the notion of *semantic equivalence* of propositions from the material on logic, where the notation

$$A \equiv B$$

is used to denote the fact that with respect to every valuation A has the same boolean interpretation as B.

Both notations are routinely used and you should become comfortable with both.

Each set has a special relation: Given a set S the **identity relation I_S on S** is given by

$$\{(s,s)\in S\times S \mid s\in S\}.$$

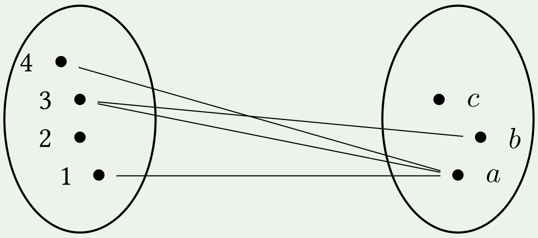
In other words, every element is related to itself, and to nothing else. The identity relation on S is the graph of the identity function id_S on S.

Given a relation R from S to T there is an easy way of turning it into a relation from T to S: The **opposite relation R^{op} of R** is given by

$$R^{\mathsf{op}} = \{(t,s)\in T\times S \mid (s,t)\in R\}.$$

In other words we 'turn the relation around' by changing the order of the pairs, and also switch the 'source' and 'target'.

Example 7.6. For the relation given in Example 7.3 above the opposite relation is given by the following picture.



Example 7.7. Consider the relation 'is a child of' as a relation on the product with itself of the set of all people, where

$$(a,b)$$

being an element of the relation means that a is a child of b. The opposite of the 'is a child of' relation is the 'is a parent of' relation, where

$$(a,b)$$

is in the relation if and only if a is a parent of b.

Note that since relations are sets we can apply set operations to them. In particular, given a relation R from S to T there is its complement,

$$(S\times T)\setminus R,$$

and given two relations R and R' from S to T we may form

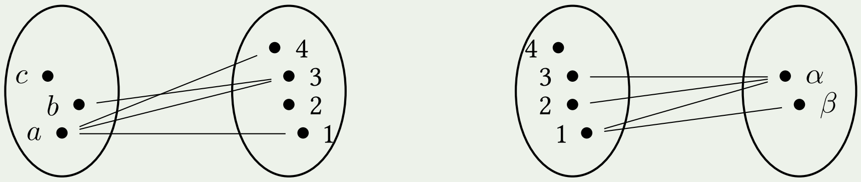
their union $R\cup R'$ and their intersection $R\cap R'$.

Relations are a little like functions in that one can define their composition.

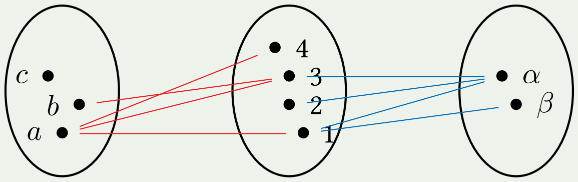
Let R be a relation from S to S', and let R' be a relation from S' to S''. The **relational composite** $R\,;R'$ is given by

$$R\,;R' = \{(s,s'')\in S\times S'' \mid \exists s'\in S'.\,((s,s')\in R \text{ and } (s',s'')\in R')\}.$$

Example 7.8. Consider the two relations given by the following picture.

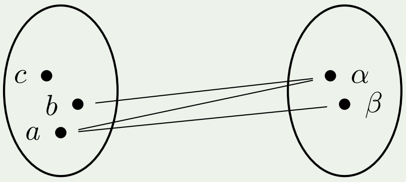


If we 'overlay' the two we can more easily see what the composite of the two relations is.



The composite connects

- a node in the left-most set with
- a node in the right-most set if and only if
- they are connected by a line through the set in the middle (that is a red line followed by a blue one).



Example 7.9. Consider the 'is a child of' relation from Example 7.7. We may form the relational composite of this relation with itself, and the result is the 'is a grandchild of' relation.

Example 7.10. Assume we have two relation schema (compare Example 7.1) in a database,

- one connecting members (given by their membership number) with borrowed books (given by their catalogue number) and
- one connecting books (given by catalogue number) and their titles.

If one wants to create a new relation scheme that connects members (given by their membership number) with the titles of the books they have on loan one has to form the relational composite of the two underlying relations.
On the left we have a table describing the relations between members and the catalogue numbers of the books they have on loan, and on the right a table describing the title of the book corresponding to a catalogue number.

Member 1	00002
Member 1	00003
Member 3	00005
Member 3	00007
Member 3	00011

00002	*The Joys of Java*
00003	*Maths for Dummies*
00005	*Higher Category Theory*
00007	*Topoi and Theories*
00011	*Multiversal Algebra*

The relational composite of the underlying relations, written again as a table, is the following:

Member 1	*The Joys of Java*
Member 1	*Maths for Dummies*
Member 3	*Higher Category Theory*
Member 3	*Topoi and Theories*
Member 3	*Multiversal Algebra*

But note that this is a special example, which makes it look as if every entry in the first table gives rise to an entry in the resulting table. This is the case because our second table defines a very special relation, namely one that is functional: For every catalogue number there exists exactly one book title (namely the title of the corresponding book). Moreover, since every catalogue number can be on loan to at most one person, the first relation is also special.

Example 7.11. Assume we have a different database. We have a relation schema connecting applicants to the school with their computer science interests, and another relation schema which suggests related interests.

Wong	AI
Wong	HCI
Kim	Maths
Anna	AI
Anna	Hardware

AI	Machine Learning
AI	Logic Programming
AI	Knowledge Representation
Security	Encryption
Maths	Logic
Maths	Probabilities
Hardware	Circuits

The relational composite of the underlying relations, written again as a table, is the following:

Wong	Machine Learning
Wong	Logic Programming
Wong	Knowledge Representation
Kim	Logic
Kim	Probability Theory
Anna	Machine Learning
Anna	Logic Programming
Anna	Knowledge Representation
Anna	Circuits

It connects students with potential interests.

Example 7.12. In COMP11212 the notion of a *simulation* between finite state automata is introduced. This is a relation that connects the states of the two automata. Only relations with particular properties are valid simulations. This is an example where relations specifically appear in a computer science context.

Optional Exercise 30. Show that if you have a relation R which is a simulation from A to A', and if R' is a simulation from A to A'' then their composite as relations, $R\,;R'$, is a simulation from A to A''.

Exercise 172. Show the following for relational composition:

(a) Given a relation R from S to T, show that for the identity relation I_S on S we have $I_S\,;R = R$ and similarly that for the identity relation I_T on T we have $R\,;I_T = R$.

(b) Assume that R, R' and R'' are relations that can be composed. Show that $(R\,;R')\,;R'' = R\,;(R'\,;R'')$.

We begin by looking at a generalization of the notion of a function, and then move to other cases of relations with particular properties.

7.2 Partial functions

Sometimes we would like to consider assignments that behave like functions, but which are not defined on the whole source set.

Example 7.13. Division is such a function from

$$\mathbb{Q} \times \mathbb{Q} \longrightarrow \mathbb{Q}.$$

It is defined everywhere with the exception of the subset

$$\mathbb{Q} \times \{0\} \qquad \text{of} \qquad \mathbb{Q} \times \mathbb{Q}.$$

This gives us the choice of defining division

- as a function with source and target

  $$\mathbb{Q} \times (\mathbb{Q} \setminus \{0\}) \longrightarrow \mathbb{Q}$$

- or as a *partial function*

  $$\mathbb{Q} \times \mathbb{Q} \rightharpoonup \mathbb{Q}\ ,$$

  which is undefined for all those pairs whose second component is 0, that is, pairs of the form $(q, 0)$.

In this example there is a choice regarding what to do, but it can be difficult, or even impossible, to calculate the domain where a particular partial function is defined. Examples of this appear in COMP11212.

Example 7.14. A more interesting example is that of subtraction for the natural numbers. We may define a partial function

$$\mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$$

which maps

$$(n, m) \longmapsto \begin{cases} n - m & n \geq m \\ \text{undefined} & \text{else.} \end{cases}$$

Alternatively a total function would have be[2]

$$\text{from } \{(n, m) \in \mathbb{N} \times \mathbb{N} \mid n \geq m\} \qquad \text{to } \mathbb{N}.$$

Example 7.15. Nothing stops us from defining, for example, for natural numbers n,

$$fn = \begin{cases} n & \text{the no of atoms in the universe is divisible by } n \\ \text{undefined} & \text{else} \end{cases}$$

Calculating where this function is defined is impossible (beyond repeating the definition). If we use this function together with a recursive definition we can define

Base case g. $\qquad g0 = 0.$

Step case g.

$$g(n+1) = \begin{cases} gn + f(n+1) & gn,\ f(n+1) \text{ both defined} \\ fn & fn \text{ defined, } gn \text{ undefined} \\ \text{undefined} & \text{else.} \end{cases}$$

Here it is even harder to work out whether g is defined for a given n, and there is no simple predicate which tells us whether gn is defined for a given n.

Example 7.16. In COMP11212 you will study the idea of a *partially decidable partial function*. This is a partial function for which we can find an algorithm to tell us where it is defined, but we demand the termination of the algorithm only in the case where the given partial function is defined at the given element. Specifically, the question is whether there is a While program which, for input n, will terminate and give the answer 1 if the function is defined at n. Such a program is known as a *partial decision procedure*.
In a computer science context a typical application of this idea is to have a program P that takes some input, for example a natural number, and produces

[2] Or we would have to turn it into a total function such as $\dot{-}$ from the previous section.

an output, say another natural number. We may then define a partial function

$$n \longmapsto \begin{cases} \text{output of } P \text{ on input } n & P \text{ produces output for input } n \\ \text{undefined} & \text{else.} \end{cases}$$

The reason why P might not produce an output for the given input could be that it attempts a division by 0, or that it runs into an infinite loop on certain inputs. You will study this idea in more detail in COMP11212.

We give a formal definition for the concept used in the examples above.

Definition 55: partial function

Let S and T be sets. A **partial function from** S **to** T is an assignment where

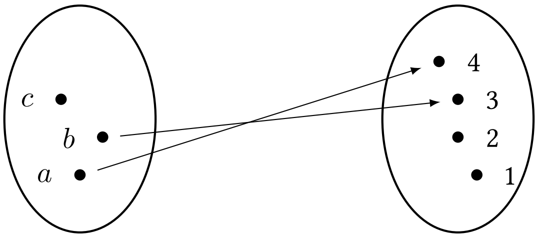
$$\text{for every } s \in S \qquad \text{there is at most one } t \in T \qquad \text{with} \quad s \longmapsto t.$$

We use a different kind of arrow, one with only 'half a tip'

$$f \colon S \rightharpoonup T$$

to indicate that the function described is partial.

When we have partial functions between small sets we can draw pictures of partial functions, similar to those used for (total) functions in Section 0.3. Here for every element of the source set we may have at most one element of the target set which it is connected with.



If f is a partial function then when we write fs we cannot be sure whether this defines an element of T, or whether this is undefined. This can be awkward when one tries to argue about partial functions. For this reason it is fairly customary to use an extra symbol[3] $\perp$.

If $f \colon S \rightharpoonup T$ is a partial function then we write

$$fs = \perp$$

in the case[4] where f is not defined at s.

Example 7.17. Every function from S to T is also a partial function from S to T.

When both, proper and partial functions are around people sometimes talk of **total functions** to distinguish proper functions, which are defined everywhere, from partial ones.

[3] This looks like the symbol we used in Chapter 3 for propositions whose boolean interpretation relative to every valuation is 0, and that is no coincidence. People usually pronounce it 'bottom'—see Definition 68 for the reason why.

[4] This only works if $\perp$ is not an element of the target set T.

Example 7.18. Recall the notion of a list over a set S from the previous chapter. We may want to define a function that returns the most recently added element of the list (if it exists). This should be a partial function head from the set of lists over S to S. It is partial because it is undefined for the empty list. There is a recursive definition which says

$$\text{head}(s\ :\ l) = s.$$

By not providing a definition for head at $[\,]$ we are implicitly stating that this function is not defined for the empty list, which means that it is partial. There really is no sensible way of extending this function to the empty list—how would we pick an element of S to return?

It is for reasons such as the function in this example (and the fact that when programming, illegal operations such as division by 0 cannot be prevented from happening in some automated way) that computer scientists have to consider partial functions.

Above we discussed the idea that we can make a partial function total by restricting its source to include only those elements for which the function is defined. This idea has a name.

The domain of definition **of a partial function** $f\colon S \rightharpoonup T$ is defined as

$$\text{dom}\, f = \{s \in S \mid fs \text{ is defined}\}.$$

Proposition 7.1

For every partial function $f\colon S \rightharpoonup T$ there is a unique total function

$$g\colon \text{dom}\, f \longrightarrow T$$

with the property that for all $s \in \text{dom}\, f$ we have

$$gs = fs.$$

Proof. We can use the desired equality as a definition for g. This clearly gives a total function with the required source and target and the condition ensures that for every element of $\text{dom}\, f$ there is a unique element of T to which to map it.

This means it is possible to work with total functions if we prefer to do so. However in theoretical computer science there exist a number of concepts around computability theory (and also in recursion theory in mathematics) which are more easily explained using partial functions. Examples occur in COMP11212.

The previous result has a counterpart where we move from total to partial functions.

Proposition 7.2

If $f\colon S \longrightarrow T$ is a (total) function from S to T and S' is a superset of S then there is a unique partial function

$$g\colon S' \rightharpoonup T$$

with the property that

$$\operatorname{dom} g = S \qquad\qquad \text{and} \qquad\qquad \forall s \in \operatorname{dom} g.\, gs = fs.$$

Exercise 173. Prove Proposition 7.2.

Partial functions can be composed just is the case for total functions, but we have to be careful about for which arguments the corresponding composite function is defined.

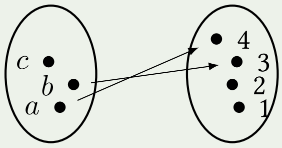
The **composite of partial functions** $f\colon S \rightharpoonup T$ and $g\colon T \rightharpoonup U$ is defined as:

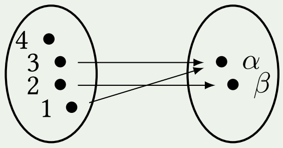
$$(g \circ f)s = \begin{cases} g(fs) & fs \text{ and } g(fs) \text{ are both defined} \\ \bot & \text{else.} \end{cases}$$

Alternatively one may say that $(g \circ f)s$ is defined if and only if s is in the domain of definition of f and fs is in the domain of definition of g.

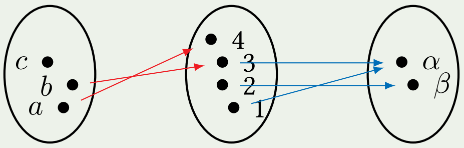
Note that this subsumes the definition of the composition of functions (defined in Section 0.3).

Example 7.19. We illustrate the notion of a composite of two partial functions by using a small example which can be described using a picture.



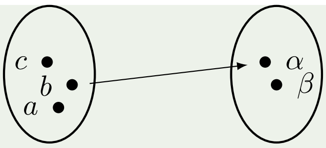


If we 'overlay' the two pictures we can more easily see what the composite has to look like.



The composite of the two partial functions can be drawn by the following procedure:

- Pick an element of the left hand set. If there is an outgoing arrow, follow that arrow. If there is an outgoing arrow from that element, follow that arrow and connect the original element with the resulting element.
- If at any point in the above procedure there is no outgoing arrow for one of the elements then the original element is not connected to anything.



CExercise 174. Consider the following partial functions with source $\mathsf{FBTrees}_S$.

(a) Give a recursive definition of a (possibly partial) function that takes a full binary tree and returns the label of the root of the tree. What is the domain of definition of your function? *Note that this is not a truly recursive function in the sense that when you define the step case you do not have to use the result of the function for the left and right subtrees.*

(b) Give a recursive definition of a (possibly partial) function that takes a full binary tree and returns its left subtree. What is the domain of definition of your function?

(c) What happens if you apply your second function, and then the first, to a tree? Describe the domain of definition of this composite.

EExercise 175. Let $f\colon S \rightharpoonup T$ and $g\colon T \rightharpoonup U$ be partial functions.

(a) Show that the domain of definition of $g \circ f$ is a subset of the domain of definition of f, that is

$$\operatorname{dom}(g \circ f) \subseteq \operatorname{dom} f.$$

(b) Give a description of the domain of definition of $g \circ f$ as a subset of $\operatorname{dom} f$, without mentioning $g \circ f$, nor the expression $g(fs)$ for some $s \in S$. *Hint: You may use* $\operatorname{dom} f$ *and* $\operatorname{dom} g$.

The *graph of a partial function* $f\colon S \rightharpoonup T$ is very similar to that of a (total) function (compare page 37). Its definition is

$$\{(s, fs) \in S \times T \mid f \text{ is defined at } s\}.$$

We can characterize those subsets of $S \times T$ which appear as the graph of a partial function (compare Proposition 2.1).

Proposition 7.3

A relation $R \subseteq S \times T$ is the graph of a partial function from S to T if and only if

$$\text{for all } s \in S \qquad \text{there exists at most one } t \in T \qquad \text{with} \qquad (s, t) \in R.$$

Proof. It is possible to make minor changes to the proof of Proposition 2.1 to obtain a proof of this result.

In the same way that we can view a function as a relation by considering its graph we may view a partial function that way, and this proposition tells us which relations are the graphs of partial functions.

Exercise 176. Show that for partial functions

$$f\colon S \rightharpoonup T \qquad \text{and} \qquad g\colon T \rightharpoonup U$$

we have that

$$\operatorname{gr}(g \circ f) = \operatorname{gr} f \mathbin{;} \operatorname{gr} g,$$

where ; is the relational composite defined on page 370, and gr applied to a function gives its graph. Note that this means in particular that composition of (total) functions is subsumed by relational composition.

7.2.1 Binary relations

For the remainder of this chapter we only consider relations from one set *to itself.* Instead of saying that R is a relation $S \longrightarrow S$ one typically says that R is a **(binary) relation on** S. Often it is convenient to drop the 'binary' in this case.

Example 7.20. Typical examples of binary relations on a set are the following.

(a) Sharing some property, such as having the same size, the same colour, the same nationality, speaking the same language, having the same digits after the decimal point, having a common divisor, evaluating to the same number,

(b) Having the same value under some function (some of the examples from above can also be viewed from this perspective, for example, one could map people to their height, or a real number to its sequence of digits after the decimal point, but one cannot do this for all these examples since a person may have more than one nationality or speak more than one language),

(c) Other kinds of connections between two members of the same set, for example one person having the contact details for another, or a number being less than or equal to another, or two subsets of a given set being included in each other.

(d) If we look at two objects of class List then we think of them as implementing the same list if the method equal (see Example 6.13) returns true. This defines a binary relation on objects of this class.

A different way of thinking of the same relation is as follows: Every object of class List can be thought of as implementing an element of $\mathsf{Lists}_{\mathbb{Z}}$ in the obvious way. This defines a function

$$\text{Objects of class List} \longrightarrow \mathsf{Lists}_S,$$

and two such objects are in the relation if and only if this function maps them to the same list.[5]

Note that for binary relations the 'source' and 'target' are identical, which means we may always form the relational composite of a binary relation with itself.

[5] Compare Example 6.13.

7.2.2 Picturing binary relations

If we have a binary relation on a small finite set S it is possible to draw[6] it in the form of a *directed graph*.

Definition 56: directed graph

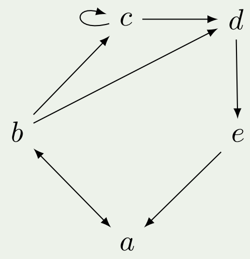
A **directed graph on a set** S is given[7] by a binary relation on S. In this context the elements of S are often called the *nodes* of the graph[8] and the pairs in the relation the *edges*.

Such graphs are often drawn as pictures when the set S is small.

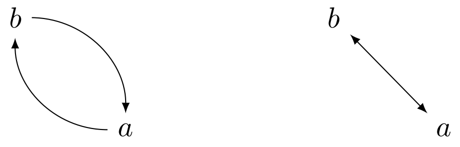
Example 7.21. Consider the following relation on the set $\{a, b, c, d, e\}$:

$$\{(a, b), (b, a)(b, c), (c, c), (c, d), (d, e), (e, a), (b, d)\}.$$

Its picture as a binary relation is as follows.



In this picture the arrow with tips at both ends is a shortcut to having arrows going each way.



Exercise 177. Draw the following binary relations.

(a) The powerset $\mathcal{P}\{a, b, c\}$ with the relation which relates a subset S of $\{a, b, c\}$ to the subset S' if and only if $S \subseteq S'$.

(b) The set $\{0, 1, 2, 3, 4, 5, 6\}$ with the relation which relates an element m to an element n if and only if they both leave the same remainder when divided by 3, that is, $m \bmod 3 = n \bmod 3$.

(c) The set $\{0, 1, 2, 3, 4, 5, 6\}$, with the relation which relates an element m to an element n if and only if $m \leq n$.

[6] Above we show how to generally draw relations from a small set S to a small set T, but this method makes use of the fact that the relation goes from a set to the same set.

[7] Some definitions of directed graph exclude connections between an element of S and itself. In that case the graph is given by a collection of pairs of the form (s, s') where s and s' are distinct elements of S. When that definition is used, our directed graphs become *directed graphs with loops*.

[8] Some people call the nodes of the graph the vertices.

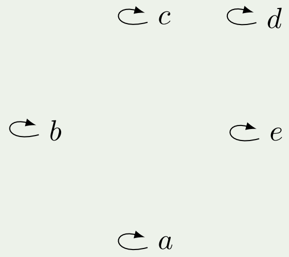
(d) The set $\{0, 1, 2, 3, 4, 5, 6\}$ with the relation which relates an element m to an element n if and only if m divides n.

(e) The compulsory course units for a student on your programme of studies, where one unit is related to another if it is a prerequisite for that unit (ignore corequisites).

(f) The students in your tutorial group where one student is related to another if they have the same gender.

The identity relation on a set can be pictured easily using these ideas. It is the relation which connects every element of the set with itself, and nothing else.

Example 7.22. For the set $\{a, b, c, d, e\}$ the identity relation is pictured as follows.

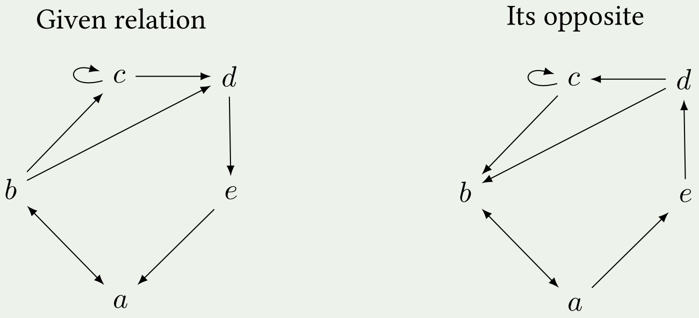


When we have a binary relation we picture its opposite slightly differently from the way illustrated in Example 7.6.

Example 7.23. For the relation on the set $\{a, b, c, d, e\}$ from Example 7.21,

$$\{(a, b), (b, a)(b, c), (c, c), (c, d), (d, e), (e, a), (b, d)\},$$

we show both, the given relation and its opposite.



The opposite relation, described as a set of pairs, is

$$\{(b, a), (a, b)(c, b), (c, c), (d, c), (e, d), (a, e), (d, b)\},$$

We can see how the opposite relation arises from the original by turning around all the arrows—which means that for loops, or for arrows with tips on both ends, nothing changes.

Further relations on a set

Very occasionally one may want to connect more than two elements of the same set, in which case one may speak of a

- ternary relation on S (a subset of $S \times S \times S$),
- a quarternary relation on S (a subset of $S \times S \times S \times S$),
- or, more generally, an n-ary relation on S (a subset of the n-fold product of S).

7.3 Equivalence relations

Sometimes when we look at all the elements of a set we don't necessarily wish to distinguish all the elements of the set. When you are building a specific structure from building blocks, in order to complete the design you don't have to consider the colour of a given block, just its shape. So from the point of view of the design, all blocks of the same shape are equivalent.[9] In COMP11212 you have met the notion of a bisimulation between two finite state automata. You can think of a bisimulation as a way of demonstrating that two given automata are equivalent, as far as the language they define is concerned.
More generally whenever we want to think of various entities as equivalent we have to make sure we do this in a safe way. Relations that allow us to do this are known as *equivalence relations*. These are relations satisfying a number of properties, and we introduce these properties one at a time.

7.3.1 Reflexivity

Definition 57: reflexive

A binary relation R on a set S is **reflexive** if and only if it is the case that we have

$$\text{for all } s \text{ in } S \qquad (s, s) \in R.$$

If we want to express properties of binary relations using first order logic we can express the relation as a two-placed predicate, where we take $R(x, y)$ to mean that R relates x and y, which we usually write as $(x, y) \in R$. Hence the first order logic formula that describes reflexivity of R is

$$\forall x. \, R(x, x).$$

We may express reflexivity in a very brief way by noting that it means that

$$I_S \subseteq R,$$

where I_S is the identity relation on the underlying set S.
This means that if our relation is reflexive then every element of the underlying set is related to itself. If we can draw a picture of the relation we can check whether the relation is reflexive by checking that every element has a connection from itself to itself, usually drawn as a little loop.
Typical examples are relations involving some kind of equality.

[9] But, of course, for aesthetic reasons, or to match an existing edifice, you may want to consider the colour when building a structure.

Example 7.24. We provide examples and non-examples for this concept.

