City University of New York (CUNY)

CUNY Academic Works

Open Educational Resources

Queensborough Community College

2021

Introduction to Discrete Mathematics: An OER for MA-471

Mathieu Sassolas

CUNY Queensborough Community College

How does access to this work benefit you? Let us know!

More information about this work at: https://academicworks.cuny.edu/qb_oers/179

Discover additional works at: https://academicworks.cuny.edu

This work is made publicly available by the City University of New York (CUNY).
Contact: AcademicWorks@cuny.edu

Introduction to Discrete Mathematics: An OER for MA-471

Mathieu Sassolas

2021-11-17

Department of Mathematics and Computer Science
Queensborough Community College
City University of New York

Student version

This Open Education Resource is licensed under the Creative Commons Attribution-NonCommercial 4.0 International License.

Contents

Foreword **iii**

I Propositional Logics **1**

I.A Introduction . . . 2

I.B Boolean Algebra . . . 3

I.C Classifying formulas . . . 14

II First-order logic (FO) **19**

II.A Predicates . . . 20

II.B Quantifiers . . . 21

II.C Equivalence in first-order logic . . . 27

III Proof systems and proof patterns **31**

III.A What is a proof? . . . 32

III.B Proof systems . . . 33

III.C From formal proofs to mathematical proofs . . . 41

IV Set Theory **49**

IV.A Sets over a domain . . . 50

IV.B Sets of sets . . . 58

IV.C Sets of tuples . . . 60

V Equivalence and Order Relations **73**

V.A Equivalence relations . . . 74

V.B Order relations . . . 80

VI Sequences and Recurrence **87**

VI.A Sequences . . . 88

VI.B Proofs by induction . . . 93

VI.C Application: Growth and complexity . . . 97

VII Number Theory **107**

VII.A Division of integers . . . 108

VII.B Prime numbers . . . 113

VII.C Greatest Common Divisor and Least Common Multiple . . . 119

VII.D Diophantine Equations . . . 123

VII.E Congruence . . . 127

VIII Introduction to Combinatorics **139**

VIII.A What is Combinatorics? . . . 140

VIII.B Choosing k among n . . . 142

Foreword

Objectives

Topics

This book was developed with the syllabus of *MA-471: Introduction to Discrete Mathematics* of Queensborough Community College (QCC) in mind. It therefore bears the same objectives, which can be summarized as follows.

The first objective is to define and discuss the meaning of *truth* in mathematics. We explore logics, both propositional (Chapter I) and first-order (Chapter II), and the construction of proofs, both formally and human-targeted (Chapter III).

Using the proof tools, this book then explores some very fundamental definitions of mathematics through set theory (Chapter IV). This theory is then put in practice in several applications. The particular (but quite widespread) case of equivalence and order relations is developed in Chapter V. Chapter VI introduces sequences and proofs by induction. Number Theory is the focus of Chapter VII. Finally, a small introduction to combinatorics is given in Chapter VIII.

As the name indicates, this book is an *introduction*, so most topics touched in this book would deserve a book in their own right.

Math or Computer Science?

The topics of this book oscillate between Computer Science and Mathematics. This is because theoretical computer science (the *science* part, that can be done with pen and paper rather than on a computer) needed in its inception to properly define truth and the mechanics of truth so that a machine could handle it. As a result, throughout the book some references are made to computer science and sometimes programming. A reader unfamiliar with these concepts can skip these parts, but for a computer programmer the relations between the theoretical concepts and their application in programming should help understand the theory better. In any case, knowing theoretical foundations helps being a better programmer, whether programming is learned before or after taking this course.

Take-home message

As the course touches lots of fundamental aspects of math, the book contains lots of definitions and vocabulary. Although it is necessary for the book to be self-contained, the vocabulary is not the most important idea of the book. What matters most is that it introduces and uses proof techniques. These techniques provide powerful tools and problem-solving strategies to approach any mathematical problem. In addition, they set a high standard for argumentation in everyday life: wrong arguments and wrong proofs are more easily detected when the mind knows what to expect from an actual well-formed proof. This protection against misleading arguments is valuable regardless of future coursework and career.

How to use this book

Writing Intensive

The Mathematics and Computer Science department at QCC designated the course MA-471 as *Writing intensive*. That means students must produce a certain amount of written material throughout the semester.

In the context of this course, the main focus of the writing will be proofs. As a result, reading the proofs in this book should provide many examples as to what is expected in writing such proofs, as it is different from writing an English essay.

Margin notes

This book uses a relatively wide margin that allows for extensive marginal notes. They are color-coded for your convenience:

- Green notes indicate an internal reference within the book. In PDF form, the links are clickable and lead to the pointed section.
- Dark red notes (all with the same text) indicate this part is more part of mathematical culture, usually some concepts that should be expanded more but do not fit within the scope of the course.
- Bright red notes indicate that you must proceed carefully.
- Yellow notes are uncategorized notes. They include (among others) historical comments and fun facts.

An internal reference.

This is for your personal culture more than to be applied in this course.

A warning.

A note.

Student *vs* Instructor version

This textbook exists in two versions: the student and instructor version. The only difference between the two is the presence of exercise answers in the Instructor version.

This version is the Student version. If you are an instructor looking for the instructor version, please contact me at `MSassolas@qcc.cuny.edu`.

Comments, bug reports, reuses

This book has been put in circulation on-line without being extensively tested (especially by students). I welcome any suggestions and comments, even if only to report a typo, by email at `MSassolas@qcc.cuny.edu`.

If you wish to reuse this book with modification and need access to the LaTeX source, you can also reach out to me. I cannot promise the source is clean, though. This work is being published under the *Creative Commons Attribution-NonCommercial 4.0 International License*, so any derivative work should “include at least the same license elements as the license applied to the original material” (`https://creativecommons.org/faq/`).

Sources and acknowledgments

This book was created from the slides used in class developed for the course in Fall 2020 and Spring 2021 semesters. The writing style of the book may be an indication of this origin, as it was intended as an extended version of the course as given in talk and slides. (The initial intent was to have enough space to write the longest proofs properly.)

While the slides themselves and therefore this book were my personal writing, they were inspired by my readings at the time. These include, but are not limited to:

- Lecture notes and exercises for this course from my colleague Dr. Kwang Hyun Kim.
- *Mathematics for Computer Science* by Eric Lehman, Tom Leighton, and Albert Meyer; `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2010/readings/`.
- Wikipedia, for historical points and notation examples.
- Stackoverflow and TEXample for LATEX/Tikz code for figures (the source is indicated in the text).

Chapter I
Propositional Logics

Chapter contents

I.A Introduction **2**

I.A.1 What is truth? 2

I.A.2 Propositions 2

I.A.3 Natural language *vs* mathematical language 3

I.B Boolean Algebra **3**

I.B.1 The building blocks: Propositional variables and operators . . 3

I.B.1.i Propositional variables 3

I.B.1.ii Boolean values and operators 4

I.B.2 Truth table for formulas 6

I.B.2.i Example: Writing the truth table for $(q \wedge r) \vee (p \wedge (q \vee r))$ 7

I.B.2.ii Logical equivalence 8

I.B.2.iii Remark: Equivalence and if-and-only-if 9

I.B.3 Putting the Algebra back into Boolean Algebra 9

I.B.3.i Neutrality and Absorption 9

I.B.3.ii Double negation and Idempotence 10

I.B.3.iii Commutativity, Associativity, Distributivity 10

I.B.3.iv De Morgan's laws, Contraposition 11

I.B.3.v Constructing operators from others 12

I.B.3.vi Calculating in the Boolean algebra 14

I.C Classifying formulas **14**

I.C.1 Formula categories 14

I.C.2 Classifying formulas, in practice 15

I.C.2.i Example: $\varphi = (p \vee q) \to (p \wedge \neg r \wedge q)$ 16

I.C.2.ii Example: $\psi = (p \wedge q) \to (p \vee q)$ 16

I.C.3 Satisfiability 17

I.C.3.i The satisfiability problem: SAT 17

I.C.3.ii The NP class 18

I.A Introduction

I.A.1 What is truth?

"What is truth?" is both a very deep and very broad question. It has been asked by philosophers for centuries (at least since the Greek antiquity), and have received some partial answers, depending on what discipline focuses on the issue. In the context of this book, we will take the mathematicians' approach, and more precisely the logicians' approach. We can summarize this as follows:

> Truth is either something we **assume** to be true, or something we can **build** as true from other truths.

Although this definition seems self-contained, it leaves two (big) opportunities for interpretation:

- What should be the truth that we assume?
- By what mechanisms can we build new truths?

There is actually no one good answer to that, and logicians are still actively working on this topic: Playing with these parameters generate a wild bunch of *logical systems*. In this book we will only see two such systems: Propositional calculus (in this chapter) and Natural Deduction system (in Chapter III).

I.A.2 Propositions

The framework described above is still too broad to be tackled formally by logics: in natural language (here English), there are lots of *context* that is implicit in a statement. For example "My name is John" is a true statement only if uttered by someone actually named John; "It is raining" may be true or false, depending on the time and place of the statement. Even mathematical statements can bear some uncertainty: "$2x - 3 = 0$" may be a true or false statement, depending on the value of x.

As a result, the statements that are studied in logics are only *propositions*: statements that are either true or false, regardless of context. It does not matter whether you actually know the truth value of the statement, what matter is that it is definite. For example "It rained on the current location of QCC on June 17th, 1076" is a proposition: it is either true or false, even though nobody knows which. Other, somewhat simpler, examples of proposition include:

- "$1 + 1 = 2$" is a true proposition.
- "$2 + 2 = 3$" is a false proposition.
- "$\int_0^1 \int_0^1 \frac{1}{1-xy} dx\, dy = \frac{\pi^2}{6}$" is a true proposition, although it is not obvious why!

The notion of *free variable* will be studied with more detail in Section II.B.6.

As noted before, statements that include external parameters (such as a free variable) are not propositions. Sentences that are not statements are not propositions either:

- "Let's go!" is an injunction, not a statement, hence not a proposition.

- “What time is it?” is a question, not a statement, therefore not a proposition.
- ”$3x^2 - 7 = 0$ depends on the value of x, so it is not a proposition.
- “This statement is false” is a paradox: it is neither true or false.

Exercise I.1

For each of the following statements, decide whether it is a “Proposition” or “Not a proposition”.

1. The exponential function is its own derivative.
2. There exists a finite quantity of prime numbers.
3. How are you doing?
4. There exists an infinite number of ways to write 3 as the sum of three cubes.
5. $x^2 - 2x + 5 = 9$
6. $x^3 - 2x^2 - 18x + 9 = 0$ when $x = 3$

I.A.3 Natural language *vs* mathematical language

Even when context is clear, the English language has ways to produce completely correct sentences (grammatically speaking) that can be interpreted in several ways. This ambiguity usually stems from internal references: connectors or pronouns can refer to more than one part of the sentence. An example of this ambiguity can be found in the song *Lola*, by The kinks (it is completely deliberate): “I'm glad I'm a man and so is Lola”. Does “so” refer to the fact that the speaker is “glad” or to the fact that he is “a man”? A simpler – and probably involuntary – ambiguity is also found in the tabloid headline “A mother beats up her daughter because she was drunk”, where the “she” can refer to either of the protagonists.

And without pronouns, parsing (i.e. performing grammatical analysis) a sentence might in itself be difficult: “John saw the man on the mountain with a telescope”. The trouble a human has to parse this shows the immense difficulty a computer would!, despite the great progress shown in the field of Natural Language Processing (which is not at all the topic of this book).

But the difficulty do not end here: in English, a word can have several meanings, which we know to interpret based on social context. Compare the sentences “You can have chicken *or* fish” with “Since you like Turner, you may like Monet *or* Pissarro”: in the first case, it is implied that you cannot have both meal options, while in the second case you may like both painters. In the mathematical context, we will give a single meaning to the word “or”, with a formal definition, in order to avoid this kind of confusion.

I.B Boolean Algebra

I.B.1 The building blocks: Propositional variables and operators

I.B.1.i Propositional variables

Although propositions represent an statement on the real world, it is not really what is the focus of logics: what matters is how thes propositions relate to each other. As a result, the actual content of the proposition can usually be abstracted away through a propositional variable.

> A *propositional variable* is a name (usually a single letter) used to represent a proposition. By habit, we mostly use the letters p, q, r.

Sometimes a letter closer to the content of the proposition is used. For example, we can write the following to define some propositional variables that represent "real-life" statements.

- Let c be the proposition "Socrates is a cat"
- Let s denote the proposition "It is snowing"
- We define t as the proposition "John is taller than Bob"

In most cases, propositional variables denote a yet undetermined proposition: we can reason on propositions before knowing (or without caring) whether it is true or false.

I.B.1.ii Boolean values and operators

I.B.1.ii.a Syntax

Propositional variables are sometimes called *atomic*[1] *propositions*, because they are the base building blocks of more complex propositions. We denote by AP the set of all atomic propositions, which are the propositional variables we will use.

What joins these building blocks are the *boolean operators*, and the result of said combination is called a *formula* (plural: *formulas*, or *formulae* to be pedantic). By habit, formulas are denoted using the Greek letters φ or ψ (and sometimes ξ).

The boolean values and operators are as follows:

Booleans are named after English logician Georges Boole.

Use $\top$ and $\bot$ when handwriting because letters **T** and **F** can easily be confused.

$\boxed{\top}$ True: the value of a true proposition. When writing in plain-text it can be replaced by the letter **T**.

$\boxed{\bot}$ False: the value of a false proposition. When writing in plain-text it can be replaced by the letter **F**.

$\boxed{\neg}$ Negation ("not"), unary operator.

$\boxed{\wedge}$ Conjunction ("and"), binary operator.

$\boxed{\vee}$ Disjunction ("or"), binary operator.

$\boxed{\oplus}$ XOR ("exclusive-or"), binary operator.

$\boxed{\rightarrow}$ Implication ("if…then"), binary operator.

$\boxed{\leftrightarrow}$ Iff ("if and only if"), binary operator.

The above definition of what a formula is is often summarized with a *grammar*:

$$\varphi := \top|\bot|p \in AP|\neg\varphi|\varphi \wedge \varphi|\varphi \vee \varphi|\varphi \oplus \varphi|\varphi \rightarrow \varphi|\varphi \leftrightarrow \varphi$$

To read these grammars, just replace φ by "a formula" and the | symbol by "or": A formula is $\top$, or $\bot$, or a propositional variable, or the $\neg$ of a formula, or the $\wedge$ of two formulas…

[1] Atomic means "which cannot be cut" in Greek.

φ	$\neg\varphi$
$\top$	$\bot$
$\bot$	$\top$

(a) Negation: $\neg$

φ	ψ	$\varphi \wedge \psi$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$

(b) Conjunction: $\wedge$

φ	ψ	$\varphi \vee \psi$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\bot$

(c) Disjunction: $\vee$

φ	ψ	$\varphi \oplus \psi$
$\top$	$\top$	$\bot$
$\top$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\bot$

(d) XOR: $\oplus$

φ	ψ	$\varphi \rightarrow \psi$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\top$

(e) Implication: $\rightarrow$

φ	ψ	$\varphi \leftrightarrow \psi$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\top$

(f) Iff: $\leftrightarrow$

Figure I.1: Truth tables for Propositional Logics.

I.B.1.ii.b Semantics

Up to now, only symbols have been given: this is called the *syntax* of the logics. But the symbols have no actual meaning: we are missing the *semantics* of the logics. In the case of propositional logics, the semantics of the operators gives the truth value of the compound formula based on the truth value of the sub-formulas. Since all combination of truth values for sub-formulas must be considered, the semantics of an operator is given in a *truth table* that lists all these possibilities.

The truth tables of the above operators are displayed in Figure I.1. As the name indicates, Negation turns true into false and false into true.

Conjunction, that we read as "and", is true only if both sub-formulas are true. Dually, disjunction, that we read as "or", is true if *at least* one of the sub-formula is. This is a different interpretation than the usual English one: it is an *inclusive or*: to the "chicken or fish" question, logicians allow themselves to answer "both". The exclusive or is closer to the English interpretation: it is true when *exactly one* of the sub-formula is. So "chicken xor fish" forces you to choose one of them only. Remark that the actual meaning of "chicken or fish" is often "not(chicken and fish)", because you are perfectly allowed to have neither, which does not work with the interpretation of $\vee$ or $\oplus$…

Implication mimics "if…then…": for $\varphi \rightarrow \psi$ to be true, whenever φ (called the *premise*) holds, then so must ψ (the *conclusion*). Note that when φ is not true, there is no obligation whatsoever on ψ: so for a false premise, the implication will always be true, as shown in the examples below:

- $2 + 2 = 4 \rightarrow 1 \neq 0$ is true because both parts are true
- $42 \times 0 = 0 \rightarrow \pi = 42$ is false because even though the left part is true, the right part is false
- $1 + 2 = 5 \rightarrow \frac{1}{2} < 3$ is true because $1 + 2 = 5$ is false ($\frac{1}{2} < 3$ happens to be true)
- $\frac{1}{2} > \frac{7}{8} \rightarrow 1 + 1 = 7$ is true because $\frac{1}{2} > \frac{7}{8}$ is false ($1 + 1 = 7$ happens to be false)

So as long as the premise is false, the conclusion can be anything. That allows for some seemingly strange statements to be true: “if 1=0, then the moon is larger than the earth” is a true sentence!

I.B.1.ii.c Precedence

Mathematical operators $+$, $\times$, etc have an order of precedence: an implicit order of operations that we don’t have to indicate with parenthesis (usually summed up as the acronym PEMDAS). Similarly, logical operators also have an order of precedence to avoid some parentheses:

- $\neg$ has the highest precedence: $\neg p \wedge q$ is to be understood as $(\neg p) \wedge q$.
- $\wedge$ has precedence over $\vee$: $p \vee q \wedge r$ is to be understood as $p \vee (q \wedge r)$ (but it is usually clearer to still use parenthesis in this case).
- $\rightarrow$ has lower precedence than $\wedge$ and $\vee$ (and $\neg$), but higher precedence than $\leftrightarrow$: $p \rightarrow q \leftrightarrow p \rightarrow q \vee r$ is to be understood as $(p \rightarrow q) \leftrightarrow (p \rightarrow (q \vee r))$.

The $\oplus$ has no real fixed precedence: it depends on the context. For example, in the C programming language (and its derivatives like C++) it is between $\wedge$ and $\vee$, but mathematical texts might use another convention. So it is better to use parenthesis when using the exclusive or.

I.B.2 Truth table for formulas

Based on the semantics of the operators, one can calculate the semantics of any formula. This is done by computing the truth table for this formula: finding for every possible combination of truth value of the atomic propositions, called a *valuation*, what is the truth value of the whole formula.

The procedure is as follows:

1. Make one column for each variable.
2. Write all possible sets of values for the variables appearing, that creates 2^n lines if there are n variables, as depicted in Figure I.2. It helps to follow some method in order not to forget any line. For example, in the tables displayed in Figure I.2, the rightmost variable has alternation of $\top$ and $\bot$, the variable to its left has blocks of 2 $\top$, then 2 $\bot$, then blocks of $4 = 2^2$, etc until the leftmost variable has blocks of 2^{n-1} $\top$ and a block of 2^{n-1} $\bot$.
3. Decompose the formula to find the *sub-formulas*. This is done is the reverse order of operations: start with the operator that would be applied last; its operand(s) are the sub-formulas. Then proceed similarly to decompose the sub-formulas until variables are reached. It helps to give names to the sub-formulas.
4. Write a column for each sub-formula, starting from the least complex and ending in the whole formula.
5. Fill the columns using the truth tables for the operators, based on the value present in the column of the operands.

p	q	$\ldots$
$\top$	$\top$	
$\top$	$\bot$	
$\bot$	$\top$	
$\bot$	$\bot$	

(a) Truth table for two variables

p	q	r	$\ldots$
$\top$	$\top$	$\top$	
$\top$	$\top$	$\bot$	
$\top$	$\bot$	$\top$	
$\top$	$\bot$	$\bot$	
$\bot$	$\top$	$\top$	
$\bot$	$\top$	$\bot$	
$\bot$	$\bot$	$\top$	
$\bot$	$\bot$	$\bot$	

(b) Truth table for three variables

Figure I.2: Line structure of truth tables for 2 or 3 variables.

			φ	ξ	$\psi = p \wedge \xi$	$\varphi \vee \psi$
p	q	r	$q \wedge r$	$q \vee r$	$p \wedge (q \vee r)$	$(q \wedge r) \vee (p \wedge (q \vee r))$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\top$	$\bot$	$\bot$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\top$	$\bot$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\top$
$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Figure I.3: Truth table for $(q \wedge r) \vee (p \wedge (q \vee r))$.

I.B.2.i Example: Writing the truth table for $(q \wedge r) \vee (p \wedge (q \vee r))$

1. There are 3 variables, so we can start by drawing a truth table with 3 columns for p, q, r.

2. We write all possible combinations in $2^3 = 8$ lines as in Figure I.2(b).

3. $(q \wedge r) \vee (p \wedge (q \vee r))$ has sub-formulas $\varphi = (q \wedge r)$ and $\psi = p \wedge (q \vee r)$, which in turns has sub-formula $\xi = q \vee r$.

4. A column is added for φ, ξ, ψ, and one column for the whole formula.

5. The lines are filled using the appropriate columns: for example, column φ and ξ are filled using only columns q and r, column p does not matter. Column ψ is filled using column p and ξ, the other ones do not matter. Note that since it is a conjunction, the truth table for conjunction is used: column ψ has a $\top$ only when both columns p and ξ have a top in this line.

The end result is displayed in Figure I.3.

p	q	r	$q \vee r$	$p \wedge (q \vee r)$	$(p \wedge q)$	$(p \wedge r)$	$(p \wedge q) \vee (p \wedge r)$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Figure I.4: Joint truth table for $p \wedge (q \vee r)$ and $(p \wedge q) \vee (p \wedge r)$.

I.B.2.ii Logical equivalence

Definition: Logical Equivalence

Two formulas using the same set of atomic variables are said to be *logically equivalent* if they are true for exactly the same combination of the truth value of the variables.

Equivalently, two equivalent formulas have the same last column in their truth table. This is written by the symbol $\equiv$, to differentiate from syntactical equality (which is denoted with the usual $=$ symbol).

For example $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$: as we can see in the truth table of Figure I.4 where both $p \wedge (q \vee r)$ and $(p \wedge q) \vee (p \wedge r)$ are gathered in the same table, the fifth and eight columns are identical.

Exercise I.2

1. Build the truth tables of the following formulas:

   a. $\neg p \vee q$
   b. $\neg p \vee \neg q$
   c. $\neg(p \vee q)$
   d. $\neg(p \wedge q)$
   e. $\neg p \wedge \neg q$
   f. $\neg q \rightarrow \neg p$
   g. $(p \rightarrow q) \wedge (q \rightarrow p)$
   h. $\neg(p \oplus q)$
   i. $(p \wedge q) \vee (\neg p \wedge \neg q)$
   j. $p \rightarrow q$[a]
   k. $p \leftrightarrow q$

2. Which of these formulas are equivalent to each other?

Practical instructions:

- Use a single (wide) truth table.
- A spreadsheet document works well for this. In this case, use `NOT`, `AND`, `OR`, `XOR`, `->`, `<->` instead of symbols and color columns in the same color if they are logically equivalent.

[a] No real work needed in this question and the next, this is for Question 2.

I.B.2.iii Remark: Equivalence and if-and-only-if

The if-and-only-if $leftrightarrow$ operator is true whenever its operands are true at the same time and false at the same time. This idea is really close from logical equivalence: both sides of the $\equiv$ symbol are true at the same time and false at the same time. And indeed these notions are related:

Here "at the same time" is an abuse of language to mean "for the same truth values of the propositional variables appearing in the formulas".

For any formulas φ and ψ, $\varphi \equiv \psi$ means $\varphi \leftrightarrow \psi \equiv \top$.

As a result, $\leftrightarrow$ is sometimes (abusively) referred to as the *equivalence* operator. This confusion must not hide from the mind that these two "equivalences" actually work at different levels:

- The $\leftrightarrow$ is an operator defined at the logical level, and its result may be true or false.
- The $\equiv$ is a relation defined at the mathematical level (i.e. the level at which mathematician communicate with each other as humans) and implicitly states that the equivalence holds.

In other terms, when $\varphi \leftrightarrow \psi$ is false, the statement "$\varphi \equiv \psi$" is a mathematical mistake, a reasoning error, that should not have been written.

We explore relations in Chapter V, and this particular type of relation in Section V.A.

I.B.3 Putting the Algebra back into Boolean Algebra

The equivalences found in Exercise I.2 show that we can rewrite some formulas into others. This is the base for the calculation rules that form *Propositional Calculus*. The structure made of the boolean values and the properties of the operator is called *Boolean Algebra*, because it shares some structural properties with real algebra (the "usual" algebra on numbers). To some extent, $\top$ can be thought to be 1, $\bot$ to be 0, $\wedge$ to be $\times$ and $\vee$ to be $+$. The correspondence is not perfect since $1 + 1 = 1$ with booleans; indeed, booleans have only two "numbers"! That does not prevent electrical engineers to use these notations instead of $\top$ and $\bot$, and seeing it that way can help in memorizing the rules.

No proofs will be provided for these rules, as it is left as an exercise to the reader to build the truth tables that show logical equivalence for the rules not covered by Exercise I.2.

I.B.3.i The special case of $\top$ and $\bot$: Neutrality and Absorption

The boolean values $\top$ and $\bot$ are not variables: they are the possible values that every formula can take. They also have a special role regarding the operators $\wedge$ and $\vee$.

Namely, whenever $\top$ is present in a conjunction, it does not affect the result. Similarly, $\bot$ does not affect the result of a disjunction. This is called *neutrality*, and we say that $\top$ *is the neutral element for conjunction* and that $\bot$ *is the neutral element for disjunction*. Remark that we have the same thing in real algebra: $x + 0 = x$ and $x \times 1 = x$ (0 is the neutral element for addition, 1 is the neutral element for multiplication).

Furthermore, whenever $\top$ is present in a disjunction, the result will always be $\top$, regardless of the other operands. And $\bot$ makes any conjunction $\bot$, regardless of the other operand. This property is called *absorption* and we say that $\top$ *is absorbing for disjunction* and that $\bot$ *is absorbing for conjunction*. This is similar to real algebra only for 0: $x \times 0 = 0$ (0 is absorbing for multiplication).

Some sources use the term *domination* instead of absorption. That vocabulary relates to the boolean algebra as a *lattice* rather than an *algebra*, so absorption will be used in this book.

For any formula φ:

$\varphi \wedge \top \equiv \varphi$ $\varphi \vee \bot \equiv \varphi$ (Neutrality)

$\varphi \vee \top \equiv \top$ $\varphi \wedge \bot \equiv \bot$ (Absorption)

I.B.3.ii Doing the same thing twice: Double negation, Idempotence, and the rest

As negation changes the truth value from true to false and vice versa, performing it twice bring us back to the starting point: therefore a double negation can be eliminated.

Another interesting case to consider is trying to apply a binary operator to the same operand twice. From the point of view of truth table, that means we need to consider the two lines $\top, \top$ and $\bot, \bot$ (the first and fourth in the truth tables as displayed in Figures I.1(b-f)). What we see is that for both conjunction and disjunction, line $\top, \top$ yield $\top$ and line $\bot, \bot$ yield $\bot$. Therefore applying a conjunction (or a disjunction) to the same value twice will always produce this very original value. This property is called *idempotence*: we say that $\wedge$ *is idempotent*, and that $\vee$ *is idempotent*.

The case of the other operators can also be considered, also is it of lesser importance in practice. In these cases, the value produced is always the same regardless of the original value.

For any formula φ:

$\neg(\neg\varphi) \equiv \varphi$ (Double negation elimination)

$\varphi \wedge \varphi \equiv \varphi$ $\varphi \vee \varphi \equiv \varphi$ (Idempotence)

$\varphi \oplus \varphi \equiv \bot$ $\varphi \to \varphi \equiv \top$ $\varphi \leftrightarrow \varphi \equiv \top$

I.B.3.iii Commutativity, Associativity, Distributivity

For an operator, *commutativity* denotes the fact that its operands can appear in any order without affecting the result of the operation. In real algebra, it is the case for both multiplication and addition: $x \times y = y \times x$ and $x + y = y + x$. In the case of boolean algebra, what need to be considered to establish commutativity are the lines of the truth tables where the operands are different: lines $\top, \bot$ and $\bot, \top$ (the second and third in the truth tables as displayed in Figures I.1(b-f)). We can see that conjunction, disjunction, exclusive-or, and iff are commutative. On the other hand, implication is not commutative as $\bot \to \top$ is true but $\top \to \bot$ is false. Note that the choice of the symbols used to represent these operators reflects this: the symbols for the commutative operators are symmetrical, but the $\to$ of the implication is not.

Commutativity

For any formulas φ and ψ:

$\varphi \wedge \psi \equiv \psi \wedge \varphi$ $\varphi \vee \psi \equiv \psi \vee \varphi$ $\varphi \oplus \psi \equiv \psi \oplus \varphi$ $\varphi \leftrightarrow \psi \equiv \psi \leftrightarrow \varphi$

Associativity refers to the fact that when two of the same operation is to be performed, the order in which they are performed does not matter. In some sense, an associative operator has no precedence with itself because order does not matter. In real algebra, both addition and multiplication are associative. In boolean algebra, proving associativity requires to look at a table with three variables, and comparing the cases where operations are performed left-to-right and right-to-left. The result, left as an exercise to the reader, is that conjunction, disjunction, exclusive-or, and iff are associative. On the other hand, implication is not associative: $(\bot \to \top) \to \bot$ is false but $\bot \to (\top \to \bot)$ is true. As a result, we ought to define the order of operation in which $\varphi \to \psi \to \xi$ should be interpreted: it is to be understood as $\varphi \to (\psi \to \xi)$, meaning implications must be calculated from right to left (but parentheses are advised anyway).

✍ We say that → is *right-associative.*

Associativity

For any formulas φ, ψ, and ξ:

$$\varphi \land \psi \land \xi \equiv (\varphi \land \psi) \land \xi \equiv \varphi \land (\psi \land \xi) \qquad \varphi \lor \psi \lor \xi \equiv (\varphi \lor \psi) \lor \xi \equiv \varphi \lor (\psi \lor \xi)$$

$$\varphi \oplus \psi \oplus \xi \equiv (\varphi \oplus \psi) \oplus \xi \equiv \varphi \oplus (\psi \oplus \xi) \qquad \varphi \leftrightarrow \psi \leftrightarrow \xi \equiv (\varphi \leftrightarrow \psi) \leftrightarrow \xi \equiv \varphi \leftrightarrow (\psi \leftrightarrow \xi)$$

Distributivity is the capacity of an operator to “enter the parenthesis of another one”. The most common example is the distributivity of multiplication over addition in real algebra: $x \times (y + z) = x \times y + x \times z$. In the case of boolean algebra, distributivity is much more widespread. Indeed, not only *conjunction is distributive over disjunction* (this was actually proved through the truth table of Figure I.4), but also the other way around: *disjunction is distributive over conjunction* (the proof is left as an exercise to the reader).

Distributivity

For any formulas φ, ψ, and ξ:

$$\varphi \land (\psi \lor \xi) \equiv (\varphi \land \psi) \lor (\varphi \land \xi) \qquad \varphi \lor (\psi \land \xi) \equiv (\varphi \lor \psi) \land (\varphi \lor \xi)$$

There are actually more distributive properties in the boolean algebra, but listing them all would be more confusing that enlightening (although one will be proved in Exercise I.3).

Remark that distributivity can work in reverse: factoring can also be performed using the same rules, but from right to left.

I.B.3.iv Playing with negation: De Morgan’s laws, Contraposition

As we should have seen in Exercise I.2, negation does not distribute directly over conjunction or disjunction. For example $\neg(\top \lor \bot)$ is false (as the negation of $\top$), while $\neg\top \lor \neg\bot)$ is true (as the disjunction of $\bot$ and $\top$). Indeed, to distribute the negation, one must also change the operator: negation transforms disjunction into conjunction and vice versa. These properties are gathered under the name of *De Morgan’s laws.*

✍ Named after the British logician Augustus De Morgan who formalized them.

The De Morgan’s laws seem trivial, but they really express the duality at play between disjunction and conjunction, which we can rephrase by describing their truth tables (Figures I.1(b-c)) with the following similar sentences:

- The conjunction is only true when both operands are true.
- The disjunction is only false when both operands are false.

De Morgan's laws

For any formulas φ and ψ:

$$\neg(\varphi \land \psi) \equiv \neg\varphi \lor \neg\psi \qquad \neg(\varphi \lor \psi) \equiv \neg\varphi \land \neg\psi$$

Negation also behaves in a special way with implication. The negation of an implication does not look like an implication, as we will see in Section I.B.3.v. What is more interesting is the implication of the negations: it is also an implication, but is reversed (the proof of this is in Exercise I.2). The implication $\neg\psi \to \neg\varphi$ is called the *contrapositive* of $\varphi \to \psi$. The fact that they are equivalent is quite useful for proofs.

↪ More details on proofs by contraposition in Section III.C.4.iii.

Contrapositive

For any formulas φ and ψ:

$$\varphi \to \psi \equiv \neg\psi \to \neg\varphi$$

I.B.3.v Constructing operators from others

The last set of rules are about rewriting operators using others. As a result, there is no one "basic" set of operators, but several equivalent presentations of the logic. Fewer symbols mean an easier formalization but expressing an actual statement would be more complex.

I.B.3.v.a From ¬, ∧, and ∨

The simplest (in cognitive rather than mathematical terms) set of "basic" operators is using only $\neg$, $\land$, and $\lor$, and building the others from them. Note that, using De Morgan's laws, using only one of $\land$ or $\lor$ would be sufficient, but it is usual to have them both for symmetry. Also remark that $\top$ and $\bot$ are not "basic" operators here, because they can be constructed. Namely, $\top$ is the disjunction $\varphi \lor \neg\varphi$, which is called the *excluded middle*: a formula is either true or false (in which case its negation is true), there is no "middle" option. Symmetrically, $\bot$ is defined as the conjunction $\varphi \land \neg\varphi$: no proposition can be both true and false (in which case its negation is true), that would be a *contradiction*.

✍ In Latin the name is *tertium non datur*, literally meaning "no third possibility is given".

The case of implication is quite useful, since it allows to replace the non-commutative, non-associative $\to$ operator by a combination of $\neg$ and the commutative and associative $\land$ and $\lor$. As proved in Exercise I.2, the implication $\varphi \to \psi$ is equivalent to $\neg\varphi \lor \psi$. Note that as a result, the negation of the implication is not an implication.

↪ See Exercise I.3 Question 1.

The exclusive-or and iff can be defined in a symmetrical way from each other: indeed, the XOR is true when the operands differ, which is when the iff is true, so they are the negation from one another. The XOR can be expressed in a way that follows its name of "exclusive or": for $\varphi \oplus \psi$ to be true, $\varphi \lor \psi$ must be true, but also, we need to exclude the case when both operands are true, i.e. the case $\varphi \land \psi$. As a result we can rewrite $\varphi \oplus \psi$ as $(\varphi \lor \psi) \land \neg(\varphi \land \psi)$.

For any formulas φ and ψ:

$$\top \equiv \varphi \lor \neg\varphi \text{ (Excluded middle)} \qquad \bot \equiv \varphi \land \neg\varphi \text{ (Contradiction)}$$

$$\varphi \to \psi \equiv \neg\varphi \lor \psi \text{ (Rewriting of } \to\text{)} \qquad \varphi \oplus \psi \equiv (\varphi \lor \psi) \land \neg(\varphi \land \psi)$$

$$\varphi \leftrightarrow \psi \equiv \neg(\varphi \oplus \psi) \equiv \varphi \to \psi \land \psi \to \varphi \equiv (\varphi \land \psi) \lor (\neg\varphi \land \neg\psi)$$

I.B.3.v.b From $\to$ and $\bot$

One of the smallest set of operators that can recreate all others is $\to$ and $\bot$. In practice this is highly inefficient, but it allows for a very succinct definition of the logic's grammar. Because of this inefficiency, once an operand has been recreated we will use it instead of its definition using only $\to$ and $\bot$.

ℹ This is for your personal culture more than to be applied in this course.

$\boxed{\top}$ We define $\top \equiv \bot \to \bot$. As we can see by looking at the truth table of the implication (Figure I.1(e)), whenever the premise (left side of the $\to$) is $\bot$, the implication is true. So in particular, $\bot \to \bot$ is always true.

$\boxed{\neg}$ We define $\neg\varphi \equiv \varphi \to \bot$:

- whenever the premise is false, the implication is true;
- whenever the premise is true, the implication is only true if the conclusion is too, but $\bot$ is never true so here the implication is false.

Another way to connect this to the usual definition is to rewrite $\varphi \to \bot$ as $\neg\varphi \lor \bot \equiv \neg\varphi$. But although that allows to understand why it works, it is not a good way to define $\neg$ in this context.

$\boxed{\lor}$ We define $\varphi \lor \psi \equiv \neg\varphi \to \psi$. One way to understand (but not really prove) this definition is, as above, by rewriting the implication: $\neg\varphi \to \psi \equiv \neg\neg\varphi \lor \psi \equiv \varphi \lor \psi$. To prove the correctness of this definition semantically, one need to consider all cases:

- If φ is true, then $\neg\varphi$ is false and the implication is true.
- If φ is false, then the implication is only true when ψ is, which is the same semantics as the disjunction.

Note that the way we tend to interpret this implication by first looking at the premise and not even considering the conclusion if the premise is false is akin to the evaluation of boolean formulas in some programming languages called *lazy evaluation* (or *short circuit*): when evaluating $\varphi \lor \psi$, when φ is true ψ is not evaluated.

$\boxed{\land}$ We define $\varphi \land \psi \equiv \neg(\neg\varphi \lor \neg\psi)$: this follows from De Morgan's laws.

The other operators can be recreated from $\neg$, $\land$, and $\lor$, as mentioned in Section I.B.3.v.a.

I.B.3.vi Calculating in the Boolean algebra

Using the rules above allow to calculate in the boolean algebra in a similar way that we would in real algebra. The only additional principle to be added it the *substitution principle*: if $\varphi \equiv \psi$, then φ can be replaced by ψ in another expression.

Calculating in the boolean algebra is often more efficient that building the truth tables in order to prove equivalences. In these calculation, indicating what rule was used allows to make sure that no mistake is made. For example:

$$\begin{aligned}
(p \to q) \lor (p \to r) &\equiv (\neg p \lor q) \lor (\neg p \lor r) && [\text{Rewriting of } \to] \\
&\equiv \neg p \lor q \lor \neg p \lor r && [\text{Associativity of } \lor] \\
&\equiv \neg p \lor \neg p \lor q \lor r && [\text{Commutativity of } \lor] \\
&\equiv (\neg p \lor \neg p) \lor (q \lor r) && [\text{Associativity of } \lor] \\
&\equiv \neg p \lor (q \lor r) && [\text{Idempotence of } \lor] \\
(p \to q) \lor (p \to r) &\equiv p \to (q \lor r) && [\text{Rewriting of } \to]
\end{aligned}$$

Exercise I.3

Using the rules[a] of Boolean Algebra:

1. Prove that $\neg(p \to q)$ is logically equivalent to $p \land \neg q$.
2. Prove that $(p \to q) \land (p \to r)$ is logically equivalent to $p \to (q \land r)$.
3. Prove that $(p \land q) \to p$ is logically equivalent to $\top$.
4. Prove that $(p \to q) \land p \land \neg q$ is logically equivalent to $\bot$.
5. Prove that $((p \to q) \to p) \to p$ is logically equivalent to $\top$ (*Peirce's Law*).
6. Prove that $(p \lor (q \land \neg r)) \to p$ is logically equivalent to $p \lor \neg q \lor r$.

[a] Indicate which rule was used.

I.C Classifying formulas

I.C.1 Formula categories

Formulas can be classified into three main categories:

- Formulas that are true regardless of the truth value of the propositional variables, called *tautologies*. These formulas are logically equivalent to $\top$, and their truth table (the last column) contains only $\top$. For example, the excluded middle $p \lor \neg p$ is a tautology; so is $(p \land q) \to p$.

- Formulas that are false regardless of the truth value of the propositional variables, called *contradictions*. These formulas are logically equivalent to $\bot$, and their truth table (the last column) contains only $\bot$. For example, the contradiction $p \land \neg p$ is a contradiction; so is $(p \to q) \land p \land \neg q$.

The statement "the contradiction $p \land \neg p$ is a contradiction" is a tautology!

- All other formulas, that are neither tautologies nor contradictions are called *contingent* formulas. The term contingent refers to the fact that the truth value of the formula is contingent (i.e. depends) on the truth value of the propositional variables. For a formula to be contingent, there needs to be at least a valuation of propositional variables that makes the formula true, and one that makes the formula false. In the truth table (the last column), that corresponds to having at least a line $\top$ and a line

$\bot$. For example, $(p \to q) \land \neg q$ is contingent: it is true for $p = q = \bot$, and false for $p = \top$ and $q = \bot$.

I.C.2 Classifying formulas, in practice

One way to classify a formula is to build the truth table. It is however often inefficient: for n propositional variables, the truth table has 2^n lines.

There are better methods for all three cases:

- To prove that a formula φ is a tautology, it is sufficient to use propositional calculus and show that $\varphi \equiv \top$.
- To prove that a formula φ is a contradiction, it is sufficient to use propositional calculus and show that $\varphi \equiv \bot$.
- To prove that a formula φ is contingent, it is sufficient to exhibit a valuation that makes the formula $\top$, and another that makes the formula $\bot$.

For contingent formulas, finding these valuations may also be done without building the truth table, but through a bit of guesswork, based on the operators in the formula. We work in a top-down approach, similar to the decomposition into sub-formulas to build the truth table. If the formula is…

$\boxed{p}$ For atomic propositions, choose $\top$ to make it true and $\bot$ to make it false. Whenever a choice of truth value is made for an atomic proposition, keep it in mind: another later choice cannot contradict it.

$\boxed{\neg\varphi}$

- To make a negation true, the operand have to be false: so try to make φ false.
- To make a negation false, the operand have to be true: so try to make φ true.

$\boxed{\varphi \land \psi}$

- To make a conjunction true, both operands have to be true: so try to make both φ and ψ true.
- To make a conjunction false, only one operand needs to be false: choose one between φ and ψ and try to make it false.

$\boxed{\varphi \lor \psi}$

- To make a disjunction true, only one operand needs to be true: choose one between φ and ψ and try to make it true.
- To make a disjunction false, both operands have to be false: so try to make both φ and ψ false.

$\boxed{\varphi \to \psi}$

- An easy way to make an implication true is to try to make the left handside false: try to make φ false. If that fails, try to make ψ true.
- The only way to make an implication false is to try to make the left handside true and the right handside false: try to make φ true and ψ false.

If a contradiction appears (for example p should be both $\top$ and $\bot$), then either come back to the last choice that was made (for example when trying to make a disjunction true) and choose the other way. Note that although this technique is more efficient in practice, it is not guaranteed to always be, and you might end up trying all combinations anyway.

✍ "(resp. …)" indicates that this paragraph is actually two paragraphs: one without the parentheses, and one where "true" is replaced by "false", "contradiction" by "tautology", etc.

It is also possible that after trying all choices, it seems impossible to make a given formula true (resp. false). Unlike exercises, which may explicitly ask to prove that a formula belongs to a certain category, you may have started this procedure without the knowledge that it was contingent. In this case, it means the formula seems to be a contradiction (resp. tautology). But, as this trial-and-error technique is not a formal exploration of all cases (which would be the truth table), it remains to be proved that the formula is a contradiction (resp. tautology), usually by reducing it to $\bot$ (resp. $\bot$) through calculus.

I.C.2.i Example: $\varphi = (p \vee q) \to (p \wedge \neg r \wedge q)$

Make φ true. It's an implication, so having $(p \vee q)$ be false is sufficient. In order to do that, set both p and q to $\bot$. There is no constraint on r. So $p, q = \bot$ and say $r = \top$ makes φ true.

Make φ false. It's an implication, so we need $p \vee q$ to be true while $p \wedge \neg r \wedge q$ is false. For $p \vee q$, we can chose one of them to be true, and see if it works, for example set p to $\top$. Now for $(p \wedge \neg r \wedge q)$, we can chose one of the operands to be false, but not p because we already assumed it to be $\top$. So if we choose $\neg r$ to be false, that means r is $\top$ (and q does not matter). So for example $p, r = \top$ and $q = \bot$ makes φ false.

As we have exhibited a valuation that makes φ true and another one that makes φ false, we can classify φ as a contingent formula.

I.C.2.ii Example: $\psi = (p \wedge q) \to (p \vee q)$

Make ψ true. It's an implication, so having $p \wedge q$ be false is sufficient. For example by having $p, q = \bot$, formula ψ is true.

Make ψ false. It's an implication, so we need $p \wedge q$ to be true while $p \vee q$ is false. So we need to set p and q to $\top$. To make $p \vee q$, they must both be false, but we have already set them $\top$, to it seems impossible to find a valuation that makes the formula false!

As we succeeded in finding a valuation that makes ψ true, but not a valuation that makes ψ false, it seems that ψ is a tautology. It remains to be proved, for example using a reduction to $\top$:

$$\begin{array}{rcll}
\psi & = & (p \wedge q) \to (p \vee q) & \\
 & \equiv & \neg(p \wedge q) \vee (p \vee q) & [\text{Rewriting of } \to] \\
 & \equiv & (\neg p \vee \neg q) \vee (p \vee q) & [\text{De Morgan's Law}] \\
 & \equiv & \neg p \vee \neg q \vee p \vee q & [\text{Associativity of } \vee] \\
 & \equiv & p \vee \neg p \vee q \vee \neg q & [\text{Commutativity of } \vee] \\
 & \equiv & (p \vee \neg p) \vee (q \vee \neg q) & [\text{Associativity of } \vee] \\
 & \equiv & \top \vee \top & [\text{Excluded middle}] \\
\psi & \equiv & \top & [\top \text{ is absorbing for } \vee]
\end{array}$$

We also could have used a truth table to prove the same fact (for two variables it is still relatively small):

p	q	$p \wedge q$	$p \vee q$	$(p \wedge q) \to (p \vee q)$
$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\top$

Exercise I.4

1. Prove that the following formulas are contingent:
   a. $p \vee (q \wedge (r \to p))$
   b. $(p \leftrightarrow r) \wedge (q \to \neg r)$
2. Prove that the following formulas are tautologies:
   a. $((p \to q) \wedge p) \to q$
   b. $(p \to q) \vee (q \to p)$
   c. $((p \to q) \wedge \neg q) \to \neg p$
3. Prove that the following formulas are contradictions:
   a. $(p \oplus q) \wedge (p \leftrightarrow q)$
   b. $(q \to p) \wedge (p \to r) \wedge q \wedge \neg r$
4. Classify the following formulas as tautologies, contradictions, or contingent formulas:
   a. $(p \to (r \vee \neg p)) \to q$
   b. $(p \to r) \vee (\neg p \to q)$
   c. $p \to ((r \vee \neg p) \to q)$

I.C.3 Satisfiability

There is a fourth category of formulas that is of interest: *satisfiable* formulas. A satisfiable formula is a formula that is not a contradiction; otherwise said it is either contingent or a tautology. That means at least for one particular valuation, the formula is true:there is at least a line in the truth table that is $\top$. So to prove that a formula is satisfiable, one has to exhibit a valuation that makes the formula $\top$.

ℹ This is for your personal culture more than to be applied in this course.

I.C.3.i The satisfiability problem: SAT

The satisfiability problem, or SAT for short, is asking, for a formula φ, whether it is satisfiable. It is an important problem in practice because lost of situations can be understood as an instance of a SAT problem. For example, the fact for a critical system to be in an error state (think: failure of a spacecraft) can be modeled into a propositional logic formula, albeit with lots of propositional variables. If this formula is satisfiable, that means there is the possibility of an error, which should be corrected before the system is actually launched.

Given the number of variables, this problem ought to be solved by a computer program rather than a human. From what we saw above, we can deduct two ways of solving this problem:

- Build the truth table, which amounts to testing all 2^n possibilities when there are n variables; this is the *brute-force* approach. This grows very fast! In critical systems, 100 variables would actually be a small number.To put things in perspective, there are about 2^{80} atoms in the universe.

- Another way would be to *guess* a valuation that does satisfy the formula. In this case, that would mean only n guesses, one for each variable. After the guesses, we just have to verify that the formula is indeed true with this valuation (it takes about as much time as reading the formula). This method is *non deterministic*: it requires the ability to guess. Although current actual computers cannot guess, theoretical computers with this ability can be conceived; and quantum computing may one day provide real computers that can make such guesses.

There are better ways to solve SAT without guessing, but they are only better in practice, and in some (very particular) instances may still require a full exploration of the whole 2^n cases. So guessing is more efficient that any technique that we know.