(a) Consider the relation between first year students in the School where two students are related if and only if they are in the same tutorial group. Since every student is in the same tutorial group as him- or herself, this relation is reflexive.

(b) Look at the relation between two members of the human population where two people are related if and only if they have the same height in centimetres. Since everybody has the same height as him- or herself, this relation is reflexive.

(c) The relation between two non-zero natural numbers where m is related to n if and only if m divides n is reflexive since every natural number other than 0 divides itself.

(d) We define the relation between first year students in the School where student A is related to student B if A's student id number is below that of student B. This relation is not reflexive since a number is not below itself.

(e) Consider the relation between members of the human population which relates person A to person B if and only if they are siblings. This relation is not reflexive since nobody is their own sibling. On the other hand, relating two people if and only if they have a parent in common is a reflexive relation.

(f) Consider the relation between two natural numbers where m is related to n if $n = 2m$. Since, for example $1 \neq 2 \cdot 1$ this relation is not reflexive.

(g) The relation between elements of Lists_S which relates two lists if and only if they have the same number of elements is reflexive.

(h) The relation which relates two objects of class List if the method equal returns true (see Example 6.13) is reflexive.

(i) The relation of semantic equivalence for propositions is reflexive.

Exercise 178. Which of the following relations are reflexive? Justify your answer.

(a) The relation where two first year students in the School are related if their last name starts with the same letter.

(b) Two first year students in the School being related if there is a university society they both belong to.

(c) The following relation on the set $\{a, b, c, d\}$:

$$\{(a, a), (b, c), (c, b), (b, b), (c, d), (d, c), (d, d)\}$$

(d) The relation on $\mathbb{N}$ where two numbers are related if and only if they have a common divisor greater than 1.

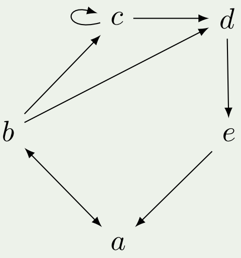
It is easy to make a relation reflexive in a unique and minimal way. Given a binary relation R on S the reflexive closure **of** R is given by

$$R \cup I_S = R \cup \{(s, s) \mid s \in S\}.$$

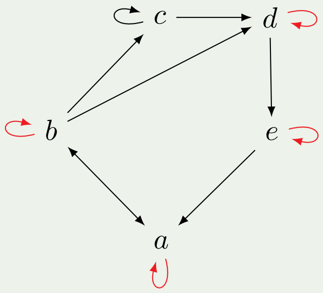
Since we add precisely those pairs to the relation which have to be present for it to satisfy reflexivity this is clearly the smallest relation we can form which contains R as a subset and which is reflexive. Optional Exercise 31 asks you to think about how to prove this.

When we consider a binary relation on a small finite set then checking whether the relation is reflexive, and drawing the reflexive closure, is easy. All we have to do is to make sure that every element has an arrow from itself to itself.

Example 7.25. If we go back to the relation from Example 7.21, we see that the relation is not reflexive since, with the exception of c, no element has a connection to itself.



The reflexive closure of this relation is pictured below.[10]



7.3.2 Symmetry

The next important property we consider for binary relations on a set is concerned with directedness: If we can go from one element to another, can we always go back?

Definition 58: symmetric

A binary relation R on a set S is **symmetric** if and only if we have

$$\text{for all } s, s' \in S \qquad (s, s') \in R \quad \text{implies} \quad (s', s) \in R.$$

The first order logic formula that describes this property for a binary predicate symbol R is

$$\forall x. \forall y. (R(x, y) \to R(y, x)).$$

[10] The new edges are drawn in red.

Relations built around the idea of equality of a property are usually symmetric, but there are plenty of relations which are not symmetric.

If we have a picture of a relation we can check whether it is symmetric by checking that every connection has an arrow at each end.

Example 7.26. We give some examples for relations which are symmetric, and some which are not.

(a) Consider the relation between first year students in the School where two students are related if and only if they are in the same tutorial group. If student A is in the same tutorial group as student B then student B is in the same tutorial group as student A, and so this relation is symmetric.

(b) Look at the relation between two members of the human population where two people are related if and only if they have the same height in centimetres. Since A having the same height as B implies that B has the same height as A this relation is symmetric.

(c) The relation between two non-zero natural numbers where m is related to n if and only if m divides n is not symmetric: 1 divides 2 but 2 does not divide 1.

(d) The relation between first year students in the School where student A is related to student B if A's student id number is below that of student B is not symmetric:[11] Indeed, if the id number for student A is below that of student B then that for student B cannot be below that for student A.

(e) The relation that relates two people if and only if they are siblings is symmetric, whereas the relation that relates two people if the first is the child of the second is not.

(f) Consider the relation between two natural numbers where m is related to n if $n = 2m$. This relation is not symmetric since 2 is related to 4 but 4 is not related to 2.

(g) The relation between elements of Lists_S which relates two lists if and only if they have the same number of elements is symmetric.

(h) The relation which relates two objects of class List if the method equal returns true (see Example 6.13) is symmetric.

(i) The relation of semantic equivalence for propositions is symmetric.

Again we can make a relation symmetric in a unique and minimal way.

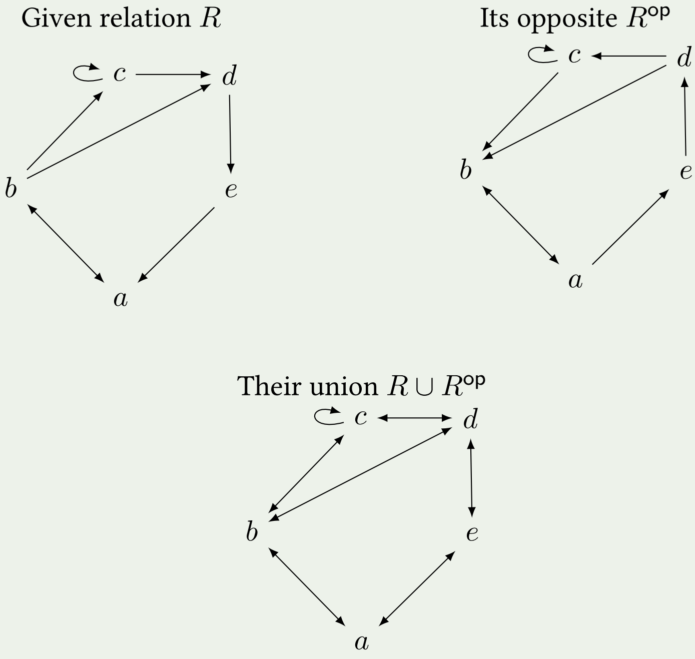
If R is a binary relation on a set S then the **symmetric closure of** R is given by

$$R \cup R^{\mathsf{op}}.$$

This is the smallest relation on S that contains R and is symmetric. Optional Exercise 31 asks you to think about how to prove this.

[11] This example seems to be the opposite of symmetric—to make that idea precise look at the notion of *anti-symmetry* defined in Section 7.4.1.

Example 7.27. If we return to Example we can see that taking the union of the given relation and its opposite we get a relation where every connection has arrows pointing both ways.



Given relation R

Its opposite R^{op}

Their union $R \cup R^{\mathsf{op}}$

This gives us a concise way of saying what it means for a relation R to be symmetric, namely

$$R = R \cup R^{\mathsf{op}}.$$

This can be simplified further by the observation that the non-trivial part of this equality is that

$$R \supseteq R \cup R^{\mathsf{op}},$$

and since R is always a superset of itself, symmetry is equivalent to demanding that

$$R \supseteq R^{\mathsf{op}}.$$

Another way of expressing symmetry using these ideas is to demand that

$$R = R^{\mathsf{op}}.$$

Above there is an argument that symmetry is equivalent to $R \supseteq R^{\mathsf{op}}$, and by applying the $(-)^{\mathsf{op}}$ operator on both sides this implies

$$R^{\mathsf{op}} \supseteq (R^{\mathsf{op}})^{\mathsf{op}} = R,$$

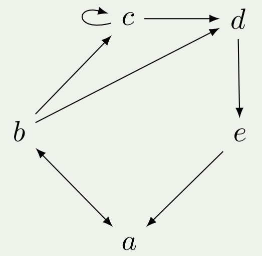
so the two must be equal.

An alternative way of describing the symmetric closure of R is given by

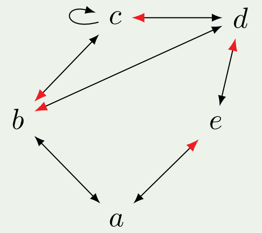
$$\{(s, s') \in S \times S \mid (s, s') \in R \text{ or } (s', s) \in R\}.$$

In other words, we add exactly those pairs to the relation which have to be present for the relation to become symmetric. If the relation is defined on a small finite set then we can once again look at the graph.

Example 7.28. Looking once again at the relation from Example 7.21 we can perform the check suggested above to see that it is not symmetric.



For example, (b, c) is in the relation but (c, b) is not. We may think of its symmetric closure as being constructed by adding all the arrow tips missing, and picture it[12] as follows. If we think of it this way we don't have to draw the opposite relation as in Example 7.27.



Again if we can draw the corresponding graph it is easy to see whether the relation is symmetric: We just have to check that every arrow that is not a loop has a tip at both ends.

Exercise 179. Which of the following relations is symmetric? Justify your answer.

(a) The relation where two first year students in the School are related if their last name starts with the same letter.

(b) The relation on first year students in the School where A is related to B if A can name student B when shown a picture.

(c) The following relation on the set $\{a, b, c, d\}$:

$$\{(a, a), (b, c), (c, b), (b, b), (c, d), (d, c), (d, d)\}$$

(d) The relation on $\mathbb{N} \setminus \{0\}$ where m is related to n if and only if m and n have a common divisor other than 1.

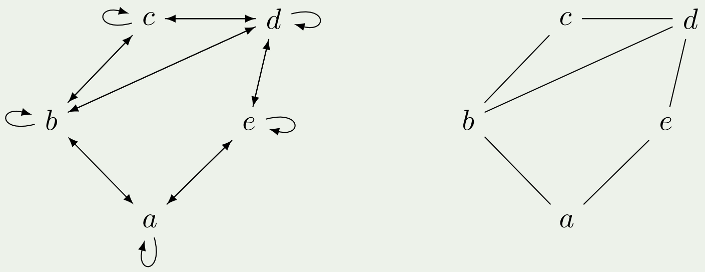
(e) The relation on $\mathbb{N} \setminus \{0, 1\}$ where m is related to n if and only if m divides a power of n.

Whenever we know a relation to be reflexive and symmetric we can picture it using an *undirected graph*. This is a graph where we only record which of the elements are connected, without worrying about the direction of that connection.

[12] New arrow tips drawn in red.

We do not record an element being connected with itself, we know they all are and so there's no reason to include that information in the picture.

Example 7.29. The symmetric closure of the reflexive closure of our example relation is drawn on the left, and on the right we draw the corresponding undirected graph where redundant information (for relations known to be reflexive and symmetric) has been removed.



7.3.3 Transitivity

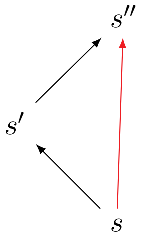
We require one additional property of relations to define the concept we are aiming for.

Definition 59: transitive

A binary relation R on a set S is **transitive** if and only if we have that

$$\text{for all } s, s', s'' \in S$$
$$(s, s') \in R \text{ and } (s', s'') \in R \qquad \text{implies} \qquad (s, s'') \in R.$$

This definition means that whenever we have a situation in the picture below, given the two black arrows we must have the red one.



The corresponding first order formula describing this property is

$$\forall x. \forall y. \forall z. \left((R(x, y) \wedge R(y, z)) \to R(x, z) \right).$$

Again, relations based on equality of a property are often transitive, but not always.

Example 7.30. (a) Consider the relation between first year students in the School where two students are related if and only if they are in the same tutorial group. If student A is in the same tutorial group as student B and student B is in the same tutorial group as student C then student A is in the same tutorial group as student C, and so this relation is transitive.

(b) Look at the relation between two members of the human population where two people are related if and only if they have the same height in centimetres. Since A having the same height as B and B having the same height as C implies that A has the same height as C this relation is transitive.

(c) The relation between first year students in the School where student A is related to student B if A's student id number is below that of student B is transitive: Indeed, if the id number for student A is below that of student B and that for student B is below that for student C then that for student A is below that for student C.

(d) The relation on the set of all humans which relates person A to person B if and only if A is a child of B is not transitive, but the relation that relates A to B if and only if B is an ancestor of A is. (See also Example 7.32.)

(e) Consider the relation between two natural numbers where m is related to n if $n = 2m$. This relation is not transitive since 2 is related to 4 and 4 is related to 8 but 2 is not related to 8.

(f) Consider the relation between human beings where person A is related to person B if there is a language they both speak. This relation is reflexive and symmetric but not transitive: Person A may speak English and Urdu, Person B may speak English and Spanish, and Person C may speak Spanish. The relation is

$$\{(A, B), (B, C)\}.$$

So A is related to B, and B is related to C, but A is not related to C.

(g) The relation between elements of Lists_S which relates two lists if and only if they have the same number of elements is transitive.

(h) The relation which relates two objects of class List if the method equal returns true (see Example 6.13) is transitive.

(i) The semantic equivalence relation between propositions is transitive.

We may define the **transitive closure of a relation R on a set S** as adding all those pairs

$$(s_1, s_n) \qquad \text{to} \qquad R$$

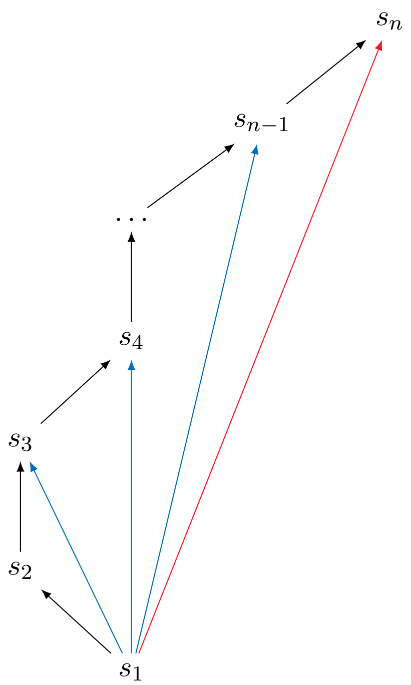
for which we can find a list of elements

$$s_1,\ s_2,\ \dots,\ s_n \text{ in } S, \qquad\qquad \text{with } n \geq 2,$$

such that

$$\text{for all } 1 \leq i \leq n-1 \qquad \text{we have} \qquad (s_i, s_{i+1}) \in R.$$

We may picture this situation as follows: Whenever we have the black arrows, we must have the blue arrows, and that means we must have the red arrow. You may understand the definition above as going straight from the black arrows to the red one.



If we look at a relation on a small finite set then transitivity is more difficult to check than reflexivity and symmetry. What we need to check here is that for all s, s' in S,

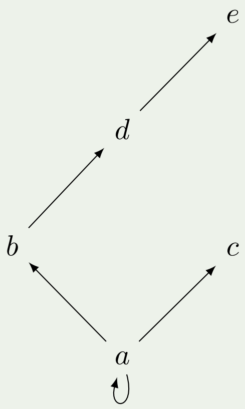
if there is a **path** from s to s' then there is an **edge** from s to s'.

In the relation from Example 7.21 the transitive closure requires us to connect *every pair of elements* since there is a path from every element to every element. This is messy to draw, so we consider a different example here.

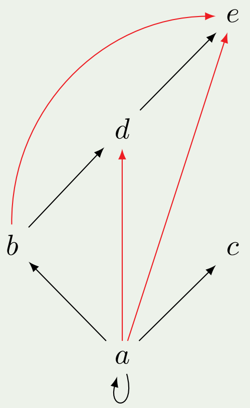
Example 7.31. Assume we have the relation

$$R = \{(a,a),(a,b),(a,c),(b,d),(d,e)\}.$$

The picture of this relation is as follows:



The transitive closure of this relation is given by[13]



Example 7.32. If we look at the relation 'is a parent of' (see Example 7.7) then its transitive closure gives us the 'is an ancestor' relation.

Instead of calculating the transitive closure of a relation in one go we can do it stepwise. For this we need to define the powers (with respect to composition of relations) of a binary relation. Consider he definition of the relational composite ; from page 370, and note that since a binary relation goes from a set to the same set we may compose such a relation with itself.
We may therefore give the following recursive definition. Let R be a binary relation on a set S.

Base case ˆ. $\quad R^0 = I_S.$

Step case ˆ. $\quad R^{n+1} = R^n \,;\, R.$

An intuitive explanation of R^n is the following:

s and s' in S are related by R^n
if and only if
one can get from s to s' by following the relation R exactly n times.

The transitive closure of R consists of all those pairs (s, s') for which it is possible to get from s to s' by following R any finite number greater than 0 times. This motivates a second recursive definition.

Base case R_n. $\quad R_0 = \emptyset.$

Step case R_n. $\quad R_{n+1} = R_n \cup R^{n+1}.$

s and s' in S are related by R_n
if and only if
one can get from s to s' by following the relation R at most n times.

Note that R^0, the identity relation on S, is not involved in computing the transitive closure of a relation.

[13] New edges drawn in red.

Example 7.33. Assume the relation R describes railway journeys, so that two stations are in the relation if and only if one can travel from the first to the second having to change trains. Then R^2 gives us pairs of stations for which it is possible to travel from the first to the second by changing exactly once, R^3 gives us pairs of stations where one has to change exactly twice, whereas R_3 gives those where one has to change at most twice.

We define

$$R_\infty = \bigcup_{n\in\mathbb{N}} R_n = \bigcup_{n\in\mathbb{N}\setminus\{0\}} R^n.$$

This is the relation which can be described as follows:

s and s' in S are related by R_∞
if and only if
one can get from s to s'
by following the relation R a finite number of times.

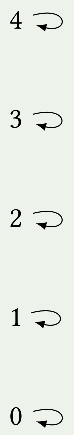
This gives us a way of stating that a relation R is transitive, since this is the case if and only if

$$\forall n \in \mathbb{N} \setminus \{0\} \qquad \text{we have} \qquad R^n \subseteq R.$$

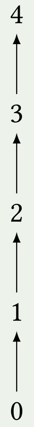
Example 7.34. If, for example, we have the set $\{0, 1, 2, 3, 4\}$ with the relation

$$R = \{(0, 1), (1, 2), (2, 3), (3, 4)\}$$

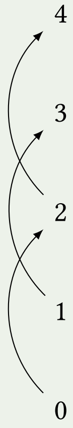
then we have the following.



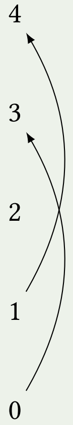
$$R^0 = \{(0, 0), (1, 1), (2, 2), (3, 3), (4, 4)\}$$



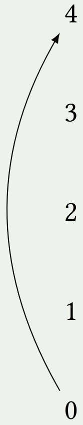
$R^1 = R$



$R^2 = \{(0,2),(1,3),(2,4)\}$



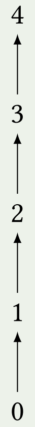
$R^3 = \{(0,3),(1,4)\}$



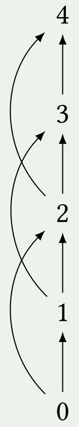
$R^4 = \{(0,4)\}$

All R^n, for $n \geq 5$, are empty.
We draw the relations R_n for this example.

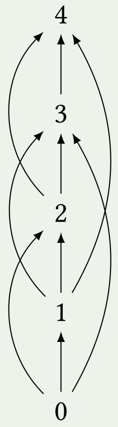
$R_0 = \emptyset$



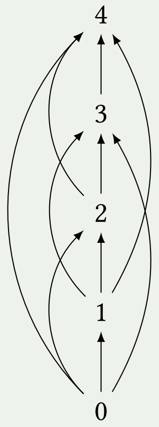
$R_1 = R^1 \cup R_0 = R$



$R_2 = R^2 \cup R_1$



$R_3 = R^3 \cup R_2$.



$R_4 = R^4 \cup R_3$

All remaining R_n, for $n \geq 5$, are equal to R_5. For this reason the last picture, showing R_4, shows the transitive closure of R.

Exercise 180. Show that R_∞ is the transitive closure of R.

CExercise 181. Which of the following relations are transitive? Justify your answers. For those which are not transitive describe the transitive closure.

(a) The relation on the set of first year students within the School which relates student A to student B if they have at least one course unit in common.

(b) The relation on the set of first year students within the School which relates student A to student B if they have the same nationality.

(c) The relation on $\mathbb{N}$ where m is related to n if and only if m divides n.

(d) The relation on $\mathbb{N}$ where m is related to n if and only if $m + n$ is even.

(e) The relation on $\mathbb{N}$ where m is related to n if and only if m and n have a common divisor greater than 1, or if $m = n = 1$.

(f) The following relation on the set $\{a, b, c, d, e\}$:

$$\{(a, a), (a, b), (b, e), (e, c)\}$$

(g) The following relation on the set $\{a, b, c, d, e\}$:

$$\{(a, a), (a, b), (c, d), (d, e)\}$$

(h) The following relation on the set $\{a, b, c, d, e\}$:

$$\{(a, a), (b, b), (c, c), (e, e)\}$$

If we want to create the transitive closure of the symmetric closure of the reflexive closure of R all we have to do to this procedure is to change what we do at the start. We have to change the relation we use

$$\tilde{R} = I_S \cup R \cup R^{\mathsf{op}}$$

This achieves two objectives:

- By adding all elements of the identity relation we add all pairs (s, s). where $s \in S$, we make sure the relation we produce is reflexive.
- By adding all elements of the relation R^{op} we add all the pairs (s', s) for which $(s, s') \in R$, which ensures that we produce a symmetric relation.

Proposition 7.4

The transitive closure of the symmetric closure of the reflexive closure of a relation R on a set S is given by $\tilde{R}_\infty$.

Proof. By Exercise 180 we know that given a relation R' the relation R'_∞ is transitive. We observe that forming $I_S \cup R$ is the reflexive closure of R, and that the symmetric closure of the result is

$$\begin{aligned}(I_S \cup R) \cup (I_S \cup R)^{\mathsf{op}} &= (I_S \cup R) \cup ({I_S}^{\mathsf{op}} \cup R^{\mathsf{op}}) \\ &= I_S \cup R \cup R^{\mathsf{op}} \qquad\qquad {I_S}^{\mathsf{op}} = I_S.\end{aligned}$$

The remainder follows from the following exercise.

Note that the procedure of forming the reflexive symmetric transitive closure is important. If somebody tells you about particular instances they want you to consider as equivalent then this allows you to generate an equivalence relation which makes all the specified entities equivalent, but does not identify anything unnecessarily. This procedure is used in Section 7.3.4.

EExercise 182. The various closures of relations defined above work well together.

(a) Show that the symmetric closure of a reflexive relation is reflexive.

(b) Show that the transitive closure of a reflexive relation is reflexive.

(c) Show that the transitive closure of a symmetric relation is symmetric.

Conclude that the transitive closure of the symmetric closure of a reflexive relation is reflexive, symmetric and transitive. It is usually referred to as the *reflexive symmetric transitive closure. Hint: You may want to use Exercise 180, but you don't have to.*

This exercise shows that if we form the transitive closure of the symmetric closure of the reflexive closure of a relation then the result will be reflexive, symmetric, and transitive.

Optional Exercise 31. Above we talk about how to add a minimal number of elements to a relation to make it reflexive, symmetric or transitive. In this exercise we make these ideas precise.

(a) Show that if R' is a relation on a set S which is reflexive and which contains the relation R as a subset then R' contains the reflexive closure of R.

(b) Show that if R' is a relation on a set S which is symmetric and which contains the relation R as a subset then R' contains the symmetric closure of R.

(c) Show that if R' is a relation on a set S which is transitive and which contains the relation R as a subset then R' contains the transitive closure of R.

(d) Conclude that the reflexive/symmetric/transitive closure of a relation R is the smallest reflexive/symmetric/transitive relation which contains R as a subset.

Optional Exercise 32. There is another way of defining the three closure operations for relations. Instead of adding elements to the give relation one may think about starting with a large relation and then removing all those pairs of elements which are not required.

(a) Show that the intersection of arbitrarily many reflexive relations is reflexive.

(b) Show that the intersection of arbitrarily many symmetric relations is symmetric.

(c) Show that the intersection of arbitrarily many transitive relations is transitive.

(d) Prove that the intersection of all reflexive/symmetric/transitive relations containing a relation R is the smallest reflexive/symmetric/transitive relation containing R and conclude (with the help of the previous optional exercise) that this intersection is equal to the reflexive/symmetric/transitive closure of R.

7.3.4 Equivalence relations defined

Definition 60: equivalence relation

A binary relation on a set is an **equivalence relation** if it is reflexive, symmetric and transitive.

A number of examples are given above, but we put them together here:

Example 7.35. We give examples of equivalence relations.

(a) From every function we get an equivalence relation by relating those elements of the source set which are mapped to the same element in the target set, see Exercise 184.

A number of the relations mentioned above fall under this idea.

(i) Considering building blocks of the same shape equivalent has the underlying function which maps a block to its shape.

(ii) Mapping people to their height in centimetres leads to identifying those that have the same height.

(iii) Mapping students to their tutorial group allows us to identify the members of the same group.

(iv) Mapping people to their nationality[14] allows us to talk about the nationalities represented in a particular group.

(v) Another example taken from programming is the following: Given a specific algorithm there are many programs which implement that algorithm in a particular programming language.[15] Typically we don't care which particular program is used, only that it implements the chosen algorithm correctly. The underlying function here maps each program to the algorithm it implements.

(b) Relating elements of a set which have a particular property is usually a special case of the previous example, because we can map the elements of the set to the corresponding property (for example nationality, or tutorial group). But one has to be careful here in cases where there is no underlying function: Speaking a common language is not an equivalence relation since this need not be a transitive relation, see above on page 389.

(c) One might want to identify all the sets which have the same size, which leads to the notion of *cardinal* in mathematics.[16]

(d) When we consider which algorithm to use to solve a particular problem we may be worried about its complexity only. In that case we typically don't worry about distinguishing between algorithms that are in the same *complexity class*.[17]

(e) In COMP11212 and COMP26120 you will learn about the notion of 'big O'. There is an underlying equivalence relation on functions where f and g are equivalent if and only if

- there exists $m \in \mathbb{N}$ such that nf eventually dominates g and

- there exists $n \in \mathbb{N}$ such that ng eventually dominates f.

This is an equivalence relation.

(f) The notion of semantic equivalence from Chapter 3 defines an equivalence relation on propositions, where we only care about the boolean interpretations of a given proposition.

(g) The relation between elements of Lists_S which relates two lists if and only if they have the same number of elements is another example.

(h) The relation which relates two objects of class List if the method equal returns true (see Example 6.13) is an equivalence relation.

CExercise 183. Which of the following relations are equivalence relations? Justify your answers.

(a) The relation where two first year students in the School are related if their last name starts with the same letter.

(b) Two first year students in the School being related if there is a university society they both belong to.

(c) The $\equiv$ relation on propositional formulae from Chapter 4.

(d) The relation on Java objects of the class java.lang.Object defined by object A being related to object B if and only if the default instance method A.equals(B) returns true.

(e) The relation on objects of the class BTree, where t1 is related to t2 if and only if the following method (compare Code Example 6.8, when called as equal(t1,t2), returns true:

```
public static boolean equal (BTree t1, BTree t2)
{
 if (t1 == null)
     return (t2 == null);
 else {
     if (t2 == null)
         return false;
     else
         return ((t1.value == t2.value)
         && equal (t1.left, t2.left)
         && equal (t1.right, t2.right));
   }
```

[14] This ignores the possibility of dual citizenship.

[15] Formally defining what this means is tricky, but you should get the general idea.

[16] Formally one has to be a bit careful since there is no such thing as the ‘set of all sets’, but the idea remains.

[17] A formal definition of what that means is given in COMP11212.

}

(f) The following relation on the set $\{a, b, c, d, e\}$:

$$\{(a, a), (b, b), (a, b), (b, a), (a, c), (c, a), (b, c), (c, b), (d, d), (e, e)\}$$

(g) The reflexive closure of the following relation on the set consisting of the elements a, b, c, d and e:

$$\{(a, b), (b, a), (b, c), (c, b)\}$$

(h) The reflexive closure of the following relation on the set $\{a, b, c, d, e\}$:

$$\{(a, b), (b, a)\}$$

(i) The relation on $\mathbb{N}$ where m is related to n if and only if $m + n$ is even.

(j) The relation on $\mathbb{N}$ where m is related to n if and only if m and n have a common divisor greater than 1 or if $m = n = 1$.

(k) The relation on $\mathbb{N} \setminus \{0\}$ where m is related to n if and only if

$$m \bmod n = 0.$$

Exercise 184. We look at the idea that functions generate equivalence relations. Assume that $f \colon S \longrightarrow T$ is a function.

(a) Show that the following defines an equivalence relation on S:

$$s \sim_f s' \qquad \text{if and only if} \qquad fs = fs'.$$

(b) Show that f is injective if and only if the corresponding equivalence relation $\sim_f$ from (a) is given by the identity relation on S,

$$I_S = \{(s, s) \mid s \in S\}.$$

(c) Show that if $g \colon T \longrightarrow U$ is a function then

$$s \sim_{g \circ f} s' \qquad \text{if and only if} \qquad s \sim_f s' \text{ or } fs \sim_g fs'.$$

Given a binary relation R on a set the **equivalence relation generated by a binary relation R** is the relation obtained by forming the transitive closure of the symmetric closure of the reflexive closure of R. The resulting relation is reflexive, symmetric and transitive by Exercise 182. On page 395 there is a description of how to calculate that closure step-by-step. See Example 7.49 for a concrete example of carrying out the procedure.

We use an equivalence relation wherever we wish not to distinguish between certain elements of a set.

Example 7.36. When we use fractions to refer to rational numbers we do this with respect to the following equivalence relation: For m, n, m', n' in $\mathbb{Z}$ we set

$$\frac{m}{n} \sim \frac{m'}{n'} \qquad \text{if and only if} \qquad mn' = m'n.$$

One typically writes

$$\frac{m}{n} = \frac{m'}{n'}$$

in that situation. How to define the rational numbers formally, and how that connects with this idea of fractions, is explained in Section 7.3.7.

7.3.5 Equivalence classes—modular arithmetic

When we have an equivalence relation we often do not wish to distinguish between elements which are equivalent. We look at one important example before we consider the general case.

When we calculate 'modulo' a given number we are not really interested in the numbers involved, just in the remainder they leave when dividing by the given number.

Example 7.37. For the simplest example assume that we are concerned only whether a number is odd or even. We know we have rules that allow us to calculate with 'even' and 'odd':

$+$	even	odd
even	even	odd
odd	odd	even

$\cdot$	even	odd
even	even	even
odd	even	odd

We may make this idea formal by defining an equivalence relation on $\mathbb{N}$, or on $\mathbb{Z}$, and then calculating with the equivalence classes.

For i and j in $\mathbb{N}$ (or in $\mathbb{Z}$) we set

$$i \sim j \qquad \text{if and only if} \qquad i \bmod 2 = j \bmod 2.$$

It is easy to check that this is an equivalence relation. We have two equivalence classes for this relation,

- the even numbers, all of which leave remainder 0 when divided by 2 and which all are in $[0]$
- the odd numbers, all of which leave remainder 1 when divided by 2 and all of which are in $[1]$.

So we have a new set where we distinguish between elements of $\mathbb{N}$ only up to odd- and evenness. This is known as the *quotient set*, and the formal notation is

$$\mathbb{N}/_{\sim} = \{[0], [1]\}.$$

This idea is formally introduced in Definition 61 below.

We can calculate with these equivalence classes by adding or multiplying them, defining[18]

$+$	$[0]$	$[1]$
$[0]$	$[0]$	$[1]$
$[1]$	$[1]$	$[0]$

$\cdot$	$[0]$	$[1]$
$[0]$	$[0]$	$[0]$
$[1]$	$[0]$	$[1]$

These tables fit those given for even and odd numbers above.

But, in fact, these operations may be derived from the addition and multiplication operations that exist on the set we started with, $\mathbb{N}$.

Note that if we have two pairs of numbers, say i, j, i' and j' in $\mathbb{N}$ with the property that

$$i \sim i' \qquad \text{and} \qquad j \sim j'$$

then both,

$$i + j \sim i' + j' \qquad \text{and} \qquad ij \sim i'j'.$$

To show this we have to go through all the possible cases:

- If i and j are both even then so are i' and j', and the sum of i and j is even, as is that of i' and j'. In that case the product of i and j is also even, as is that of i' and j'.
- If i is even and j is odd, then i' is also even, and j' is also odd. The sum of i and j is odd, as is that of i' and j', and the product of i and j is even, as is that of i' and j'.
- If i is odd and j is even then by commutativity of addition and multiplication on $\mathbb{N}$ the argument from the previous case applies.
- If i and j are both odd then so are i' and j', and the sum of i and j is even, as is that of i' and j'. The product of i and j is odd as is that of i' and j'.

What this means is that in order to find the result of

$$[i] + [j]$$

all we have to do is pick any element of $[i]$, say i', and any element of $[j]$, say j', and calculate $i' + j'$—the result $[i] + [j]$ we are looking for is $[i' + j']$.

In other words, if we define

$$[i] + [j] = [i + j],$$

then this definition works. This is a non-trivial observation for the following reason:

What we have proved above is that, for example, we can pick any number in $[0]$, say 6, and any number in $[1]$, say 17, we can add them to each other and the result will tell us the result of $[0] + [1]$:

$$[0] + [1] = [6] + [17] = [6 + 17] = [23] = [1].$$

We might say that our equivalence relation is well-behaved for our given operations of addition and multiplication, and mathematicians might say that it is a *congruence relation with respect to both* $+$ *and* $\cdot$.
We refer to calculating 'modulo 2' when we think in this way. There is nothing special about 2, and below we consider calculations modulo other numbers.

Exercise 185. Determine the properties of both, multiplication and addition, for $\mathbb{N}/_\sim$. Are they associative or commutative? Do they have a unit? Do inverses exist for them? How does this compare with the properties of the corresponding operations for $\mathbb{N}$?

Mathematicians have a name for the set of equivalence classes from Example 7.37 with these operations, they call it the two-element field $\mathbb{F}_2$.
Calculating modulo 2, that is, with odd and even numbers, is just one of infinitely many such cases.

Example 7.38. We may look at calculating modulo 3, for i and j in $\mathbb{N}$ setting

$$i \sim j \qquad \text{if and only if} \qquad i \bmod 3 = j \bmod 3.$$

Like many relations based on an equality this is an equivalence relation as well. There are now three equivalence classes, namely

- the numbers divisible by 3, all of which are in $[0]$,
- the numbers which leave a remainder of 1 when divided by 3, all of which are in $[1]$ and
- the numbers which leave a remainder of 2 when divided by 3, all of which are in $[2]$.

Again we would like to add and multiply these equivalence classes, but it is a bit tedious to show separately for each n that calculating modulo n is safe, so we do it once and for all.

Proposition 7.5

Let n, k, l, k' and l' be natural numbers with the property that

$$k \bmod n = k' \bmod n \qquad \text{and} \qquad l \bmod n = l' \bmod n.$$

Then

$$\begin{aligned} (k+l) \bmod n &= (k'+l') \bmod n \\ \text{and} \qquad kl \bmod n &= k'l' \bmod n. \end{aligned}$$

Proof. We know from Fact 2 that there exist unique m, m', i, i', j and j' in $\mathbb{N}$ with

$$0 \leq m, m' \leq n-1$$

[18] Note that we are using the operations $+$ and $\cdot$ in two different senses, once for numbers and once for equivalence classes.

and

$$k = in + m \qquad k' = i'n + m \qquad l = jn + m' \qquad l' = j'n + m'.$$

We calculate

$$k + l = in + m + jn + m' = (i + j)n + m + m'$$

and conclude[19] that

$$(k + l) \bmod n = (m + m') \bmod n.$$

We further calculate

$$k' + l' = i'n + m + j'n + m' = (i' + j')n + m + m'$$

and conclude that

$$(k' + l') \bmod n = (m + m') \bmod n = (k + l) \bmod n.$$

We next calculate

$$\begin{aligned} kl &= (in + m)(jn + m') \\ &= ijn^2 + im'n + jmn + mm' \\ &= (ijn + im' + jm)n + mm', \end{aligned}$$

and conclude that

$$kl \bmod n = mm' \bmod n.$$

Similarly we calculate

$$\begin{aligned} k'l' &= (i'n + m)(j'n + m') \\ &= i'j'n^2 + i'm'n + j'mn + mm' \\ &= (i'j'n + i'm' + j'm)n + mm', \end{aligned}$$

and so

$$k'l' \bmod n = mm' \bmod n = kl \bmod n$$

as required.

Hence for every number n we may calculate with equivalence classes modulo n. Moreover, the resulting operations are commutative and associative, and have a unit. To see commutativity, note that for all i and j in $\mathbb{N}$ we have

$$[i] + [j] = [i + j] = [j + i] = [j] + [i].$$

The argument for associativity is similar. The unit for addition is $[0]$ since for all i in $\mathbb{N}$ we have

$$[i] + [0] = [i + 0] = [i] = [0 + i] = [0] + [i].$$

The unit for multiplication is $[1]$, and the proof is very similar.

[19] This uses that for all k, i and n in $\mathbb{N}$ we have $k \bmod n = (in + k) \bmod n$, see Exercise 25.

Example 7.39. We continue Example 7.38 by giving the tables for calculating modulo 3:

$+$	$[0]$	$[1]$	$[2]$
$[0]$	$[0]$	$[1]$	$[2]$
$[1]$	$[1]$	$[2]$	$[0]$
$[2]$	$[2]$	$[0]$	$[1]$

$\cdot$	$[0]$	$[1]$	$[2]$
$[0]$	$[0]$	$[0]$	$[0]$
$[1]$	$[0]$	$[1]$	$[2]$
$[2]$	$[0]$	$[2]$	$[1]$

CExercise 186. Carry out the following studies.

(a) Give tables for the addition and multiplication of equivalence classes when calculating modulo 4. Identify units for addition and multiplication if they exist. Determine whether you have inverses for addition and multiplication.

(b) Give tables for the addition and multiplication of equivalence classes when calculating modulo 5. Identify units for addition and multiplication if they exist. Determine whether you have inverses for addition and multiplication.

(c) Give tables for the addition and multiplication of equivalence classes when calculating modulo 6. Identify units for addition and multiplication if they exist. Determine whether you have inverses for addition and multiplication.

Hint: You may want to go back to Section 2.5 to remind yourself of the notion of unit and inverse element for a binary operation.

Modular arithmetic is important in cryptography. It also appears as an important example in COMP26120. For this reason we look at more of the properties it has. From now on we use

$$k \sim_n l \qquad \text{to mean} \qquad k \bmod n = l \bmod n.$$

Proposition 7.6

The following hold for calculating modulo n.

(i) For all $n \in \mathbb{N} \setminus \{0, 1\}$, addition is commutative and associative for $\mathbb{N}/_{\sim_n}$; the additive unit is given by $[0]$ and every element of the set $\mathbb{N}/_{\sim_n}$ has an additive inverse.

(ii) For all $n \in \mathbb{N} \setminus \{0, 1\}$, multiplication is commutative and associative for $\mathbb{N}/_{\sim_n}$; the multiplicative unit is given by $[1]$, and the element m of $\mathbb{N}/_{\sim_n}$ has a multiplicative inverse if and only if the greatest common divisor of m and n is 1.

In particular, if n is a prime number then every number has both, an additive and a multiplicative inverse when calculating modulo n.

Proof. We show the two parts.

(i) Associativity and commutativity of addition follow immediately from

Proposition 7.5 since, for example,

$$\begin{aligned}[k]+[l]&=[k+l] && \text{def } + \text{ on } \mathbb{N}/{\sim_n}\\ &=[l+k] && \text{addition commutative on } \mathbb{N}\\ &=[l]+[k] && \text{def of } + \text{ on } \mathbb{N}/{\sim_n}.\end{aligned}$$

The fact that $[0]$ is the additive unit also follows immediately from Proposition 7.5. Let k be an arbitrary element of $\mathbb{N}$. By Fact 2 we know that $k \bmod n < n$. Hence we can set $l = n - k \in \mathbb{N}$. Using the same proposition we calculate that

$$[k]+[l]=[k \bmod n]+[l]=[k+l]=[n]=[0]$$

and

$$[l+k]=[n]=[0].$$

(ii) Commutativity and associativity of multiplication follows in the same way as for addition. The fact that $[1]$ is the multiplicative unit also follows from the Proposition 7.5. Regarding multiplicative inverses below we give a method for calculating the multiplicative inverse by giving employing Euclid's algorithm (see Example 6.42) and from the definition of the method it is clear that such an inverse is produced whenever the two numbers we start with have 1 as their largest common divisor.

This completes the proof.

For public key cryptography the concept of *modular exponentiation* is particularly important. If we take the powers of a natural number then we eventually reach numbers that are too large for whichever format we are using. This cannot happen in modular arithmetic, because when calculating modulo n we always get a number which is below n.

When performing these kinds of calculation then using the square brackets for equivalence classes becomes a bit tedious. What is typically done is to merely use the numbers

$$0, 1, 2, \ldots, n-2, n-1,$$

which represent their respective equivalence classes. Typical notation is then

$$2+3=5==0 \pmod 5 \qquad \text{or} \qquad 2\cdot 4=8=3 \pmod 5,$$

where previously we would have written

$$[2]+[3]=[0] \qquad \text{or} \qquad [2]\cdot[4]=[3],$$

where we didn't have a good way of specifying modulo which number we were doing our calculations.

Note that when we are carrying out calculations in modular arithmetic we may, at any point, move to a different representative of the equivalence class.

Example 7.40. By moving to different elements of some equivalence class we can make sure that we always calculate with the smallest numbers possible.

$$\begin{aligned}4\cdot 16+2\cdot 17 &= 64+34 \pmod 3\\ &= 1+1 \pmod 3\\ &= 2 \pmod 3.\end{aligned}$$

Note that it is customary to ensure that the final result is the canonical representative of its equivalence class, that is, when calculating modulo n, a number from 0 to $n-1$.

Multiplicative inverses. Finding multiplicative inverses modulo some given number is possible (where they exist), and we describe how this works.

Assume we have a number $n\in\mathbb{N}\backslash\{0,1\}$, and we are looking for the multiplicative inverse of m, modulo n, that is we want to determine

$$m^{-1} \pmod n.$$

This means we are asking for a number l, the multiplicative inverse, with the property that

$$m\cdot l = 1 \pmod n,$$

which is equivalent to there being a number i such that

$$m\cdot l = i\cdot n+1.$$

Assume we know that the greatest common divisor of m and n is 1. If we apply Euclid's algorithm to n and m, see Example 6.42, we can see that, for

$$r_0 = n \qquad \text{and} \qquad r_1 = m$$

we get equalities

$$\begin{aligned}r_0 &= k_1\cdot r_1+r_2\\ r_1 &= k_2\cdot r_2+r_3\\ &\dots\\ r_{n-2} &= k_{n-1}r_{n-1}+1\\ r_{n-1} &= k_n\cdot 1+0\end{aligned}$$

that is at some point we have

$$r_n = 1 \qquad\qquad r_{n+1} = 0,$$

since the r_i that appears before one obtains 0 is the greatest common divisor of $r_0 = n$ and $r_1 = m$.

We take the equalities and isolate the r_i with the highest index appearing in each we obtain the following.

$$\begin{aligned}r_2 &= r_0 - k_1\cdot r_1\\ r_3 &= r_1 - k_2\cdot r_2\\ &\dots\\ r_n &= 1 = r_{n-2} - k_{n-1}\cdot r_{n-1}\end{aligned}$$

We can now recursively substitute each r_i in the final equality according to the previous ones, which gives us an equality that expresses

$$1 \qquad \text{in terms of} \qquad r_0 = n \text{ and } r_1 = m.$$

This gives the desired solution for the original problem.

Example 7.41. We would like to calculate the multiplicative inverse for 5 modulo 7. Following Euclid's algorithm we calculate as follows

$$\begin{aligned} 7 &= 1 \cdot 5 + 2 \\ 5 &= 2 \cdot 2 + 1 \end{aligned}$$

We reorganize the two equalities as instructed above to obtain

$$\begin{aligned} 2 &= 7 - 1 \cdot 5 \\ 1 &= 5 - 2 \cdot 2. \end{aligned}$$

We insert the right hand side of the first equality for the right hand 2 in the second equality and get

$$1 = 5 - 2 \cdot (7 - 1 \cdot 5) = 3 \cdot 5 - 2 \cdot 7,$$

and so

$$3 \cdot 5 = 2 \cdot 7 + 1.$$

Hence the multiplicative inverse of 5 modulo 7 is 3. And indeed,

$$3 \cdot 5 = 15 = 2 \cdot 7 + 1 = 1 \pmod 7.$$

Example 7.42. We carry out another example calculation. The multiplicative inverse of 4 modulo 11 can be found as follows.

$$\begin{aligned} 11 &= 2 \cdot 4 + 3 \\ 4 &= 1 \cdot 3 + 1 \end{aligned}$$

Rearranging gives

$$\begin{aligned} 3 &= 11 - 2 \cdot 4 \\ 1 &= 4 - 1 \cdot 3 \end{aligned}$$

Inserting the first into the second shows that

$$1 = 4 - (11 - 2 \cdot 4) = 3 \cdot 4 - 11.$$

Hence the multiplicative inverse for 4 is 3, and indeed,

$$4 \cdot 3 = 12 = 1 \pmod{11}.$$

Example 7.43. In cases where the multiplicative inverse does not exist the algorithm proceeds as follows. Note that Euclid's algorithm calculates the greatest common divisor of the two given numbers, Compare Example 6.42. If the greatest common divisor of the two given numbers is a number other than 1 we cannot use this idea, and indeed in that case there is no multiplicative inverse for the chosen modular arithmetic. We wish to find a multiplicative inverse for 4 modulo 6.

$$\begin{aligned}6 &= 1 \cdot 4 + 2\\ 4 &= 2 \cdot 2 + 0.\end{aligned}$$

In other words we reach 0 as the remainder without having reached 1 first.

Example 7.44. We give one final example. We wish to calculate the multiplicative inverse for 11 modulo 17.

$$\begin{aligned}17 &= 1 \cdot 11 + 6\\ 11 &= 1 \cdot 6 + 5\\ 6 &= 1 \cdot 5 + 1\end{aligned}$$

Rearranging give

$$\begin{aligned}6 &= 17 - 1 \cdot 11\\ 5 &= 11 - 1 \cdot 6\\ 1 &= 6 - 1 \cdot 5.\end{aligned}$$

Inserting the first two into the final equality tells us that

$$\begin{aligned}1 &= 6 - 1 \cdot 5 && \text{given}\\ &= 6 - (11 - 1 \cdot 6) && \text{snd equality into third}\\ &= 2 \cdot 6 - 11 && \text{simplification}\\ &= 2 \cdot (17 - 1 \cdot 11) - 11 && \text{fst equality into result so far}\\ &= 2 \cdot 17 - 3 \cdot 11 && \text{simplification.}\end{aligned}$$

So the multiplicative inverse of 11 modulo 17

$$-3 = 14 \pmod{17}.$$

Again we can verify that

$$14 \cdot 11 = 154 = 9 \cdot 17 + 1 = 1 \pmod{17}.$$

Make sure that the answers you give are indeed numbers from 0 to $n-1$ when calculating modulo n in this notation. If in the previous example you give -3 as an answer you will lose marks since this number is not an element of the set we are considering.

Note that 0 cannot have a multiplicative inverse as is shown in the following.

Proposition 7.7

Let N be a set with two associative binary operations, multiplication and addition, in such a way that there is a unit 0 for addition and 1 for multiplication, with $0 \neq 1$, and such that every element has an additive inverse. Finally we assume the following distributivity law for all k, m and n in N:

$$k \cdot (m+n) = k \cdot m + k \cdot n.$$

Then we have for all $n \in N$ that

$$0 \cdot n = 0 = n \cdot 0,$$

and in particular 0 cannot have a multiplicative inverse.

Proof. Let n be an element of N.

$$\begin{aligned} 0 \cdot n &= (0+0) \cdot n && \text{0 unit for } + \\ &= (0 \cdot n) + (0 \cdot n) && \text{distributivity law} \end{aligned}$$

If we add the additive inverse of $0 \cdot n$ on both sides of this equality we get

$$0 = 0 \cdot n.$$

Similarly we can show $n \cdot 0 = 0$ in this situation.
Hence in this situation 0 cannot have a multiplicative inverse *unless* 0 is both, the additive and multiplicative unit, so $0 = 1$, and that requires that $S = \{0\}$ and that $+$ and $\cdot$ are the same operation.

Exponentiation. The modular operation of most interest in cryptography is that of exponentiation, and we look at the basics here.
If we look at the powers of $2 \pmod 5$ we find that they are

n	0	1	2	3	4	5	6	7	8	9	...
$2^n \pmod 5$	1	2	4	3	1	2	4	3	1	2	...

Of course the numbers used in cryptography are considerably larger than 5. For RSA, for example, two large primes are selected, and then exponentiation modulo their product is performed. For this and other applications it becomes important to be able to carry out exponentiation efficiently.
We look at the question of how to calculate

$$a^b \pmod n.$$

Code Example 7.1. The really naive method is to calculate a to the power of b as integers, and then calculate the remainder when dividing by n, that is[20]

```
public static int modpower (int a, int b, int n)
{
  if (b==0)
    return 1;
  else {
    int power = 1;
    for (int count=1, count <= b, count++)
```

```
            power = (power * a);
          return power % n;
      }
  }
```

Using this code quickly runs out of memory[21] despite the fact that we know that the result is a number from 0 to $n-1$. We look at a sample calculation of

$$13^5 \pmod{197}.$$

The table gives the value of power as the loop progresses.

count	1	2	3	4	5
power	13	169	2197	28561	371293

We then take the final result and calculate

$$371293 \bmod 197 = 145.$$

Code Example 7.2. The slightly less naive algorithm is to just keep multiplying by the base and calculating the remainder when dividing by n at each step.

```
public static int modpower (int a, int b, int n)
{
    if (b==0)
        return 1;
    else {
        int power = 1;
        for (int count=1, count <= b, count++)
            power = (power * a) % n;
        return power;
    }
}
```

This works better but still is not very efficient. It performs b many multiplications and remainder operations. Again, here is a sample calculation of the same number as before.

count	1	2	3	4	5
power	13	169	30	193	145

This program does not run out of memory as the first one inevitably does.

Note that this works because of Proposition 7.5: We want to calculate

$$a^b \pmod{n},$$

[20] Note that one could improve efficiency by checking whether a is equal to 0 or 1, but this would not improve matters much and we are looking for a short program.