I.C.3.ii The NP class

Problems like SAT, that can be solved by guessing a “small” number of times are said to be *in NP*, which stands for *Non deterministic Polynomial.* Formally: the number of guesses and the checking time must be smaller than a polynomial on the size of the input. In the case of SAT, the size of the input is the length of the formula l, which is bigger than the number of variables n, and we need n guesses and l calculation steps to check the guess. So SAT is an NP problem.

Furthermore, every problem that requires guesses can actually be interpreted as a satisfiability problem for a given formula: we say that SAT is *NP-complete.*

So far, it is unknown whether SAT could be solved with a polynomial number of instructions but without guesses (a.k.a a deterministic polynomial-time algorithm). Answering this question would mean giving an answer to the “Is $\mathsf{P} = \mathsf{NP}$?” question (and winning a million dollars in the process). If there is a deterministic polynomial-time algorithm, that would mean that every problem that can be reduced to SAT (such as checking for potential error in a critical system) could be solved efficiently in all cases. It would also have less desirable consequences: for example, many currently used cryptographic protocols rely on the fact that it is not possible to guess a solution, and using a brute-force approach would take too long. Having a polynomial-time procedure to find a solution would wreak havoc on security systems.

It is, however, unlikely that this is the case, and most computer scientists believe that $\mathsf{P} \neq \mathsf{NP}$ But until there is a proof of that, every result that uses that must start with a caveat. It is actually not uncommon to find in the literature results that start with the phrase “Assuming $\mathsf{P} \neq \mathsf{NP}$...” and proofs that end by “if this is false then $\mathsf{P} = \mathsf{NP}$”.

Chapter II
First-order logic (FO)

Chapter contents

II.A Predicates **20**

II.A.1 Interpretations and semantics 20

II.A.2 Notations and vocabulary 21

II.A.3 Predicates and boolean operators 21

II.B Quantifiers **21**

II.B.1 Universal quantifier: Always 21

II.B.2 Existential quantifier: Sometimes 22

II.B.3 Negated quantifiers: Never, Not always 22

II.B.4 Finding the truth value of quantified formula 23

II.B.5 Quantification on several variables 24

II.B.6 Variable scope, freeness and boundedness 25

II.B.7 Remarks 25

II.B.7.i A new look on the satisfiability problem 25

II.B.7.ii Abbreviations for easier maths 26

II.B.7.iii Second- (or more) -order logic 27

II.C Equivalence in first-order logic **27**

II.C.1 Syntactical calculation 27

II.C.1.i Calculation rules 27

II.C.1.ii Quantifier alternation 29

II.C.2 Semantic proofs 29

II.C.2.i Proving non-equivalence 29

II.C.2.ii Proving equivalence 30

II.A Predicates

Propositions, that were the focus of Chapter I are weak, in the sense that they can express very little. Since they can only represent something that is definitely true or false, there is no place for the nuance of "it depends". For example "$x^2 - 2x + 5 = 9$" is not a proposition because *it depends on x*: this statement falls out of the scope of propositional logics, but it is still worth considering.

Since the truth value of the statement depends on the value of x, we can write it as a function R that takes a value x (a real number) and assigns the truth value $\top$ or $\bot$: this is called a *predicate*. For example $R(42)$ is $\bot$, $R(1+\sqrt{5})$ is $\top$, and $R(\pi)$ is $\bot$.

Remark that as soon as a value is given for x, $R(x)$ become a proposition, and can be used as such with the boolean algebra operators.

Definition: Predicate

A predicate is a function that associates to n values in a *domain* a truth value in $\{\top, \bot\}$.

To use a programmer's point of view, a predicate is a function that takes some arguments and returns a boolean value.

II.A.1 Interpretations and semantics

Providing the domain, which is what kind of values can be used for the arguments, is essential to actually evaluating a predicate. Technically, providing the domain is not sufficient: one has to give a meaning to all the operators in the predicate. In this book, and unless otherwise specified, we will assume that mathematical symbols are used with their usual meaning: $+$ means addition, $\times$ means multiplication. And when dealing with numbers, we will use the usual order predicates $<$ (strictly less than), $\leq$ (less than or equal to), $\geq$ (greater than or equal to), $>$ (strictly greater than).

They are indeed predicates: given two values, these operators may be true or false.

Also note that, in its most abstract definition, we do not need to provide what exactly is the meaning of a predicate to be able to write formulas using it, for example $P(x) \wedge P(y) \rightarrow Q(x, y)$ is a formula that is *syntactically* correct. But to give its *semantics*, i.e. its meaning, one need to provide a domain for x and y, the definition of predicates P and Q, as well as the definition of any symbols used in the definition o P and Q. Together, they are called an *interpretation* of the formula.

It may be the case that a predicate is not defined by a mathematical formula but by a description, although it is less formal. For example, let $D(n, k)$ be the predicate over integers that is true when n is a multiple of k; in this case $D(-42, 7)$ is $\top$ and $D(12, 5)$ is $\bot$, for instance.

In most cases, the informal definition is only given as intuition to the reader, and the formal definition is still present: let $P(a, b, c)$ be the predicate over integers that is true when (a, b, c) is a Pythagorean triple, i.e. when $a^2 + b^2 = c^2$. In this case $P(1, 2, 3)$ is $\bot$ because $1^2 + 2^2 = 1 + 4 = 5 \neq 9 = 3^2$; but $P(3, 4, 5)$ is $\top$ because $3^2 + 4^2 = 9 + 16 = 25 = 5^2$.

II.A.2 Notations and vocabulary

It is customary to denote predicates using a capital letter. P, Q, and R, are mostly used (in similarity with p, q, r for propositional variables), but other letters that relate more to the meaning of the predicate can also be used (for example, above D is used because $D(n, k)$ is true when k *divides* n).

Section VII.A.1 is devoted to the concept of divisibility.

In addition, when the domain is known, variables used in predicates usually follow the cultural habits of the domain: so x, y, z would be used for real numbers, n, k, p, m for integers, etc.

Predicates are categorized by the number of variables they use.

Arity of a predicate

- A predicate with no variable is called a *proposition*.
- A predicate with a single variable is called a *unary* predicate.
- A predicate with a two variables is called a *binary* predicate.
- A predicate with a three variables is called a *ternary* predicate.
- A predicate with n variables is called an *n-ary* predicate.

So in the predicates given above, R is unary, D is binary, and P is ternary.

II.A.3 Predicates and boolean operators

As remarked above, predicates can be used as atomic formulas (like propositions). Truth value of the whole formula depends on the truth value of all predicates and the rules (truth tables) for each operators (see Figure I.1). For example, over the domain of integers, with D and P defined as above:

- $P(a, b, c) \vee D(c, a)$ is true for $a = 1$, $b = 2$, $c = 3$ (because 3 is a multiple of 1), but false for $a = 7$, $b = 8$, $c = 9$ (it is not a Pythagorean triple and 9 is not a multiple of 7).

- $(P(a, b, c) \wedge D(a, k) \wedge D(b, k)) \rightarrow D(c, k)$ is actually true for all values a, b, c, k: this formula states that if a, b, c is a Pythagorean triple where a and b are multiple of the same number k, then so is c.

II.B Quantifiers

II.B.1 Universal quantifier: Always

In the example above, it does not really matter what the values of a, b, c, k are: the formula will *always* be true. So we can actually evaluate the truth value of this formula even before values for the variables are assigned. It is actually the goal of lots of mathematical statements: mathematics are about general truths, that hold "for every x", or, in the context of predicates, regardless of the value of the variables in the predicate.

This is denoted using a *universal quantifier*, written by the symbol ∀, which is read "*for all*". This symbol is followed by the variable it quantifies.

For example: $\forall x, P(x)$ is a formula which is true if *for all* value x in the domain, $P(x)$ is true. This formula is actually in itself a proposition: it does not depend on the value of variable x, which is *bound* by the quantifier.

A statement φ is *stronger* than ψ if φ implies ψ, but not the other way around.

The universally quantified statement is a *stronger* statement than having to say individually that all versions of the predicate are true. For example, let $P(x)$ be the predicate defined on real numbers by $x+1>x$. The proposition $\forall x, P(x)$ is true. Saying so is stronger than saying $0+1>0$ is a true proposition, and so is $\frac{1}{2}+1>\frac{1}{2}$, and $\pi+1>\pi\ldots$ (that has no end!).

II.B.2 Existential quantifier: Sometimes

The satisfiability problem is described in Section I.C.3.

As in the satisfiability problem, it is interesting to know when a predicate is true for some values. One does not have to actually know what are the values that make a predicate true to use the fact that is sometimes true. In some sense, this is what is done when we define numbers such as $\sqrt{2}$: its exact value is not given, but this number that is positive and such that its square is two exists (somewhere between 1.4 and 1.5, to give a very broad approximation); we just name it $\sqrt{2}$ to facilitate notations.

This is denoted using a *existential quantifier*, written by the symbol $\exists$, which is read as "*there exists*". This symbol is followed by the variable it quantifies.

For example: $\exists x, P(x)$ is a formula, which is true if *there exists* a value x in the domain such that $P(x)$ is true. This formula is actually in itself a proposition: it does not depend on the variable, which is *bound* by the quantifier.

The existentially quantified statement is a *weaker* statement than actually giving a value for which the predicate is true. For example, let $P(x)$ be the predicate $2x+3=0$. The proposition $\exists x, P(x)$ is true over the reals. Saying so gives less information than saying that $2\times\left(-\frac{3}{2}\right)+3=0$, which actually provides the value for x.

II.B.3 Negated quantifiers: Never, Not always

There are two ways to express that something is never true.

- We can say that it is always false: $\forall x, \neg P(x)$.
- We can say that there is no way to make it true: $\neg\exists x, P(x)$.

Similarly, there are two ways to express that something is not always true.

- We can say that it is not the case that it is always true: $\neg\forall x, P(x)$.
- We can say that there at least one way to make it false: $\exists x, \neg P(x)$.

That gives us a version of De Morgan's laws for quantifiers.

De Morgan's Laws for quantifiers

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

On finite domains, we can relate that to the De Mogan's laws for conjunction and disjunction as follows. Assume the domain contains values $\{x_1,\ldots,x_n\}$. Formula $\forall x, P(x)$ is, in that case, the same as $P(x_1)\wedge\cdots\wedge P(x_n)$: it must be true for all the values $x_1,\ldots,x_n$.

On the other hand, $\exists x, P(x)$ is, in that case, the same as $P(x_1) \vee \cdots \vee P(x_n)$: it must be true for at least on of the the values $x_1, \ldots, x_n$. So we can write:

$$\neg\forall x, P(x) \equiv \neg(P(x_1) \wedge \cdots \wedge P(x_n)) \equiv \neg P(x_1) \vee \cdots \vee \neg P(x_n) \equiv \exists x, \neg P(x)$$

$$\neg\exists x, P(x) \equiv \neg(P(x_1) \vee \cdots \vee P(x_n)) \equiv \neg P(x_1) \wedge \cdots \wedge \neg P(x_n) \equiv \forall x, \neg P(x)$$

II.B.4 Finding the truth value of quantified formula

Given an interpretation (domain and meaning of the predicates), the truth value of a quantified formula can be exhibited through a proof. In some sense, calculating the truth value of a propositional formula is also a proof, albeit being a purely calculating one using the truth tables of the operators. In the case of first order logic, the proof must take into account the quantifiers. and the structure of the proof will change depending on what is being proved.

- To prove that $\exists x, P(x)$ is true, it is sufficient to exhibit one value a in the domain and show that $P(a)$ is true. The choice of a is completely up to the proof-writer, and is made keeping in mind what P is to ensure that it will actually be true. For example, to prove that $\exists x, x^2 = x$ over the reals, one can chose $x = 1$ and show that $1^2 = 1$, or do the same with 0. But it would not work with another value: so if you chose $x = 42$, you won't be able to prove this statement.

- To prove that $\forall x, P(x)$ is true, it must be shown that $P(x)$ holds for any value of x. Such proofs usually start with the words "Let x be an element of the domain". For example to prove $\forall x, x^2 + 1 > x$ over reals, the proof would start by "Let x be a real number". When such a value is chosen, we have no say in the choice of x (we say it is chosen by "the universe"), and cannot make careless assumptions about it.

- Proving that $\forall x, P(x)$ is false, is like proving that $\neg\forall x, P(x) \equiv \exists x, \neg P(x)$ is true, so it is sufficient to exhibit one value a in the domain and show that $P(a)$ is false. This value a is then called a *counterexample*.

- Proving that $\exists x, P(x)$ is false, is like proving that $\neg\exists x, P(x) \equiv \forall x, \neg P(x)$ is true, so it must be shown that $P(x)$ does not hold for any value of x. As in the above case, the proof would start with "Let x be an element of the domain"; in this case it must then be proved that $P(x)$ is false.

In Section III.C.4.v we discuss how to prove this kind of statement using *careful* assumptions in a proof by cases.

Exercise II.1

In the following exercises, you have to prove your claim (or at least give the proof structure).

1. Let $P(x)$ be the predicate "$x^2 \geq 0$". What is the truth value of $\forall x, P(x)$...
   a. when the domain is the integers?
   b. when the domain is the reals?
   c. when the domain is the complex numbers?
2. Let $Q(x)$ be the predicate "$4x - 3 = 0$". What is the truth value of $\exists x, Q(x)$...
   a. when the domain is the integers?
   b. when the domain is the reals?

II.B.5 Quantification on several variables

There can be more than one quantified variable, hence more than one quantifier per formula. For example, with a single binary predicate, we can have the following formulas:

$$\forall x, \forall y, Q(x,y) \qquad \exists x, \forall y, Q(x,y) \qquad \forall x, \exists y, Q(x,y) \qquad \exists x, \exists y, Q(x,y)$$

$$\forall y, \forall x, Q(x,y) \qquad \exists y, \forall x, Q(x,y) \qquad \forall y, \exists x, Q(x,y) \qquad \exists y, \exists x, Q(x,y)$$

When the quantifier is the same, the order does not actually matter:

- $\forall x, \forall y, Q(x,y)$ is the same as $\forall y, \forall x, Q(x,y)$ can be abbreviated $\forall x, y, Q(x,y)$
- $\exists x, \exists y, Q(x,y)$ is the same as $\exists y, \exists x, Q(x,y)$ and can be abbreviated $\exists x, y, Q(x,y)$

This is because in the proof of these statements, the choices are always made by the same "person": the proof-writer (to prove that $\exists x, y, Q(x,y)$ is true or that $\forall x, y, Q(x,y)$ is false), or the universe (to prove that $\forall x, y, Q(x,y)$ is true or that $\exists x, y, Q(x,y)$ is false). the order in which the choices are performed is therefore irrelevant.

This is not the case when the quantifiers are different. To illustrate this, let us consider four cases using predicate $Q(x,y)$ being $x + y = 0$ over the integers.

$\boxed{\forall x, \forall y, Q(x,y)}$ This formula is false. We can prove it using the counterexample $x = 42$ and $y = 0$.

$\boxed{\exists x, \forall y, Q(x,y)}$ This formula is false: let x be an integer; then choosing $y = x + 1$ we have $x + y = x + x + 1 = 2x + 1$ which can't be 0 for any integer.

$\boxed{\forall x, \exists y, Q(x,y)}$ This formula is true: let x be an integer; then choosing $y = -x$ we have $x + y = x - x = 0$.

$\boxed{\exists x, \exists y, Q(x,y)}$ This formula is true: choosing $x = y = 0$ yields $x + y = 0 + 0 = 0$.

Note that in this particular case, x and y have the same role in the predicate so $\exists y, \forall x, Q(x,y)$ and $\exists x, \forall y, Q(x,y)$ are similar; this is not the case in general!

In the example above, the choice of y after the choice of x allowed the proof to work: the value of y depends on the value of x. In the proof of $\exists x, \forall y, Q(x,y)$, since we claim that it is false, we are playing the role of the universe in a proof trying to prove that it is true: for any choice of x by the proof-writer, the universe could have chosen a value of y that makes $Q(x,y)$ false.

Exercise II.2

Let $R(x,y)$ be the predicate "$x \times y = 1$".

1. What is the truth value of $\exists x, \exists y, R(x,y)\ldots$
   a. when the domain is the integers?
   b. when the domain is the reals?
2. What is the truth value of $\exists x, \forall y, R(x,y)\ldots$
   a. when the domain is the integers?
   b. when the domain is the reals?
3. What is the truth value of $\forall x, \exists y, R(x,y)\ldots$
   a. when the domain is the integers?

b. when the domain is the reals?

4. What is the truth value of $\forall x, \forall y, R(x, y) \ldots$
   a. when the domain is the integers?
   b. when the domain is the reals?

II.B.6 Variable scope, freeness and boundedness

Variable that are not bound by a quantifier are called *free* variables. A formula with free variables is not a proposition, but it can be seen as a predicate on these variables.

A quantifier only binds in its *scope*: by default, the scope of a quantifier is everything that appears after the quantifier; but it can be restricted using parentheses. For example:

- $\forall x, P(x) \rightarrow Q(x, y)$ has free variable y only; could be viewed as a unary predicate $R(y)$.
- $(\forall x, P(x)) \wedge Q(x, y)$ has free variables x and y, because outside of the parentheses x is not bound anymore; could be viewed as a binary predicate $R'(x, y)$.

The concept of scope is similar to the one used in programming: variables declared in a function only exist until the end of this function's code; a variable defined inside a loop only exists within the loop.

A bound variable can be renamed throughout its scope (this is called *α-renaming*) without affecting the truth value of the formula: the change is purely syntactic. For example, we can rename the bound x into z in the above example:

$$(\forall x, P(x)) \wedge Q(x, y) \overset{\alpha}{\rightsquigarrow} (\forall z, P(z)) \wedge Q(x, y).$$

This renaming is useful in this case so that human readers distinguish better the scope of the quantified variable, but it doesn't change the definition of $R'(x, y)$.

Exercise II.3

Determine the free variables in the following formulas:

1. $P(x, y) \wedge (\forall z, Q(z, y))$
2. $\forall x, Q(x) \rightarrow (\exists y, R(x, y))$
3. $(\forall y, Q(y)) \wedge \exists x, P(x, y)$

II.B.7 Remarks

II.B.7.i A new look on the satisfiability problem

A boolean formula φ using free variables $p_1, \ldots, p_n$ is actually a n-ary predicate interpreted over the domain $\{\top, \bot\}$, and using the semantics of Boolean algebra for operators. This point of view shines a new light on the problem of classifying formulas (see Section I.C). Namely:

- A formula is satisfiable if *for some valuation* its truth value is $\top$, so φ is satisfiable if $\exists p_1, \ldots, p_n, \varphi(p_1, \ldots, p_n)$ is true.

- A formula is a tautology if *for all valuation* its truth value is $\top$, so φ is a tautology if $\forall p_1, \ldots, p_n, \varphi(p_1, \ldots, p_n)$ is true.

- A formula is a contradiction if *for all valuation* its truth value is $\bot$, so φ is a contradiction if $\forall p_1, \ldots, p_n, \neg\varphi(p_1, \ldots, p_n)$ is true; or $\neg\exists p_1, \ldots, p_n, \varphi(p_1, \ldots, p_n)$ is true, i.e. $\exists p_1, \ldots, p_n, \varphi(p_1, \ldots, p_n)$ is false.

The proof structure that is used when classifying propositional formulas simply follows from the structure of the quantifiers in the corresponding first-order formula. For example, to prove that a formula $\varphi(p_1, \ldots, p_n)$ is contingent, it must be proved that it is not a tautology and not a contradiction: It is not a tautology if $\forall p_1, \ldots, p_n, \varphi(p_1, \ldots, p_n)$ is false. This is proved by finding a counter-example, *i.e.* a value in the domain $\{\top, \bot\}$ for each variable $p_1, \ldots, p_n$ (which is called a valuation) such that $\varphi(p_1, \ldots, p_n)$ is false. It is not a contradiction if $\forall p_1, \ldots, p_n, \neg\varphi(p_1, \ldots, p_n)$ is false (or, equivalently, if $\exists p_1, \ldots, p_n, \varphi(p_1, \ldots, p_n)$ is true). This is also proved using a counter-example: a value in the domain $\{\top, \bot\}$ for each variable $p_1, \ldots, p_n$ such that $\varphi(p_1, \ldots, p_n)$ is true.

II.B.7.ii Abbreviations for easier maths

Quantifiers are used, sometimes in hiding, throughout mathematical statements. To simplify the writing and remain closer to what the statement would look in natural language, some abbreviations are often used. It is however necessary to keep in mind what is the exact meaning of these abbreviations, as it guides the proof structure.

II.B.7.ii.a Restricting the domain

It is useful to restrict the domain to which the quantifiers actually apply: instead of considering the full domain on which a universal quantifier must hold, or from which a value can be chosen for a existential quantifier, only a part of it is used.

Parts of the domain are called *sets*, a notion fully developed in Chapter IV.

This restriction is denoted using the symbol $\in$, which is read as "in" or "belonging to": $\forall x \in A \ldots$ and $\exists x \in A$ mean the domain of this variable is restricted to A. Formally A is given by a predicate $A(x)$ that is true when x is in this sub-part A. Then the meaning of this restricted domain is as follows:

- $\forall x \in A, P(x)$ is actually $\forall x, A(x) \to P(x)$.

- $\exists x \in A, P(x)$ is actually $\exists x, A(x) \wedge P(x)$.

Remark that it differs depending on the type of quantifier, but that De Morgan's laws still work with these "restricted domain quantifiers". It can be proved using the rules of propositional calculus and De Morgan's laws for quantifiers:

$$\begin{aligned}
\neg\forall x \in A, P(x) &\equiv \neg\forall x, A(x) \to P(x) && \text{[Expansion of the abbreviation]} \\
&\equiv \exists x, \neg(A(x) \to P(x)) && \text{[De Morgan's Law for } \forall\text{]} \\
&\equiv \exists x, \neg(\neg A(x) \vee P(x)) && \text{[Rewriting of } \to\text{]} \\
&\equiv \exists x, \neg\neg A(x) \wedge \neg P(x) && \text{[De Morgan's Law]} \\
&\equiv \exists x, A(x) \wedge \neg P(x) && \text{[Double negation elimination]} \\
\neg\forall x \in A, P(x) &\equiv \exists x \in A, \neg P(x) && \text{[Contraction of the abbreviation]}
\end{aligned}$$

II.B.7.ii.b Uniqueness

To denote that a *single* element of the domain makes the predicate true, the symbol $\exists!$ is used. It is read as "there exists a unique". The formula $\exists! x, P(x)$ is a shorthand for $\exists x, P(x) \wedge (\forall y, P(y) \rightarrow x = y)$: there is an element x that satisfies the predicate, and any element of the domain that satisfies the predicate is x.

II.B.7.iii Second- (or more) -order logic

Propositional logics is logics without any quantifiers. In first-order logic, there is quantification on variables of the domain, that are used in predicates.

We can go one step further, and in second-order logic, we can quantify on variables of the *predicates*, to write formulas such as this one: $\forall P, \exists Q, \forall x, P(x) \rightarrow \exists y, Q(x, y)$. So technically, the first-order formula $\forall x, P(x) \rightarrow \exists y, Q(x, y)$ is a second-order predicate with free variables P, and Q, which are instantiated when an interpretation is provided.

This is for your personal culture more than to be applied in this course.

But why stop here? In third-order logic, there can also be quantification over second order predicates. This could go on; if it goes as high as we want, we obtain *higher-order logic* (HOL).

In practice, above second-order, the properties of n-order and HOL are somewhat similar, and third-order and above are not studied individually. Second order is however worth studying on its own, in particular the fragment that only allows unary predicates, which is called *Monadic Second Order*. Unfortunately, discussing MSO in more depth is way beyond the scope of this course.

II.C Equivalence in first-order logic

II.C.1 Syntactical calculation

In the same manner that syntactical calculation is possible through the algebraic rules of propositional calculus, it is possible to perform syntactical calculations in first-order logic. The fact that it is syntactical means that:

- There is no domain set.
- We don't get to see the meaning of the predicates.
- Hence there is no functions (operators) in there, except the logical operators.

Equivalences that are thus proved are valid regardless of the interpretation.

II.C.1.i Calculation rules

Since first-order is built on propositional logic, the rules of calculations for FO also uses the rules of propositional calculus. In order to deal with the quantifiers, calculus for FO also uses:

- De Morgan's Laws for quantifiers ($\neg\forall \equiv \exists\neg$ and $\neg\exists \equiv \forall\neg$)
- Rules regarding the scope of the bounded variables.
- Rules regarding how quantifiers interact with other operators.