[21] even if you pick one of the large integer classes in Java instead of **int**.

but we know that

$$i \cdot j \pmod{n} \qquad \text{has the same result as} \qquad (i \bmod n) * (j \bmod n),$$

and so taking the remainder when dividing by n at each step of the exponentiation process is the same as taking it at the end:

$$\begin{aligned} a^b \bmod n &= (a \cdot a^{b-1}) \bmod n \\ &= (a \cdot (a^{b-1} \bmod n)) \bmod n \qquad \text{Prop 7.5.} \end{aligned}$$

Indeed, one way of paraphrasing that proposition is to say that when adding or multiplying $\pmod{n}$ one may form the remainder when dividing by n at any time![22]

Code Example 7.3. In COMP26120 you will look at a third method for performing this calculation.

```
public static int modpower (int a, int b, int n)
{
    if (b==0)
        return 1;
    else {
        int power = 1;
        while (b != 0) {
            if (b % 2 == 1)
                power = (power * a) % n;
            a = (a*a) % n;
            b = b/2;
        }
        return power;
    }
}
```

Again we perform a sample calculation, but this time we have to track not just of the value held in the variable power, but also of what is held in the variables a and b, the latter of which controls the **while** loop. If that variable is even then nothing happens in the **while** loop, which we denote in the table by an empty cell for power. We repeat the above calculation, so at the start we have $a = 13$, $b = 5$ and $n = 197$. The method sets the variable power to 1, so we use that as our first value for that variable.

b		5	2	1	0
power	1	13		145	
a	13	169	193	16	

Because this is a short calculation it looks as if we have just as many steps to perform as before, since we now have to calculate changing values for a as well. But you can see that when b is large the fact that the **while** loop is carried out only $\log b$ many times becomes more important than the fact that each time we have between one and two multiplications to carry out.

The idea behind this algorithm is quite simple: Instead of multiplying with a the required number of times the number of multiplications is brought down by the squaring operation. To do this one may think of b in base 2.

Example 7.45. We illustrate how this works in an example: For $b = 5$, which is 101 in binary, we have

$$a^5 = a^4 \cdot a^1 = (a^2)^2 \cdot a^1 = a^{1 \cdot 2^2} \cdot a^{0 \cdot 2^1} \cdot a^{1 \cdot 2^0},$$

that is, we have one count of an iterated square of a for each position where b in binary has 1. A larger example is $b = 21$, where we have

	4	3	2	1	0
b	1	0	1	0	1
powers	$(((a^2)^2)^2)^2$		$(a^2)^2$		a

and so

$$a^{21} = a^{1 \cdot 2^4} \cdot a^{0 \cdot 2^3} \cdot a^{1 \cdot 2^2} \cdot a^{0 \cdot 2^1} \cdot a^{1 \cdot 2^0}.$$

The algorithm ensures that what is multiplied is the appropriate selection of iterated squares.

Example 7.46. We carry out one more example that illustrates how the two algorithms work with numbers where it's a bit easier to follow the calculation. We calculate

$$7^7 \pmod{11}.$$

We follow the second algorithm.

n	1	2	3	4	5	6	7
$7^n \pmod{11}$	7	5	2	3	10	4	6

For the third algorithm at the start we have $a = 7$, $b = 7$, $n = 11$.

b	7	7	3	1	0
power	1	7	2	6	
a	7	5	3	9	

EExercise 187. Carry out the following tasks.

(a) Find the multiplicative inverses, where they exist, for

$$7 \pmod{11}, \qquad 15 \pmod{17}, \qquad 8 \pmod{12}, \qquad 5 \pmod{9}.$$

Hint: See Examples 7.41 to 7.44.

(b) Calculate the following using the two algorithms from Code Examples 7.2

[22] In COMP11212 you will learn about *loop invariants* and then you can make a rigorous step-by-step argument that the two programs compute the same number.

and 7.3.

$$4^5 \pmod{13}, \qquad 7^4 \pmod{9}, \qquad 6^5 \pmod{11}, \qquad 8^6 \pmod{16}.$$

(c) Depending on b, how many times is the loop in the algorithm from Code Example 7.2 carried out? What about the other algorithm?

(d) For which of the following can you find an exponent different from 0 so that the given number is equal to 1 in the relevant modular arithmetic? Give the smallest such exponent if possible, argue why there is none otherwise.

$$7 \pmod{11}, \qquad 15 \pmod{17}, \qquad 8 \pmod{12}, \qquad 5 \pmod{9}.$$

Hint: There is no algorithm that determines the required number for all cases.

7.3.6 General equivalence classes

In the previous section we looked at a particular situation where we want to treat equivalent elements at the same. In general we may want to do this when we have an equivalence relation. In this situation we construct another set where there is only one representative for each class of equivalent elements, just as in $\mathbb{N}/_{\sim_2}$ there are only two elements, $[0]$ and $[1]$.
Given a set S with an equivalence relation R and an element s of S the **equivalence class with respect to R generated by s**, $[s]$, is the subset of S consisting of all elements of S which are related to s by R, that is

$$[s] = \{s' \in S \mid (s, s') \in R\}.$$

Note that since an equivalence relation is symmetric this is the same as

$$\{s' \in S \mid (s', s) \in R\}.$$

If the equivalence relation in question is not clear from the context, the equivalence class generated by s for a relation R is written as $[s]_R$. Usually equivalence relations are written in infix notation and for the remainder of this section that is the notation we use.

Example 7.47. We go through the examples given in Example 7.35.

(a) For the relations given by functions we have the following equivalence classes:

(i) building blocks of the same shape,

(ii) groups of people of the same height (up to the nearest centimetre),

(iii) tutorial groups,

(iv) all people of the same nationality,

(v) implementations of the same algorithm.

(b) This example is too general to give the equivalence classes.

(c) All sets which have the same size as a given form an equivalence class (for example, all sets with 5 elements, or all countably infinite sets).

(d) Algorithms which belong to the same complexity class form an equivalence class.

(e) All propositions with the same boolean interpretation (with respect to every valuation) form an equivalence class.

(f) Functions are in the same equivalence class if they eventually dominate each other, up to a factor.

(g) Objects of class List are in the same equivalence class for equal if and only if the underlying elements of Lists$_{\mathbb{Z}}$ have the same elements in the same order.

Alternatively we can describe equivalence classes for this relation as follows. Two objects l1 and l2 of the class List are in the same equivalence class if and only if

- their instance variables l1.value and l2.value give the same integer value and
- their instance variables l1.next and l2.next are references to lists which are in the relation.

Example 7.48. The last example deserves another look, in particular if we look at the second description. This suggests that there is a recursive procedure for deciding whether two objects are in this relation, and indeed there is. We use the idea of defining a set recursively, see Section 6.4.3.

Base case E_{List}. $(\textbf{null}, \textbf{null}) \in E_{\text{List}}$.

Step case E_{List}. $(\text{l1}, \text{l2}) \in E_{\text{List}}$ and m==n for m:**int** and n:**int** implies

$$(\textbf{new}\ \text{List}\ (\text{m,l1}), \textbf{new}\ \text{List}\ (\text{n,l2})) \in E_{\text{List}}.$$

The base case tells us that two List objects that are a **null** reference are considered equivalent, and using the step case repeatedly we can build up to longer and longer lists being considered equal.

Equivalence classes split the given set into disjoint blocks of equivalent elements and we say that they *partition* the set. In other words, they give us a new set where we no longer distinguish between equivalent elements.

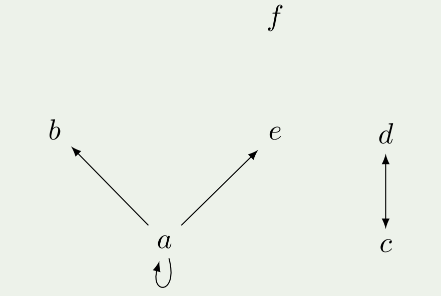
Definition 61: quotient set

Given a set S witn an equivalence relation $\sim$, the **quotient set of S with respect to $\sim$** consists of the equivalence classes of S with respect to $\sim$. This is written as $S/_\sim$.

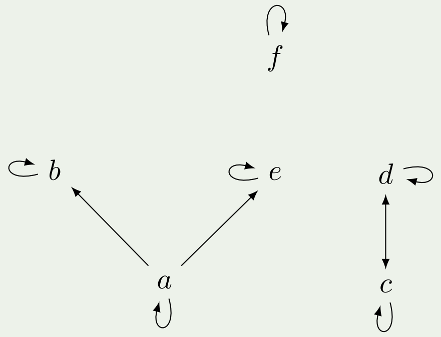
Example 7.49. We begin our study of formal examples with a relation on a small finite set $\{a, b, c, d, e, f\}$, namely

$$\{(a, a), (a, b), (a, e), (c, d), (d, c)\}.$$

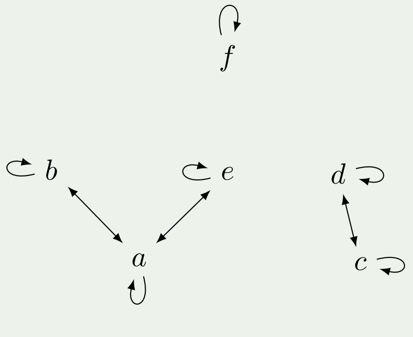
As before we picture the relation using a directed graph.



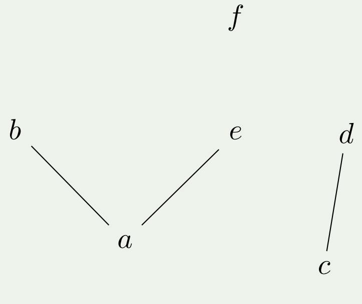
The reflexive closure of this relation adds connections to itself for each element:



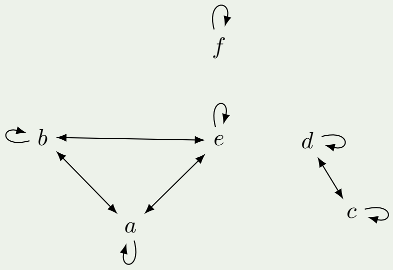
The symmetric closure of the relation turns all edges between different elements into double-tipped ones:

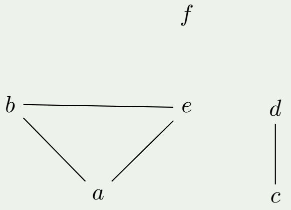


If we draw the result as an undirected graph we lose some of the now redundant information:

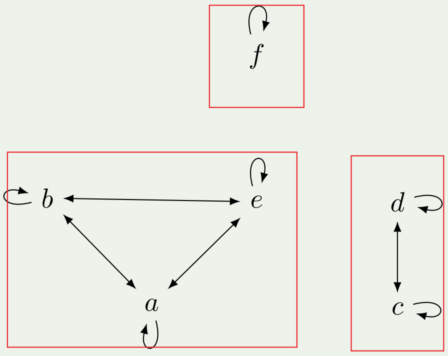


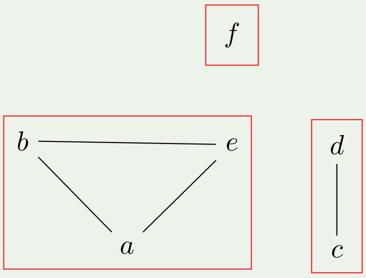
The transitive closure of this relation, both as a directed and as an undirected graph:





We draw the equivalence classes in both graphs:





We have three equivalence classes:

$$[a] = [b] = [e] = \{a, b, e\}, \qquad [c] = [d] = \{c, d\}, \qquad [f] = \{f\}.$$

Note the following: If we pick any pair of nodes in an equivalence class then there is a connection between them (and this is a two-sided connection in the case of a directed graph). This is true for all equivalence classes in all equivalence relations.

Example 7.50. We look at another example. In Chapter 3, an algorithm is described which, given a propositional formula, arrives at a conjunctive normal form for that formula. We may think of this as defining a relation on propositional formulae.
If we also allow the rules for simplifying formulae, and take the reflexive

symmetric transitive closure of the resulting relation, we obtain an equivalence relation for propositional formulae.
Two formulae are equivalent for that relation if and only if, starting with both given formulae, we can apply the rules in such a way that we arrive at the same CNF.
This is the same equivalence relation as that which considers two formulae equivalent if and only if for every valuation they give the same truth table.
The equivalence class of a formula, say P, is then the set of all formulae that have the simplified CNF P, such as $P \wedge P$ or $P \vee P \vee \bot$.

Example 7.51. If you are writing a programme where queues are implemented, and there is one central resource that all existing queues need to access (for example processing time) then you might want to implement a procedure which allocates the resource to the longest queue. From the point of view of that program, it is only important how long the queues are, and not what elements they have. We use this idea, but for our previously defined type of list.
We define a binary relation on Lists_S where

$$l \sim l' \qquad \text{if and only if} \qquad \mathrm{len}\, l = \mathrm{len}\, l',$$

using the len function from Exercise 141.
Two elements of Lists_S are in the relation $\sim$ if and only if they have the same number of elements. This means that there are infinitely many equivalence classes, one for each natural number. The empty list is the only element of its equivalence class, but as long as S has more than one element there is more than one element in all the other equivalence classes. For example, for the list $[s]$ the equivalence class consists of all the $[s']$ for which $s' \in S$.
One can define a corresponding relation on objects of class List where

l1 is related to l2

if and only if the following method, called as

length(l1)==length(l2),

where length is the method from Section 6.1.3 defined in detail in Exercise 141.

Note that whenever a set is *partitioned*, that is, split into disjoint subsets, there is an equivalence relation at the heart: Given a set S, all we have to do is to define

$$s \sim s' \qquad \text{if and only if} \qquad s \text{ and } s' \text{ are in the same partition.}$$

In this case the equivalence classes are exactly the partitions. This means that partitioning a set is exactly the same thing as forming the equivalence classes for an equivalence relation.

CExercise 188. For the following relations, calculate the equivalence relation they generate, try to[23] describe the resulting equivalence classes, and count their number.

(a) The reflexive symmetric transitive closure of the following relation on the set consisting of the elements a, b, c, d, e and f:

$$\{(a, b), (b, a), (b, c), (d, e), (e, f)\}.$$

(b) The reflexive symmetric transitive closure of the following relation on the same set as in the previous part:

$$\{(a, a), (b, c)\}.$$

(c) The reflexive symmetric transitive closure of the following relation on the same set as in the previous part:

$$\{(a, a), (b, c), (c, d), (d, e)\}.$$

(d) On the set $\mathbb{N}$ the relation $m \sim n$ if and only if $m \bmod 4 = n \bmod 4$. What is $[1]$ in this example? Can you describe $\mathbb{N}/_{\sim}$?

(e) Consider the set D of decimal numbers. We assume here that such a number consists of a finite number of digits from 0-9, followed by a decimal point, followed by an infinite number of digits 0–9. (Note that this is not how we usually write decimal numbers—we drop all (or most) of the infinitely many 0s that appear.) Take the reflexive symmetric closure $\sim$ of the relation where two numbers are related if and only if:

- the first number ends with infinitely many 0s,
- the second number ends with infinitely many 9s and
- the (finitely many) digits to the left of these are equal, with the exception of the right-most such digit, which is one less for the second number.

What is $[1]$ in this example? Can you describe $D/_{\sim}$?

(f) On the natural numbers $\mathbb{N}$ the relation where $m \sim n$ if $m + n$ is even (compare Exercise 183). Can you describe $\mathbb{N}/_{\sim}$?

(g) On the complex numbers the relation where $a + bi$ is related to $a' + b'i$ if and only if $a = a'$.

(h) On the complex numbers the relation where z is related to z' if and only if $z\overline{z} = z'\overline{z'}$.

Example 7.52. We recursively define a binary relation on the set Lists_S of lists over the set S as follows.

Base case $\sim$. $[\,] \sim [\,]$

Step case $\sim$. For $l \sim l'$ and $s, s' \in S$ we have

$$s : l \sim s' : l'.$$

[23] For the finite examples you should list all equivalence classes.

The remainder of this example is concerned with understanding what this relation does. We show first of all that it is a reflexive relation, that is, that each list is related to itself. This is a proof by induction.

Base case $\sim$. We note that

$$[\,] \sim [\,] \qquad \text{base case } [\,].$$

Ind hyp. We assume the statement holds for the list l, that is $l \sim l$.

Step case $\sim$. We see that given $s \in S$ we have that

$$s\,:\,l \sim s\,:\,l \qquad \text{by step case } \sim.$$

We can also show by induction that this relation is also symmetric and transitive. We add a proof for the former.

Base case $\sim$. We note that

$$[\,] \sim [\,] \qquad \text{base case } [\,],$$

and so the only instance of using the base case of the definition of $\sim$ results in a symmetric relation.

Ind hyp. We assume the statement holds for the lists l and l', that is $l \sim l'$ implies that $l' \sim l$.

Step case $\sim$. We see that the only way of building further instances of the relation $\sim$ is to take note that, for $l \sim l'$, as well as s and s' in S we get

$$s\,:\,l \sim s'\,:\,l' \qquad \text{by step case } \sim,$$

but we also get, from the induction hypothesis, that $l' \sim l$, and so

$$s'\,:\,l' \sim s\,:\,l \qquad \text{by step case } \sim,$$

and so the relation remains symmetric as we add additional instances.

So what does this relation do? One possibility is to look at some examples, and let's assume that the underlying set S is $\mathbb{N}$. We know that we start with

$$[\,] \sim [\,].$$

The step case tells us that we can now add any one element to $\emptyset$ and all the resulting lists are related, so

$$[0] \sim [1] \sim [2], \ldots$$

and all lists of length one are related. We can take any two of these, and add an element to each, and get two more related lists, which means that all lists of length 2 are related.
This gives rise to the conjecture that for all lists l and l' we have

$$l \sim l' \qquad \text{if and only if} \qquad \operatorname{len} l = \operatorname{len} l'.$$

We give a formal proof of this. This is an 'if and only if' statement and we show it by proving each part separately.
We first show that if $l \sim l'$ then l and l' have the same length.

Base case $\sim$. We have the base case of the relation,

$$[\,] \sim [\,],$$

and we can see that both sides are equal, so applying the length function gives the same result.

Ind hyp. For the lists l and l' we have that $l \sim l'$ implies $\operatorname{len} l = \operatorname{len} l'$.

Step case $\sim$. If we have lists l and l' with $l \sim l'$ and for s and s' in S we use the step case of $\sim$ to derive that

$$s : l \sim s' : l'$$

we may conclude that

$$\begin{aligned} \operatorname{len}(s : l) &= 1 + \operatorname{len} l && \text{step case len} \\ &= 1 + \operatorname{len} l' && \text{ind hyp} \\ &= \operatorname{len}(s' : l') && \text{step case len} \end{aligned}$$

In the other direction we want to show that if for two lists l and l' we know that $\operatorname{len} l = \operatorname{len} l'$ then we have $l \sim l'$. How do we know that the two lengths are equal? This can only happen by another inductive process: The length of a list can only be 0 if the list is empty, which gives the base case. If we have two lists l and l' of equal lengths, then for s and s' in S we have

$$\begin{aligned} \operatorname{len}(s : l) &= 1 + \operatorname{len} l && \text{step case len} \\ &= 1 + \operatorname{len} l' && \text{assumption} \\ &= \operatorname{len}(s' : l'). \end{aligned}$$

Base case $\sim$. We have the base case of equal length

$$\operatorname{len} [\,] = \operatorname{len} [\,],$$

and we can see that

$$[\,] \sim [\,]$$

by the base case of $\sim$.

Step case. If we have lists l and l' of equal length, which allows us, for s and s' in S, to conclude that

$$\text{len}(s : l) = \text{len}(s' : l'),$$

then we get that $l \sim l'$ from the induction hypothesis and we may conclude that

$$s : l \sim s' : l'$$

by the step case of $\sim$.

Hence the equivalence relation defined in this example is the same as that from Example 7.51.

Understanding recursive definitions is not easy, and typically one wants to find an alternative description that is easier to grasp.

CExercise 189. Consider the following relation on $\mathsf{FBTrees}_S$.

Base case $\sim$. For all $s, s' \in S$ we have

$$\texttt{tree}\, s \sim \texttt{tree}\, s'.$$

Step case $\sim$. For $t \sim t'$, $t'' \sim t'''$ and $s, s' \in S$ we have

$$\texttt{tree}_s(t, t'') \sim \texttt{tree}_{s'}(t', t''').$$

(a) Which of the following trees are $\sim$-related?

(i) $\texttt{tree}_2(\texttt{tree}\,1, \texttt{tree}\,3)$

(ii) $\texttt{tree}_3(\texttt{tree}\,2, \texttt{tree}_3(\texttt{tree}\,2, \texttt{tree}\,1))$

(iii) $\texttt{tree}_3(\texttt{tree}_2(\texttt{tree}\,1, \texttt{tree}\,1), \texttt{tree}_2(\texttt{tree}\,3, \texttt{tree}\,2))$.

(iv) $\texttt{tree}_3(\texttt{tree}_2(\texttt{tree}\,1, \texttt{tree}\,3), \texttt{tree}_2(\texttt{tree}\,3, \texttt{tree}\,1))$.

(v) $\texttt{tree}_3(\texttt{tree}_2(\texttt{tree}\,1, \texttt{tree}\,3), \texttt{tree}\,1))$.

(vi) $\texttt{tree}_3(\texttt{tree}_2(\texttt{tree}\,1, \texttt{tree}\,1), \texttt{tree}_2(\texttt{tree}\,3, \texttt{tree}\,2))$.

(vii) $\texttt{tree}_3(\texttt{tree}\,2, \texttt{tree}\,3)$

Hint: You may want to draw them and then think about which trees are related. Begin with trees related by the base case, and then apply the step case once or twice to see how it all works

(b) Prove by induction that the relation is reflexive. In fact, it is an equivalence relation and you may use this to answer the following part.

(c) Informally describe the equivalence classes of this relation. How would you describe all the trees that are equivalent to a given one?

(d) How would you implement a corresponding relation for objects of class BTree? You are trying to write a method with the following first line:

public static boolean similar (BTree t1, BTree t2)

The method should return true exactly when the two input trees are related by $\sim$.

Exercise 190. Show the following for an arbitrary equivalence relation $\sim$ on a set S.

(a) For all $s \in S$ we have $s \in [s]$.

(b) For all s, s', s'' in S we have

$$s', s'' \in [s] \qquad \text{implies} \qquad s' \sim s''.$$

(c) For all s and s' in S we have that

$$s \sim s' \qquad \text{implies} \qquad [s] = [s'].$$

(d) For all s and s' in S we have

$$[s] \cap [s'] \text{ is either empty or equal to } [s].$$

(e) For every element of S there is an equivalence class it belongs to.

(f) The equivalence classes for the relation split S into a pairwise disjoint collection of sets.

Hence the equivalence classes of S partition the whole set into disjoint subsets.

While it may be useful to determine a new set which has only one representative for each equivalence class, this construction becomes significantly more useful when we have sets with operations on them. The following sub-section cover relevant examples.

7.3.7 Important examples

The integers

The point of this and the following account of the rationals is to show you how to formally define these numbers with their operations. The material is not examinable as such, but does provide more examples for using equivalence relations and understanding it may be helpful for answering other exam questions.
Consider the following set:

$$\mathbb{N} \times \mathbb{N} = \{(m, n) \mid m, n \in \mathbb{N}\}.$$

On this set we define a relation, namely

$$(m, n) \sim (m', n') \qquad \text{if and only if} \qquad m + n' = m' + n.$$

It is easy to see that this is an equivalence relation: Since for all m and n in $\mathbb{N}$ we have that $m + n = m + n$ it is reflexive, and symmetry is obvious from the definition. Transitivity is obtained as follows:

$$(m, n) \sim (m', n') \qquad \text{and} \qquad (m', n') \sim (m'', n''),$$

imply

$$m + n' = m' + n \qquad \text{and} \qquad m' + n'' = m'' + n'$$

respectively, and together these imply that

$$m + n' + m' + n'' = +m' + n + m'' + n'$$

which by commutativity and assoiativity of addition is equivalent to

$$(m + n'') + (n' + m') = (m'' + n) + (m' + n')$$

which by the final statement of Fact 1 implies that

$$m + n'' = m'' + n.$$

which means that we have

$$(m, n) \sim (m'', n'').$$

We define an operation we call ‘addition’ on this set, by setting

$$(m, n) \oplus (m', n') = (m + m', n + n').$$

We next show that

$$\begin{array}{ll} (k,l) \sim (m,n) \text{ and } (k',l') \sim (m',n') & \\ \text{implies} \qquad (k,l) \oplus (k',l') \sim (m,n) \oplus (m',n'), & \end{array}$$

by calculating

$$(k,l) \oplus (k',l') = (k+k', l+l') \qquad \text{and} \qquad (m,n)+(m',n') = (m+m', n+n')$$

and

$$\begin{aligned} k + k' + n + n' &= (k + n) \oplus (k' + n') && \text{comm, ass } + \\ &= (m + l) \oplus (m' + l') && (k,l) \sim (m,n),\ (k',l') \sim (m',n') \\ &= m + m' + l + l' && \text{comm, ass } + . \end{aligned}$$

This allows us to define an addition operation *on the set of equivalence classes with respect to* $\sim$, $(\mathbb{N} \times \mathbb{N})/_{\sim}$.

Given two arbitrary equivalence classes for our set, if we pick two respective elements, say (k, l) and (m, n) we may set

$$[(k, l)] + [(m, n)] = [(k, l) \oplus (m, n)]$$

since our previous calculation assures us that no matter which elements of each equivalence class we pick, the result of adding the two always determines the same equivalence class.

It is fairly easy to show that addition is commutative and associative for our original set, $\mathbb{N} \times \mathbb{N}$, and the same is then true for the derived operation on $(\mathbb{N} \times \mathbb{N})/_{\sim}$. We can also check that there is a unit, namely the equivalence class of $(0, 0)$: Given $(m, n) \in \mathbb{N} \times \mathbb{N}$ we have

$$[(m, n)] \oplus [(0, 0)] = [(m + 0, n + 0)] = [(m, n)] = [(0, 0)] \oplus [(m, n)].$$

This raises the question of whether we have inverses with respect to this operation. In other words, given $(m, n) \in \mathbb{N} \times \mathbb{N}$, can we find an element whose equivalence class we can add to $[(m, n)]$ such that the result is $[(0, 0)]$? To find the answer to this question it is a good idea to first think about what the elements of $[(0, 0)]$ are. An element (m, n) of our set is equivalent to $(0, 0)$ if and only if

$$m + 0 = 0 + n,$$

that is, if $m = n$.
Given $(m, n) \in \mathbb{N} \times \mathbb{N}$ we note that

$$[(m, n) \oplus [(n, m)] = [(m + n, n + m)] = [(0, 0)],$$

and so we do indeed have an inverse element, namely $[(n, m)]$ for $[(m, n)]$. In other words, $(\mathbb{N} + \mathbb{N})/_\sim$ is a commutative group with respect to addition. Indeed, this is nothing but an alternative description for a group you are all familiar with. As a preliminary calculation we note that for $(m, n) \in \mathbb{N} \times \mathbb{N}$ with $m \geq n$ we have

$$[(m, n)] = [(m - n, 0)],$$

since

$$m + 0 = m = m - n + n$$

and so

$$(m, n) \sim (m - n, 0).$$

Consider the following function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{Z}$:

$$(m, n) \longmapsto m - n.$$

This function is constant on equivalence classes, since if we have

$$(m, n) \sim (m', n')$$

then we have

$$m - n = m + n' - n' - n = m' + n - n' - n = m' - n'.$$

Hence we may define a function f from $(\mathbb{N} \times \mathbb{N})/_\sim$ to $\mathbb{Z}$ by setting

$$f\colon [(m, n)] \longmapsto m - n,$$

without worrying which element of $[(m, n)]$ we may be referring to. We further define a function g from $\mathbb{Z}$ to $(\mathbb{N} \times \mathbb{N})/_\sim$ by setting

$$g\colon i \longmapsto \begin{cases} [(i, 0)] & i \geq 0 \\ [(0, -i)] & \text{else.} \end{cases}$$

We claim that this defines a bijection between the two sets: Given $i \in \mathbb{Z}$ we calculate

- if $i \geq 0$ we have $f(gi) = f[(i, 0)] = i - 0 = i$, and
- if $i < 0$ we have $f(gi) = f[(0, -i)] = 0 - (-i) = i$.

Given $(m, n) \in \mathbb{N} \times \mathbb{N}$ we have that $f[(m, n)] = m - n$, and

- if $m - n \geq 0$ we have that
$$g(f[(m, n)]) = g(m - n) = [(m - n, 0)] = [(m, n)],$$
- if $m - n < 0$ we have that
$$g(f[(m, n)]) = g(m-n) = [(0, -(m - n))] = [(0, -m + n))] = [(m, n)].$$

This shows that the two functions are mutual inverses. But much more is true: These two functions *preserve the addition operation*, that is, we have, for (k, l) and (m, n) in $\mathbb{N} \times \mathbb{N}$ that

$$\begin{aligned} f[(k, l)] + f[(m, n)] &= (k - l) + (m - n) && \text{def } f \\ &= (k + m) - (l + n) && \text{ass, comm } +, \text{ Ex } 34 \\ &= f[(k + m, l + n)] && \text{def } f \\ &= f([(k, l)] + [(m, n)]) && \text{def addition,} \end{aligned}$$

and for i and j in $\mathbb{Z}$ we have

$$gi + gj = [(i, 0)] + [(j, 0)] = [(i + j, 0)] = g(i + j).$$

This tells us that our set $(\mathbb{N} + \mathbb{N})/_\sim$ really is describing the set $\mathbb{Z}$ with its addition operation by simply giving a different name to all the elements.
Indeed, the *formal definition of the set of integers*, $\mathbb{Z}$, is $(\mathbb{N} \times \mathbb{N})/_\sim$, where we use the integer $m - n$ as a name for the equivalence class of (m, n).
We can also define multiplication on our 'formal integers', $(\mathbb{N} \times \mathbb{N})/_\sim$ by setting

$$[(m, n)] \cdot [(m', n')] = [(mm' + nn', mn' + m'n)].$$

Exercise 191. This exercise is concerned with checking that this definition of multiplication makes sense and fits with the usual one on $\mathbb{Z}$.

(a) Show that for (k, l), (k', l'), (m, n) and (m', n') in $\mathbb{N} \times \mathbb{N}$ we have that $(k, l) \sim (k', l')$ and $(m, n) \sim (m', n')$ implies

$$(k, l) \cdot (m, n) \sim (k', l') \cdot (m', n').$$

This means that our definition of multiplication works properly on equivalence classes with respect to $\sim$.

(b) Show that $[(1, 0)]$ is the unit for this multiplication.

(c) Show that for $i, j \in \mathbb{Z}$ we have $(gi) \cdot (gj) = g(ij)$.

(d) Show that for $(k, l), (m, n) \in \mathbb{N} \times \mathbb{N}$ we have

$$f[(k, l)]f[(m, n)] = f([(k, l)] \cdot [(m, n)]).$$

Hence our formal integers can play the role of $\mathbb{Z}$ for all the usually considered operations, namely addition (which gives subtraction thanks to the existence of inverses) and multiplication. Mathematicians say that the two are *isomorphic as rings*.

Optional Exercise 33. In the formal definition of $\mathbb{Z}$ we use a number of properties that hold on a very general level. This exercise is about understanding those properties better. Assume you have an equivalence relation $\sim$ on a set S.

(a) Define a function q from S to $S/_\sim$ which maps each element to its equivalence relation. This is known as the *quotient map*.

(b) Assume you have a function f from S to a set T with the property that, for $s, s' \in S$ we have $s \sim s'$ implies $fs = fs'$. Define a function $f_\sim$ from $S/_\sim$ to T with the property that $f_\sim \circ q = f$. Can you argue that $f_\sim$ is uniquely determined by these properties?

(c) Assume you have a binary operation $\circledast$ on S such that for $s, s', t, t' \in S$ we have $s \sim t$ and $s' \sim t'$ implies $s \circledast s' \sim t \circledast t'$ (we may think of this as saying that the operation is well-behaved with respect to the equivalence relation)[24]. Define a binary operation $\circledast$ on S with the property that for all $s, s' \in S$ you have

$$qs \circledast qs' = q(s \circledast s').$$

Can you show that this operation is uniquely determined by these properties?

(d) Assume that we have the same situation as in the previous part. Show that if the given operation is commutative or associative, then the newly defined operation has the same property. Further show that if e is a unit for the operation $\circledast$ then $[e]$ is a unit for the operation $\circledast$. Lastly prove that if s' is an inverse for s with respect to $\circledast$ then $[s']$ is an inverse for $[s]$ with respect to $\circledast$.

(e) Check that the addition and multiplication operations for $(\mathbb{N} \times \mathbb{N})/_\sim$ given in the text above are special cases of such binary operations.

Definition 62: integers

The commutative ring $\mathbb{Z}$ of integers is (formally) defined as the quotient of $\mathbb{N} \times \mathbb{N}$ with respect to the equivalence relation given above, and with addition and multiplication operations also given above.

The rationals

We can use a similar construction to obtain a formal definition of the rational numbers. As a basis we use the set

$$\mathbb{Z} \times (\mathbb{Z} \setminus \{0\}) = \{(i, m) \mid i,\, m \in \mathbb{Z},\, m \neq 0\}.$$

We use the following equivalence relation on this set:

$$(i, m) \sim (j, n) \qquad \text{if and only if} \qquad in = jm.$$

[24] If we think of the operation as the more fundamental item we say that the relation is a *congruence relation for the given operation.*

Exercise 192. Show that the relation defined above is an equivalence relation.

We may then form the set $(\mathbb{Z} \times (\mathbb{Z} - \setminus\{0\}))/_\sim$ of $\sim$-equivalence classes. As before we are concerned with defining operations on this new set. This time we begin with multiplication. For i, j in $\mathbb{Z}$ and m, n in $\mathbb{Z} \setminus \{0\}$ we set[25]

$$[(i, m)] \cdot [(j, n)] = [(ij, mn)].$$

This defines a commutative and associative operation with unit $[(1, 1)]$. We have inverses for this operation: Assume we have $i, m \in \mathbb{Z}$ with $m \neq 0$. We claim that the inverse for $[(i, m)]$ is $[(m, i)]$ and verify this by calculating

$$[(i, m)] \cdot [(m, i)] = [(im, mi)] = [(1, 1)],$$

and

$$[(m, i)] \cdot [(i, m)] = [(mi, im)] = [(1, 1)].$$

We still owe the connection between our set of equivalence classes and the set of rational numbers $\mathbb{Q}$. This is a bit trickier to describe cleanly than the case of the integers above because effectively the rational numbers have been treated as a quotient all along: Given a rational number we know that we may find integers m and n, with $n \neq 0$ such that our number is equal to the fraction m/n. But m and n are not defined uniquely, and so we effectively use an equivalence relation on fractions, where

$$\frac{m}{n} \approx \frac{m'}{n'} \qquad \text{if and only if} \qquad mn' = m'n.$$

This suggests that the proper translation between our set

$$(\mathbb{Z} \times (\mathbb{Z} \setminus \{0\}))/_\sim \qquad \text{and} \qquad \mathbb{Q}$$

is as follows. We define a function f from $(\mathbb{Z} \times (\mathbb{Z} \setminus \{0\}))/_\sim$ to $\mathbb{Q}$ by setting

$$[(m, n)] \longmapsto \frac{m}{n},$$

and a function g in the opposite direction by setting

$$\frac{m}{n} \longmapsto [(m, n)].$$

Exercise 193. This exercise is concerned with checking that the claims made above are true.

(a) Show that for (i, m), (i', m'), (j, n) and (j', n') in $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$ we have that $(i, m) \sim (i', m')$ and $(j, n) \sim (j', n')$ implies $(ij, mn) \sim (i'j', m'n')$. This means that our definition of multiplication works properly on equivalence classes with respect to $\sim$.

(b) Show that $[(1, 1)]$ is the unit for this multiplication.

(c) Observe that for i, j, m, n from $\mathbb{Z}$ with $m, n \neq 0$ we have

$$(i, m) \sim (j, n) \qquad \text{if and only if} \qquad \frac{i}{m} \approx \frac{j}{n}$$

and conclude that $\approx$ is an equivalence relation. Further note that this implies

[25] Again one has to check that this makes sense—see the exercise below.

that the definition of g makes sense by mapping equivalent fractions to the same equivalence class for $\sim$ in $(\mathbb{Z} \times (\mathbb{Z} \setminus \{0\}))$.

(d) Show that for $i, j, m, n \in \mathbb{Z}$ with $m, n \neq 0$ we have $g(\frac{i}{m}) \cdot g(\frac{j}{n}) = g(\frac{i}{m}\frac{j}{n})$.

(e) Show that for $(i, m), (j, n) \in \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$ we have

$$f[(i, m)]f[(j, n)] = f([(i, m)] \cdot [(j, n)]).$$

This means that our $(\mathbb{Z}\times(\mathbb{Z}\setminus\{0\}))/_\sim$ really is a way of talking about the rationals cleanly, that is, we know exactly what this set looks like, where before we did not have a clean way of referring to its elements. But so far we have only considered multiplication and still owe the addition operation.
We set, for i, j, m and n in $\mathbb{Z}$, with $m, n \neq 0$,

$$[(i, m)] + [(j, n)] = [(in + jm, mn)].$$

This defines a commutative associative operation with unit $[(0, 1)]$, and with inverses where the additive inverse of $[(i, m)]$ is given by $[(-i, m)]$.

Exercise 194. This exercise is concerned with checking that this definition of addition makes sense and fits with the usual one from $\mathbb{Q}$.

(a) Show that for (i, m), (i', m'), (j, n) and (j', n') in $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$ we have that $(i, m) \sim (i', m')$ and $(j, n) \sim (j', n')$ implies

$$(i, m) + (j, n) \sim (i', m') + (j', n').$$

This means that our definition of addition works properly on equivalence classes with respect to $\sim$.

(b) Show that $[(0, 1)]$ is the unit for this addition.

(c) Show that for $i, j, m, n \in \mathbb{Z}$ with $m, n \neq 0$ we have

$$g(\frac{i}{m}) + g(\frac{j}{n}) = g(\frac{i}{m} + \frac{j}{n}).$$

(d) Show that for $(i, m), (j, n) \in \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$ we have

$$f[(i, m)] + f[(j, n)] = f([(i, m)] + [(j, n)]).$$

Hence we may use $(\mathbb{Z}\times(\mathbb{Z}\setminus\{0\}))/_\sim$ as a formal definition of the rational numbers (with addition and multiplication)[26], and that is what mathematicians do:. Instead of assuming that m/n has a predefined meaning, they use (m, n) to express the same number, being aware of the fact that an equivalence relation is required since there is more than one way of representing a given number. A mathematician would say that what we have defined above is a *field*.

Optional Exercise 34. Carry out the last part of Optional Exercise 33 for

[26] Since subtraction and division are defined via inverses for addition and multiplication, respectively, these are also included.

multiplication and addition on $(\mathbb{Z} \times (\mathbb{Z} \setminus \{0\}))/_\sim$ as defined in this section.

Definition 63: rationals

The field $\mathbb{Q}$ of rational numbers is (formally) defined as the quotient of $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\}$ with respect to the equivalence relation given above, and with addition and multiplication operations also given above.

Further examples

Polynomial functions (see also Section 0.3.5) often come up as describing the complexity of various algorithms or programs. These functions exist over all our sets of numbers. Assume we have fixed a set of numbers N.

The formal definition of a polynomial is that of a function[27]from N to N for which we can find $n \in \mathbb{N}$ and $a_0, a_1, \ldots, a_n$ in N such that the function is given by

$$\sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

For our purposes we *exclude* the polynomial all of whose coefficients are 0. It is possible to extend the ideas that follow below to this polynomial, but it creates quite a lot of exceptions which make this account more complicated, and which distract from the ideas presented below.

The a_i are the **coefficients**. We define the **degree** of the polynomial to be the largest n for which $a_n \neq 0$, and write

$$\deg\left(\sum_{i=0}^{n} a_i x^i\right)$$

for this number. The corresponding coefficient a_n is called the *leading coefficient of the polynomial* in question.

Example 7.53. We calculate some degrees.

$$\begin{aligned} \deg\left(3x^4 + 2x^2 - 5\right) &= 4 \\ \deg\left(0x^5 + 3x^3 - 2x^1\right) &= 3 \\ \deg\left(0x^5 + 0x^2 - 4\right) &= 0. \end{aligned}$$

The leading coefficients of these polynomials are 3, 3, and -4 respectively.

When a coefficient is 0 it is customary not to write the corresponding part of the term, that is, we omit expressions of the form $0x^i$.

We can add such polynomial functions, and we can multiply them, in an obvious way. We first look at an example for each, and then give a general definition.

[27] A more formal mathematical definition of polynomials works without considering them as functions, but merely as formal expressions involving coefficients and powers of some variable, say x. The definitions of the addition and multiplication operations for polynomials given below still work in that case.

Example 7.54. We look at how to add polynomial to another. In the example below we have a polynomial of degree 3 which we want to add to one of degree 2, for example,

$$\begin{array}{ccccc}
 & x^3 + & & 2x + & 1 \\
+ & & 3x^2 + & 5x + & 10 \\
= & x^3 + & 3x^2 + & 7x + & 11.
\end{array}$$

In general we would like to write

$$\sum_{i=0}^{m} a_i x^i + \sum_{i=0}^{n} b_i x^i = \sum_{i=0}^{\max\{m,n\}} (a_i + b_i) x^i,$$

but it may be the case that some of the coefficients are not defined. If we go back to the concrete example above we have

$$\begin{array}{lllll}
a_3 = 1 & a_2 = 0 & a_1 = 2 & a_0 = 1 & \text{and} \\
 & b_2 = 3 & b_1 = 5 & b_0 = 10. &
\end{array}$$

Here the coefficient b_3 is not defined, so the convention is to assume that such coefficients are equal to 0 to make the formal definition work.

Multiplication is slightly more complicated.

Example 7.55. Again we start with an example.

$$\begin{aligned}
&(x^3 + 2x + 1)(3x^2 + 5x + 10) \\
&= (1 \cdot 3)x^5 + (1 \cdot 5)x^4 + (1 \cdot 10 + 2 \cdot 3)x^3 + (2 \cdot 5 + 1 \cdot 3)x^2 \\
&\qquad + (2 \cdot 10 + 1 \cdot 5)x + 1 \cdot 10 \\
&= 3x^5 + 5x^4 + 16x^3 + 13x^2 + 25x + 10.
\end{aligned}$$

In general, the definition is

$$\left(\sum_{i=0}^{m} a_i x^i\right)\left(\sum_{i=0}^{n} b_i x^i\right) = \sum_{i=0}^{m+n}\left(\sum_{j=0}^{i} a_{i-j} b_j\right) x^i,$$

which multiplied out looks something like

$$\begin{aligned}
a_0 b_0 + (a_0 b_1 + a_1 b_0)x + (a_2 b_0 + a_1 b_1 + a_0 b_2)x^2 \\
+ \cdots + (a_{m-1} b_n + a_m b_{n-1})x^{m+n-1} + a_m b_n x^{m+n}.
\end{aligned}$$

These operations on polynomials are associative and commutative. The unit for addition is the constant 0 polynomial, all of whose coefficients are 0, while the unit for multiplication is the constant 1 polynomial, where $a_0 = 1$ and all other coefficients are 0.

When we consider the complexity of an algorithm we often are only interested in the degree of the polynomial, that is, the highest power of x that occurs in the term.

We define an equivalence relation on polynomial functions, whereby

$$\sum_{i=0}^{m} a_i x^i \sim \sum_{i=0}^{n} b_i x^i \qquad \text{if and only if} \qquad \deg\left(\sum_{i=0}^{m} a_i x^i\right) = \deg\left(\sum_{i=0}^{n} b_i x^i\right).$$

In other words we consider two polynomial functions equivalent if and only if they have the same degree, that is if the largest n for which x^n has a non-zero coefficient, is the same for both. This means that two polynomials are in the same equivalence class if and only if they define the same 'big O' class of functions, compare Example 7.35. You are asked to think about some aspects of this idea in Exercise 197.

Now when we want to determine the complexity of an algorithm or program, and we know this is a polynomial function, we can concentrate on determining the degree. For this we need to be able to calculate with this equivalence relation, and the following exercise invites you to work out how this works. You will study these ideas in more generality in COMP11212 and COMP26120.

Exercise 195. For the following polynomials, give the degrees and leading coefficients, and calculate their sum and product.

- $2x^5 + x^3 - 1$ and
- $x + 1$.

What are the degrees and coefficients of the two resulting polynomials? How do they arise from the degrees and leading coefficients of the two polynomials you started with?

EExercise 196. Assume you are trying to implement polynomials with coefficients from $\mathbb{N}$ in Java by using the List class from Section 6.1, see Code Example 6.1. You want to think of the list as giving you the coefficient of the polynomial, so that

$$[3, 5, 2]$$

encodes the polynomial

$$3 + 5x + 2x^2.$$

The empty list (or **null** reference) does not correspond to a polynomial via this encoding. We are going to use it as another name for the polynomial of degree 0 whose (only) coefficient is 0, that is, it encodes the same polynomial as the list $[0]$.

Give definitions for the following operations, either by recursive definition for elements of $\mathsf{Lists}_{\mathbb{Z}}$ or by giving a recursive Java method for the class List. You have to stick to one of these for all the parts.

(a) Addition of polynomials.

(b) Multiplication of polynomials.

This is quite complicated. You may either develop your own way of doing this, or you may work according to the following idea.

When we multiply a given polynomial, say

$$2 + x + 3x^2$$

with another polynomial, say p, we have

$$\begin{align}(2+x+3x^2)\cdot p &= 2\cdot p \tag{7.1}\\ &+x\cdot p \tag{7.2}\\ &+3x^2\cdot p. \notag\end{align}$$

Hence in order to define multiplication of polynomials it is sufficient to use addition provided we also define a method that takes a polynomial and multiplies it by a polynomial of the form

$$mx^n,$$

where m and n are elements of $\mathbb{N}$.

- For multiplying a given polynomial by a polynomial of the form x^n you may find it convenient to have an operation that takes as input an integer n and returns a list which consists of n many 0s.
- It is useful to have an operation that takes a polynomial and multiplies is by a natural number (by multiplying all the coefficients by that natural number).
- Using the previous two parts one may define an operation that multiplies a given polynomial by a polynomial of the form

  $$mx^n,$$

  where $m, n \in \mathbb{N}$.
- Using the previous part and the addition operation for polynomials you should now be able to define multiplication of polynomials.

You may use previously defined operations for lists at any point.

EExercise 197. Let $P_{\mathbb{N}}$ be the set of polynomials with coefficients in $\mathbb{N}$.

(a) Above there is the definition of an equivalence relation $\sim$ on the set of polynomials which relates two polynomials if they have the same degree. Find a way of representing these so that you can describe the set $P_{\mathbb{N}}/_{\sim}$. Which polynomials are in $[x^2]$?

(b) Define an addition operation $\oplus$ on $P_{\mathbb{N}}/_{\sim}$. Ensure that for polynomials p and q you have $[p] \oplus [q] = [p+q]$—in other words, the equivalence class we obtain when adding the class of p to the class of q is the class of $p+q$.

(c) Define a multiplication $\circledast$ operation on $P_{\mathbb{N}}/_{\sim}$. Ensure that for polynomial functions p and q you have $[p] \circledast [q] = [pq]$.

Hint: In Exercise 186 you are asked to define addition and multiplication operations on equivalence classes when calculating modulo some number. You can think of this exercise as asking you to define addition and multiplication operations on equivalence classes of polynomials given by their degrees.

If you are struggling with this then start by carrying out Exercise 195. To get the marks for preparing the exercise that's a requirement if you cannot do anything else.

Finite state automata have an interesting equivalence relation on them, given by bisimulations.

Exercise 198. Argue that the following relation is an equivalence relation: The relation on finite state automata, where

$$A \sim A'$$

if and only if there is a bisimulation between them. There's no need to give a formal proof, just explain in English why you think the given relation has the required properties.

When programming we sometimes want to have a notion that two objects are equivalent for some equivalence relation. For example in Java the standard class `java.lang.Object` contains an instance method `equals()` which is designed to model the notion that two objects may be equivalent. See *Java: Just in Time* for more details in Section 20.4. This can only work properly if the relation in question is an equivalence relation.

7.4 Partial orders

Many structures we consider carry an *order* on them which allows us to *compare their elements*. There is a standard way of axiomatizing those properties that make a relation into an order in a way that agrees with our intuitions.

7.4.1 Posets

When comparing two elements s and s' we write

$$s \leq s'$$

to state that s is less than, or equal to, s', and

$$s' \geq s$$

to state that s' is greater than or equal to s. We consider the two equivalent. We further write

$$s < s'$$

if and only if

$$s \leq s' \qquad \text{and} \qquad s \neq s'.$$

We are using $\leq$ to give the infix notation for a *binary relation on* a set S. In this section we study the properties a binary relation needs to have in order to be something that behaves as we expect a comparison to behave.

First of all we expect each element s of S to be less than or equal to itself, so we expect

$$s \leq s$$

to hold for all $s \in S$. This means that we expect $\leq$ to be a reflexive relation.

Secondly we expect to be able to combine comparisons, that is if

$$s \leq s' \qquad\qquad \text{and} \qquad\qquad s' \leq s''$$

for elements s, s', s'' from S then we expect to be able to conclude that

$$s \leq s''.$$

In other words, we expect $\leq$ to be transitive.

These two properties together are sufficient to ensure that $\leq$ has most of the properties we expect from a comparison (see Optional Exercise 35), but it is customary to demand more.

When we know that

$$s \leq s' \qquad\qquad \text{and} \qquad\qquad s' \leq s$$

hold for elements s and s' of S we want to be able to conclude that

$$s = s'.$$

Definition 64: antisymmetric

A binary relation R on a set S is **anti-symmetric** if for all elements s and s' from S we have that

$$(s, s') \in R \text{ and } (s', s) \in R \qquad \text{implies} \qquad s = s'.$$

We cannot describe this property without having a binary equality predicate, say E. If we use that then the first order formula expressing anti-symmetry is

$$\forall x. \forall y. \left((R(x, y) \wedge R(y, x)) \rightarrow E(x, y) \right).$$

We may describe this property by noting that it says that

$$R \cap R^{\mathsf{op}} \subseteq I_S.$$

Anti-symmetry means that two elements cannot be less than or equal to each other unless they are equal. This property is the opposite of demanding symmetry of a relation, see Exercise 199, which explains the name.

In other words the only way we could have a two-sided arrow in the picture of an anti-symmetric relation is for that arrow to be a loop.[28]

Exercise 199. Show that a symmetric reflexive relation on a set S is anti-symmetric if and only if it is the identity on S.