The first set of rules regards the change of the scope of a quantifier. As these rules work both ways, and the scope can be extended or reduced. A scope reduction, which can be reduced to nothing, eliminating the quantifier is only possible if the quantifier was actually useless: this is why these rules are known as the *null quantification* rules. For a scope extension to be allowed (i.e. to be without effect on the truth value), this extension must not *capture* a variable: that means that the variable whose scope is extended must not have appeared free beforehand in its new scope. Note that if the variable appeared but is bound, it can be α-renamed, and therefore not appear anymore. Formally:

Null quantification rules

Assume φ is a formula where x is not a free variable. Then:

$$\forall x, \varphi \equiv \varphi \qquad \forall x, (P(x) \vee \varphi) \equiv (\forall x, P(x)) \vee \varphi$$

$$\exists x, \varphi \equiv \varphi \qquad \exists x, (P(x) \wedge \varphi) \equiv (\exists x, P(x)) \wedge \varphi$$

The second set of rules regards the interaction of quantifiers with the conjunction and disjunction operators. As previously noted, on a finite domain, a universal quantifier can be seen as a conjunction, while an existential quantifier can be seen as a disjunction. It is therefore not a surprise that universal quantifier behaves well with conjunction and universal quantifier with disjunction, even for infinite domains.

Quantifier distribution

$$(\forall x, P(x)) \wedge (\forall x, Q(x)) \equiv \forall x, (P(x) \wedge Q(x))$$

$$(\exists x, P(x)) \vee (\exists x, Q(x)) \equiv \exists x, (P(x) \vee Q(x))$$

In addition, syntactical rules can be used as well (such as $\exists x \exists y$ rewritten as $\exists x, y$).

All together, these rules are used to prove equivalences as in algebra. For readability, we indicate which rule was used in each step.

For example, we can show that $\forall x, (P(x) \to \neg\exists y, Q(y)) \equiv \forall x, y, \neg P(x) \vee \neg Q(y)$.

$$\begin{aligned}
\forall x, (P(x) \to \neg\exists y, Q(y)) &\equiv \forall x, (\neg P(x) \vee \neg\exists y, Q(y)) && \text{[Rewriting of } \to\text{]}\\
&\equiv \forall x, (\neg P(x) \vee \forall y, \neg Q(y)) && \text{[De Morgan's Law for } \exists\text{]}\\
&\equiv \forall x, ((\forall y, \neg Q(y)) \vee \neg P(x)) && \text{[Commutativity of } \vee\text{]}\\
&\equiv \forall x, (\forall y, (\neg Q(y) \vee \neg P(x))) && \text{[Null quantification]}\\
&\equiv \forall x, (\forall y, (\neg P(x) \vee \neg Q(y))) && \text{[Commutativity of } \vee\text{]}\\
\forall x, (P(x) \to \neg\exists y, Q(y)) &\equiv \forall x, y, (\neg P(x) \vee \neg Q(y))) && \text{[Syntactical shortcut]}
\end{aligned}$$

Exercise II.4

Assume φ is a formula where x is not a free variable. Prove the following equivalences using the rules[a] of first-order calculation:

1. $(\forall x, P(x)) \wedge \varphi \equiv \forall x, (P(x) \wedge \varphi)$
2. $(\exists x, P(x)) \vee \varphi \equiv \exists x, (P(x) \vee \varphi)$
3. $\forall x, (\varphi \to P(x)) \equiv \varphi \to (\forall x, P(x))$
4. $\exists x, (\varphi \to P(x)) \equiv \varphi \to (\exists x, P(x))$

5. $\forall x, (P(x) \to \varphi) \equiv (\exists x, P(x)) \to \varphi$
6. $\exists x, (P(x) \to \varphi) \equiv (\forall x, P(x)) \to \varphi$
7. $\neg\forall x, (P(x) \to Q(x)) \equiv \exists x, P(x) \wedge \neg Q(x)$

Remark: Once you have proved that an equivalence holds, it can be used in the following ones.

[a] Indicate which rule was used.

II.C.1.ii Quantifier alternation

This is for your personal culture more than to be applied in this course.

Using the equivalences of Exercise II.4, we can see that it is possible to use these rules (and composite rules) to put any first-order formula in a given form. It is done as follows:

- Put all quantifiers to the front. Using α-renaming, we can make it so all the quantifiers have distinct variables.
- Pushed all negations to the inside (at least after all quantifiers), using De Morgan's law as necessary.
- Gather (syntactically) similar quantifiers that are together.

The end result is a formula of the following form, called *prenex normal form*:

$$\exists x_{1,1}, \ldots, x_{1,n_1}, \forall y_{1,1}, \ldots, y_{1,p_1}, \exists x_{2,1}, \ldots, x_{2,n_2},$$

$$\forall y_{2,1}, \ldots, y_{2,p_2} \cdots \exists x_{k,1}, \ldots, x_{2,n_k}, \forall y_{k,1}, \ldots, y_{k,p_k},$$

$$P(x_{1,1} \ldots, x_{1,n_1}, y_{1,1}, \ldots, x_{k,1}, \ldots, x_{2,n_k}, y_{k,1}, \ldots, y_{k,p_k})$$

Not that it is possible that the first quantifier is actually a $\forall$, or the last a $\exists$.

The number k in the prenex normal form is called the number of *quantifier alternation*. It is an indicator of how complex the formula is. Intuitively, to prove a formula with k quantifier alternations, the prover will make a choice, then take unknown variables from the domain (chosen by the universe, and which can depend on the prover's choice), then the prover again choses, taking into account the value chosen by the universe, etc k times. The choice of the first values has a great influence on the later choices, and it is possible, when writing such a proof, that it turns out later that the choice was not good, and you need to backtrack.

II.C.2 Semantic proofs

II.C.2.i Proving non-equivalence

Although proving equivalence is often more useful, proving non-equivalence has one particular application: showing that would-be rules of calculations are actually not valid. For example, is $(\forall x, P(x)) \vee (\forall x, Q(x))$ equivalent to $\forall x, (P(x) \vee Q(x))$?

To prove non-equivalence in propositional logic, what is needed is a valuation for which one formula is true while the other is false. This amounts to finding a line in the truth table where the formulas differ. In first-order, there is no truth table, but the principle is the same, although what must be chosen is not the truth value of propositional variables, but the interpretation, i.e. a domain and a meaning for all predicates. So one must find an interpretation that makes one of the formula $\top$ while the other is $\bot$.

For example, to prove $(\forall x, P(x)) \vee (\forall x, Q(x)) \not\equiv \forall x, (P(x) \vee Q(x))$, we can choose the domain as the integers, $P(x)$ meaning "x is even", and $Q(x)$ meaning "x is odd".

On the left handside, we have that $(\forall x, P(x)) \vee (\forall x, Q(x))$ is $\bot$ because:

- $\forall x, P(x)$ is false since $P(1)$ is false.
- $\forall x, Q(x)$ is false since $P(0)$ is false.

On the right handside, we have $\forall x, (P(x) \vee Q(x))$ is $\top$ because for any integer, it is either even or odd. In this particular case, the left handside always implies the right handside. Why it doesn't work in the other direction is because on the left, there are two different variables (we could α-rename one: $(\forall x, P(x)) \vee (\forall y, Q(y)))$, which can take two different values, while on the right handside there is a single variable. If $(\forall x, P(x)) \vee (\forall y, Q(y))$ is true, it is true in particular for the case where x and y are instantiated with the same value.

Exercise II.5

Prove that $(\exists x, P(x)) \wedge (\exists x, Q(x)) \not\equiv \exists x, (P(x) \wedge Q(x))$

II.C.2.ii Proving equivalence

This is for your personal culture more than to be applied in this course.

Proving equivalence of two first-order formulas using the semantics is also possible. It is actually the only way to prove the basic Null quantification and Quantifier distribution rules given above.

Since the equivalence must hold for any interpretation, the proof must consider "an interpretation" without actually knowing what it is. Then for this unspecified interpretation, one must prove that if one formula is true in an interpretation, so is the other; if it is false, so is the other.

For example let's prove *semantically* that $\forall x, (P(x) \rightarrow \neg\exists y, Q(y)) \equiv \forall x, y, \neg P(x) \vee \neg Q(y)$:

Let D be a domain and P, Q predicates.

- Assume $\forall x, (P(x) \rightarrow \neg\exists y, Q(y))$ is true. We can α-rename it into $\forall t, (P(t) \rightarrow \neg\exists z, Q(z))$ Let $x, y \in D$. We have by hypothesis $P(x) \rightarrow \neg\exists z, Q(z)$.
  - Assume we have $P(x)$; then $\neg\exists z, Q(z)$, in particular $Q(y)$ is false so we have $\neg Q(y)$, hence $\neg P(x) \vee \neg Q(y)$.
  - Assume we don't have $P(x)$, then we have $\neg P(x) \vee \neg Q(y)$.
- Assume $\forall x, (P(x) \rightarrow \neg\exists y, Q(y))$ is false. Then for some $x \in D$, $P(x) \rightarrow \neg\exists y, Q(y))$ is false. That implication can only be false if $P(x)$ is true and $\neg\exists y, Q(y)$ is false, meaning $\exists y, Q(y)$ is true. Let's take such a y. Then for these particular x and y, we have $P(x)$, so $\neg P(x)$ is false; and we have $Q(y)$, so $\neg Q(x)$ is false. As a result, $\neg P(x) \vee \neg Q(y)$ is false, and we have a counterexample to $\forall x, y, \neg P(x) \vee \neg Q(y)$, so this formula is false.

So regardless of the domain and predicate interpretation, these formulas are equivalent.

Chapter III
Proof systems and proof patterns

Chapter contents

III.A What is a proof? **32**

III.A.1 Formal *vs* human proofs 32

III.A.2 Vocabulary 32

III.B Proof systems **33**

III.B.1 Rule-based systems 33

III.B.2 Hilbert-style *Natural Deduction* system for propositional logic . 33

III.B.2.i The rules 33

III.B.2.ii Writing proofs 35

III.B.2.iii Examples 35

III.B.2.iv Tricks for creating proofs in ND 37

III.B.3 Natural deduction for first-order logic 38

III.B.4 Other proof systems 39

III.B.4.i Some equivalent systems 39

III.B.4.ii Intuitionistic logic 39

III.B.4.iii Sequent calculus 40

III.B.4.iv All the other proof systems 40

III.C From formal proofs to mathematical proofs **41**

III.C.1 Human proofs 41

III.C.2 Decomposition of the statement 42

III.C.3 The art of writing a proof 42

III.C.4 Proof patterns 43

III.C.4.i Proving a quantified statement 43

III.C.4.ii Direct proof by implication 44

III.C.4.iii Proof by contrapositive 44

III.C.4.iv Proof by contradiction 44

III.C.4.v Proof by cases 45

III.C.4.vi Proving iff as two implications 46

III.C.4.vii Chains of iffs 47

III.C.4.viii Chains of iffs using implications 47

III.C.5 Remarks 47

III.C.5.i All the other rules 47

III.C.5.ii On the boldness of arrows 48

III.C.5.iii Practicing proof techniques 48

III.A What is a proof?

A proof is a formal way to produce true statements from other true statements. These original true statements may have been proved in a similar manner, or just assumed true: they are *hypotheses*. The construction of mathematics in layers of proofs from the initial hypotheses, which are named *axioms*, explains why math is in itself cumulative: the math curriculum of Kindergarten is still used when studying calculus.

For example, when applying the Pythagorean identity, to *prove* that $x = 5$ in Figure III.1 on the right, the truth of the Pythagorean theorem is used. The assumption that the Pythagorean theorem holds is valid as long as we accept the initial axioms of geometry (known as Euclid's axioms) and mathematics. These axioms are often implicit.

In another example, when proving that the diagonals of a rectangle have equal length, the fact that it is a rectangle is used as an hypothesis. In this case the hypothesis is very local: not all shapes are rectangles, we just *assume* that the shape under consideration is one. Of course, the usual axioms are also needed in order to produce this proof.

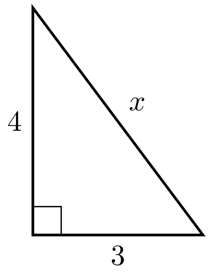


Figure III.1: A right rectangle

III.A.1 Formal *vs* human proofs

In this chapter, we will first consider proofs in the logical sense, for both propositional and first-order logics. These proofs are really a calculation, and are well suited for a formal setting. Indeed, some of the proof systems mentioned here are used in automatic theorem provers, or rather proof assistant, which are computer-aided mathematical proofs.

When discussing between humans, these formal systems that do not allow any detail to be spared are harder to understand than natural language. So in the second part of this chapter, we will discuss the writing of proofs in English. Although technically less formal, writing proofs in natural language actually mimics the calculation rules of the proof systems. As a result, the techniques used to create a formal or a human-readable proof are somewhat similar, in the sense that both proofs rely on an observation of the hypotheses and the structure of the statement being proved.

III.A.2 Vocabulary

As we have seen, assumptions can come in two flavors, which are technically the same, but philosophically quite different. *Axioms* are assumptions that are expected to be always true regardless of the context, for example: "there is a single line that goes through two distinct points". *Hypotheses*, on the other hand, are not necessarily true in general, for example: "Let ABC be a right rectangle". So, as axioms are not expected to change, hypotheses can be introduced at will.

The new true statement that results of a proof without hypotheses is called a *Theorem*. Note that hypotheses can actually be contained within the theorem. For example the Pythagorean Theorem "If ABC is a right triangle in C, then $AC^2 + BC^2 = AB^2$" implicitly contains the hypothesis that the triangle is right as the premise of an implication but the full sentence does not require any hypothesis (beyond the axioms of mathematics and geometry).

The term theorem literally means “a thing from God” in Latin. In math this term is used for results deemed important (this being a purely subjective notion). For less important results, other terms are often used:

Lemma An intermediate result not important in itself, but used to proved a theorem.

Corollary An easy consequence of a theorem. What is easy being subjective also; a good example is when the corollary is a particular case of the theorem.

Proposition A theorem of lesser importance; it is implied that it is proven to be a true proposition.

Scholia Literally a “comment”: an intermediate result proved during the proof of a theorem that deserves a statement in its own right (as an afterthought).

III.B Proof systems

III.B.1 Rule-based systems

Formal proof systems are based on *deduction rules*: from one or more statement, called the *premises*, the rule allows to produce a new statement, called the *conclusion*. Rules that have no premises are called *axioms*: they are deemed always true.

The rules are written using the $\vdash$ symbol, which is read as “proves”. So $\varphi_1, \ldots, \varphi_n \vdash \psi$ means that ψ can be proved when $\varphi_1, \ldots, \varphi_n$ are assumed. Here $\varphi_1, \ldots, \varphi_n$ forms the premise, while ψ the conclusion.

This notation is used to denote a statement proved under some hypotheses and sometimes the rules themselves, as is the case in the Natural Deduction (ND) system.

III.B.2 Hilbert-style *Natural Deduction* system for propositional logic

This system was introduced by mathematician and logician David Hilbert in the beginning of the 20th century to formalize the human-made proofs.

III.B.2.i The rules

In this system there 11 deduction rules and no axioms:

The rules of Natural Deduction system

Negation introduction, ¬-I $p \to q, p \to \neg q \vdash \neg p$

Negation elimination, ¬-E $\neg p \vdash p \to q$

Double negation elimination, ¬¬-E $\neg\neg p \vdash p$

Conjunction introduction, ∧-I $p, q \vdash p \wedge q$

Conjunction elimination (left and right), ∧-E$_l$ and ∧-E$_r$ $p \wedge q \vdash p$ and $p \wedge q \vdash q$

Disjunction introduction (left and right), ∨-I$_l$ and ∨-I$_r$ $p \vdash p \vee q$ and $q \vdash p \vee q$

Disjunction elimination, $\vee$-E $p \vee q, p \rightarrow r, q \rightarrow r \vdash r$

Iff introduction, $\leftrightarrow$-I $p \rightarrow q, q \rightarrow p \vdash p \leftrightarrow q$

Iff elimination, $\leftrightarrow$-E $p \leftrightarrow q \vdash p \rightarrow q$ and $p \leftrightarrow q \vdash q \rightarrow p$

Implication introduction (conditional proof), $\rightarrow$-I $p \vdash q$ becomes $\vdash p \rightarrow q$.

Implication elimination (Modus ponens), $\rightarrow$-E $p, p \rightarrow q \vdash q$

All rules have abbreviated named. They are formed with the symbol for the operator and I for introduction or E for elimination. Rules that have two versions (conjunction elimination and disjunction introduction) have two abbreviations to indicate whether the left or right version is being considered;

Let's consider all the rules and the intuition behind them. The negation introduction rule states that if a proposition p implies both a proposition q and its negation, then we have proved the negation of p. The idea is that p would imply a contradiction.

The negation elimination rule simply states that if the negation of p has been proved, then p implies anything.

Double negation elimination states that the proof of the negation of the negation of a proposition proves the proposition itself.

Conjunction introduction states that two independent proofs of p and q provide a proof of their conjunction $p \wedge q$. On the other hand, a conjunction elimination takes a proof of $p \wedge q$ and uses it to make *in particular* a proof of p (or a proof of q).

Disjunction introduction states that a proof of p proves the weaker statement $p \vee q$.

Disjunction elimination is more involved. There is no way to just eliminate a disjunction on its own: when $p \vee q$ has been proved, one cannot know which of p or q to chose. This rule bypasses this difficulty by using a third proposition r implied by both p and q, independently. So it does not matter which of p or q would be chosen, because r would be proved anyway.

The Iff rules are just a translation of the fact that Iff is implication in both directions.

The *conditional proof* rule allows to convert a premise of the proof into a premise of the statement. It allows to make theorems that implicitly contain hypotheses.

The *modus ponens* allows to apply an implication: when the implication has been proved and the premise of said implication has been proved as well, that proves the conclusion. This is what is at hand when applying a theorem (usually an implication): it must be proved that all the premises hold in order to be able to apply it and obtain the conclusion of the theorem.

Remarks

Note that these rules are about what is *provable*, not what is *true*. This is why the explanations above refrain from using phrases like "if p is true". This subtle distinction is developed further in Section III.B.4.iv.

Also these rules provide *deduction*, not equivalence. Some of the rules produce a conclusion that is a weaker statement than the premise, that is to say a statement that is implied by the premise. Examples of these are the conjunction elimination and disjunction introduction rules.

III.B.2.ii Writing proofs

A proof in the Natural Deduction (ND) system consists of lines of propositions that have been proved or assumed. To create a new line, hence a new proposition, one has to use one of the 11 rules above, or add an hypothesis. The rule used must be indicated next to the proposition, with reference to the rule's premises. The $\vdash$ ("proves") symbol is not used within the proof itself.

There are two ways to introduce an hypothesis. The first is as an hypothesis of what is being proved. Namely, if in the end the goal is to prove $\varphi_1, \ldots, \varphi_n \vdash \psi$, then each of the φ_i can be introduced using the (meta-)rule *Premise*.

The second way to introduce an hypothesis is through a local assumption, with the rule *Assumption*. Anything can be assumed in this manner, but every assumption must be *discharged* by the end of the proof. Discharging is done through the conditional proof rule: the assumed hypothesis become the premise of an implication. It must however be noted that when an hypothesis is discharged, everything that was proved *under this assumption* cannot be used anymore. In order to highlight what was under which assumption, it is common to use indentation or adequate numbering (the latter is the approach taken here). Crossing out the lines no longer usable is also possible, but it hinders readability (see Figure III.2).

Another "rule" that can be used when writing proofs is the *Reiteration* rule, that simply repeats a proposition proved earlier. Using this rule is never necessary, it only enhances the readability of the proof in some cases.

There are no additional rules. In particular, it is not possible to use rules of propositional calculus in this setting. For example, the rewriting of the implication arrow is not possible; technically, it can be allowed but only after being proved once (which is done in Exercise III.1). Then normally it should be proved again each time it is being used.

III.B.2.iii Examples

III.B.2.iii.a $\vdash p \rightarrow p$

This first example is very simple. What we can see is first that is does require a proof. Now to prove the statement $p \rightarrow p$, which is an implication, the natural way is to assume the premise and try to prove the conclusion under the assumption of the premise (which is trivial here). Finally, the conditional proof discharges our assumption of the premise and provides a proof of the implication.

(1)	p	[Assumption, discharged in (2)]
(1.1)	p	[Reiteration of (1)]
(2)	$p \rightarrow p$	[Conditional proof on (1) and (1.1)]

III.B.2.iii.b $(\neg p \rightarrow \neg q) \vdash (q \rightarrow p)$

In this case, we can start by writing all the hypotheses as premises. Then, because we need to prove an implication, we assume the premise q and try to prove p. In this case, we have little choice but to try to use our premise.

As an implication, it could be used in a modus ponens, disjunction elimination, iff introduction, or negation introduction. Disjunction elimination would be hard to use, since we don't have a disjunction to start with. Similarly, iff introduction would require $\neg q \rightarrow \neg p$, and also would provide an iff which is not really needed.

(1)	$\neg p \to \neg q$	[Premise]	
(2)	~~q~~	~~[Assumption]~~	discharged
(2.1)	~~$\neg p$~~	~~[Assumption]~~	discharged
~~(2.1.1)~~	~~q~~	[Reiteration of (2)]	
~~(2.2)~~	~~$\neg p \to q$~~	[Conditional proof on (2.1) and (2.1.1)]	
~~(2.3)~~	~~$\neg\neg p$~~	[Negation introduction on (1) and (2.2)]	
~~(2.4)~~	~~p~~	[Double negation elimination on (2.3)]	
(3)	$q \to p$	[Conditional proof on (2) and (2.4)]	

Figure III.2: Proof of $(\neg p \to \neg q) \vdash (q \to p)$ using crossing-out.

The modus ponens requires a bit more attention: if we assume $\neg p$, we could have $\neg q$. That seems to be a contradiction with the fact that we have assumed q. But there is no rule that says that having both q and $\neg q$ is an issue per se.

On the other hand, we can show that $\neg p$ implies both q and $\neg q$, then use the negation introduction rule, which was our last choice. Then having $\neg\neg p$ will yield p, which was our goal.

In ND, it translates as follows:

(1)	$\neg p \to \neg q$	[Premise]
(2)	q	[Assumption, discharged in (3)]
(2.1)	$\neg p$	[Assumption, discharged in (2.2)]
(2.1.1)	q	[Reiteration of (2)]
(2.2)	$\neg p \to q$	[Conditional proof on (2.1) and (2.1.1)]
(2.3)	$\neg\neg p$	[Negation introduction on (1) and (2.2)]
(2.4)	p	[Double negation elimination on (2.3)]
(3)	$q \to p$	[Conditional proof on (2) and (2.4)]

Note that reading the ND proof on its own might seem very artificial. But this proof was constructed by starting from both ends at the same time: what we have (the hypothesis) and what we want. Then we considered the rules available to use, selecting the ones that go us closer to the goal.

III.B.2.iii.c $(p \wedge \neg q) \to q \vdash p \to q$

In this case, once again the start and end of the proof is given by the structure of the hypothesis and conclusion we want to prove: we write down the hypothesis as a premise, and we assume p, to be discharged in the end once we have proved q.

Intuitively, we can see why q should be true under hypotheses $(p \wedge \neg q) \to q$ and p: for both these to hold, either $\neg q$ is true then it implies q, or it is not then that means we have q. This intuition based on the possible truth values for q does not however yield a proof in ND: it would have to assume the excluded middle $q \vee \neg q$, which is not a base rule. Part of this reasoning nonetheless can give us an idea: if $\neg q$ is true, then it implies q. That is sufficient to be able to prove $\neg q \to q$ (under the aforementioned hypotheses). While not a contradiction, the only way for this to hold is for q to be true; again, this does not constitute a proof in ND. One must first prove $\neg q \to \neg q$, then to introduce the negation to obtain $\neg\neg q$.

The resulting proof in the Natural Deduction system is:

(1)	$(p \land \neg q) \to q$	[Premise]
(2)	p	[Assumption, discharged in (3)]
(2.1)	$\neg q$	[Assumption, discharged in (2.2)]
(2.1.1)	$p \land \neg q$	[Conjunction introduction on (2) and (2.1)]
(2.1.2)	q	[Modus Ponens on (1) and (2.1.1)]
(2.2)	$\neg q \to q$	[Conditional proof on (2.1) and (2.1.2)]
(2.3)	$\neg q$	[Assumption]
(2.3.1)	$\neg q$	[Reiteration of (2.3)]
(2.4)	$\neg q \to \neg q$	[Conditional proof on (2.3) and (2.3.1)]
(2.5)	$\neg\neg q$	[Negation introduction on (2.2) and (2.4)]
(2.6)	q	[Double negation elimination on (2.5)]
(3)	$p \to q$	[Conditional proof on (2) and (2.6)]

III.B.2.iii.d $\vdash p \lor \neg p$ (excluded middle)

First, let's once again remark that we cannot just use a truth table to prove the excluded middle. We have to rely entirely on the available rules.