Note that unlike for the other properties of binary relations studied in these notes, there is no such thing as an 'anti-symmetric closure' of a relation. For the other properties it is possible to *add* elements to the relation to obtain the desired property. But to make a relation anti-symmetric we would have to remove one of (s, s') or (s', s) if both are elements of the relation, and there's no sensible way to decide which one to take away. One could remove both, but then one constructs something that is a long way removed from the starting relation.

[28] But we don't draw loops with arrows at both ends!

Exercise 200. Show that the transitive closure of the reflexive closure of an anti-symmetric relation is not necessarily anti-symmetric. Hence we cannot generate partial orders from relations in the way we can do with equivalence relations. *Hint: The smallest counterexample is a binary relation on a set with three elements.*

Now we have all the properties required to define a partial order.

Definition 65: partial order

A **partial order** on a set S is a binary relation which is reflexive, anti-symmetric and transitive. A set together with a partial order is known as a **partially ordered set**, or in short, a **poset**.

We often write $(S, \leq)$ for the poset consisting of the set S with the partial order $\leq$.

Example 7.56. The standard definition for $\leq$ on sets of numbers between $\mathbb{N}$ and $\mathbb{R}$ are all partial orders.[29]

All these orders on sets of numbers satisfy an extra condition, see Definition 66, and you may find that you are assuming properties because you are guided by the examples you already know. Nonetheless we use the symbol $\leq$ for all partial orders in this section, so be careful not to make assumptions about such relations you make.

In order to check that a given relation on a small finite set is a partial order we may draw it as a directed graph, and then check reflexivity and transitivity as before. To check anti-symmetry we just have to make sure that there's no connection with arrow tips on both ends. To put it differently, we make use of the transitivity of the relation when interpreting the diagram.

It is customary, however, to draw a partial order on a small finite set as an undirected graph that is *oriented* on the page. Since a partial order is always reflexive one usually does not draw this part of the relation. Since a partial order is typically not symmetric (indeed, it has a property which is in a way the opposite, see Exercise 199), you might think that one should draw a directed graph, but instead the convention is to use the orientation of the paper to provide direction: Whenever two elements are connected by a line then the one that sits lower on the page is considered to be less than or equal to the other.[30] Furthermore, in order to avoid drawing superfluous lines, the assumption is that if a is connected to b which is higher up on the page, and b is connected to c, which is higher up again on the page, then the relation connects a and c even if no connection is drawn between the two.

The resulting graph is known as the *Hasse diagram* of the given partial order.

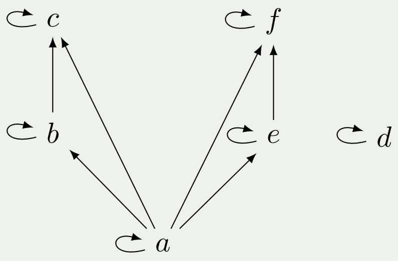
[29] Moreover, the operations on these sets of numbers, namely multiplication and addition, have specific properties which mean they are compatible with that order.

[30] It should go without saying that this means that we have to draw our Hasse diagrams in such a way that two elements on the same level are never connected.

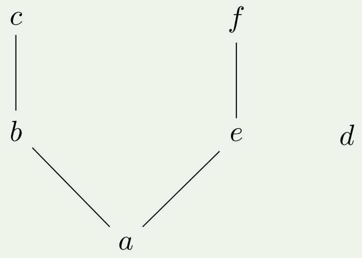
Example 7.57. Consider the partial order on $\{a, b, c, d, e, f\}$ which is the transitive reflexive closure of

$$\{(a, b), (b, c), (a, e), (e, f)\}$$

Previously we would have drawn this relation as follows.



But for relations known to be partial orders we draw the corresponding *Hasse diagram* as explained in the preceding paragraph, which for the given poset looks like this:

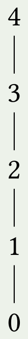


The picture tells us (because of the orientation on the page) that a is less than or equal to b which in turn is less than or equal to c, and that a is also less than or equal to e, which in turn is less than or equal to f. The element d is not comparable with any element other than itself.

Example 7.58. If we go back to the example from page 395, and form the transitive closure of the reflexive closure of

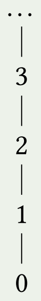
$$\{(0, 1), (1, 2), (2, 3), (3, 4)\}$$

then the Hasse diagram of this partial order is simply



Note that this is a total order (see Definition 66). In the Hasse diagram of a totally ordered set all the elements are lined up.

Example 7.59. One particular partial order should be familiar.



On the natural numbers $\mathbb{N}$ we have the 'usual order' we consider most of the time. We picture this as on the right, but because we are considering an infinite set we can only draw the Hasse diagram of a small part of the whole. The expression $m \leq n$ is one that you will have seen before, with this meaning. Note that this is a total order.

Example 7.60. Consider the following partial order on the natural numbers without 0, $\mathbb{N} \setminus \{0\}$. We set

$$m \leq n \qquad \text{if and only if} \qquad m \text{ divides } n.$$

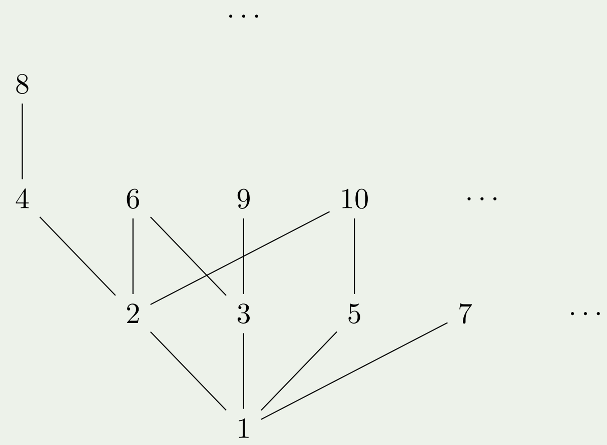
We show that this is a partial order by checking the properties required.[31]

- Clearly the relation defined above is reflexive since every natural number other than 0 divides itself.
- The relation is anti-symmetric since, given two natural non-zero numbers m and n,

  $$m \text{ divides } n \qquad \text{and} \qquad n \text{ divides } m$$

  imply $m = n$.
- The relation is transitive—you can find a proof in Exercise 23 (c).

Drawing a picture of (part of) this partial order, for the numbers from 1 to 10, gives a bit of an idea of what it looks like.



[31] Note that these properties are established above in examples or exercises but we repeat some of the arguments here for completeness' sake.

Note that in the preceding Examples, 7.59 and 7.60, we use the same symbol, $\leq$, for two *distinct* partial orders. Which partial order is intended in any given situation has to be clear from the context. If you find it confusing to use $\leq$ for an order other than the 'usual' one, you may instead use symbols such as $\sqsubseteq$ or $\preccurlyeq$ to make a visible distinction.

Example 7.61. Classes in Java are given a partial order by *inheritance*. You can define that one class is less than or equal to another if and only if it is a subclass.

Example 7.62. Many rankings come close to being partial orders. But typically it is possible for two entities to be considered equal in ranking, and so they often are not anti-symmetric. For example, ranking tennis, golf or chess players by the points they have accumulated has this property, as does ranking students by their overall averages. In the case where such a relation is anti-symmetric it is a total order, compare Definition 66. See Optional Exercise 35 for a look at relations that only lack anti-symmetry to be partial orders.

Example 7.63. We can define something that is almost a partial order on finite state machines by setting

$$A \leq A' \qquad \text{iff} \qquad \text{there is a simulation from } A \text{ to } A'.$$

This is reflexive since the identity relation is always a simulation. It is transitive since the relational composite of two simulations is a simulation (see Optional Exercise 30. It is not anti-symmetric since it is possible for two automata to have simulations going each way without the automata being identical. Optional Exercise 35 encourages you to think about relations that have these properties.

Example 7.64. Scheduling is typically performed based on information that Task A has to be performed before Task B. This information has to be

- anti-symmetric, since otherwise it is impossible to schedule all tasks;
- transitive since if Task A has to come before Task B, which in turn has to happen before Task C can be performed, then Task A has to happen before Task C.

Hence the reflexive closure of such a relation is a partial order. If we have to schedule tasks in a linear fashion, maybe because there is only one person (or robot or machine) to carry out all the tasks, then a valid schedule is one where

Task A

can only be scheduled when

$$\text{all tasks } B \qquad \text{with } B < A \qquad \text{have been scheduled.}$$

We identify a special property shared by the partial orders on sets of numbers, see Example 7.56. Note that the following notion is defined in Section 20.3 of *Java: Just in Time*.

Definition 66: total order

A **total order** on a set S is a partial order $\leq$ with the property that for all s and s' in S we have

$$s \leq s' \qquad \text{or} \qquad s' \leq s.$$

In this situation we say that the poset $(S, \leq)$ is **totally ordered**.

Again we give the first order proposition for this condition, where we assume a binary predicate symbol R for the partial order.

$$\forall x. \forall y. \left(R(x, y) \vee R(y, x) \right)$$

In a total order every two elements are comparable, so we may think of them all as being lined up, the way we usually think of $\mathbb{N}$ or $\mathbb{R}$. When we do not have this extra property then you may find that (partial) orders don't quite behave in the way you expect.

Example 7.65. Examples of totally ordered sets are $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ with the usual order.

Example 7.66. The partially ordered set from Example 7.58 shows a total order.

CExercise 201. Which of the following relations are partial orders? Try to sketch a Hasse diagram for each[32], and determine whether it is total. Justify your answers.

(a) The relation on complex numbers where $a + bi$ is less than or equal to $a' + bi'$ if and only if $a \leq a'$ and $b \leq b'$.

(b) The relation on complex numbers where $a + bi$ is less than or equal to $a' + b'i$ if and only if $a \leq a'$.

(c) The relation on $\mathbb{N} \times \mathbb{N}$ where (m, n) is less than or equal to (m', n') if and only if $m + n \leq m' + n'$.

(d) The relation on $\mathbb{N} \times \mathbb{N}$ where (m, n) is less than or equal to (m', n') if and only if

$$m < m' \qquad \text{or} \qquad m = m' \text{ and } n \leq n'.$$

(e) The relation on functions from $\mathbb{N}$ to $\mathbb{N}$ where f is less than or equal to g if and only if (compare Section 5.1)

$$g \text{ dominates } f,$$

that is

$$\text{for all } n \in \mathbb{N} \text{ we have } fn \leq gn.$$

This is known as the *pointwise order* on $\mathsf{Fun}(\mathbb{N}, \mathbb{N})$ since we look at each input point.

(f) The subset relation on the powerset of a set X.

(g) The relation on strings over a set S where string l is less then or equal to string l' if and only if l is a *prefix* of l'—in other words, if the beginning of l' is l.

(h) Assume we have a poset $(S, \leq)$. Then we obtain an order on the set of strings over the set S, known as the *lexicographic order* by setting[33]

$$\begin{array}{rll} & s_1s_2\cdots s_m \preccurlyeq s'_1s'_s\dots s'_n & \text{if and only if} \\ & \text{for } i \text{ smallest number in } \mathbb{N} \text{ with } s_i \neq s'_i & \\ \text{we have} & s_i < s'_i \text{ or } s_i \text{ not defined.} & \end{array}$$

Show that the relation defined in this way is indeed reflexive, anti-symmetric and transitive.

(i) The relation on the set of first year students in the School where student A is related to student B if and only if student B is taller than student A.

(j) The relation on the set of first year students in the School where student A is related to student B if and only if the registration number of student A is less than or equal to that of student B.

(k) The relation on all valid Java programs where programme P is less than or equal to program Q if and only if as a string, in the lexicographic order (also known as the 'dictionary order') program P comes before program Q.

(l) The following relation on the set $\{a, b, c, d, e\}$:

$$\{(a, a), (b, b), (c, c), (d, d), (e, e)\}.$$

(m) The reflexive closure of the following relation on $\{a, b, c, d, e\}$:

$$\{(a, b), (c, d)\}.$$

(n) The reflexive transitive closure of the following relation on the same set.

$$\{(a, b), (b, c), (c, d), (d, e)\}.$$

(o) The reflexive transitive closure of the following relation on the same set.

$$\{(a, b), (b, a), (b, c)\}.$$

(p) The subset relation on the powerset of the set $\{0, 1, 2\}$.

In parts (a) to (e) you have an existing relation $\leq$ on a set of numbers ($\mathbb{N}$ or $\mathbb{R}$), and a new relation is defined based on that. You may find it less confusing to use a new symbol for the new relation, maybe $\sqsubseteq$ or $\preccurlyeq$ as I do in part (h).

Example 7.67. We show how to define a partial order on $\mathsf{FBTrees}_S$, where S is an arbitrary set. We would like to define[34]

Base cases $\leq$. For all $s \in S$, we have $\texttt{tree}\, s \leq \texttt{tree}\, s$.

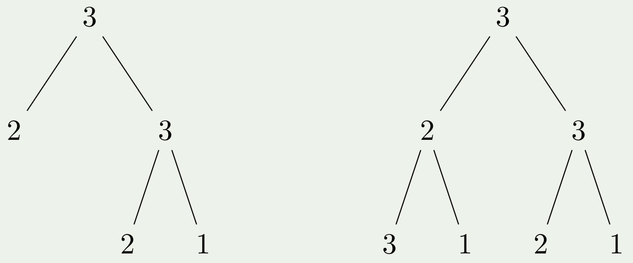
For all $s \in S$, t, $t' \in \mathsf{FBTrees}_S$, $\qquad \texttt{tree}\, s \leq \texttt{tree}_s(t, t')$.

Step case $\leq$. If $t \leq t''$ and $t' \leq t'''$ then $\texttt{tree}_s(t, t') \leq \texttt{tree}_s(t'', t'')$.

This relates two trees if and only if the second arises from the first by repeatedly extending the tree, that is

- pick a leaf with label s (that is a subtree of the form $\texttt{tree}\, s$) and
- replace that leaf by a tree whose root label is s, that is, a tree of the form $\texttt{tree}_s(t, t')$.

So, for example, we have that the tree on the left is less than or equal to the one on the right.



Alternatively we can think of the relation $\leq$ as relating a tree t to a tree t' if and only if we can obtain the first tree from the second tree by cutting off some of the branches.

We can show that this is a partial order. For reflexivity we have an inductive proof.

Base cases $\leq$. For $s \in S$ we know that $\texttt{tree}\, s \leq \texttt{tree}\, s$ by the first base case of $\leq$.

Ind hyp. We know that $t \leq t$ and $t' \leq t'$ for some full binary trees t and t' over S.

Step case $\leq$. From the induction hypothesis we may use the step case of $\leq$ to deduce that for every $s \in S$ we have

$$\texttt{tree}_s(t, t') \leq \texttt{tree}_s(t, t').$$

We can show anti-symmetry by a second induction proof.

[32] For infinite or large sets just try to get a general idea what such a diagram might look like and draw a part of it.

[33] Any dictionary gives you the words in this order. Also, the Linux list function presents the contents of some directory in this order.

Base cases $\leq$. For $t, t' \in \mathsf{FBTrees}_S$ if $t \leq t'$ by one of the base cases of the definition then it must be the case that we can find $s \in S$ such that $t = \texttt{tree}\, s$.

In order for $t' \leq t$ to also hold it must be the case, since we know that $t = \texttt{tree}\, s$, that this also occurs by the first base case of the definition of $\leq$, and so we must have that there is $s' \in S$ with $t' = \texttt{tree}\, s'$, but But if they are related by the base case of the definition of $\leq$ then it must also be the case that $s = s'$, and so

$$t = \texttt{tree}\, s = \texttt{tree}\, s' = t'.$$

Ind hyp. If $t \leq t'$ and $t' \leq t$ then $t = t'$, and if $t'' \leq t'''$ and $t''' \leq t'$ then $t'' = t'''$.

Step case $\leq$. The only remaining case is that

$$\texttt{tree}_s(t, t') \leq \texttt{tree}_s(t'', t''') \qquad \text{and} \qquad \texttt{tree}_s(t'', t''') \leq \texttt{tree}_s(t, t')$$

both by the step case, and so

$$t \leq t'' \text{ and } t' \leq t''' \qquad \text{and} \qquad t'' \leq t \text{ and } t''' \leq t'$$

and by the induction hypothesis

$$t = t'' \qquad \text{and} \qquad t' = t'''.$$

Note that this is a more complicated induction proof than others contained in these notes. This is required to ensure that all possible cases are covered.
The proof of transitivity is a similar induction proof, see the following exercise.

Exercise 202. This exercise studies in more detail the partial order on trees described in the preceding example.

(a) Rewrite the recursive definition of this partial order to define a set

$$R_{\leq} \subseteq \mathsf{FBTrees}_S \times \mathsf{FBTrees}_S.$$

(b) Show by induction that for

$$(t, t') \in R_{\leq}$$

we have that the roots of the two trees have the same label.

(c) Show that we have for $t, t' \in \mathsf{FBTrees}_S$ and the partial order $\leq$ from Example 7.67 that

$$t \leq t' \qquad \text{if and only if} \qquad (t, t') \in R_{\leq}.$$

(d) Show that $\leq$ as defined in Example 7.67 is transitive.

[34] Note that there are many base cases here, one for each tree!

(e) Write a program that takes two objects of class BTree and returns true if and only if the first is less than or equal to the second in this order.

Example 7.68. We can define a partial order on the set Lists_S of lists over a set S, where one list is less than or equal to another if and only if the second list is an extension of the first list, that is, it arises from the first list by adding more elements (from the left as usual). Note that while this has many similarities to the previous example there are subtle differences.
So, for example, if $S = \mathbb{N}$ we have

$$[4, 3, 2, 1] \leq [7, 9, 3, 4, 4, 3, 2, 1],$$

because we have that

$$[7, 9, 3, 4, 4, 3, 2, 1]$$

may be obtained from

$$[4, 3, 2, 1]$$

by adding the elements

$$4, \ 3, \ 9, \ \text{and } 7$$

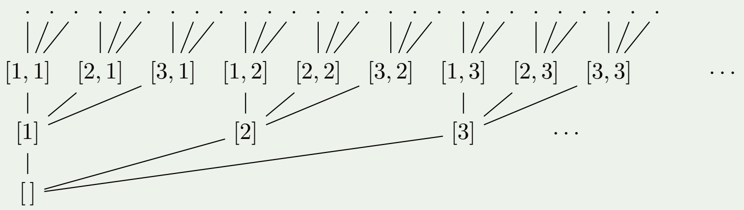
in that order. On the other hand, the list

$$[4, 3, 2]$$

is not less than or equal to

$$[4, 3, 2, 1]$$

because the two lists don't start out in the same way.
We can begin to sketch a Hasse diagram for this poset.



We can give a recursive definition for the set of all pairs that belong to this relation $R_{\leq}$ on Lists_S. In other words, we recursively define

$$R_{\leq} \subseteq \mathsf{Lists}_S \times \mathsf{Lists}_S$$

similar to the way we defined the subset of $\mathbb{N}$ that consists of all even numbers (Example 6.43), or of all powers of 2 (Example 6.44).

Base cases $R_{\leq}$. $\quad l \in \mathsf{Lists}_S$ implies $(l, l) \in R_{\leq}$.

Step case $R_{\leq}$. $\quad (l, l') \in R_{\leq}$, $s \in S$ implies $(l, s : l') \in R_{\leq}$.

If we are allowed to apply the reflexive transitive closure operation afterwards we can get away with a simpler definition of a relation, say $\hat{R}$:

- For $l \in \mathsf{Lists}_S$ and $s \in S$ we have $(l, s : l) \in \hat{R}$.

The relation R is the reflexive transitive closure of the relation $\hat{R}$.
Alternatively we can write a method which takes as input two objects of class List and returns true if and only if the first list is less than or equal to the second list. (You may want to compare this to Example 163.) Here we use the reverse method from Code Example 6.6.

```
public static boolean lessthan (List l1, List l2)
{
    if (l1 == null)
        return true;
    else {
        if (l2 == null)
            return false;
        else {
            List r1 = reverse (l1);
            List r2 = reverse (l2);
            if (r1.value == r2.value)
                return lessthan (l1.next, l2.next);
            else
                return false;
        }
    }
}
```

If you wonder why we reverse the two lists before comparing them, have a go at trying to write a method which does not require this step.
In some ways it is easier to recursively define the relation than to define a method to check whether it holds. In other words, defining an algorithm that checks whether two lists are in the relation is harder than defining the relation. Indeed, finding *decision procedures* (which you encounter in COMP11212) can be difficult even for properties that can be described in a simple manner. [35]

Exercise 203. Show the following for the relation $R_{\leq}$ defined in the preceding example:

(a) The relation $R_{\leq}$ is reflexive.

(b) If $(l, l') \in R_{\leq}$ then $\operatorname{len} l \leq \operatorname{len} l'$.

(c) If $(l, l') \in R_{\leq}$ and $\operatorname{len} l = \operatorname{len} l'$ then $l = l'$.

(d) The given relation is anti-symmetric.

(e) Show that if we have l and l' in Lists_S, and $n \in \mathbb{N}$, and s_1, s_2, $\ldots s_n$ in S,

[35] However, one can take that recursive definition of the relation and turn it into the decision procedure given following a specific set of rules! This idea takes us beyond the scope of this course unit.

with

$$l' = s_n : (s_{n-1} : (\dots(s_1 : l)))$$

then

$$(l, l') \in R_{\leq}.$$

(f) Show the converse of the previous part, that is, given $(l, l') \in R_{\leq}$ then we can find elements in S to satisfy the given condition.

(g) Show that the given relation is transitive.

For any given part you may need to use a previous part to give the desired proof.

Note that when programming, we sometimes come across ways of comparing elements which are *not* partial orders. For example, the CompareTo method for many classes in Java typically implements a comparison that is *not* anti-symmetric. It is an instance of a formal notion explored in the following (optional) exercise.

Optional Exercise 35. A *pre-order* $\preceq$ is a binary relation on a set S which is reflexive and transitive. For such a pre-order it makes sense to define a relation $\approx$ by, for s and s' in S,

$$s \approx s' \qquad \text{if and only if} \qquad s \preceq s' \text{ and } s' \preceq s.$$

(a) Can you think of an example of a pre-order? Having an example at hand my help you with the following parts.

(b) Show that $\approx$ is an equivalence relation.

(c) Use $\preceq$ to define a partial order on $S/_{\approx}$ and show that your relation is indeed a partial order. Conclude that for every pre-order there is a corresponding partial order on a related set.

(d) The CompareTo() method in Java for many classes is concerned with a pre-order, rather than a partial one. If you want to think more about pre-orders you could verify this claim, and think about why the method was designed in this way.

(e) Another pre-order is given by the relation between sets where S is related to T if and only if the size of S is at most as bit as that of T. Why is this not a partial order?

Exercise 204. Show that the set of all functions

$$\mathsf{Fun}(X, N),$$

where

- X is any set, and
- N is a set of numbers between $\mathbb{N}$ and $\mathbb{R}$

carries a partial order given by the 'is dominated by' relation from Section 5.1, which is known as the *pointwise order* (compare Exercise 201 (e)).

Exercise 205. Show that if $(P, \leq)$ is a poset then we obtain another partial order by the following means.

(a) By setting

$$p \sqsubseteq p' \qquad \text{if and only if} \qquad p' \leq p.$$

This is known as the *opposite (partial) order* of the poset, since $\sqsubseteq$ is $\leq^{\mathsf{op}}$. It means turning the poset upside-down.

(b) By taking a subset Q of P and defining, for q and q' in Q,

$$q \sqsubseteq q' \qquad \text{if and only if} \qquad q \leq q' \text{ as elements of } P.$$

This is known as the *subset (partial) order* on Q.

Optional Exercise 36. The total orders on $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ you are used to are very well behaved. For example, for numbers m, n, m' and n' we have

$$m \leq m' \text{ and } n \leq n' \qquad \text{implies} \qquad m + n \leq m' + n'.$$

(a) Prove this property for $\mathbb{N}$ using induction, and the recursive definitions of $\leq$ and $+$.

(b) Prove this property for $\mathbb{Z}$ and $\mathbb{Q}$ using their formal definitions from the previous section.

(c) Can you think of a similar property that holds for multiplication? *Hint: Have a look at Fact 7!*

(d) There is no standard partial order on the complex numbers $\mathbb{C}$. While it is possible to define a number of partial orders on this set (compare Exercise 201) most of these are not total,[36] nor will the operations be well-behaved with respect to these partial orders. Conduct some experiments by defining orders on $\mathbb{C}$ and exploring their properties.

7.4.2 Maximal, minimal, greatest, least

Having a partial order on a set allows us to find elements with distinctive properties.

Definition 67: maximal/minimal

An element p of a poset $(P, \leq)$ is a **maximal element of** P if and only if for all $p' \in P$,

$$\text{if } p \leq p' \qquad \text{then} \qquad p = p'.$$

An element p of a poset $(P, \leq)$ is a **minimal element of** P if and only if for

[36] What would a total order on the set $\mathbb{C}$ look like?

all $p' \in P$ we have

$$\text{if } p' \leq p \qquad \text{then} \qquad p' = p.$$

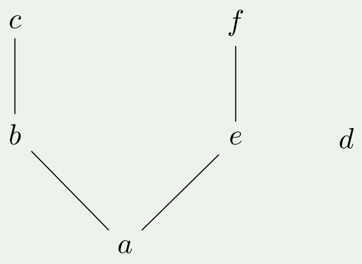
The propositional formula in the predicate calculus that expresses this property requires a constant p to be a maximal element for a partial order, a binary predicate symbol R for the partial order, and a binary relation E for equality. It looks as follows.

$$\forall x. \, (R(p, x) \to E(p, x))$$

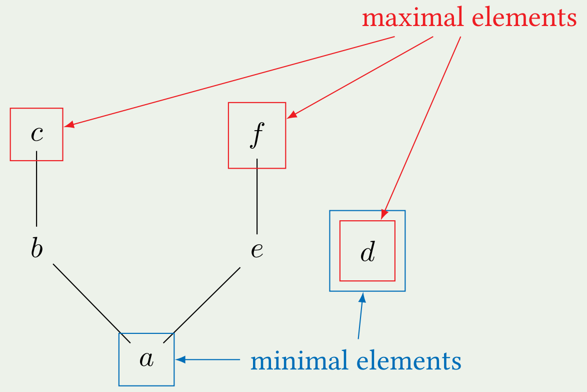
The way to understand this definition is to note that for a maximal element p of a poset there cannot be another element above it (because any element above has to be equal to p). Analogously, a minimal element cannot have another element below it.