This proof is more advanced, and harder to develop from scratch without having the central idea: we will do a proof by contradiction. Namely, we will prove that assuming the negation of our formula allows to prove the formula itself: $\neg\varphi \to \varphi$, where $\varphi = p \lor \neg p$. Then because $\neg\varphi \to \neg\varphi$ trivially, by introduction of negation we obtain $\neg\neg\varphi$, then φ.

Proving $p \lor \neg p$ from $\neg(p \lor \neg p)$ is a bit tricky in itself. Since it is a disjunction, one must choose in advance which part will be proved. In this case, we will prove $\neg p$, by proving that p implies both φ and its negation (by assumption). Note that in this case we could have chosen to prove p by proving that $\neg p$ implies both φ and its negation.

The resulting ND proof is as follows:

(1)	$\neg(p \lor \neg p)$	[Assumption, discharged in (2)]
(1.1)	p	[Assumption, discharged in (1.2)]
(1.1.1)	$p \lor \neg p$	[Disjunction introduction on (1.1)]
(1.2)	$p \to (p \lor \neg p)$	[Conditional proof on (1.1) and (1.1.1)]
(1.3)	p	[Assumption, discharged in (1.4)]
(1.4)	$p \to \neg(p \lor \neg p)$	[Conditional proof on (1.3) and (1)]
(1.5)	$\neg p$	[Negation introduction on (1.2) and (1.4)]
(1.6)	$p \lor \neg p$	[Disjunction introduction on (1.5)]
(2)	$\neg(p \lor \neg p) \to p \lor \neg p$	[Conditional proof on (1) and (1.6)]
(3)	$\neg(p \lor \neg p)$	[Assumption, discharged in (4)]
(4)	$\neg(p \lor \neg p) \to \neg(p \lor \neg p)$	[Conditional proof on (3) and (3)]
(5)	$\neg\neg(p \lor \neg p)$	[Negation introduction on (2) and (4)]
(6)	$p \lor \neg p$	[Double negation elimination on (5)]

III.B.2.iv Tricks for creating proofs in ND

It is often difficult to start writing a proof in the Natural Deduction system directly. As in the above example, it is easier to start by making a plan of the proof, starting from both what we have and what we want, and paying good attention to the structure of these formulas in order to mentally list all the rules that could be applied at this point.

Do not use the rules of propositional calculus!

Exercise III.1

Prove in the Natural Deduction system:

1. $p \vee s$ from hypotheses $(p \wedge q) \vee r$ and $r \rightarrow s$
2. $(\neg p \wedge q) \rightarrow t$ from hypotheses $r \rightarrow p$, $\neg r \rightarrow s$, and $s \rightarrow t$.
3. $\neg p \vee q$ from hypothesis $p \rightarrow q$ (You don't have to do both versions.)
   a. Easy version: add $p \vee \neg p$ as an hypothesis.
   b. Hard version: get inspiration from the proof of $p \vee \neg p$, but don't use it directly.

III.B.3 Natural deduction for first-order logic

Extension from propositional logic to first-order logic was done through the introduction of variables and quantifiers. The extension to proof systems is similar: we need to introduce some rules to deal with the quantifiers. As is the case for other operators, there is an introduction rule, named *generalization*, and an elimination rule, named *instantiation*, for each quantifier.

In order to prevent scoping issues, some restrictions apply. A quantifier can be introduced only if it uses a new variable. And this new variable does not replace one that appeared in an hypothesis: that would amount to performing the replacement only partially, and that could mean assuming a variable could take two different values (see the examples in Section II.C.2.i).

Such a new variable is called a *fresh* variable

In addition, we will use the following notation for *variable substitution*: $P(a/x)$ means P where every occurrence of x is replaced by a.

The rules of Natural Deduction system, extension to first-order

Existential generalization, $\exists$-I $P(a) \vdash \exists x, P(x/a)$ if a does not appear free in any premise or assumption and x does not appear in P.

Existential instantiation, $\exists$-E $\exists x, P(x) \vdash P(y/x)$ where y is a new symbol (think: constant).

Universal generalization, $\forall$-I $P(y) \vdash \forall x, P(x/y)$ if y does not appear free in any premise or assumption and x does not appear in P.

Universal instantiation, $\forall$-E $\forall x, P(x) \vdash P(a/x)$.

Example: $(\forall x, P(x)) \rightarrow (\exists y, P(y))$

(1)	$\forall x, P(x)$	[Assumption]
(1.1)	$P(z)$	[Universal instanciation on (1)]
(1.2)	$\exists y, P(y)$	[Existential generalization on (1.1)]
(2)	$(\forall x, P(x)) \rightarrow (\exists y, P(y))$	[Conditional proof on (1) and (1.2)]

Exercise III.2

Prove in the Natural Deduction System: $(\exists x, \neg P(x) \wedge Q(x))$ from hypotheses $\exists x, \neg P(x)$ and $\forall y, Q(y)$

III.B.4 Other proof systems

The Natural Deduction system presented above is just *a* deduction system. There are lots of possible variations, which may or may not yield an equivalent system, two systems being equivalent if they can prove exactly the same statements.

This is for your personal culture more than to be applied in this course.

III.B.4.i Some equivalent systems

One simple variation that would provide the same proof system would be to keep only the the conjunction introduction and the implication introduction and elimination rules (modus ponens and conditional proof), and replace all other rules by an axiom of the corresponding implication. For example, conjunction elimination would be replaced by the axioms $\vdash p \wedge q \to p$ and $\vdash p \wedge q \to q$. And Negation introduction would be replaced by the axiom $((p \to q) \wedge (p \to \neg q)) \to \neg p$. So instead of applying a rule, the modus ponens would be applied on the premises joined by a conjunction if more than one premise is required.

This change is purely syntactical, and it is relatively easy to see why the system thus produced is equivalent. For other systems, it is not as easy to see (and no proof will be provided here). For example, the system defined by Jan Łukasiewicz only uses the modus ponens $(p \to q, p \vdash q)$ and replaces the other axioms for propositional logic by the following few axioms:

$$p \to (q \to p) \qquad (\neg p \to \neg q) \to (q \to p) \qquad (p \to (q \to r)) \to ((p \to q) \to (p \to r))$$

III.B.4.ii Intuitionistic logic

It is possible to use the same rule system, but removing the Double negation elimination rule. The logic thus created is called *intuitionistic logic*, in opposition to the *classical logic* used up to now.

In this system, it is impossible to prove the excluded middle or the contraposition rule, or to use proofs by contradiction: proving $\neg\neg\varphi$ does not prove φ. That means that in order to prove $p \vee \neg p$, you have to either prove that p holds or that $\neg p$ holds. This illustrates the difference between the *truth* and the *provability*.

Proving "the impossibility of a proof" is not straightforward, as one need to consider *all possible proofs*!

As a consequence, to prove an existential statement $\exists x, P(x)$ one cannot prove that there is a contradiction if we assume $\forall x, \neg P(x)$: an actual value for x that satisfies $P(x)$ has to be exhibited. This approach is called *constructive*.

Classical logic seems more natural to modern mathematicians who have been using it all their lives, but when Hilbert proposed his system with this double negation rule, some logicians and philosophers deemed it unrealistic because it allowed non-constructive proofs. This is where the term intuitionistic comes from: only constructive proofs were considered compatible with the intuition that to prove the existence of something one need to show it.

III.B.4.iii Sequent calculus

The structure of the Natural Deduction system mimics how proofs are written on paper: linearly. But when planning the proof, we often see that there are different parts of the proof that are independent. For example, when proving a conjunction $\varphi \wedge \psi$, the proofs of φ and of ψ may be completely independent, have local assumptions, etc.

As a result, it makes more sense to write the proofs not as a succession of lines but as a *tree*. This is the approach taken in the *sequent calculus*, invented by Gerhard Gentzen in the 1930s.

A sequent is $\Gamma \vdash \Delta$ that keep the list of all hypothesis and provable statements (under these assumptions). So if $\Gamma = \{\varphi_1, \ldots, \varphi_n\}$ and $\Delta = \{\psi_1, \ldots, \psi_k\}$, the sequent $\Gamma \vdash \Delta$ should be understood as $\varphi_1 \wedge \cdots \wedge \varphi_n$ proves $\psi_1 \vee \cdots \vee \psi_k$. The are written in a vertical way: on top are what is proved, on the bottom the new sequent: since the sequent contains both hypothesis and conclusion, assumptions are always indicated on the left handside of the sequent.

For example the rule corresponding to modus ponens is Cut, while the conditional proof is *implication right* $(\to_r)$:

$$\frac{\Gamma \vdash p, \Delta \qquad \Gamma', p \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ \textsc{Cut} \qquad\qquad \frac{\Gamma, p \vdash q, \Delta}{\Gamma, \vdash p \to q, \Delta}\ \to_r$$

This tree structure allows to make proofs about proofs. The most well-know result about proofs in sequent calculus is the *Cut elimination Theorem* that states the Cut rule is actually not necessary. This result mimics the idea that in math, you can do without intermediate results as long as you prove them again from scratch.

The other beautiful result about sequent calculus is that to obtain the intuitionistic version of the logic, the only restriction that is needed is to require the right handside of the sequent to be a single formula.

III.B.4.iv All the other proof systems

Whenever a proof system is introduced, equivalence to the one shown here is not the most important feature. What is matter is whether the system is *sound* and *complete*.

The term *validity* is used rather than truth.

Soundness means that all statements than can be proved in this systems are true. For propositional logic, validity corresponds to what truth tables provide. For first-order logic, validity means true for any interpretation (domain and meaning of predicate).

Completeness means that for every true statement can be proved. In the case of statements with premises, that means that if $\varphi \to \psi$ is a true statement, then $\varphi \vdash \psi$ is provable.

Proving soundness is usually relatively easy. The plan of the proof is as follows:

- for each axiom, prove that it is true.
- for each rule, prove that if the premise is true, so is the conclusion.
- since everything deemed provable was through these axioms and rules, if it is proved it is correct.

Completeness is more complex to prove, because all possible statements have to be considered. Or rather, since these proofs are often by contradiction, we assume that for a given statement, there is no proof, and from that extract a valuation that makes the statement false.

The Natural deduction system developed above is sound and complete both for propositional and first-order logics.

Gödel Incompleteness Theorem

The fact that the ND system in this chapter is both sound and complete may sound contradictory if you have heard about Gödel Incompleteness Theorem:

Gödel Incompleteness Theorem

Any deduction system that can express *second-order arithmetic* cannot be at the same timed defined by a finite number of axioms and rules, sound, and complete.

But this not a contradiction: it only applies to systems that allows second-order arithmetic, so more expressive than first order. In addition, the notion of completeness used in this theorem is slightly different: it means that for any formula φ, either φ or its negation $\neg\varphi$ can be proved. Soundness can also be defined in an analogous way by requiring that for no formula φ both φ and its negation $\neg\varphi$ can be proved.

This theorem however applies to any logical model that might be created to formalize the logics behind mathematical reasoning. It forbids any finite axiomatization: or any set of finite axioms we admit, there will be some statements that cannot be neither proved nor disproved. This statement (or its negation for that matter) could be added as a new axiom in the theory, which will enable proofs of more statements. It is for example frequent to have mathematical theorems that state "Assuming the axiom of choice...". But regardless of how many axioms are added, there will always be some other unprovable statement, and this process will never stop.

III.C From formal proofs to mathematical proofs

III.C.1 Human proofs

Despite being called *natural deduction*, the proof system described in he first part of this chapter is not completely natural. In fact, it was actually created in order to formalize what were he proofs written at the time, in order to formally verify them. In this chapter, the reverse approach has been taken: now that we have seen the formal system, we can get inspiration from it, in particular paying attention to the structure of the statement in order to devise and write up a proof.

It must be noted that human proofs go beyond first order logic (we allow ourselves to reason in higher order), and that the set of axioms we are using is not entirely clear. As remarked above, there will always be some axiom that is missing. It didn't prevent mathematicians to try to approach an axiomatization of the mathematics we write. One such axiomatization is called *Zermelo–Fraenkel set theory with the Axiom of choice*, a.k.a. *ZFC*.

When writing proofs, we allow for a less formality than for ND or other proof systems. It must be kept in mind that the reader of the proof is in this case also a human! For example, complete sentences are used instead of merely pointing to a rule (which is often implicit). In first (or higher) order, the domain is often implicit: even when it is not mentioned, the statements proved are usually only valid in this given interpretation (for example real

✍ It is tempting to call the proofs humans write as *natural*, but they are the product of our mathematical culture. The question of whether math is *natural* is left open for the consideration of philosophers.

numbers with the usual operators). In addition, to improve readability, the proofs can be cut in several lemmas that are used later. Some lemmas are just theorems that you learned in school that can be used without being proved again.

So in the end, what remains from natural deduction? As the structure of the statement guides the proof, the deduction rules provide *proof patterns* that can be applied in order to write a proof.

III.C.2 Decomposition of the statement

Although the statement to prove is often in English (or another human language) and so will be the proof, formalizing into a mathematical statement highlights its internal structure and may help choosing which of the proof pattern (described next) to apply.

In addition, exhibiting the structure can also avoid errors such as quantifier confusion and scope mistakes.

For example, the statement “Every real number that is non-zero has a multiplicative inverse” can be translated into the formula: $\forall x, (x \neq 0) \to \exists y, x \times y = 1$, keeping in mind that the domain is the real numbers. This translation highlights that it is a universal statement, so it can be proved using the usual pattern for a universal statement, and the proof will start as follows: “Let x be a real number.” Then we need to prove the statement $(x \neq 0) \to \exists y, x \times y = 1$, which is an implication, so it should be proved using a pattern that deals with implication, and so on until the statement is fully proved.

Exercise III.3

Exhibit the structure of the following statements by transforming them into a first-order formula, indicating the interpretation of predicates and the domain. For example the statement *There is no greatest integer* can be transformed into $\neg\exists x, \forall y, P(x, y)$ where $P(x, y)$ is the predicate $x \geq y$, the domain being the integers.
Do not try to prove them (it may not be possible!).

1. Every integer can be written as the sum of 2 squares.
2. Every positive real number has a square root.
3. The cosine function has zeroes.
4. The cosine function has at least two distinct zeroes.
5. There is a neutral element[a] for multiplication in real numbers.
6. Every odd square can be written as the sum of three odd numbers.

[a] Remember for example that $\bot$ is neutral for $\lor$ in Boolean algebra because $\bot$ does not affect a disjunction.

III.C.3 The art of writing a proof

Writing a proof, which is a text intended for the explanation of a reasoning, is a process that requires some form of literary skill, although it falls into the category of technical writing, so style is not of utmost importance.

There are however some codes and techniques that can be applied to make a proof more readable.

The first is that a proof can mix formal and informal statements: having both a formula and a sentence saying the same thing helps the reader use the one he prefers to understand said statement. Usually, the sentence is easier to understand, but the formula relieves any ambiguity that the natural language may bring.

When writing a proof, it is usually not created in the order that it is read. As there are subformulas inside a formula, a proof will consist of several parts. These parts are usually obvious when the proof is being devised, so it helps the writer to draft a plan of the proof first. But this plan can also help the reader understand the proof! It is therefore a good idea to keep the plan in the final proof. In that sense, technical writing differs from essays, where the plan has to bu concealed in the text. So when writing a proof, do not refrain from using intermediate titles or bullet points. In this book I often use framed subformulas in order to help the reader understand which part is being treated. See for example the proofs that formulas are contingent in Section I.C.2.

Throughout the writing of a proof, or even right after having written it, the proof may appear too big to be easily understood in one go. In that case, it is worthwhile (if possible) to write part of the proof as the proof of a lemma: the proof of the lemma will be separate from the proof of the theorem, and the lemma will only be applied in the proof of the theorem. This is akin to, in programming, separating the code of a big function into an auxiliary function that is called by the main function. In both cases it is not always possible and it requires a bit of experience to perform this separation elegantly.

It is common to end the proof by writing *q.e.d.*, which stands for the Latin phrase *Quod erat demonstrandum*, meaning “what was to be shown”. It is often symbolized by a square at the bottom right of the proof (seen here at the end of this paragraph). In research papers, using this square is a visual cue that allows the reader to skip the proof without missing any of the text of the paper. □

III.C.4 Proof patterns

III.C.4.i Proving a quantified statement

The techniques and patterns for quantifiers was actually already covered in Section II.B.4, but we revisit them here both for exhaustiveness and to highlight the link with natural deduction.

III.C.4.i.a Existential quantifiers

To prove $\exists x, P(x)$, it is sufficient to exhibit one value a in the domain and show that $P(a)$ is true. That corresponds to the *Existential generalization* rule $P(a) \vdash \exists x, P(x/a)$.

Proving that $\neg\forall x, P(x)$ means proving the equivalent existential formula $\exists x, \neg P(x)$, so it is sufficient to exhibit one value a in the domain and show that $P(a)$ is false (a counterexample).

III.C.4.i.b Universal quantifiers

To prove $\forall x, P(x)$, it must be shown that $P(x)$ holds for any value of x. That corresponds to the *Universal generalization* rule $P(y) \vdash \forall x, P(x/y)$. In practice, we use the same symbol for the quantified variable and the uncontrolled value used in the proof. So the proof of a statement of the form prove that $\forall x, P(x)$ would start with the words “Let x be an element of the domain”.

The proof could start with “Let y be an element of the domain”, but it is less confusing to use the same symbol as the quantifier so that the reader can match it.

For example to prove $\forall n, 2n^2 + 1 \geq 3x$ over integers, the proof would start by “Let n be an integer”.

Proving $\neg\exists x, P(x)$ means proving the universal statement $\forall x, \neg P(x)$: it must be shown that $P(x)$ does not hold for any value of x.

III.C.4.ii Direct proof by implication

Implications are the most common form of statements. As explained before, most theorems are implications, since the premise of the implication contains the hypotheses. Implications can be proved in a variety of manners.

The most straightforward, and hence named the *direct proof*, relies on the conditional proof rule: to prove $p \to q$, assume p and prove q.

For example, let's prove the following statement: "For an integer n, if n is even, then n^2 is even". First, we can exhibit the formula corresponding to this statement: $\forall n, n$ is even $\to$ n^2 is even. And since it is a universal statement the proof starts by "Let n be an integer". Then the statement that remains to be proved is the implication n is even $\to$ n^2 is even, which can be proved by a direct proof: "Assume n is even. We need to prove that n^2 is even". In this case the proof of n^2 is even is obtained by calculation. The full proof is therefore as follows:

Note that here we restate what needs to be proved to remind the reader (and ourselves) of the plan of the proof.

Proof of "For an integer n, if n is even, then n^2 is even"

Let n be an integer. Assume n is even. We need to prove that n^2 is even Then n can be written $n = 2p$. So $n^2 = (2p)^2 = 4p^2 = 2(2p^2)$, hence n^2 is even. □

III.C.4.iii Proof by contrapositive

To prove an implication $p \to q$, it is possible to prove $\neg q \to \neg p$ instead. This relies not on a base rule of the deduction system, but rather on the fact that$\neg q \to \neg p \vdash p \to q$ (proved in Section III.B.2.iii.b).

Proofs usually begin by the words "Let's prove the contrapositive, namely that…" to indicate to the reader the plan of the proof.

This type of proof when the conclusion seems to have more information or manipulable variables than the premise of the implication. Typically, proving that for any integer n, if n^2 is even, then n is even: it is easier to reason on n than on a perfect square n^2.

Proof of "For an integer n, if n^2 is even, then n is even"

Let's prove the contrapositive, namely that if n is odd, then n^2 is odd. Assume n is odd, then $n = 2p + 1$ for some p. Hence $n^2 = (2p + 1)^2 = 4p^2 + 4p + 1 = 2(2p^2 + 2p) + 1$ is odd. □

III.C.4.iv Proof by contradiction

This technique relies on assuming the *negation* of what needs to be proven, and deduce a *contradiction*. For readability, these proofs usually indicate that this is the approach taken: "Assume, by contradiction, that…" or "We will use a proof by contradiction. Assume that…"

The contradiction can take two forms: either assuming $\neg p$ allows to prove p, then the idea is that what was proved is $\neg p \to p$. Then the proof would go as follows : we can trivially prove that $\neg p \to \neg p$, so by negation introduction we have $\neg\neg p$ and by double negation elimination we have p. In actual proofs the latter part is omitted and replaced by the symbol ↯before concluding that the original (non negated) proposition is now proved.

In other cases, the contradiction is proved by proving both q and $\neg q$ for some other proposition. The rest of the proof would follow the same idea as before; and as before it is replaced by the ↯symbol.

Note that since this proof scheme uses the double negation elimination, it only works because we reason in classical logics, but would not be allowed in a constructionist (intuitionistic) mathematical world.

(that is often omitted, and the proof ends at this point uses the same idea as the negation introduction rule $p \to q, p \to \neg q \vdash \neg p$, combined with double negation elimination.

This pattern is quite efficient when dealing with quantified statements, especially when proving non-existence: to prove $\neg\exists x, P(x)$, assume $\exists x, P(x)$ and exhibit a contradiction? What makes the proof easier in this case is that it is usually easier to prove a contradiction when there is an x to calculate on. This can also be seen as proving $\forall x, \neg P(x)$ by taking an x and then assuming by contradiction that $P(x)$ holds (by De Morgan's Law on quantifiers).

A very famous example of a proof by contradiction is proving that $\sqrt{2}$ is irrational, i.e. *there does not exist* integers p and q such that $\frac{p}{q} = \sqrt{2}$. Proving non-existence is an arduous task, so we use a proof by contradiction in order to have values p and q on which to reason and calculate.

Proof that $\sqrt{2}$ is irrational

Assume, by contradiction, that $\sqrt{2}$ is rational.
Then it can be written as an irreducible fraction: there are two integers p and q that do not have a common divisor such that $\sqrt{2} = \frac{p}{q}$.
Therefore $\sqrt{2}^2 = \left(\frac{p}{q}\right)^2$ so $2 = \frac{p^2}{q^2}$, and $2q^2 = p^2$ (Eq.1).
So p^2 is even. By the Lemma that was proved in Section III.C.4.iii, p is also even: it can be written as $p = 2p'$.
In that case, $p^2 = (2p')^2 = 4p'^2$, and (Eq.1) becomes $2q^2 = 4p'^2$, which is equivalent to $q^2 = 2p'^2$.
That means q^2 is even, so q is even, which contradicts the fact that p and q had no common divisor. ↯
As a result our assumption that $\sqrt{2}$ is rational is false, so $\sqrt{2}$ is irrational. □

III.C.4.v Proof by cases

As the name indicates, this proof pattern consists in separating the hypotheses into cases, then proving the theorem for each of the cases. This is akin to using the *disjunction elimination* rule: $p \vee q, p \to r, q \to r \vdash r$. In practice there may be more than two cases: $p_1 \vee p_2 \vee \cdots \vee p_n, p_1 \to r, p_2 \to r, \ldots, p_n \to q \vdash r$.

When the theorem that is to be proven is an implication, the hypotheses is the premise of the implication (that is assumed): instead of proving $p \to q$, prove $p \to r_1 \vee r_2 \vee \cdots \vee r_n$. Then prove $r_1 \to q$, $r_2 \to q$, ..., and $r_n \to q$.

When the statement is not an implication, it is possible to split in cases using the knowledge of the domain, for example using the excluded middle on a well chosen predicate. The difficulty is to know how to chose this predicate: for integers there is even/odd, for all numbers there is positive/negative/zero. In other cases the statement may provide some indication.

For example, to prove that "For an integer n, if n is not a multiple of 3, then $n^2 = 3k+1$ for some k", which we can write as the formula $\forall n, P(n) \to \exists k, Q(n,k)$, with $P(n)$ being "n is not a multiple of 3" and $Q(n,k)$ being "$n^2 = 3k+1$". This being a universally quantified formula, we can deal with this quantifier by taking an arbitrary n: "Let n be an integer". Then we need to find predicates $R_1(n)$ and $R_2(n)$ such that $P(n) \to R_1(n) \vee R_2(n)$, $R_1(n) \to \exists k, Q(n,k)$, and $R_2(n) \to \exists k, Q(n,k)$. In this case, the premise gives us a hint: if n is not a multiple of 3, then we can try to create cases based on what happens when n is divided by 3. That provides two cases: remainder is 1 or 2, and each case is treated using calculation. The full proof is as follows:

Division is understood here as Euclidean division with quotient and remainder. This concept is explained in details in Section VII.A.2.

Proof of "For an integer n, if n is not a multiple of 3, then $n^2 = 3k+1$ for some k"

Let n be an integer. Since n is not a multiple of 3, the remainder in the Euclidean division of n by 3 is either 1 or 2. So n can be written either as $3p+1$ or $3p+2$ for some p

- Assume $n = 3p+1$, then $n^2 = (3p+1)^2 = 9p^2+6p+1 = 3(3p^2+2p)+1$; we choose $k = 3p^2+2p$.
- Assume $n = 3p+2$, then $n^2 = (3p+2)^2 = 9p^2+12p+4 = 9p^2+12p+3+1 = 3(3p^2+6p+1)+1$; we choose $k = 3p^2+6p+1$.

In both cases we can find k such that $n^2 = 3k+1$, which concludes the proof. □

III.C.4.vi Proving iff as two implications

There are several possibilities to prove $p \leftrightarrow q$. The first one is to see this statement as the conjunction of two statements: $p \to q$ and $q \to p$ (called the *converse* of $p \to q$). The converse $q \to p$ is also written $p \leftarrow q$, so we speak of the *right-to-left implication* (and $p \to q$ is the *left-to-right* implication). This mimics the approach of the iff introduction rule in ND.

Since there are two independent parts to these proofs, it helps to keep track of what is being proved by highlighting the plan of the proof. One way to do that is to indicate which direction is being proved.

Then there are many choices on how to prove these statements. They need not be proved in the same manner, as the examples below show.

III.C.4.vi.a Two direct proofs

The blunt approach is to use two direct proofs: prove $p \to q$ and $q \to p$. For example we can prove the simple statement "n is even iff $n+1$ is odd" on integers this way.

Proof of “For any integer n, n is even iff $n+1$ is odd”

$\boxed{\rightarrow}$ Assume n is even: $n = 2p$, so $n+1 = 2p+1$ is odd.

$\boxed{\leftarrow}$ Assume $n+1$ is odd: $n+1 = 2p+1$, so $n = 2p$ is even.

III.C.4.vi.b One direct and one contrapositive proof

The contrapositive can be used to prove the converse: prove $p \rightarrow q$ and $\neg p \rightarrow \neg q$. This pattern is the source of the name “*if and only if*” comes from: if p then q; only if p then q.

Proof of “For any integer n, n is even iff n^2 is even”

$\boxed{\rightarrow}$ $n = 2p$, so $n^2 = 4p^2 = 2(2p^2)$ is even.

$\boxed{\leftarrow}$ By contrapositive, see Section III.C.4.iii.

Or the contrapositive is used for the right-to-left implication: $q \rightarrow p$ and $\neg q \rightarrow \neg p$

Proof of “For any real x, $\leftrightarrow x^2 > x \leftrightarrow (x > 1 \vee x < 0)$”

$\boxed{\rightarrow}$ By contrapositive assume $\neg(x > 1 \vee x < 0)$, so $x \leq 1$ and $x \geq 0$, hence $x^2 \leq x$.

$\boxed{\leftarrow}$ By cases $x > 1 \rightarrow x^2 > x$ and $x < 0 \rightarrow x < 1 \rightarrow x^2 > x$.

III.C.4.vii Chains of iffs

Proving an iff $p \leftrightarrow q$ can be done by introducing intermediate propositions that are also equivalent to p and q: $p \leftrightarrow r_1 \leftrightarrow r_2 \leftrightarrow \cdots \leftrightarrow r_n \leftrightarrow q$ So in this case we have a chain of iffs.

The idea is that every equivalence $p \leftrightarrow r_1$, $r_i \leftrightarrow r_{i+1}$, $r_n \leftrightarrow q$, should be easy, or even trivial.

This is what happens when solving equations: $2x + 3 = 0 \leftrightarrow 2x = -3 \leftrightarrow x = -\frac{3}{2}$

III.C.4.viii Chains of iffs using implications

In the case of a proof of $p \leftrightarrow q \leftrightarrow r$ (or more than 3), the several iffs can be proved using a *chain of implications* that loops back to the first proposition:

$$p \rightarrow q \rightarrow r \rightarrow p$$

Note that in this case all the propositions are equivalent to each other. It is also implied that all equivalences are interesting (which is not the case of using a chain of trivial iffs to prove a single interesting iff).

III.C.5 Remarks

III.C.5.i All the other rules

Not all rules of ND have been used in the presentation of the proof patterns. One such rule is the conjunction introduction. It can actually be presented as a proof pattern, but it is not really interesting: to prove $p \wedge q$, prove p and prove q.

Other rules are actually used as *instantiation*: all these rules are used when using another result (usually a theorem). It may be the quantifier instantiation rules or the modus ponens, or even the weakening rules of conjunction elimination (if you have $p \wedge q$, you have p *in particular*) and disjunction introduction (if you have p you have $p \vee q$).

III.C.5.ii On the boldness of arrows

Some texts use $\Rightarrow$ to mean implication, instead of $\rightarrow$, and the symbol $\Leftrightarrow$ to mean iff, instead of $\leftrightarrow$.

This difference comes from the difference of worlds that these arrows live in: in mathematics, the “double” version is used, while the simple version denotes the logical connectors (as in proof systems). They are indeed linked, but *technically* they are different.

In the remainder of this book, I'll use $\Rightarrow$ and $\Leftrightarrow$ in math (and $\rightarrow$ and $\leftrightarrow$ in formulas only)

III.C.5.iii Practicing proof techniques

Apart from the exercise below, applying the proof techniques discussed in this chapter make little sense without an actual application domain. As a result, the remainder of this book will be the applications of the proof techniques: every proof written will try to emphasize the proof patterns used, and it is expected that exercises are solved in the same manner.

Exercise III.4

For each of the following statement:

- Identify the statement's structure by transforming it into a first-order formula.
- Prove the statement, indicating the proof technique being used.

1. There is a solution to the system $\begin{cases} 4x + 5y = 2 \\ -2x + y = 6 \end{cases}$
2. For any real x, there is a value y such that $\cos^2(x) + y^2 = 1$.
3. There are no integers n and p such that $6n + 3p = 2$.
4. For any integers n and p, such that one is even and the other is odd, their sum is odd.

Chapter IV
Set Theory

Chapter contents

IV.A Sets over a domain **50**

IV.A.1 Definition, vocabulary, notations 50

IV.A.1.i The writing of sets 50

IV.A.1.ii Usual domains 50

IV.A.2 Boolean algebra on sets 51

IV.A.2.i Sets and first-order logic 51

IV.A.2.ii Boolean operators for sets 51

IV.A.2.iii Venn Diagrams 53

IV.A.2.iv Proving theorems about sets 53

IV.A.3 Cardinality 56

IV.A.4 Disjointedness, coverings, and partitions 57

IV.A.4.i Disjoint sets 57

IV.A.4.ii Coverings 57

IV.A.4.iii Partitions and Disjoint Unions 58

IV.B Sets of sets **58**

IV.B.1 The nature of the elements 58

IV.B.2 The powerset 59

IV.B.3 Defining natural numbers from sets 60

IV.C Sets of tuples **60**

IV.C.1 Cartesian products 61

IV.C.1.i Definition 61

IV.C.1.ii Cardinality 61

IV.C.1.iii Examples 62

IV.C.2 Relations 62

IV.C.2.i Definition 62

IV.C.2.ii Examples 63

IV.C.2.iii Projection 63

IV.C.2.iv Relations in relational databases 64

IV.C.3 Functions 64

IV.C.3.i Definition and notations 64

IV.C.3.ii Special cases for domains and co-domain 65

IV.C.3.iii Injection, surjection, bijection 67

IV.C.3.iv Injection, surjection, bijection and cardinality 68

IV.A Sets over a domain

IV.A.1 Definition, vocabulary, notations

The notion of set is fundamental to mathematics. It is actually possible to build all current mathematics from the concept of sets. So let's focus a little bit on the theory of set manipulation, starting with the very definition of sets.

Definition: Set

A *set* is an unordered collection of elements.

The elements of the set are written enclosed between { and }.

As such, this definition may bring more questions than it answers. So let's give more details on the three essential features of a set mentioned in the above definition:

Unordered means that order does not matter. So the set $\{1, 2, 3\}$ could also be written $\{2, 1, 3\}$, $\{1, 3, 2\}$, or using any other order of numbers 1, 2, and 3.

Collection means that every element can only appear once. So $\{1, 2, 1, 3\}$ is *not* a set.

✍ Sets with repetition are called multi-sets.

Elements are objects that belong to the domain. Which domain is being used is usually given as part of the definition of the set itself.

We write $x \in A$, read as "x belongs to A" to denote that x is an element of the set A. We write $x \notin A$ to denote that x is *not* an element of the set A. For example $2 \in \{1, 2, 3\}$, and $5 \notin \{1, 2, 3\}$.

IV.A.1.i The writing of sets

Defining and denoting a set can be done in several ways. The simplest is to simply give the list of all the elements of the set: this is called the *roster method*. For example: $A = \{3, 2, 12, 42\}$. One obvious limitation to this notation is that it only works with finite sets, and can easily get tedious for big sets.

It is therefore common to use ellipsis to denote a big, or even infinite, set. For example $I_{42} = \{1, 2, \dots, 42\}$ is the set of integers from 1 to 42; $M_7 = \{0, 7, 14, 21, \dots\}$ is the set of positive integers that are multiples of 7. It is up to the reader to replace the "…" with the actual elements. As a result, it can lead to ambiguity and should be used with care.

A better (because more formal) way to define sets, whether finite or infinite, is to use the *set-builder notation*: $\{x \mid P(x)\}$ where P is a unary predicate. In English, this notation is read as "the set of elements x such that $P(x)$". For example, $\{x \mid x > 5\}$ is "the set of numbers x such that $x > 5$".

IV.A.1.ii Usual domains

In this notation, the domain should also be given (so the above example is not completely correct as such). Usual domains are given abbreviations in order to fit in the notation and are separated from the variable using the $\in$ ("in") symbol.

$\mathbb{N}$, **Natural numbers:** $\{0, 1, 2, 3, \dots\}$ are whole positive numbers (and yes zero is included in the natural numbers). Using this domain, one can define the set of positive multiples of 7 as $M_7 = \{n \in \mathbb{N} \mid \exists k \in \mathbb{N}, n = 7k\}$. They are called natural because they are the simplest numbers that are used to count actual objects.

$\mathbb{Z}$, **Integers:** $\{0, 1, -1, 2, -2, 3, \dots\}$. Positive or negative whole numbers. Using this domain, one can define the set of perfect cubes as $C = \left\{n \in \mathbb{Z} \;\middle|\; \exists k \in \mathbb{Z}, n = k^3\right\}$.

$\mathbb{Q}$, **Rational numbers:** $\left\{\frac{p}{q} \;\middle|\; p, q \in \mathbb{Z} \text{ and } q \neq 0\right\}$, the numbers that can be written as the ratio of two integers. These are also numbers that can be written with a repeating pattern after the decimal point: for example $\frac{2}{7} = 0.285714285714\cdots = 0.\overline{285714}$. Using this domain, one can define the set of reciprocals of powers of two as follows: $\left\{q \in \mathbb{Q} \;\middle|\; \exists n \in \mathbb{N}, q = \frac{1}{2^n}\right\}$.

$\mathbb{R}$, **Real numbers:** formally, they are defined limits of sequences of rational numbers. These are also numbers that can be written with a decimal point, not necessarily with a repeating pattern: $\pi = 3.14159265358979323846264\,3\dots$ Using this domain, one can define the set of numbers greater than 5 as $\{x \in \mathbb{R} \mid x > 5\}$.

$\mathbb{C}$, **Complex numbers:** real numbers enriched with $i = \sqrt{-1}$, making it a two-dimensional domain. Complex numbers are outside the scope of this course, so this notation is mentioned for completeness' sake but will rarely be used.

Fun fact: the original notations for these domains are bold letters **N**, **Z**, **Q**, **R**, **C**. They were transcribed on the board of classrooms with double bars, and this notation is now common in typeset texts such as this one.

In this book I may also use $\mathbb{D}$ to denote an unspecified domain.

Exercise IV.1

Convert these informally described sets into a set-builder notation (you may need to extrapolate):

1. The set of real numbers between $-\sqrt{2}$ (included) and 3π (excluded).
2. Rational positive numbers.
3. $\{1, 3, 5, 7, 9, 11, 13, 15, \dots, 79\}$.

IV.A.2 Boolean algebra on sets

IV.A.2.i Sets and first-order logic

Sets and first-order predicates are intertwined, to the point that it isn't uncommon to read that "a first-order unary predicate *is* a set". Why is that so? In the set-builder notation, a predicate is used to define the set. But the relation also goes the other way around: a set A defines a unary predicate: $P_A(x)$ interpreted as $x \in A$: the predicate is true whenever x is an element of A.

For example, the set M_7 of positive multiples of 7 defines the predicate P_7 that is true whenever the number is a multiple of 7: $P_7(42)$ is true because $42 \in M_7$; $P_7(29)$ is false because $29 \notin M_7$.

IV.A.2.ii Boolean operators for sets

Since sets are unary predicates, whatever can be done with predicates can be done on sets. In particular, sets can be combined using boolean operators. Slightly different notations are used in order not to confuse the set and the underlying predicate, but the process really mimics the propositional Boolean Algebra of Section I.B. Technically, the operators below create another Boolean Algebra; the domain being the sets instead of the truth values $\top$ and $\bot$.

$\emptyset$, **The empty set.** It corresponds to predicate $\bot$: $\emptyset = \{x \in \mathbb{D} \mid \bot\}$.

$\mathbb{D}$, **The whole domain.** It corresponds to predicate $\top$: $\mathbb{D} = \{x \in \mathbb{D} \mid \top\}$.

$\cap$, **Intersection.** It corresponds to the conjunction: $A \cap B = \{x \in \mathbb{D} \mid x \in A \wedge x \in B\}$. An element belongs to the intersection if it belongs to both sets, and the intersection is the set of all such elements. For example $3 \in \{1, 2, 3\} \cap \{2, 3, 5, 7\}$, and $\{1, 2, 3\} \cap \{2, 3, 5, 7\} = \{2, 3\}$.

$\cup$, **Union.** It corresponds to the disjunction: $A \cup B = \{x \in \mathbb{D} \mid x \in A \vee x \in B\}$. An element belongs to the union if it belongs to at least one of the sets (like disjunction, union is inclusive), and the union is the collection of elements of both these sets. For example, $7 \in \{1, 2, 3\} \cup \{2, 3, 5, 7\}$, and $\{1, 2, 3\} \cup \{2, 3, 5, 7\} = \{1, 2, 3, 5, 7\}$.

$\overline{\ \ }$, **Complement.** It corresponds to the negation: $\overline{A} = \{x \in \mathbb{D} \mid \neg(x \in A)\}$. An element belongs to the complement if it does not belong to the set.

$\subseteq$, **Inclusion.** It corresponds to an implication: $A \subseteq B$ iff $\forall x, x \in A \rightarrow x \in B$. Contrary to the above operators, this operation returns a truth value, not a set. For example $\{3, 5, 7\} \subseteq \{1, 2, 3, 4, 5, 6, 7\}$ because every element of $\{3, 5, 7\}$ is also an element of $\{1, 2, 3, 4, 5, 6, 7\}$.

$=$, **Equality** It corresponds to iff: $A = B$ iff $\forall x, x \in A \leftrightarrow x \in B$. As for inclusion, this operator returns a truth value, not a set.

The operators above are the most common ones, that match the operators of boolean algebra. Other operators, which are more syntactical shortcuts than new definitions, are also used:

$\supseteq$, **Superset:** the reverse of inclusion: $A \supseteq B$ iff $B \subseteq A$.

$\not\subseteq$, **Non-inclusion:** the negation of inclusion $A \not\subseteq B$ iff it is not the case that $A \subseteq B$, i.e. $\exists x, x \in A \wedge x \notin B$.

$\subsetneq$, **Strict inclusion:** inclusion but not equality: $A \subsetneq B$ iff $A \subseteq B$ and $B \not\subseteq A$.

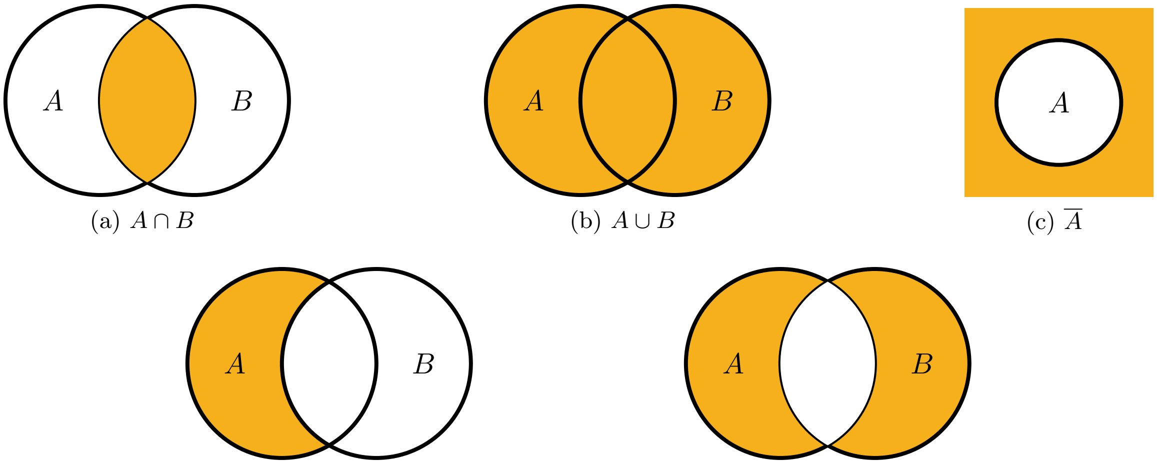
$\setminus$, **Relative complement:** $A \setminus B = \{x \in \mathbb{D} \mid x \in A \wedge \neg x \in B\} = A \cap \overline{B}$ removes from A all elements of B. For example $\{1, 2, 3, 4, 5, 6\} \setminus \{2, 3, 5, 7\} = \{1, 4, 6\}$.

$\triangle$, **Symmetric difference:** $A \triangle B = \{x \in \mathbb{D} \mid x \in A \oplus x \in B\} = \{x \in \mathbb{D} \mid (x \in A \vee x \in B) \wedge \neg(x \in A \wedge x \in B)\} = (A \cup B) \setminus (A \cap B)$ contains elements that are in A or in B, but not both.

Exercise IV.2

Write the result of the following set operations as a roster or set-builder notation:

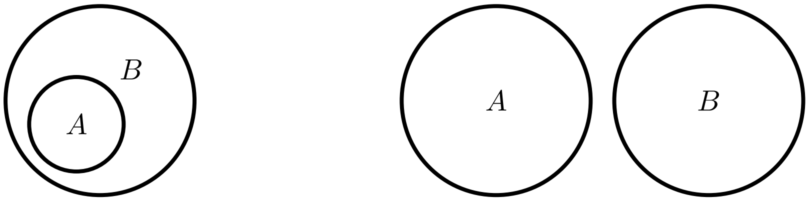
1. $A = \{2, 3, 5, 7, 11, 13, 17, 19\} \cap \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19\}$
2. $B = \overline{\{x \in \mathbb{R} \mid x^2 > 5\} \cup \{x \in \mathbb{R} \mid x^2 < 5\}}$



(a) $A \cap B$ (b) $A \cup B$ (c) $\overline{A}$

(d) $A \setminus B$ (e) $A \triangle B$

Figure IV.1: Venn diagrams for operations on sets. In gold is the result of the operation.



(a) $A \subseteq B$ (b) $A \cap B = \emptyset$

Figure IV.2: Venn diagrams for relations of sets.

IV.A.2.iii Venn Diagrams

Venn diagrams are circles (or other shapes) that represent sets. Looking at unions or intersections of sets on such diagrams can help understand the problem at hand. It does not replace a proof but drawing such diagrams on draft paper may help writing it.

In Figure IV.1 are represented in Venn diagram form the result of the main operations for two sets. In some cases there are more than two and the drawing becomes more complex, but also more necessary to understand the situation!

It is common that some hypotheses are given about sets. These situations can also be drawn as Venn diagrams, as is done for two example in Figure IV.2.

IV.A.2.iv Proving theorems about sets

Proving things about sets can be done using this underlying structure and the relation with the Boolean operations. Namely, whenever an inclusion $A \subseteq B$ needs to be proved, a universally quantified implication must be proved: take an element x, assume $x \in A$, and prove that $x \in B$. For a set equality, the inclusion must be proved in both directions.

In the case of sets built through operators, it is useful to rewrite the set definition as boolean operations on predicates. For example if there is an hypothesis that $x \in A_1 \cup A_2$, then $x \in A_1$ or $x \in A_2$, and in this case a proof by cases is probably the way to go.

While most proofs involve some domain-specific assumptions (usually implied or gathered in theorems that are known by both the writer and the reader), a couple of results can be proved in general about sets. Below are some examples and exercises of such proofs.

IV.A.2.iv.a $A \subseteq A$

The statement $A \subseteq A$ is a universally quantified implication: we need to prove $\forall x, x \in A \to x \in A$. As usual, we need to take an element of the domain, then assume the premise, and finally prove the conclusion, which is trivial in this case (which make the proof sound a bit strange). The proof is as follows:

Proof of $A \subseteq A$

Let x be an element of the domain. Assume $x \in A$. Then $x \in A$. □

IV.A.2.iv.b $A \subseteq \mathbb{D}$

As in the previous case, $A \subseteq \mathbb{D}$ is a universally quantified implication: we need to prove $\forall x, x \in A \to x \in \mathbb{D}$. The proof is also quite trivial:

Proof of $A \subseteq \mathbb{D}$

Let x be an element of the domain. Assume $x \in A$. Since x is in the domain we have $x \in \mathbb{D}$. □

IV.A.2.iv.c $\emptyset \subseteq A$

Expanding the implication, what needs to be proved is $\forall x, x \in \emptyset \to x \in A$.

Proof of $\emptyset \subseteq A$

Let x be an element of the domain. Assume $x \in \emptyset$. This is false ($\perp$), so it implies anything, in particular $x \in A$. □

IV.A.2.iv.d $A \cap B \subseteq A$

We need to prove $\forall x, x \in A \cap B \to x \in A$.

$A \cap B \subseteq A$

Let x be an element of the domain. Assume $x \in A \cap B$. By definition $x \in A$ and $x \in B$, so in particular $x \in A$. □

IV.A.2.iv.e Identities

Most identities of Boolean algebra carry over to set algebra. No proof will be given for these (some are asked as exercises), only the idea behind it by mapping the identity to the corresponding property of boolean algebra.

Identities for set algebra

- $A \cup B = B \cup A$ (by commutativity of $\vee$).
- $A \cap B = B \cap A$ (by commutativity of $\wedge$).
- $A \cup \overline{A} = \mathbb{D}$ (by the excluded middle).
- $A \cap \overline{A} = \emptyset$ (by contradiction).
- $\overline{\overline{A}} = A$ (by double negation).
- $\overline{A \cup B} = \overline{A} \cap \overline{B}$ (by De Morgan's Law).
- $\overline{A \cap B} = \overline{A} \cup \overline{B}$ (by De Morgan's Law).

IV.A.2.iv.f $A \cap B \subseteq A$

$A \cap B \subseteq A$ is a quantified implication $\forall x, x \in A \cap B \rightarrow x \in A$. The conclusion is proved from the premise using what resembles the conjunction elimination rule of Natural Deduction (see Chapter III).

Proof of $A \cap B \subseteq A$

Let x be an element of the domain. Assume $x \in A \cap B$. Then $x \in A$ and $x \in B$. In particular, $x \in A$. □

The similar (dual) $A \subseteq A \cup B$ is left as exercise.

IV.A.2.iv.g $A \subseteq B \Leftrightarrow \overline{B} \subseteq \overline{A}$

$A \subseteq B \Leftrightarrow \overline{B} \subseteq \overline{A}$ is an equivalence. If we break down both sides of the equivalence, what needs to be proved is that $(\forall x, x \in A \rightarrow x \in B) \leftrightarrow (\forall x, x \notin B \rightarrow x \notin A)$. Rewriting the $\notin$ as the negation of $\in$, we obtain: $(\forall x, x \in A \rightarrow x \in B) \leftrightarrow (\forall x, \neg(x \in B) \rightarrow \neg(x \in A))$. Using the contrapositive inside the right handside, we actually have to prove $(\forall x, x \in A \rightarrow x \in B) \leftrightarrow (\forall x, x \in A \rightarrow x \in B)$, which holds trivially (syntactically even).