Note that a poset can have more than one maximal or minimal element, see the following example.

Example 7.69. If we go back to our running example from above,

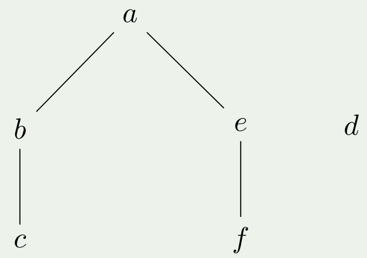


we can see that the maximal elements are c, f and d, and that there are two minimal elements, namely a and d.

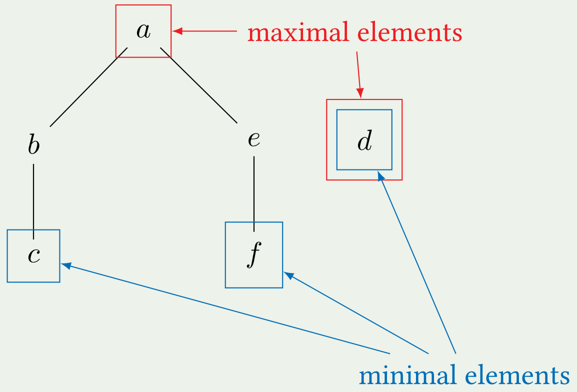


This example illustrates that a maximal element does not have to be above all the other elements (for example, f is maximal but not above c, b or d, and d is maximal but not above anything other than itself).

Example 7.70. By Exercise 205 we may turn our given poset upside-down to obtain another poset:



In this poset there are three minimal elements, c and f and d, and two maximal elements, a and d.



It is immediately clear from the definition that when we turn a poset upside-down then maximal elements become minimal, and vice versa.
We look at another concept which is more restricted than being minimal or maximal.

Definition 68: greatest/least

An element p is the[37] **greatest element of a poset** $(P, \leq)$ if and only if it is the case that

$$\text{for all } p' \in P \qquad \text{we have} \qquad p' \leq p.$$

The greatest element is usually called[38] $\top$.
An element p is the[37] **least element of a poset** $(P, \leq)$ if and only if it is the case that

$$\text{for all } p' \in P \qquad \text{we have} \qquad p \leq p'.$$

The least element is usually called[39] $\bot$.

Using the same convention as before the proposition corresponding to a parameter p describing a largest element for a partial order given by a binary predicate symbol R is

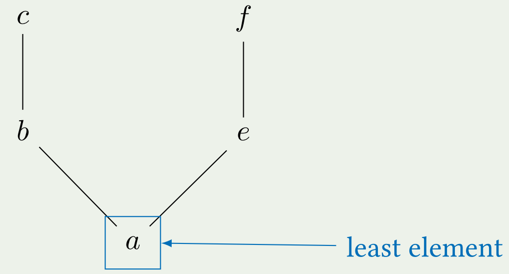
$$\forall x.\, R(x, p).$$

Example 7.71. Our original example poset has no greatest or least element. If we remove the element d from the set we obtain the following.

[37] The use of the definite article here is justified by the following exercise.

[38] Yet another usage of this symbol! It is shaped like a t, short for 'top', and that is the name of the symbol.

[39] Yet another usage of this symbol! It is shaped like the opposite of $\top$. It is known as 'bottom'.



This poset has no greatest element; a least element exists, namely a. Note that this is also a minimal element, and that there is only one minimal element. See the following exercises for this and other important properties of least and greatest elements.

Example 7.72. Consider the set $\mathbb{N} \setminus \{0\}$ where

$$m \leq n \qquad \text{if and only if} \qquad m \text{ divides } n,$$

compare Example 7.60.

Since 1 divides every number it is the least element. This means that it is the only minimal element.

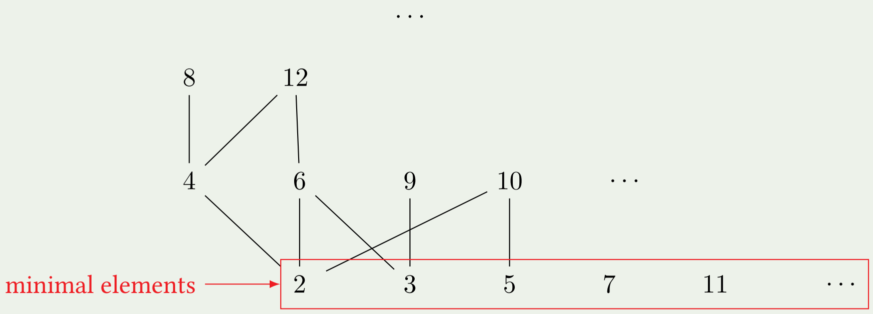
Since no number is divided by all natural numbers there is no largest element.

But there are no maximal elements either: For every element $n \in \mathbb{N} \setminus \{0\}$ we have that n divides $2n$, and so for any given $n \in \mathbb{N} \setminus \{0\}$ we can always find a larger element.

Example 7.73. We take the previous example but restrict to the set $\mathbb{N} \setminus \{0, 1\}$ where we again define

$$m \leq n \qquad \text{if and only if} \qquad m \text{ divides } n,$$

compare Example 7.60. Compared to that example we have removed the number 1 from consideration.



There is no number in $\mathbb{N} \setminus \{0, 1\}$ that divides all numbers, so there is no least element. There are a lot of minimal elements however: In order for an element n to be minimal we must have that $m \leq n$ implies $m = n$; in other words, we are are looking for numbers n with the property that for all $m \in \mathbb{N} \setminus \{0, 1\}$,

$$m \text{ divides } n \qquad \text{implies} \qquad m = n.$$

This is the case exactly when n is a prime number, and so all prime numbers are minimal.

Since no number is divided by all natural numbers there is no largest element. But there are no maximal elements either: As in the previous example we have that for every element $n \in \mathbb{N} \setminus \{0\}$ we have that n divides $2n$, and so we can always find a larger element.

Example 7.74. For the partial order on Lists_S from Example 7.68 we have a least element, namely the empty list $[\,]$. There are no maximal elements since given an arbitrary list we can always make it bigger by adding another element.

CExercise 206. For the following posets, try to draw a Hasse diagram and determine any maximal, minimal, least and greatest elements.

(a) The real numbers with the usual order.

(b) The natural numbers with the usual order.

(c) The negative integers with the usual order.

(d) For the partial order on $\mathsf{FBTrees}_S$ from Example 7.67.

(e) The powerset of a set X with subset inclusion as the order.

(f) The set $\mathsf{Fun}(\mathbb{N}, \mathbb{N})$ of functions from $\mathbb{N}$ to $\mathbb{N}$ with the pointwise order, given by:

$$f \le g \qquad \text{if and only if} \qquad \text{for all } n \in \mathbb{N} \text{ we have } fn \le gn.$$

(g) The following relation on $\{a, b, c, d, e\}$.

$$\{(a,a), (b,b), (c,c), (d,d), (e,e)\}.$$

(h) The reflexive closure of the following relation on $\{a, b, c, d, e\}$:

$$\{(a,b), (c,d)\}.$$

(i) The reflexive transitive closure of the following relation on that set.

$$\{(a,b), (a,c), (b,d), (b,e), (c,d), (c,e)$$

(j) The reflexive transitive closure of the following relation on that set.

$$\{(a,b), (b,c), (c,d), (d,e)\}.$$

(k) For your programme of study, and the set of compulsory COMP units on that programme (through years 1 to 3), the partial order which is the transitive reflexive closure of the 'is a prerequisite of' relation (ignoring corequisites).

(l) For Chapter 6, take all operations defined in that chapter (including exercises) for $\mathbb{N}$ and for lists, and use the transitive reflexive closure of the 'is required to define' relation (for example, the addition operation is required to define multiplication of natural numbers).

EExercise 207. Greatest and least elements have special properties.

(a) Show that there is at most one greatest, and at most one least, element in each poset.

(b) Show that if p is the greatest element of a poset then it is maximal. Similarly, show that if p is the least element of a poset then it is minimal.

(c) Show that if a poset has a greatest element then it has exactly one maximal element. Similarly show that if a poset has a least element then it has exactly one minimal element.

(d) Show that if $(P, \leq)$ is a totally ordered poset then every maximal element is a greatest element, and every minimal element is a least element. Conclude that a totally ordered set can have at most one maximal, and one minimal, element.

7.4.3 Upper and lower bounds

When looking at posets we often care about whether a collection of elements can be safely 'overestimated' (or 'underestimated') by using a single element. For example, when we are looking at the complexity of a problem we may be interested in the fact that it is 'as most as complicated as some function f'—that means that there is an algorithm with complexity function less than or equal to f. Or maybe we have a group of algorithms or programs we would like to discuss in one go, and we can state that the complexity function for each of these lies below some 'upper bound' given by some function f.

Definition 69: upper/lower bound

Let S be a subset of a poset $(P, \leq)$. We say that an element p of[40] P is an **upper bound for** S if and only if it is the case that

$$\text{for all } p' \in S \qquad \text{we have} \qquad p' \leq p.$$

We say that an element p of[41] P is a **lower bound for** S if and only if

$$\text{for all } p' \in S \qquad \text{we have} \qquad p \leq p'.$$

Again we give a first order proposition to describe this definition. Assume we have

- a binary predicate symbol R for the partial order,
- a unary predicate S that describes the elements of the set S, so $S(x)$ is interpreted as 1 if and only if x is in S,

[40] Note that p need not be an element of S.

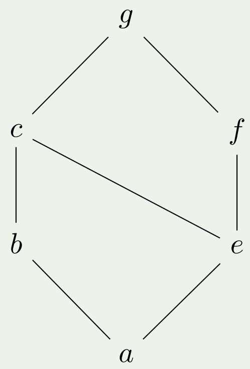
[41] Again note that p need not be an element of S.

then the proposition establishing p as an upper bound of S is

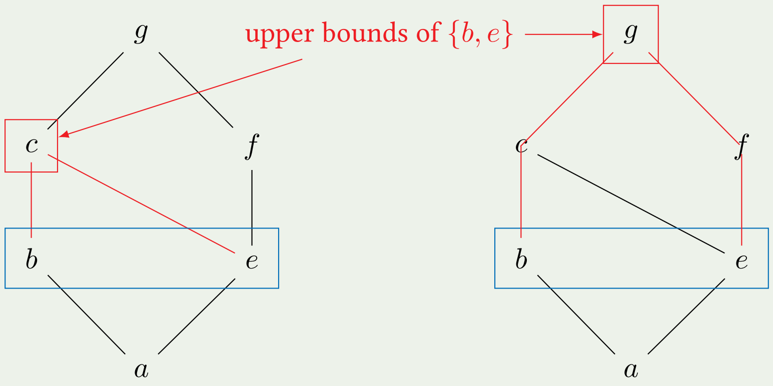
$$\forall y.\,(S(y) \to R(y,p)).$$

The example poset we have been using so far is too simple to contain many interesting upper or lower bounds. For the version of the example poset that appears in Example 7.71, the only interesting observation is that a is the only lower bound for the sets $\{c,f\}$, $\{b,f\}$, $\{c,e\}$ and $\{b,e\}$.

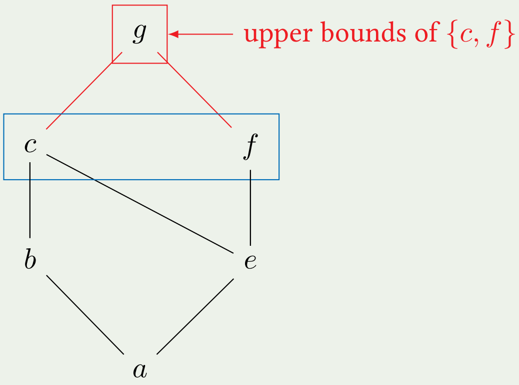
Example 7.75. Consider the poset drawn below.



What are the upper bounds for the set $\{b,e\}$? They are the elements which are above both, b and e.



We can see that the upper bounds are c and g.
What are the upper bounds of $\{c,f\}$?



The element g is the only upper bound of the sets $\{c, f\}$ (and also of $\{b, f\}$).

Example 7.76. In Java we can order classes by the subclass relation. For this partial order two classes have a common upper bound if and only if there is a class which is a superclass for both.

Exercise 208. For the poset from Example 7.75, find all lower bounds for the sets $\{b, e\}$, $\{g, b\}$, $\{g, e\}$, $\{c, f\}$, $\{b, f\}$.

Sometimes we can find a 'best' upper bound for a subset of a poset, that is, amongst all the upper bounds there may be a least one.

Definition 70: least upper/greatest lower bound

Let S be a subset of the poset $(P, \leq)$. We say that an element p of P is the[42] **least upper bound (or supremum) for** S if and only if

- p is an upper bound for S and
- if p' is an upper bound for S then $p \leq p'$.

The least upper bound of two elements p and p' is often written as $p \vee p'$.
We say that an element p of P is the[42] **greatest lower bound (or infimum) for** S if and only if

- p is a lower bound for S and
- if p' is a lower bound for S then $p' \leq p$.

The greatest lower bound of two elements p and p' is often written as $p \wedge p'$.

This statement is fairly complicated if we look at the number of quantifiers involved when we unravel the statements from above. In an appropriate formal system[43] for p to be the least upper bound of S we need

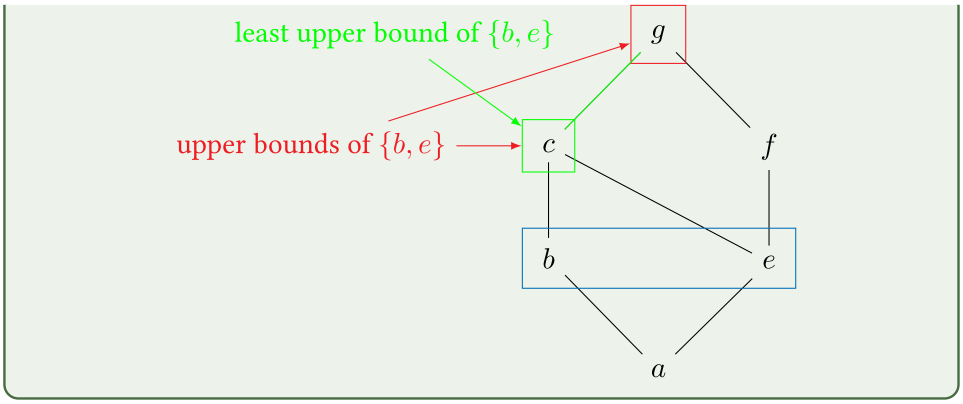
$$\begin{array}{ll} & \forall y.\, (S(y) \to R(y, p)) \\ \wedge & \qquad \forall y.\, ((\forall z. S(z) \to R(z, y)) \to R(p, y)). \end{array}$$

You can see that by building up the definitions slowly we have a final definition which looks simpler. But taking it apart reveals that the concept is quite complicated, as can be seen from looking at the corresponding proposition above.

Example 7.77. Going back to the example from above: The set $\{b, e\}$ has two upper bounds, namely c and g. This set has a least element, namely c, and so c is the least upper bound of $\{b, e\}$.

[42] The definite article is justified by Exercise 210.

[43] We need a one-placed predicate symbol for 'is an element of S' and a binary predicate symbol R for the partial order.



An alternative way of looking at least upper bonds is the following: We can form a new set from S, namely

$$\{p' \in P \mid p' \text{ upper bound for } S\},$$

and then

$$p \qquad \text{is a least upper bound for} \qquad S$$

if and only

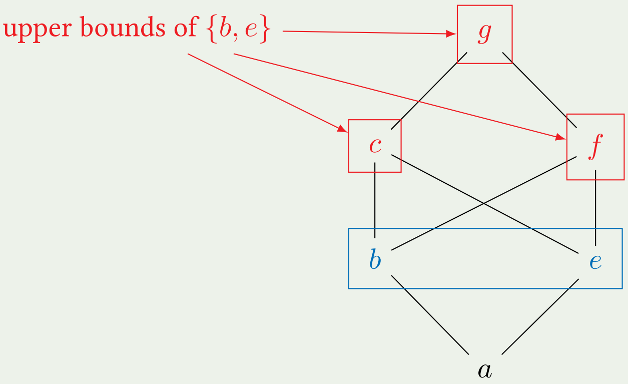
$$p \qquad \text{is the least element of} \qquad \{p' \in P \mid p' \text{ upper bound for } S\}.$$

In particular,

$$\{p' \in P \mid p' \text{ upper bound for } S\}$$

must have a least element for S to have a least upper bound.

Example 7.78. If, on the other hand, we change our underlying poset slightly to



then c, f and g are all upper bounds of $\{b, e\}$, but none of them is the least upper bound since the set

$$\{c, f, g\}$$

has no least element.

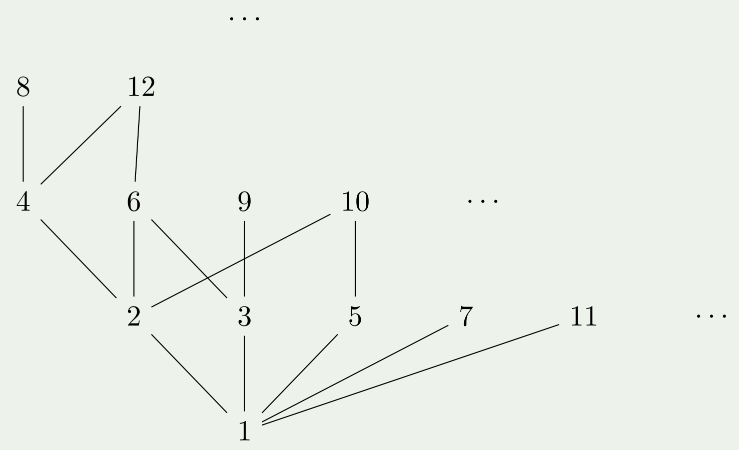
Note that the least upper bound of a set may be an element of that set. In the preceding Example 7.78 the least upper bound of $\{c, f, g\}$ is g. In the natural numbers with the usual order, the least upper bound of $\{2, 4, 8, 16\}$ is 16. Similar remarks apply to the greatest lower bound.

Exercise 209. Let $(P, \leq)$ be a partial order, and let S be a subset of P. Show that if S has a greatest element then it is the least upper bound of S.

Example 7.79. We return to another of our running examples. Consider once again the set $\mathbb{N} \setminus \{0\}$ with the partial order given by

$$m \leq n \qquad \text{if and only if} \qquad m \text{ divides } n,$$

compare Examples 7.60 and 7.72. We repeat the Hasse diagram.



If we have two numbers m and n then their lower bounds are the numbers which are below both of them, that is, all those numbers which divide both, m and n. For example, for

$$12 \qquad \text{and} \qquad 18$$

these numbers are

$$\{1, 2, 3, 6\}.$$

The set of common divisors of two numbers m and n always has a largest element for the order under consideration (and this is also the largest element when we consider the usual order on $\mathbb{N}$, namely the *greatest common divisor* of m and n, compare Example 6.42. Hence for every two elements there is a greatest lower bound, given by their greatest common divisor.

Similarly, given two numbers m and n, their upper bounds are those numbers which have both of them as divisors, that is, all the *common multiples* of the two. For example, for

$$6 \qquad \text{and} \qquad 4$$

these are all the numbers that are multiples of both,[44] 4 and 3.

$$\{12,\ 24,\ 36,\ \ldots\}$$

which means they are all the multiples of 12,

$$\{12n \in \mathbb{N} \mid n \in \mathbb{N}\}.$$

This set always has a smallest element, in this case 12. In general, the least upper bound of two numbers m and n is *their smallest common multiple.*

Example 7.80. If we return to the partial order on the set Lists_S of lists over a set S given in Exercise 7.68 we note the following:
Given any two lists, their greatest lower bound is the longest list they both have in common, so for

$$[4, 3, 2, 1] \qquad\qquad \text{and} \qquad\qquad [5, 4, 2, 1],$$

the greatest lower bound is

$$[2, 1].$$

Once those two elements have been added to the list the paths of getting to the two lists diverge. Note that there is always such a largest list (it might be the empty list), so for every pair of element a greatest lower bound exists. If you look at the Hasse diagram for this poset, which is drawn in Example 7.68 you can see that this looks like a tree. To find the greatest lower bound for two lists, you go down the branches of the tree which contain them until these branches come together.
On the other hand if we look at least upper bounds for two lists we find that they almost never exist. This is because two lists which are not related by the order do not have any upper bounds. For example, an upper bound for both

$$[2, 1] \qquad\qquad \text{and} \qquad\qquad [3, 1]$$

would have to be obtainable by adding elements to both these lists, and clearly this cannot happen. Hence the only case in which any upper bounds exist is in the case where

$$l \leq l',$$

in which case l' is the least upper bound for the two.

Exercise 210. Show that every subset of a poset P has at most one least upper, or greatest lower, bound in P.

So least upper (greatest lower) bounds of sets exist, or don't, but where they exist they are unique.

[44] You may want to convince yourself that all numbers that are multiples of both, 4 and 3, are also multiples of both, 6 and 4.

CExercise 211. This exercise is concerned with calculating greatest lower and least upper bounds. Note that all the settings described here also appear in Exercise 201 and that you may want to solve the corresponding part of that exercise first.

(a) For the natural numbers with the usual order, how would you describe the greatest lower, and least upper, bounds for sets of the form $\{m, n\}$? Can you extend this to all finite subsets of $\mathbb{N}$? Do you need finiteness? Try to extend your result to all subsets of $\mathbb{N}$ as far as possible.

(b) For the powerset of the set $\{0, 1, 2\}$ how does one calculate the greatest lower, and least upper, bound of a subset? *Hint: You may want to draw the Hasse diagram. What are the greatest lower and least upper bounds of* $\{\{0, 1\}, \{1, 2\}\}$

(c) Consider the powerset $\mathcal{P}X$ of a set X, with subset inclusion providing a partial order. Show that for every subset $\mathcal{S}$ of $\mathcal{P}X$ we have

- The least upper bound of $\mathcal{S}$ exists and is given by

$$\bigcup \mathcal{S} = \bigcup \{S \subseteq X \mid S \in \mathcal{S}\} = \{x \in X \mid \exists S \in \mathcal{S}. x \in S\}.$$

- The greatest lower bound $\mathcal{S}$ exists and is given by

$$\bigcap \mathcal{S} = \bigcap \{S \subseteq X \mid S \in \mathcal{S}\} = \{x \in X \mid \forall S \in \mathcal{S}. x \in S\}.$$

(d) Consider the set of strings made from symbols from a set S. Consider the partial order where a string is less than or equal to another if it is a prefix of the latter. What are the greatest lower, and least upper, bounds of sets of strings (where they exist)? *Hint: You have already seen a partial Hasse diagram for this set. Start from there by looking at the two-element subsets.*

(e) Consider the set $\mathsf{FBTrees}_S$ of full binary trees over a set S with the partial order from Example 7.67. When do the least upper, and greatest lower, bounds of two trees exist and what are they? For this part it is sufficient to describe your answer in English and to give an example.

Exercise 212. Consider the following relation for the set $\mathsf{FBTrees}_{\mathbb{N}}$ of full binary trees over the set of labels $\mathbb{N}$. Recall the function k_1 from Exercise 149 which maps a natural number to the number 1. Also recall the map function from the same exercise, which has the property that

$$(\text{map}\, k_1)t$$

replaces every label in the tree t by the number 1. Finally recall the partial order $\leq$ on $\mathsf{FBTrees}_S$ described in Example 7.67. We use this to define a new partial order on the same set.
For two trees t and t' we set

$$t \sqsubseteq t'$$

if and only if

$$(\text{map}\, k_1)t \leq (\text{map}\, k_1)t'.$$

(a) Try to describe in English under which circumstances we have that $t \sqsubseteq t'$.

(b) Modify the code for the lessthan method from Example 7.67 in such a way that it implements this new relation.

(c) Is this relation a partial order?

(d) For the set $S = \{1\}$, consider the partial order $\leq$ on $\mathsf{FBTrees}_S$ used above. Describe the greatest lower bound of two elements of this set in English.

(e) For the same poset as in the previous part describe the least upper bound of two elements in English.

EExercise 213. Consider the following relation on the set of partial functions from a set S to a set T:

$$\begin{aligned} f \preccurlyeq g \qquad & \text{if and only if} \\ & \qquad \operatorname{dom} f \subseteq \operatorname{dom} g \text{ and } s \in \operatorname{dom} f \text{ implies } fs = gs \end{aligned}$$

(a) Show that this relation is a partial order.

(b) What are the minimal elements of this poset? Is there a least element?

(c) What are the maximal elements of this poset? Is there a greatest element?