The formal proof consists in these rewritings (although the above proof is completely valid and probably clearer since it includes English explanations, the proof below is technically sufficient):

Proof of $A \subseteq B \Leftrightarrow \overline{B} \subseteq \overline{A}$

$$\overline{B} \subseteq \overline{A} \Leftrightarrow (\forall x, x \notin B \rightarrow x \notin A) \Leftrightarrow (\forall x, \neg(x \in B) \rightarrow \neg(x \in A)) \Leftrightarrow (\forall x, x \in A \rightarrow x \in B) \Leftrightarrow A \subseteq B$$

□

Exercise IV.3

Prove that for any set A and B:

1. $A \subseteq A \cup B$
2. $\overline{A \cap B} = \overline{A} \cup \overline{B}$

Exercise IV.4

Prove that the four following statements are equivalent to each other:

$$A \subseteq B \qquad A \cap B = A \qquad A \cup B = B \qquad \overline{B} \subseteq \overline{A}$$

Hint: You must prove the following chain of equivalences:

$$A \subseteq B \Leftrightarrow A \cap B = A \Leftrightarrow A \cup B = B \Leftrightarrow \overline{B} \subseteq \overline{A}.$$

IV.A.3 Cardinality

The cardinality of a set is its size:

Definition: Cardinality

The *cardinality* of a set is the number of elements it contains. The cardinality of A is written $|A|$.

For example $|\{1, 2, 5, 42\}| = 4$; $|\{\top, \bot\}| = 2$. The empty set, with no elements, has cardinality 0: $|\emptyset| = 0$. A set with cardinality 1, i.e. containing a single element (for example $|\{78\}| = 1$) is called *singleton*.

While it is easy to find the cardinality of finite sets in roster notation (it is only a matter of counting, it may become harder when the set is given in the set-builder notation. For example, for the set $\{x \in \mathbb{N} \mid x > 12 \land x \leq 73\}$, we must reason that we have all integers between 12 excluded and 73 included. That means $73 - 12 = 61$ element, therefore $|\{x \in \mathbb{N} \mid x > 12 \land x \leq 73\}| = 61$.

For finite sets, the counting eventually stops to an actual number (which may be very big). For infinite sets, the counting of the element never stops. The cardinality of an infinite set is not a number, but $+\infty$. That is actually the definition of finite or infinite set:

Finite *vs* infinite set

A finite set is a set whose cardinality is not $+\infty$: a set A is finite if $|A| \in \mathbb{N}$.

All the usual domains are infinite: $|\mathbb{N}| = +\infty$, $|\mathbb{Z}| = +\infty$, $|\mathbb{Q}| = +\infty$, $|\mathbb{R}| = +\infty$, $|\mathbb{C}| = +\infty$. It can also be the case for some other sets, for example set $M_7 = \{n \in \mathbb{N} \mid \exists k \in \mathbb{N}, n = 7k\}$ is infinite, so $|M_7| = +\infty$. There are also infinitely many elements in the interval of reals between 0 and 1 (usually written $[0, 1]$): $|\{x \in \mathbb{R} \mid 0 \leq x \leq 1\}| = +\infty$.

Intuitively, there should be fewer elements in M_7 than in the whole domain $\mathbb{N}$. Is that really the case? And how dos the size of $[0, 1]$ compare to $\mathbb{R}$? and to $\mathbb{N}$? In Section IV.C.3.iv, we will discuss how to compare cardinalities of infinite sets.

Exercise IV.5

Give the cardinality of all the sets of Exercises IV.1 and IV.2.

IV.A.4 Disjointedness, coverings, and partitions

IV.A.4.i Disjoint sets

Sets that don't have any common element are called disjoint. This notion can also be generalized to more than two sets, requiring than no element is in more than one of the set. This situation is illustrated with a Venn diagram in Figure IV.2(b).

Definition: Disjoint sets

- Two sets A and B are *disjoint* if $A \cap B = \emptyset$.
- A collection of sets $A_1, A_2, \ldots, A_n$ are *pairwise disjoint* if for any $i, j \in \{1, \ldots, n\}$, $A_i \cap A_j = \emptyset$.

For example $\{n \in \mathbb{Z} \mid n < 0\}$ and $\{n \in \mathbb{Z} \mid \exists p \in \mathbb{Z}, n = p^2\}$ are disjoint. And intervals $[-3, 5)$, $[5, 12]$, $[4\pi, 42)$ are pairwise disjoint.

Proofs of disjointedness can be either direct or by contradiction. In a direct proof, assume that an element is in one of the set and prove that it is not in the other. In a proof by contradiction, assume that there is an element that is in two sets and prove a contradiction. When there are more than two sets there are more proofs required: one for each pair of sets; although in some cases a proof by contradiction will work for all pairs at once.

IV.A.4.ii Coverings

A notion dual to disjointedness is coverability: several sets cover a single set X if all the elements of X can be found in at least one of the sets. This notion can also be defined for more than two sets.

Definition: Covering

- Two sets A and B *cover* a set X if $A \cup B \supseteq X$.
- Two sets A and B *exactly cover* a set X if $A \cup B = X$.
- A collection of sets $A_1, A_2, \ldots, A_n$ *cover* a set X if $A_1 \cup A_2 \cup \cdots \cup A_n = \bigcup_{i=1}^{n} A_i \supseteq X$.
- A collection of sets $A_1, A_2, \ldots, A_n$ *exactly cover* a set X if $A_1 \cup A_2 \cup \cdots \cup A_n = \bigcup_{i=1}^{n} A_i = X$.

✍ The $\bigcup_{i=1}^{n} A_i$ notation is read "the union for i from 1 to n of the A_is".

For example, $\{0, 1, 2, 4\}$ and $\{1, 3, 5\}$ cover $\{1, 2, 3, 4, 5\}$. And $\{1, 4\}$, $\{1, 2, 3\}$ and $\{2, 5\}$ exactly cover $\{1, 2, 3, 4, 5\}$.

To prove that a set is covered, any element must be shown to be in at least one of the sets. As a result, proofs by cases that will match the sets are usually the way to go for these properties.

IV.A.4.iii Partitions and Disjoint Unions

When a group of sets are both disjoint and exactly cover a set, they form a partition of the set.

Definition: Partition and Disjoint Union

- Two disjoint non-empty sets A and B that exactly cover X form a *partition* of X. We then write $X = A \uplus B$ and X is called the *disjoint union* of A and B.
- A collection of pairwise disjoint non-empty sets that exactly cover X form a *partition* of X: $X = A_1 \uplus A_2 \uplus \cdots \uplus A_n = \biguplus_{i=1}^{n} A_i$ is called the *disjoint union* of all the A_is.

For example $\{0\}$ and $\{x \in \mathbb{Z} \mid x \neq 0\}$ form a partition of $\mathbb{Z}$. And $\{x \in \mathbb{N} \mid x \text{ is even}\}$ and $\{x \in \mathbb{N} \mid x \text{ is odd}\}$ is a partition of $\mathbb{N}$. With more than two sets, $\{0\}$, $\{x \in \mathbb{R} \mid x > 0\}$, $\{x \in \mathbb{R} \mid x < 0\}$ is a partition of $\mathbb{R}$.

The extra condition that excludes empty sets is added to the definition mostly to facilitate combinatorics theorems such as counting the number of partitions of a finite set.

See Chapter VIII for an introduction to combinatorics.

Exercise IV.6

Let $\mathbb{Z}$ be the domain of work. For each collection of sets below, determine whether they are:

- pairwise disjoint, $(A \cap B = \emptyset, (A \cap C = \emptyset,\ B \cap C = \emptyset))$
- covering $\mathbb{Z}$, $(A \cup B\ (\cup\ C) = \mathbb{Z})$ It is always the case that $A \cup B\ (\cup\ C) \subseteq \mathbb{Z}$; what remains to be proved to be a covering is that $\mathbb{Z} \subseteq A \cup B\ (\cup\ C)$.
- conclude about whether they form a partition of the domain.

1. $A = \{n \in \mathbb{Z} \mid n > 37\}$ and $B = \{n \in \mathbb{Z} \mid n \leq -12\}$
2. $A = \{n \in \mathbb{Z} \mid n < 37\}$ and $B = \{n \in \mathbb{Z} \mid n \geq -12\}$
3. $A = \{n \in \mathbb{Z} \mid n > 37\}$, $B = \{n \in \mathbb{Z} \mid n \leq -12\}$, and $C = \{-11, -10, \ldots, 36, 37\}$

IV.B Sets of sets

IV.B.1 The nature of the elements

The elements in a set can be of any nature. Although the most common objects in mathematics and the first studied in the curricula are numbers, it is possible to have other values: for example the set $\{\top, \bot\}$ contains the boolean values.

The objects need not be very precisely defined, as long as they have a name. Hence it is perfectly possible to declare a set $\{a, b, c, d\}$ containing four objects.

Or the objects can be sets themselves. In terms of logics, as a set is a predicate, a set of set is a second-order predicate. Since we allow also to have sets of sets of sets, etc, this means we are working in higher-order logic.

See Section II.B.7.iii for a discussion of the orders of logic.

Let's take an example: $X = \big\{\{1,2,3,4\}, \{2,3,4,5\}, \{3,4,5,6\}\big\}$. This set X has 3 elements ($|X| = 3$), that happen to be sets. We can think of it as $X = \{A, B, C\}$ with

$A = \{1,2,3,4\}$, $B = \{2,3,4,5\}$, and $C = \{3,4,5,6\}$. So $\{1,2,3,4\} \in X$, since this set is an element of X. But $\{1,2,3,4\} \not\subseteq X$! because 1 for example, is not an element of X.

One must pay attention to what the elements are in a set. Sometimes the size of the brackets can help, but the typesetting is not guaranteed to be precise enough to discriminate. And it can be tricky, since non-homogeneous sets are allowed as well (even though their use is quite rare): $\Big\{\{1,3\}, 2, \{5,7,13,\{2,8,16\}\}\Big\}$ is a completely valid set, containing elements that are sets (some of them also containing sets. . .) and some numbers.

Exercise IV.7

For each of the following statements, determine whether it is true or false (justify your answer):

1. $\emptyset \in \{\{0,1\},\{3,7\}\}$
2. $\emptyset \subseteq \{\{0,1\},\{3,7\}\}$
3. $3 \in \{\{0,1\},\{3,7\}\}$
4. $\{0,1\} \subseteq \{\{0,1\},\{3,7\}\}$
5. $\{\{3,7\}\} \subseteq \{\{0,1\},\{3,7\}\}$
6. $\{0,1\} \in \{\{0,1\},\{3,7\}\}$

IV.B.2 The powerset

One particular set of sets, is, given a set, the set of all its subsets.

Definition: Powerset

For a set A, the *powerset* of A, written $\mathcal{P}(A)$ is the set of subsets of A: $\mathcal{P}(A) = \{X \mid X \subseteq A\}$.

For example $\mathcal{P}(\mathbb{N}) = \{A \mid A \subseteq \mathbb{N}\}$ is the set of subsets of $\mathbb{N}$. We have $\emptyset \in \mathcal{P}(\mathbb{N})$, $\mathbb{N} \in \mathcal{P}(\mathbb{N})$, $\{2,3,5,7,11\} \in \mathcal{P}(\mathbb{N})$, $\{k \in \mathbb{N} | k \text{ is even}\} \in \mathcal{P}(\mathbb{N})$, $\{k \in \mathbb{N} | \exists j \in \mathbb{N}, j^2 = k\} \in \mathcal{P}(\mathbb{N})$ (among others).

While the powerset of A depends on the elements of A, in all cases $\emptyset \in \mathcal{P}(A)$ and $A \in \mathcal{P}(A)$: this was proved when showing that $\emptyset \subseteq A$ (Section IV.A.2.iv.c) and $A \subseteq A$ (Section IV.A.2.iv.a), respectively.

In general, an element X of $\mathcal{P}(A)$ can be constructed by choosing, for each element of A, whether it will be in X or not. For the finite set $\{1,2,3\}$, if none are chosen to be in the set then the empty set is created, if only 1 is chosen then $\{1\}$ is created, if 1 and 3 are chosen then $\{1,3\}$ is created, and so on until $\{1,2,3\}$ is created when all elements are chosen to be in the set. As a result we build $\mathcal{P}(\{1,2,3\}) = \big\{\emptyset, \{1\}, \{2\}, \{3\}, \{2,3\}, \{1,3\}, \{1,2\}, \{1,2,3\}\big\}$.

This construction also provides the cardinality of the powerset: since for each element there is one choice (two options), there are $2^{|A|}$ possible elements in the powerset. Knowing this cardinality can also help control that there is no missing element (set) when building the powerset. In the example above, we count $8 = 2^3$ elements in $\mathcal{P}(\{1,2,3\})$, which is consistent with $|\{1,2,3\}| = 3$.

Properties of the powerset

For any set A:

- $\emptyset \in \mathcal{P}(A)$
- $A \in \mathcal{P}(A)$
- $|\mathcal{P}(A)| = 2^{|A|}$

Exercise IV.8

Calculate $\mathcal{P}(\{\top, \bot\})$. Verify your answer using the cardinality.

IV.B.3 Defining natural numbers from sets

ℹ This is for your personal culture more than to be applied in this course.

It was mentioned in this chapter's introduction that sets are at the basis of mathematics. Indeed, there is a way to define natural numbers from sets.

This construction, due to John Von Neumann, does not appear really natural at first if given directly. It may be clearer if we go backwards from what we want: the natural numbers. One way to see natural numbers is to consider that you start from nothing, 0, then add 1, to obtain 1. If you add 1 again, you get 2, and so on. So natural numbers are constructed from zero and the ability to get to the next number: the *successor* operation, usually noted S: $S(0) = 1$, $S(1) = S(S(0)) = 2$, and so on. This approach, called Peano arithmetic, does construct the natural numbers, but requires the zero to start the process.

The idea of Von Neumann is to use sets instead of numbers: the number n will just be a set of cardinality n. So number 0 is the empty set $\emptyset$.

Because numbers are sets, the successor operation S needs to be defined using these sets as well: an operation that adds one element to the set, in order to have the cardinality increase by one. That element is the set itself: $S(x) = x \cup \{x\}$.

The construction starts as follows:

$$\begin{aligned}
&0 = \emptyset \\
&1 = S(0) = \emptyset \cup \{\emptyset\} = \{\emptyset\} = \{0\} \\
&2 = S(1) = \{0\} \cup \{\{0\}\} = \{0, \{0\}\} = \{0, 1\} \\
&3 = \{0, 1\} \cup \{\{0, 1\}\} = \{0, 1, \{0, 1\}\} = \{0, 1, 2\} \\
&\ \vdots
\end{aligned}$$

Note that the sets thus produced are highly non-homogeneous: 0 is a set, 1 is a set of sets, 2 is a set containing both sets and sets of sets, etc.

And since all numbers are created from the natural numbers, we can see all numbers as being just sets.

IV.C Sets of tuples

Among the variety of objects that can go in a set are *couples*, and more generally *tuples*.

A couple is an ordered pair of elements, noted between parentheses: each of the two *component* has as distinct role, and cannot be swapped with its neighbor. For example the couple $(1, 2)$ is different from the couple $(2, 1)$. One of the best know use of couples is for coordinates on a plane: the first component is the x-coordinate (a.k.a abscissa, horizontal) while the second component is the y-coordinate (a.k.a. ordinate, vertical), and point $(1, 2)$ is different from point $(2, 1)$.

Tuples are a generalization of couples to more than two components. Other name for tuples include n-uples, where n can either be replaced with an actual number (then couples are 2-uples) or left unspecified.

IV.C.1 Cartesian products

IV.C.1.i Definition

Since each component of a tuple is relatively independent from the other, it is not uncommon for each component to take value over a different domain. For example, $(\top, 3)$ is a couple where the first component is a boolean value and the second a number.

The tuples are therefore elements of a domain which is built as the *Cartesian product* of the domains for each component. Sometimes referred to as the *cross product*, the name Cartesian is a reference to philosopher and mathematician René Descartes.

This idea also works for sets in the domains: a tuple is an element of the Cartesian product of several sets if each component belongs to the set in the product. Formally:

Definition: Cartesian product

- For two sets A and B, the Cartesian product of A and B is

  $$A \times B = \{(x, y) \mid x \in A \wedge y \in B\},$$

  the set of couples where the first component is an element of A and the second component an element of B.

- For n sets $A_1, \ldots, A_n$, the Cartesian product of the A_is is

  $$A_1 \times \cdots \times A_n = \{(x_1, \ldots, x_n) \mid \forall i \in \{1, \ldots, n\}, x_i \in A_i\},$$

  the set of tuples where the i-th component belongs to A_i.

Cartesian products over domains are frequently used to specify what goes into each component. For example, to define a triple containing an integer, a real, and a natural number, one can just write: “‘let $t \in \mathbb{R} \times \mathbb{Z} \times \mathbb{N}$”, or “let $(x, k, n) \in \mathbb{R} \times \mathbb{Z} \times \mathbb{N}$” in order to give a name to each of the components. This can be done for variables, which is often the case in proofs of universally quantified formula, for instance, or just to particular value, for example $(42, \pi, \top) \in \mathbb{N} \times \mathbb{R} \times \{\top, \bot\}$.

IV.C.1.ii Cardinality

In a Cartesian product, each component is independent. So in order to build an element of the product, one can choose an element in the first set, when there are $|A|$ choices, then an element in the second set, where there are $|B|$ choices. As a result, there are $|A| \cdot |B|$ choices for elements of the product. This can be generalized for products of more than two sets:

Cardinality of Cartesian Products

The cardinality of the Cartesian product is the product of the cardinalities of the sets. Namely:

- For two sets A and B, $|A \times B| = |A| \cdot |B|$.
- For n sets $A_1, \ldots, A_n$, $|A_1 \times \cdots A_n| = |A_1| \cdots |A_n| = \prod_{i=1}^{n} |A_i|$.

The $\prod_{i=1}^{n}$ notation is read “the product for i from 1 to n of…”.

IV.C.1.iii Examples

- $\{-1, 0, 1\} \times \{\top, \bot\} = \{(-1, \top), (0, \top), (1, \top), (-1, \bot), (0, \bot), (1, \bot)\}$. We can check that cardinalities match: $|\{-1, 0, 1\} \times \{\top, \bot\}| = 3 \cdot 2 = 6$
- The coordinates of a point is the plane is an element of $\mathbb{R} \times \mathbb{R}$ (also written $\mathbb{R}^2$). In space it is in $\mathbb{R} \times \mathbb{R} \times \mathbb{R} = \mathbb{R}^3$.
- The product of two finite sets:

$$\begin{array}{rcllll} \{0, 1, \ldots, n\} \times \{0, 1, \ldots, p\} & = & \{(0, 0), & (0, 1), & \ldots, & (0, p), \\ & & (1, 0), & (1, 1), & \ldots, & (1, p), \\ & & \vdots & \vdots & \ddots & \vdots \\ & & (n, 0), & (n, 1) & \ldots, & (n, p)\} \end{array}$$

  Although we used an ellipsis, we can see based on the rectangular display of the set that the cardinality is indeed $(n + 1) \cdot (p + 1)$.
- $\{p \in \mathbb{N} \mid p \text{ is even}\} \times \{p \in \mathbb{N} \mid p \text{ is odd}\} = \{(p_1, p_2) \in \mathbb{N} \times \mathbb{N} \mid p_1 \text{ is even and } p_2 \text{ is odd}\}$, contain for example the elements $(0, 1)$, $(2, 7)$, $(42, 93)$.

Exercise IV.9

1. Calculate $\{\top, \bot\} \times \{a, b, c\}$.
2. Give three (distinct) elements of $\mathbb{Z} \times \{x \in \mathbb{N} \mid \exists y \in \mathbb{N}, y^2 = x\}$.
3. How many elements are in $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \times \{a, b, c, \ldots, z\}$? (Do not build the whole set!)

IV.C.2 Relations

IV.C.2.i Definition

As remarked above, in Cartesian products each component is independent from the others. That is useful when *all* tuples of a certain type have to be considered, but there is relatively few information in it.

In *relations*, on the other hand, there are only *some* of the tuples. The choice of these tuples carries some information that goes beyond the type of each component. As for sets over a domain, choosing which values are included and which are not can be done through a predicate. In this case, the arity of the predicate will depend on the number of components. Binary relations are the most common relations and some particular types of binary relations will be studied in more details in Section IV.C.3 and Chapter V.

Definition: Relation

- A *binary relation* is a subset of a Cartesian product: $R \subseteq \mathbb{D}_1 \times \mathbb{D}_2$. It can be defined by a binary predicate: $\{(x, y) \in \mathbb{D}_1 \times \mathbb{D}_2 \mid P(x, y)\}$.
- A *relation* is a subset of a Cartesian product : $R \subseteq \mathbb{D}_1 \times \cdots \times \mathbb{D}_n$. It can be defined by an n-ary predicate: $\{(x_1, \ldots, x_n) \in \mathbb{D}_1 \times \cdots \times \mathbb{D}_n \mid P(x_1, \ldots, x_n)\}$.

For binary relations, $(x, y) \in R$ is read "x is in relation with y by R". Sometimes an *infix* notation is used and we write xRy. Usually for infix notations the relation is named with a symbol rather than a letter (see Chapter V for several examples of such relations).

As for sets, the link between relations and predicates goes both ways: a relation defined a predicate that is true when the tuple is in the relation.

Note that by definition a Cartesian product is a relation, although a relation need not be a Cartesian product.

IV.C.2.ii Examples

- $Pol = \{(n, p) \in \mathbb{Z} \times \mathbb{Z} \mid n^3 = 3p^2 + 5\}$ relates only the integers n and p that verify the specific equality $n^3 = 3p^2+5$. One such pair is $(2, 1)$, because $2^3 = 8$ and $3 \cdot 1^2 + 5 = 8$. Another one is $(8, 13)$, because $8^3 = 512$ and $3 \cdot 3 \cdot 13^2 + 5 = 3 \cdot 169 + 5 = 507 + 5 = 512$. (Other pairs include $(2, -1)$ and $(8, -13)$; there does not seem to be any other pair in this example, although it may not be easy to prove formally.)

- Relation $Rec = \{(n, q) \in \mathbb{Z} \times \mathbb{Q} \mid n \neq 0 \wedge q = \frac{1}{n}\}$ relates non-zero integers and their reciprocal.

- $L = \{(x, y) \in \mathbb{R} \times \mathbb{R} \mid y = 2x + 3\}$ defines a line in the plane.

- $Pyth = \{(a, b, c) \in \mathbb{N}^3 \mid a^2 + b^2 = c^2\}$ is a ternary relation that contains all Pythagorean triples: integer values that could be legs and hypotenuse of a right triangle.

- $Abs = \{(n, p) \in \mathbb{Z}^2 \mid n^2 = p^2\}$ relates integers that are either equal or opposite ($n = p$ or $n = -p$); otherwise said, it relates integers with the same absolute value.

- Relation $Leq = \{(n, p) \in \mathbb{Z}^2 \mid \exists k \in \mathbb{N}, n + k = p\}$ relates n to any integer that can be obtained by adding another positive integer to n. This is a convoluted way of saying that it relates n to any number greater than or equal to n. So n and p are in this relation if and only if $n \leq p$. Note that this is a way to formally define what it means to be greater than or equal to, and this relation Leq is represented by the infix symbol $\leq$.

IV.C.2.iii Projection

Projection of a tuple onto some of its components means ignoring the others. For example projecting $(1, 2)$ over the first component yields 1. And projecting $(\pi, 42, \frac{2}{3})$ over it's first and third components yields $(\pi, \frac{2}{3})$.

This notion is extended to sets: the projection of a relation onto some components is the relation made of the projected tuples over this component. To know whether a tuple is the projection of another one, the original tuple must have been in the relation. The missing components have to be filled with *some* values in order to build a tuple of the original relation. This is done through an existential quantifier. In the case of binary relations, this formally means that the projection $R \subseteq \mathbb{D}_1 \times \mathbb{D}_2$ over the first component is the set $\{x \in \mathbb{D}_1 \mid \exists y \in \mathbb{D}_2, (x, y) \in R\}$.