(d) Given two partial functions f and g from our set, under which circumstances is their greatest lower bound $f \wedge g$ defined? Can you describe this partial function?

(e) Given two partial functions f and g from our set, under which circumstances is their least upper bound $f \vee g$ defined? Can you describe this partial function?

The mathematical discipline of *order theory* is concerned with studying partially ordered sets and their properties.

Chapter 8

Applications to Computer Science

The eventual aim of this chapter is to collect examples and exercises from computer science where mathematical techniques are routinely employed, but without stating explicitly which areas of mathematics might be involved. Currently a number of suitable exercises are located in various chapters. While they serve there to motivate some concepts and techniques, from a didactic point of view it would be good if students gained some experience in solving such problems without too many hints regarding what techniques to employ.
Examples are:

- The use of precise language, and the predicate calculus, in particular first order logic, to describe specifications for a computer program (some examples appear below).
- To verify the correctness of programs and protocols using techniques from logic, see for example COMP31111.
- Programming with logical formulae in the language Prolog which is taught in COMP24412.
- Solving optimization problems with tools from logic in COMP21111.
- The use of the expected value of a random variable to calculate the complexity of a given algorithm.
- The use of probabilities and expected values to evaluate risks.
- The notion of one function growing at most as fast as another to talk about complexity classes of problems or algorithms—this material is covered in COMP11212, COMP26120 and COMP36111.
- The use of the notion of the size of a set to answer questions from computer science, for example: Are there functions from $\mathbb{N}$ to $\mathbb{N}$ which cannot be implemented using a computer?
- The use of recursion to solve particular problems. Various aspects of this idea are covered in COMP16212, COMP26120 and COMP36111.
- The use of recursively defined function to work out the complexity of a particular algorithm using recurrence relations.
- The use of induction as a technique to prove properties of many entities that appear in various course units.

- Using relations to express properties of programming constructs, or of elements of some other domain of discourse.
- The use of vectors and matrices in particular in the context of graphics, which appears in COMP27112.

For the moment only a small number of exercises appear in this chapter. They are all concerned with analysing a particular problem with a view to creating a precise specification for the desired solution.
Various exercises that appear in other chapters in these notes could be included in this chapter, but I did not want to postpone your tackling them until the end of term.

- **Chapter** 2. Various questions are concerned with the properties of Java operations, and with the properties of functions that assign user names.
- **Chapter** 4. Bayesian updating is a technique from Machine Learning and is covered in Exercises 103 to 105. Using expected values to calculate average complexities for algorithms is also a computer science topic, which is the topic of Exercise 122. We look a little bit at the reliability of hardware in Exercise 114.
- **Chapter** 6. All the exercises asking you to produce code are applications of the mathematical idea of recursion to programming. Proofs by induction that a given recursively defined function has certain properties can be transplanted to recursive programs, and are required if you want to show that your code behaves as expected/specified.
- **Chapter** 7. We look at algorithms that calculate the powers of some number in modular arithmetic in Exercise 187. There are also exercises asking you to produce code for some problems in this chapter.

Much of the time programmers do not make the effort to precisely define what they want their code to do, but under some circumstances this is a very useful tool, for example

- when code already written shows unexpected behaviour or
- when one is trying to define the behaviour of a compiler of a programming language[1] or
- for safety-critical applications, such as the control of a space or aircraft, when one would ideally have verification that the code implemented has particular properties.[2]

CExercise 214. Imagine you are a programmer tasked with writing a program that takes as its input an array of natural numbers, and returns a sorted version of that array. The aim of this exercise is to specify what exactly this means. For example, you would expect for the input array a

[1] You may be surprised by this, but a number of programming languages do not have precise definitions of their behaviour and different compilers (or interpreters) can behave differently for some programs.

[2] A unit that looks at this issue is COMP31111, *Verified Development.*

0	1	2	3	4	5	6	7
9	35	0	5	58	7	1	42

to produce the following output array, say b:

0	1	2	3	4	5	6	7
0	1	5	7	9	35	42	58

For the conditions (a) and (b) below, do the following:

- Write out as precise a description as possible using the English language.
- Write a formula in first order logic that captures the required property. You may assume the following:
  - The domain of interpretation is the set of natural numbers $\mathbb{N}$.
  - You have (unary) function symbols a and b to describe the input and output arrays respectively.
  - You have a parameter n that describes the final index of the input array (so the size of the input array is $n+1$, with indices from 0 to n).
  - You have a binary 'less than or equal' predicate L so that the interpretation of $L(x, y)$ is $Ix \leq Iy$, where I is the interpretation function.
  - You have a binary equality predicate E so that the interpretation of $E(x, y)$ is $Ix = Iy$, where again I is the interpretation function.

(a) The output array b is sorted.

(b) The array b contains exactly the numbers from array a. You may assume that all the entries in a are different. *Hint: Split the property into two parts.*

(c) Does your statement from part b) still work if a may contain duplicate entries? Check whether you can construct an example of arrays a and b which satisfy your condition, but where b has fewer elements than a.

(d) Can you see how to improve your previous so that it also works for arrays with duplicate entries? Can you transfer your condition to first order predicate logic? If not, could this be dealt with by expanding the system?

Exercise 215. Assume you are given an array of natural numbers. You are tasked with writing a program that finds the *median* of all these numbers, that is the number you would find at the halfway point of the array if it were sorted. You may think of this as the 'middle number' of all those present.[3]

(a) Write down the properties required to ensure that the number your program finds is indeed the desired median. You may assume that all the numbers in your array are different. *Hint: You may want to use the function* $|\cdot|$ *that maps a finite set to the number of elements of that set.*

(b) Now amend your properties so that it can also cope with the case where the same number may appear more than once in the array. *Hint: Consider the set you used in the previous part. Can you think of changing that to a different set that will give you the desired property?*

Exercise 216. Assume you are tasked with writing a program that constructs an individual timetable for each student (better than the one on the Student System) and which therefore needs to know about all the undergraduate students within the School. This means you will require some way of representing each student in your program.

(a) You have to define a function whose source is the set of all undergraduate students in the school, and whose target is something that your programme knows about. What would a suitable target for such a function be?

(b) Which properties should your function have?

(c) Define the function that you would use.

(d) Justify your choice of function by arguing that it satisfies the properties stipulated in Part (b).

When when we have a recursively defined procedure we can create a *recurrence relation* (compare Examples 6.47, 6.48 and Exercise 169) that describes its behaviour.

Example 8.1. Consider the function that sums all the elements from a list of numbers, given in Example 6.11.

Base case sum. $\quad \mathrm{sum}\,[\,] = 0.$[4]

Step case sum. $\quad \mathrm{sum}(s:\ l) = s + \mathrm{sum}\, l.$

We want to count the number of additions the function has to carry out for a list with n elements, which defines a function

$$f\colon \mathbb{N} \longrightarrow \mathbb{N}.$$

We can read off that

$$f0 = 0,$$

since in the base case, where the list has 0 elements, no additions are required. The step case shows us that

$$f(n+1) = 1 + fn.$$

Hence we have found a recurrence relation as studied in Section 6.4.5. Using the methods from that chapter we can show that the function we have defined is the assignment

$$n \longmapsto n,$$

that is, our function is the identity function on $\mathbb{N}$.

[3] The median income of all the employees of a company, for example, is the one with the property that half the employees earn as most as much, and half the employees earn at least as much.

Exercise 217. For the following algorithms, create recursive function definitions similar to those from Examples 6.47, 6.48 and Exercise 169. Is it possible to solve them in the same way as those examples? If not, why not?

(a) The function that calculates the factorial of a number from Example 6.41. Count the number of multiplications the function has to carry out for a given input.

(b) The function that reverses a list from Example 6.12. For n use the number of elements in the list and count the number of calls the function makes to itself.

(c) The binary search algorithm from Example 4.98. Set n to be the size of the array and count the number of array look-ups required.

[4] If you were wondering how to deal with the empty list then note that adding no elements at all is usually taken to describe the number 0. More generally, for an operation with unit e, applying the operation to 0 many elements should return e.

Produce another glossary for total. comment in/out as needed

Glossary[5]

σ-algebra 149

The set of events of a probability space. Contains the whole set of outcomes and is closed under the complement operation and forming unions of countable collections of sets.

absolute, $|\cdot|$ 18, 63, 66

Defined for various sets of numbers, here extended to complex numbers. Given a complex number $a + ib$ we have $|a + ib| = \sqrt{a^2 + b^2}$.

and 76

Connects two properties or statements, both of which are expected to hold.

anti-symmetric 407

A binary relation R on a set S is anti-symmetric if (s, s') and (s', s) both being in the relation implies $s = s'$.

argument 66

The argument of a complex number is the angle it encloses with the positive branch of the real axis.

associative 93

A binary operation is associative if and only if it gives the same result when applied two three inputs, no matter whether it is first applied to the first two, or first applied to the last two of these.

Bayes's Theorem 174

The equality which says that, given events A and B, the probability that B given A is the probability that A given B, multiplied by the probability of B and divided by that of A.

bijective 117

A function is bijective if and only if it is both, injective and surjective. A bijective function is called a bijection.

binary operation 42, 92

A function of the type $S \times S \longrightarrow S$, which takes two elements of a set S as input and produces another element of S.

[5] Note that page numbers for Chapters 1–4 will not match the printed notes for these chapters.

binary relation 57

A connection between a source set S and a target set T which is not necessarily a function. It is specified by the collection of all pairs of the form (s, t) in $S \times T$ that belong to it.

binary tree with labels from a set S 307

This is a tree where each node has a label from S and where each node has either 0, 1 or 2 children. Formally this is another recursively defined notion.

$\mathbb{C}$ 61

The complex numbers as a set with a number of operations.

coefficient 402

The coefficients of a polynomial are the numbers that appear as factors in front of a power of the variable.

commutative 96

A binary operation is commutative if and only if it gives the same result when its two inputs are swapped.

complement 32

The complement of a set S is always taken with respect to an underlying set, and it consists of those elements of the underlying set which do not belong to S.

composite of functions 45

The composite of two functions is defined provided the target of the first is the source of the second. It is the function resulting from taking an element of the source of the first function, applying the first function, and then applying the second to the result.

composite of partial functions 350

Similar to the composite of two functions, but the result is undefined if either of the two functions is not defined where required.

conditional probability 171

Given two events A and B, where B has non-zero probability, the conditional probability of A given B is the probability of $A \cap B$ divided by the probability of B.

conjugate, $\overline{\cdot}$ 70

The conjugate $\overline{z}$ of a complex number number $z = a + ib$ is $a - ib$.

continuous 198

A random variable is continuous if and only if it is not discrete.

countable 265

A set is countable if and only if there is an injective function from it to $\mathbb{N}$.

countably infinite 265

A set is countably infinite if it is both, countable and infinite.

cumulative distribution function (cdf) 211

The cdf of a random variable maps each element r of $\mathbb{R}$ to the probability that the random variable has a value less than or equal to r.

definition by cases 54

A way by piecing together functions to give a new function.

degree of a polynomial 402

The degree of a polynomial is the largest index whose coefficient is unequal to 0.

directed graph 353

A set (of *nodes*) connected by edges; can be described using a binary relation on the set.

discrete 198

A random variable is discrete if and only if its range is countable.

disjoint 32

Two sets are disjoint if they have no elements in common.

div 19

The (integer) quotient of two numbers when using integer division.

divides 17, 20

A number m divides a number n in some set of numbers if there exists a number k with the property that $n = km$.

divisible 17, 20

We say for natural numbers (or integers) that n is divisible by m if and only if n leaves remainder 0 when divided by m using integer division.

domain of definition 349

For a partial function it is the set consisting of all those elements of the source set for which the partial function is defined.

dominate 253

A function f from a set X to $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$ dominates another g with the same source and target if and only if the graph of f lies entirely above the graph of g (graphs touching is allowed).

empty set, $\emptyset$ 29

A set which has no elements.

equivalence class with respect to R generated by s, $[s]$ 386

The set of all elements which are related to s by the equivalence relation R.

equivalence relation 370

A binary relation on a set is an equivalence relation if it is reflexive, symmetric and transitive.

equivalence relation generated by a binary relation R 372

The transitive closure of the symmetric closure of the reflexive closure of R.

even 17, 20

An integer (or natural number) is even if and only if it is divisible by 2.

eventually dominate 255

A function f from $\mathbb{N}$ to $\mathbb{N}$ eventually dominates another g with the same source and target if and only if there is some number beyond which the graph of f lies above that of g (graphs touching is allowed). The analogous definition works for functions with source and target $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$.

expected value 224

The expected value of a random variable can be thought of as the average value it takes. It is given by the integral of the product of a number which the probability that it is the value of the random variable. If the random variable is discrete then this is given by a sum.

for all 82

Expresses a statement or property that holds for all the entities specified.

full binary tree with labels from a set S 298

This is a tree where each node has a label from S and where each node has either 0 or 2 children. Formally this is another recursively defined notion.

function 44

A function has a source and a target, and contains instructions to turn an element of the source set into an element of the target set. Where partial functions are discussed sometimes known as **total function**.

graph of a function 48

The graph of a function f with source S and target T consists of all those pairs in $S \times T$ which are of the form (s, fs).

greatest element, $\top$ 421

An element which is greater than or equal to every element of the given poset.

greatest lower bound, infimum 426

An element of a poset $(P, \leq)$ is a greatest lower bound of a given subset of that poset if it is both, a lower bound and greater than or equal to every lower bound of the given set.

group 103

A set with an associative binary operation which has a unit and in which every element has an inverse.

identity function 45

The identity function on a set is a function from that set to itself which returns its input as the output.

identity relation 343

The identity relation on a set S relates every element of s to itself, and to nothing else.

if and only if 80

Connects two properties or statement, and it is expected that one holds precisely when the other holds.

image of a set, $f[\cdot]$ 46

The image of a set consists of the images of all its elements, and one writes $f[S]$ for the image of the set S under the function f.

image of an element 46

The image of an element under a function is the output of that function for the given element as the input.

imaginary part 61

Every complex number $a + bi$ has an imaginary part b.

implies 79

Connects two properties or statements, and if the first of these holds then the second is expected to also hold.

independent 167, 220

Two events are independent if and only if the probability of their intersection is the product of their probabilities. Two random variables are independent if and only if for every two events it is the case that the probability that the two variables take values in the product of those events is the product of the probabilities that each random variable takes its value in the corresponding event..

infinite 261

A set is infinite if and only if there is an injection from it to a proper subset.

injective 105

A function is injective if and only if the same output can only arise from having the same input. An injective function is called an injection.

integer 18

A whole number that may be positive or negative.

integer division 16, 19

Integer division is an operation on integers; given two integers n and m where $m \neq 0$, we get an integer quotient $n \operatorname{div} m$ and a remainder $n \bmod m$.

intersection, $\cap$ 31

The intersection of two sets S and T is written as $S \cap T$, and it consists of all the elements of the underlying set that belong to both, S and T, The symbol $\bigcap$ is used for the intersection of a collection number of sets.

inverse 101

One element is the inverse for another with respect to a binary operation if and only if when using the two elements as inputs (in either order) to the operation the output is the unit.

inverse function 118

A function is the inverse of another if and only if the compose (either way round) to give an identity function.

law of total probability 177

A rule that allows us to express the probability of an event from probabilities that split the event up into disjoint parts.

least element, $\perp$ 421

An element which is less than or equal to every element of the given poset.

least upper bound, supremum 426

An element of a poset $(P, \leq)$ is a least upper bound of a given subset of that poset if it is both, a upper bound and less than or equal to every upper bound of the given set.

list over a set S 279

A list over a set S is a recursively defined concept consisting of an ordered tuple of elements of the given set.

lower bound 424

An element of a poset $(P, \leq)$ is a lower bound for a given subset of P if it is less than or equal to every element of that set.

maximal element 419

An element which does not have any elements above it.

measurable 194

A function from the sample set of a probability space to the real numbers is measurable if and only if for every interval it is the case that the set of all outcomes mapped to that interval is an event.

minimal element 419

An element which does not have any elements below it.

mod 19

The remainder when using integer division.

monoid 101

A set with an associative binary operation which has a unit.

multiplication law 174

The equality which says that given events A and B, the probability of the intersection of A and B is that of A given B multiplied with that of B.

$\mathbb{N}$ 14, 319

The natural numbers as a set with a number of operations. This set and its operations are formally defined in Section 6.4.

natural number 14

One of the 'counting numbers', 0, 1, 2, 3,. . . .

odd 17, 20

An integer (or natural number) number is odd if it is not even or, equivalently, if it leaves a remainder of 1 when divided by 2.

opposite relation of R, R^{op} 343

The relation consisting of those pairs (t, s) for which (s, t) is in R.

or 78

Connects two properties or statements, at least one of which is expected to hold.

ordered binary tree with labels from a set S 308

Such a tree is ordered if the set S is ordered, and if for every node, all the nodes in the left subtree have a label below that of the current node, while all the nodes in the right subtree have a label above.

pairwise disjoint 150

A collection of sets has this property if any two of them have an empty intersection.

partial function 348

An assignment where every element of the source set is assigned at most one element of the target set; one may think of this as a function which is undefined for some of its inputs.

partial order 408

A binary relation on a set is a partial order provided it is reflexive, anti-symmetric and transitive.

polar coordinates 66

A description for complex numbers based on the absolute and an angle known as the argument..

polynomial equation 26

An equation of the form $\sum_{i=0}^{n} a_i x^i$.

polynomial function 50

A function from numbers to numbers whose instruction is of the form x is mapped to $\sum_{1i=1}^{n} a_i x^i$ (where the a_i are from the appropriate set of numbers).

poset 408

A set with a partial order, also known as a *partially ordered set*.

powerset, $\mathcal{P}$ 43

The powerset of a set S is the set of of all subsets of S.

prime 89

A natural number or an integer is prime if its dividing a product implies its dividing one of the factors.

probability density function 159

A function from some real interval to $\mathbb{R}^+$ with the property that its integral over the interval is 1 and whose integral over subintervals always exists.

probability distribution 150

A function from the set of events that has the property that the probability of a countable family of pairwise disjoint sets is the sum of the probabilities of its elements.

probability mass function (pmf) 210

The pmf of a discrete random variable maps each element of the range of that random variable to the probability that it occurs.

probability space 150

A sample set together with a set of events and a probability distribution.

product of two sets 41

A way of forming a new set by taking all the ordered pairs whose first element is from the first set, and whose second element is from the second set.

proper subset 30

A set S is a proper subset of the set T if and only if S is a subset of T and there is at least one element of T which is not in S.

$\mathbb{Q}$ 22, 402

The set of all rational numbers together with a variety of operations, formally defined in Definition 0.1.3.

$\mathbb{R}^+$ 25

The set of all real numbers greater than or equal to 0.

$\mathbb{R}$ 24

The set of all real numbers.

random variable 194

A random variable is a measurable function from the set of outcomes of some probability space to the real numbers.

range of a function 46

The range of a function is the set of all elements which appear as the output for at least one of the inputs, that is, it is the collection of the images of all the possible inputs.

rational number 22

A number is rational if it can be written as the fraction of two integers. A formal definition is given on page 428 (and the preceding pages).

real number 24

We do not give a formal definition of the real numbers in this text.

real part 61

Every complex number $a + bi$ has a real part a.

reflexive 355

A binary relation on a set is reflexive if it relates each element of the set to itself.

reflexive closure 357

The reflexive closure of a binary relation on a set S is formed by adding all pairs of the form (s, s) to the relation.

relational composite 343

A generalization of composition for (partial) functions.

remainder for integer division, mod 16

The integer $n \bmod n$ is defined to be the remainder left when dividing n by m in the integers.

set difference, $\setminus$ 33

The set difference $S \setminus T$ consists of all those elements of S which are not in T.

size of a set 258, 260

A set is smaller than another if there exists an injective function from the first to the second. They have the same size if they are both smaller than the other.

standard deviation 237

The standard deviation of a random variable is given by the square root of its variance.

string over a set S 310

A formal word constructed by putting together symbols from S.

surjective 111

A function is surjective if and only if every element of the target appears as the output for at least one element of the input. This means that the image of the function is the whole target set. A surjective function is called a surjection.

symmetric closure 358

The symmetric closure of a binary relation on a set is formed by taking the union of the relation with its opposite.

there exists 84

Expresses the fact that a statement or property holds for at least one of the entities specified.

total order 412

A total order is a partial order in which every two elements are comparable.

transitive 361

A binary relation on a set S is transitive provided that (s, s') and (s', s'') being in the relation implies that (s, s'') is in the relation.

transitive closure 362

The transitive closure of a binary relation on a set is formed by adding all pairs of elements (s_1, s_n) for which there is a list of elements s_1, s_2 to s_n in S such that (s_i, s_{i+1}) is in the relation.

union, $\cup$ 31

The union of two sets S and T is written as $S \cup T$. It consists of all elements of the underlying set that belong to at least one of S and T. The symbol $\bigcup$ is used for the union of a collection of sets.

uncountable 265

A set is uncountable if it is not countable.

unique existence 85

A more complicated statement requiring the existence of an entity, and the fact that this entities is unique with the properties specified.

unit 98

An element of a set is a unit for a binary operation on that set if and only if applying the operation to that, plus any of the other elements, returns that other element.

upper bound 424

An element of a poset $(P, \leq)$ is an upper bound for a given subset of P if it is greater than or equal to every element of that set.

variance 237

The variance of a random variable with expected value e is given by the expected value of the random variable constructed by squaring the result of subtracting e from the original random variable.

$\mathbb{Z}$ 18, 398, 399

The integers with various operations, see Definition 0.1.2 for a formal account.

COMP11120, Semester 2

Exercise Sheet 11

For examples classes in Week 2

Core Exercises for this week

Exercise 136 on page 292. Do any part.

Exercise 141 on page 319.

Exercise 140 on page 318.

Extensional Exercises for this week

Exercise 132 on page 287. Do any one part.

Exercise 143 on page 320. Do any three parts.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

You should now be ready to do all the exercises in Chapter 5 and Section 6.1.

COMP11120, Semester 2

Exercise Sheet 12

For examples classes in Week 3

Core Exercises for this week

Exercise 146 on page 324. Do any two parts.

Exercise 147 on page 332.

Exercise 149 on page 333. Do (a) and (b) and any of the remaining parts.

Extensional Exercises for this week

Exercise 150 on page 334.

Exercise 154 on page 336.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

You should be able to do all exercises in Section 6.2.

COMP11120, Semester 2

Exercise Sheet 13

For examples classes in Week 4

Core Exercises for this week

Exercise 156 on page 338.

Exercise 159 on page 350.

Exercise 161 on page 351. Do part (a) and one of (c) or (d).

Extensional Exercises for this week

Exercise 157 on page 338.

Exercise 164 on page 354. Do one part from (a)–(c) and one from (d)–(f)

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

You should be able to do all exercises up to and inluding Section 6.4.2 now.

COMP11120, Semester 2

Exercise Sheet 14

For examples classes in Week 5

Core Exercises for this week

Exercise 167 on page 360. Do any two parts.

Exercise 169 on page 363. Do one part from (a)–(c) and one from (d)–(g).

Exercise 174 on page 378.

Extensional Exercises for this week

Exercise 171 on page 365. Do any part.

Exercise 175 on page 378.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

You should now be able to do Exercises up to Section 7.2.

COMP11120, Semester 2

Exercise Sheet 18

For examples classes in Week 9

Core Exercises for this week

Exercise 181 on page 394. Do one part from (a)–(b), one from (c)–(e) and one from (f)–(h).

Exercise 183 on page 398. Do one part from (a)–(b), one from (c)–(e), one from (f)–(h) and one from (i)–(k).

Exercise 186 on page 404. Do any one part.

Extensional Exercises for this week

Exercise 182 on page 395.

Exercise 187 on page 412. Do all parts; it is sufficient to carry out two calculations each time instead of all four.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

You should now be ready to do all exercises up to and including Section 7.3.6.

COMP11120, Semester 2

Exercise Sheet 19

For examples classes in Week 10

Core Exercises for this week

Exercise 188 on page 417. Do one from (a)–(c),one from (d)–(f) and one from (g)–(h).

Exercise 189 on page 421.

Exercise 201 on page 439. Do one part from (a)–(d),one from (e)–(h), one from (i)–(k) and one from (l)–(o).

Extensional Exercises for this week

Exercise 197 on page 432.

Exercise 196 on page 431.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

You should now be ready to do all exercises up to and including Section 7.4.1.

COMP11120, Semester 2

Exercise Sheet 20

For examples classes in Week 11

Core Exercises for this week

Exercise 206 on page 450. Do one part from (a)–(c), one from (d)–(f), one from (g)–(j) and one from (k)–(l).

Exercise 211 on page 457. Do one part from (a)–(c) and one from (d)–(e).

Exercise 214 on page 460.

Extensional Exercises for this week

Exercise 207 on page 451. Do all parts, but in each part you only have to do one of the statements (for greatest/least, or maximal/minimal).

Exercise 213 on page 458.

Remember that

- if you are stuck on an exercise move on to the next one after ten minutes, but write down why you got stuck so that the GTA can see what you were trying to do, and consider coming back to that exercise again later;
- you may only use concepts which are defined in these notes (Chapter 0 establishes concepts for numbers), and for every concept you do use you should find the definition in the notes and work with that;
- you should justify each step in your proofs;
- the GTA will assign a rough score—in the examples classes you will find out more about constructing better solutions;
- in the examples classes you have an opportunity to ask the GTA questions, and you should think of those in advance;
- the GTA may ask you to explain some of your solution.

You should now be ready to do all exercises in the notes.