In the case of more than two components, the projection is defined the same way, although the notations are a bit more involved in order to separate the components that are preserved through projection and the ones that are not. The definition below is given more for completeness' sake that to be actually applied in the context of this book or course. The

$T =$

firstName	lastName	phoneNumber
John	Doe	1234567890
Jane	Doe	3456789120
Bob	Smith	1723456890
⋮	⋮	⋮

Figure IV.3: A table (relation) in a relational database.

projection of $R \subseteq \mathbb{D}_1 \times \cdots \times \mathbb{D}_n$ onto the set of components $B = \{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$ (with $i_1 < \cdots < i_k$) is the relation

$$\{(x_{i_1}, \dots, x_{i_k}) \in \mathbb{D}_{i_1} \times \cdots \times \mathbb{D}_{i_k} \mid \exists y_1 \in \mathbb{D}_1, \dots, \exists y_n \in \mathbb{D}_n, (\forall i \in B, x_i = y_i) \wedge (y_1, \dots, y_n) \in R\}$$

Note that for the special case of a Cartesian product, the projection of $A \times B$ over the first component is the whole set A. Or for more than two components, the projection of a Cartesian product $\mathbb{D}_1 \times \cdots \times \mathbb{D}_n$ onto the set of components $B = \{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$ (with $i_1 < \cdots < i_k$) is the Cartesian product $\mathbb{D}_{i_1} \times \cdots \times \mathbb{D}_{i_k}$.

In this book, the projection on the i-th component is written σ_i (although notations can vary, this one is pretty frequent in the literature). For example, $\sigma_2(\{(x, y) \in \mathbb{N}^2 \mid x^2 = y\}) = \{y \in \mathbb{N} \mid \exists x \in \mathbb{N}, x^2 = y\}$ is the set of squares.

Exercise IV.10

1. Give three (distinct) elements of $\{(n, x) \in \mathbb{N} \times \mathbb{R} \mid n = x^2 + 1\}$.
2. Calculate the projection on the first component of $\{(x, y) \in \mathbb{N} \times \mathbb{N} \mid x = 2y + 1\}$. Give a set-builder notation and an English description.

IV.C.2.iv Relations in relational databases

This is for your personal culture more than to be applied in this course.

Some database systems, among which all the system of the SQL family are called *relational databases*. Although it is convenient to think of the data in these databases as tables where the data is organized in columns (see Figure IV.3), a table is actually a relation.

The type of each column is given by a domain (words , numbers, numbers with k digits…). The table itself is a subset of the Cartesian product of the domains, i.e. a relation: $T \subseteq \mathit{words} \times \mathit{words} \times \mathit{digits}^{10}$, where T is the table of Figure IV.3. As a result, a line of such a table is called a tuple in the database nomenclature.

Extracting data from such a database by finding tuples that satisfy a criterion (for example **lastName** being "Doe") using an SQL query is like building a subset based on that relation: the SQL query `SELECT * FROM T WHERE lastName="Doe"` corresponds to the set $\{(f, l, p) \in T \mid l = \text{“Doe”}\}$.

IV.C.3 Functions

IV.C.3.i Definition and notations

The great strength of relation is that it allows to relate values together. One particular way to relate values is to attach to each value of one domain to a single corresponding value in the other domain. That is the principle of *functions*: to each value associate, usually through calculation, one particular *image*.

Definition: (Partial) Function

A function is a binary relation $f \subseteq \mathbb{D}_1 \times \mathbb{D}_2$ where each element of $\mathbb{D}_1$ is in relation with *at most one* element of $\mathbb{D}_2$: $\forall x \in \mathbb{D}_1, \exists y \in \mathbb{D}_2, (x, y) \in f \to \forall z \in \mathbb{D}_2, (x, z) \in f \to z = y$.

Examples of relations that are functions include $Rec = \{(n, q) \in \mathbb{Z} \times \mathbb{Q} \mid n \neq 0 \wedge q = \frac{1}{n}\}$ relates non-zero integers and their reciprocal and the linear function $L = \{(x, y) \in \mathbb{R} \times \mathbb{R} \mid y = 2x + 3\}$. On the other hand, relation $Abs = \{(n, p) \in \mathbb{Z}^2 \mid n^2 = p^2\}$ is not a function, since for example 1 is in relation both with 1 and -1.

In graphical form, this is illustrated in Figure IV.4. The relation of Figure IV.4(a) is not a function since 1 is in relation both with A and B. In Figures IV.4(b-f), the represented relations are function since every element of $\mathbb{D}_1$ is related to at most a single element of $\mathbb{D}_2$.

Functions are so widespread in mathematics that specific notations and vocabulary. The only value y that is in relation with x through f is called the *image* of x and is written $f(x)$. In the notation $f(x)$, x is the *argument* of function f. In that case x is then called a *pre-image* of y. There can be several pre-images; the *set* of preimages of y is denoted $f^{-1}(y)$.

Domain $\mathbb{D}_1$ is the *domain* of f. The projection $\sigma_1(f)$, the "first components" of f, is the *domain of definition*. It may not match exactly the domain. For example in function *Rec*, 0 is an element of the domain but not the domain of definition.

Domain $\mathbb{D}_2$ is the co-domain of f. The projection $\sigma_2(f)$, the "second components" of f, is called the *range*.

To indicate the domain and co-domain, it is customary to write: $f : \mathbb{D}_1 \to \mathbb{D}_2$ which is read as "f is a function from $\mathbb{D}_1$ to $\mathbb{D}_2$".

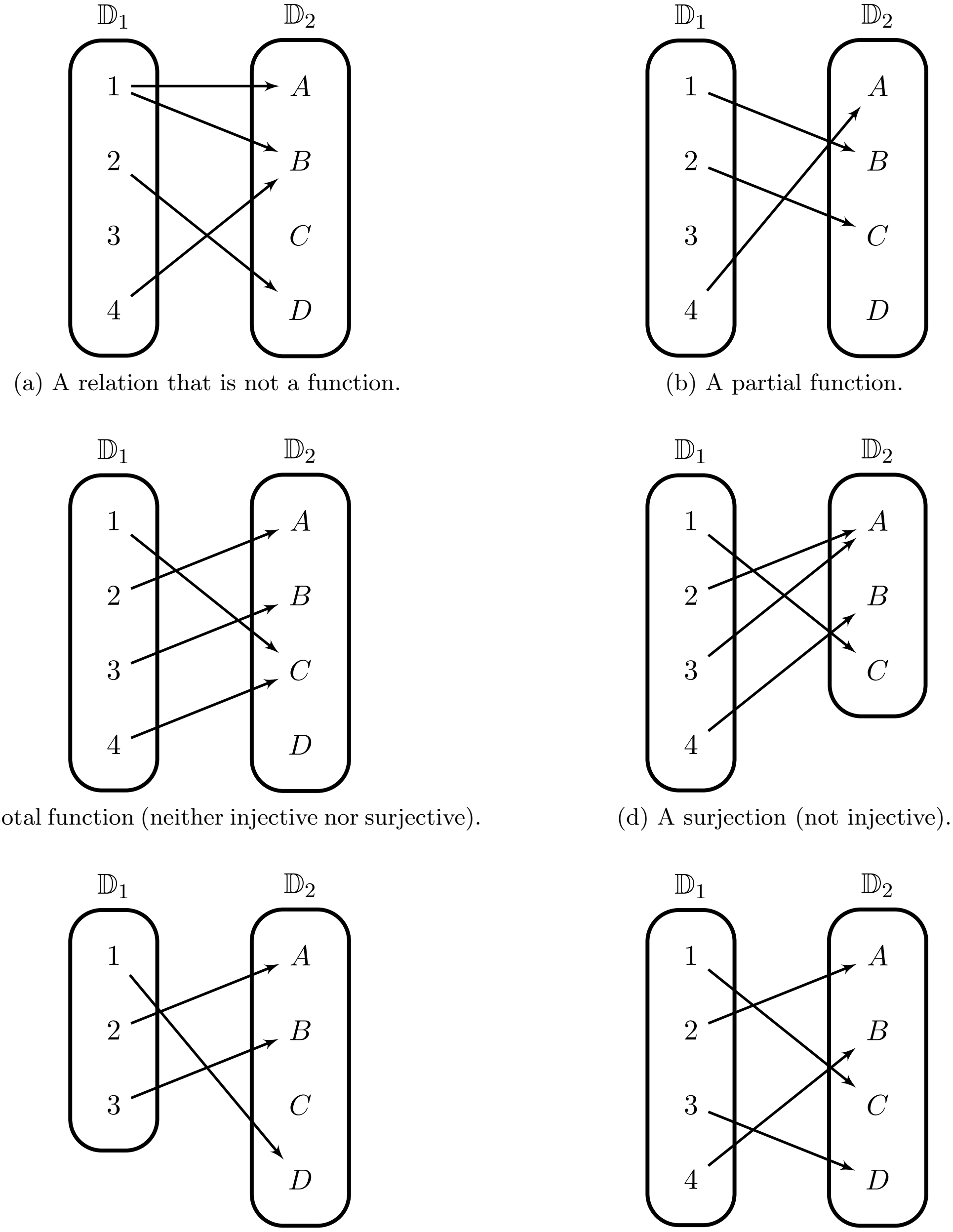
The arrow here is **not** an implication!

The notion of image is extended to subsets of the domain: $f(A) = \{f(x) \mid x \in A\}$. So in particular $f(\mathbb{D}_1)$ is the range of f. Similarly, the set of pre-images of a subset B of the codomain is $f^{-1}(B) = \{x \mid f(x) \in B\}$. So in particular $f^{-1}(\mathbb{D}_2)$ is the domain of definition of f.

IV.C.3.ii Special cases for domains and co-domain

IV.C.3.ii.a Functions from $\mathbb{R}$ to $\mathbb{R}$

The first relations encountered in the mathematics curriculum are relations over real numbers with real co-domain: relations in $\mathbb{R} \times \mathbb{R}$. In this case it is possible to graph all the points (ordered pairs of coordinates) in the plane. Using the *vertical line test*, it is the possible to check whether the relation is a function: a vertical line must not intersect the graph in more than one point. x-coordinates where a vertical line does intersect the graph once provide the domain of definition. Using a horizontal line can provide the range of the function: y-coordinates where the horizontal line intersects the graph at least once are in the range.



(a) A relation that is not a function.

(b) A partial function.

(c) A total function (neither injective nor surjective).

(d) A surjection (not injective).

(e) An injection (not surjective)

(f) A bijection: both injective and surjective.

Figure IV.4: Illustration of functions: total, injective, surjective, bijective.

IV.C.3.ii.b Functions with multiple arguments

Although functions are a special case of binary relations, it is possible to have more than one argument, and have a tuple as an image. In this case, the domain and co-domain are understood as a Cartesian product of other domains. A function f with n arguments can be defined as $f : \mathbb{D}_1 \times \cdots \times \mathbb{D}_n \to \mathbb{D}'_1 \times \cdot \times \mathbb{D}'_k$. In this case it means that for every tuple $(x_1, \ldots, x_2)$ there is at most one image $(y_1, \ldots, y_k)$. So although, technically, writing $f \subseteq \mathbb{D}_1 \times \cdots \times \mathbb{D}_n \times \mathbb{D}'_1 \times \cdot \times \mathbb{D}'_k$ and $f(\subseteq \mathbb{D}_1 \times \cdots \times \mathbb{D}_n) \times (\mathbb{D}'_1 \times \cdot \times \mathbb{D}'_k)$ is the same, separating domain and co-domain allows to determine on which domain the criterion must apply.

IV.C.3.iii Injection, surjection, bijection

In this section, we assume that $f : \mathbb{D}_1 \to \mathbb{D}_2$ is a function.

IV.C.3.iii.a Total functions

When the domain and the domain of definition coincide, the function is *total*. Formally, f is total if $\forall x \in \mathbb{D}_1, \exists y \in \mathbb{D}_2, (x, y) \in f$. Otherwise it is a *partial* function.

When f is total, we can also write $f \in \mathbb{D}_2^{\mathbb{D}_1}$ since for every element of $\mathbb{D}_1$ we have the choice of any element of $\mathbb{D}_2$. So for a finite domain A, with $|A| = n$, and co-domain B, with $|B| = p$, there exists p^n functions from A to B.

Function *Rec* above is partial since domain and domain of definition do not coincide: the function is not defined for 0. Function L, on the other hand, is total.

In Figure IV.4(b), the function is not total since 3 has no image. Functions represented in Figures IV.4(c-f), on the other hand, are all total since every element of the domain has an image.

IV.C.3.iii.b Surjections

When the co-domain and the range coincide, the function is *surjective*. Formally, $\forall y \in \mathbb{D}_2, \exists x \in \mathbb{D}_1, (x, y) \in f$.

Function *Rec* is not surjective, since not all rational numbers are reciprocals of an integer. For example $\frac{2}{5}$ is not in the range. Function L is surjective, since every number $y \in \mathbb{R}$ has a pre-image; namely, the preimage of y is the singleton $\left\{\frac{y-3}{2}\right\}$.

Functions represented in Figures IV.4(b,c,e) are not injective: there is at least an element (D, D, and C, respectively) that has no preimage. In Figures IV.4(d,f), the function is surjective.

IV.C.3.iii.c Injections

A total function is *injective* if every element of the co-domain has at most one pre-image. Otherwise said, two values of the domain cannot share an image. This is formalized by the formula $\forall x_1, x_2 \in \mathbb{D}_1, f(x_1) = f(x_2) \to x_1 = x_2$, which states that the only way for two values x_1, x_2 to have the same image is to be equal.

When a function is injective, the pre-image of any element of the range is a singleton. In that case it is customary, although a slight abuse of notation, to write $f^{-1}(y)$ for the pre-image itself. So for function L that is injective, one could write $L^{-1}(y) = \frac{y-3}{2}$.

In Figures IV.4(c,d), the functions are not injective since C has pre-images 1 and 4, and A has pre-images 2 and 3, respectively. In Figures IV.4(e,f), the depicted functions are injective.

IV.C.3.iii.d Bijections

A total function both injective and surjective is called *bijective*, (sometimes also referred to as a *one-to-one mapping*).

Bijective functions have an *inverse*: relation $f^{-1} : B \to A$ defined as $\{(y, x) \in B \times A \mid (x, y) \in f\}$. Relation f^{-1}:

- is a function because f is injective;
- is total because f is surjective;
- is surjective because f is total (which is a condition to be injective).

Since the inverse function associates to any element of B its pre-image, the same notation is used. The term *inverse* relates to the fact that if f is combined with f^{-1}, the identity function is created. In other terms, the effect of f and f^{-1} cancel each other. Formally: $\forall x \in A, f^{-1}(f(x)) = x$ and $\forall y \in B, f(f^{-1}(y)) = y$.

Function L, being both injective and surjective, is bijective. Its inverse function $L^{-1} : \mathbb{R} \to \mathbb{R}$ is defined as for any $y \in \mathbb{R}$, $L^{-1}(y) = \frac{y-3}{2}$. In Figure IV.4(f), the function is bijective.

Exercise IV.11

For each of the following relations, determine whether it is a function, and if so:

- whether it is a total function
  - if not give the domain of definition
  - if so whether it is injective
- whether it is surjective, if not give the range.

All your answers must be justified by a proof!

1. $\{(x, y) \in \mathbb{R} \times \mathbb{R} \mid 6x + 2y = 5\}$
2. $\{(x, y) \in \mathbb{R} \times \mathbb{R} \mid x \cdot y = 42\}$
3. $\{(x, y) \in \mathbb{R} \times \mathbb{R} \mid y = \cos(x)\}$
4. $\{(x, y) \in \mathbb{R} \times [-1, 1] \mid y = \cos(x)\}$
5. $\{(x, y) \in [0, \pi] \times [-1, 1] \mid y = \cos(x)\}$
6. $\{(x, y) \in \mathbb{R} \times \mathbb{R} \mid x = \cos(y)\}$

IV.C.3.iv Injection, surjection, bijection and cardinality

IV.C.3.iv.a Functions for cardinality comparison

In this section, assume $f : A \to B$ is a total function.

If f is injective, then every element of A points to a different element of B. So there must be at least as many elements in B as there are in A: $|A| \leq |B|$.

If f is surjective, then every element of B is pointed by at least an element of A. So there must be at least as many elements in A as there are in B: $|A| \geq |B|$.

If f is bijective, then we have both $|A| \leq |B|$ and $|A| \geq |B|$, therefore $|A| = |B|$.

Note that this idea works both for finite and infinite sets. And that provides a way to compare the cardinality of infinite sets.

Let A and B be two sets.

- $|A| \leq |B|$ iff there exists an injection from A to B.
- $|A| \geq |B|$ iff there exists a surjection from A to B.
- $|A| = |B|$ iff there exists a bijection between A and B.

IV.C.3.iv.b Proof techniques for cardinality comparison

So in order to prove that sets A and B have the same cardinality, there are several possibilities.

- Exhibit a bijection between A and B: find a function $f : A \to B$, prove that it is bijective.
- Or exhibit an injection from A to B and an injection from B to A: find a function $g : A \to B$ and a function $h : B \to A$; prove that both are injective.

Note that if $f : A \to B$ is bijective, then both f and f^{-1} are injective. Choosing $g = f$ and $h = f^{-1}$ would therefore follow the second approach. The advantage of the second approach is that h need not be the inverse of g, which is sometimes easier.

Proving that one set is strictly larger is often more involved: to prove $|A| > |B|$, it must be proved that *there is no injection* from A to B. Proving the absence of an object is difficult in a direct proof. The usual approach is to assume that there is an injection and show a contradiction.

IV.C.3.iv.c The pigeonhole principle

The above theorem on cardinality can also be used the other way round: if set A has more elements than set B, then there cannot be an injection from A into B. This idea is what is formalized in the *pigeonhole principle*: If there are n pigeons and $p < n$ pigeonholes, then there is at least a hole with two or more pigeons.

Formally, it can be reformulated as follows. Assume $|A| = n$ and $|B| = p$ with $p < n$, then:

- There is no injection from A to B.
- For any total function $f : A \to B$, there exists $x_1, x_2 \in A$ such that $x_1 \neq x_2$ and $f(x_1) = f(x_2)$.

This principle is a useful proof tool. For example, assume a set X is partitioned into n sets: $X = A_1 \uplus A_2 \uplus \cdots \uplus A_n$. For any $n+1$ elements of X, at least two belong to the same set A_i. This is because mapping $n+1$ elements into the subset they belong to cannot be an injection.

IV.C.3.iv.d Countability

The usual domains $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ are all infinite. But to compare these infinites, one must use injections (or prove they do not exist).

First, we can remark that we have the inclusions $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R}$, so we have $|\mathbb{N}| \leq |\mathbb{Z}| \leq |\mathbb{Q}| \leq |\mathbb{R}|$. (To exhibit an injection when there is inclusion, simply use the identity function $Id(x) = x$, which is clearly injective.)

$\mathbb{N}$ being the smallest of our usual domains, we will try to compare cardinalities to $|\mathbb{N}|$.

Definition: Countable

A set A is countable if $|A| \leq |\mathbb{N}|$.

The term *countable* stems from the fact that the elements of a countable set can be attributed a unique number, called an index (which is what is done when counting), even if that continues to infinity. Not that in particular all finite sets are countable.

The first result, that might seem counterintuitive at first, is as follows:

Theorem

$\mathbb{Z}$ and $\mathbb{Q}$ are countable sets: $|\mathbb{N}| = |\mathbb{Z}| = |\mathbb{Q}|$.

This theorem is proved by providing injections into $\mathbb{N}$.

Inject $\mathbb{Z}$ in $\mathbb{N}$ We need to find a function $f : \mathbb{Z} \to \mathbb{N}$ such that no two elements of $\mathbb{Z}$ have the same image. The problem is to fold a number line that extends in both direction into a number line that only extends into the positive integers. The solution is to insert a negative number between two positive numbers, and to count as follows:

Value (in $\mathbb{Z}$)	0	1	-1	2	-2	...
Index (in $\mathbb{N}$)	0	1	2	3	4	...

Formally, our function f will map strictly positive numbers go to odd numbers while negative numbers (and 0) will go to even numbers. For $p \in \mathbb{Z}$, if $p > 0$, $f(p) = 2p - 1$ and if $p \leq 0$, $f(p) = -2p$.

Note that a similar idea can be used in order to prove that the set of even natural numbers is a big as the whole set of natural numbers, by injecting $\mathbb{N}$ into the even numbers.

This is for your personal culture more than to be applied in this course.

Inject $\mathbb{Q}$ into $\mathbb{N}$ This is actually performed in two steps:

- Inject $\mathbb{Q}$ into $\mathbb{Z} \times \mathbb{Z}$. This part is rather straightforward. We can define f by taking, for any rational number written as a fraction $\frac{p}{q}$, $f\left(\frac{p}{q}\right) = (p, q)$. Not that technically there are more than one choice for p and q, but this can be resolved either by asking that $\frac{p}{q}$ is an irreducible fraction, or by using the Axiom of Choice which states that it is possible to choose one value from the infinitely many fractions that can represent $\frac{p}{q}$. Function f is an injection: any pair (p, q) corresponds to at most one rational number $\frac{p}{q}$.
- Inject $\mathbb{Z} \times \mathbb{Z}$ in $\mathbb{N}$. This is more involved, and will not actually be shown here in full formality. One way to perform this injection, is to index $\mathbb{Z} \times \mathbb{Z}$ by $\mathbb{N}$ using a spiral in the plane of integers, as depicted in Figure IV.5. While this is, in my opinion, convincing enough to understand why this injection is possible, it does not constitute a formal proof. It is however not easy to formally define a function from this idea, as calculating the index from the coordinates is not direct. Other proofs exist that

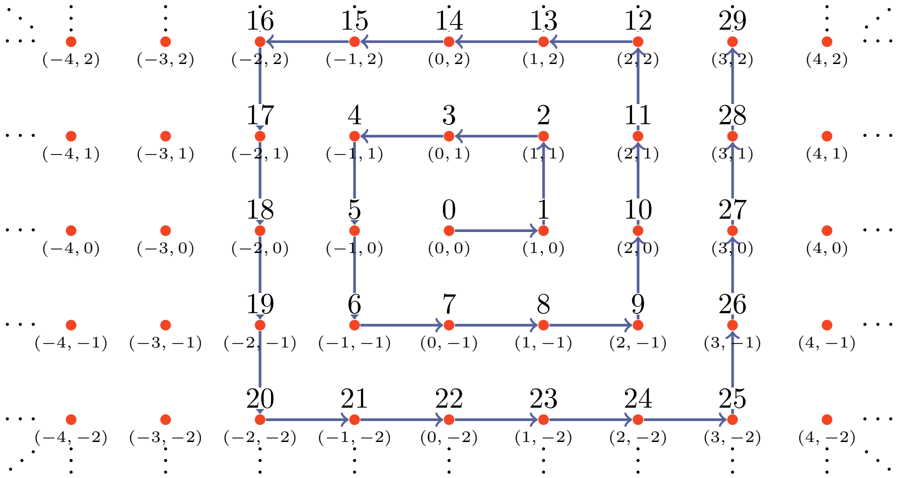


Figure IV.5: Injecting $\mathbb{Z} \times \mathbb{Z}$ into $\mathbb{N}$, a graphical idea.

do not use this spiral idea exist but use more involved tools and therefore will not be provided here.

IV.C.3.iv.e Uncountability

The case of $\mathbb{R}$ is different from $\mathbb{Q}$. Intuitively, the difference between rational numbers and real numbers can be interpreted as a difference in the quantity of information they carry.

The decimals in a rational number is a repeating pattern, so it is sufficient to remember the pattern to be able to reconstruct the number with arbitrary precision. For real irrational numbers, there is no repeating pattern, so all the decimals (infinitely many of them!) have to be stored in order to achieve arbitrary precision.

So for any rational number, even with a very long repeating pattern, there will be an infinity of irrational number slightly different around it. This idea of slightly changing the numbers is what makes $\mathbb{R}$ much bigger than $\mathbb{N}$:

Theorem

The set of real numbers is uncountable: $|\mathbb{R}| > |\mathbb{N}|$.

The proof of uncountability of $\mathbb{R}$ that follows is due to Georg Cantor. It is a proof by contradiction called the *diagonalization* proof (or diagonalization argument), and goes as follows.

This is for your personal culture more than to be applied in this course.

Assume, by contradiction, that $|\mathbb{R}| = |\mathbb{N}|$. Then we have a bijection f between $\mathbb{N}$ and $\mathbb{R}$.
Note that here f is not the indexing function, but f^{-1} is. For a real number x, $f^{-1}(x)$ give the index of the real number x. On the other hand, for any natural number n, $f(n)$ gives the n-th real number.
We can therefore write all the real numbers in the order they are indexed: $f(1)$, $f(2)$, ..., aligning them on the decimal point. Then consider the real number $z =$

$$z = 0.950631026\ldots \notin f(\mathbb{N})$$

$$\begin{aligned}
f(0) &= 677.\boxed{8}042309129237\ldots\\
f(1) &= 741.3\boxed{4}88631711009\ldots\\
f(2) &= -592.63\boxed{9}7656848368\ldots\\
f(3) &= -230.203\boxed{5}616009134\ldots\\
f(4) &= 639.8153\boxed{2}15851188\ldots\\
f(5) &= -570.38823\boxed{0}8538340\ldots\\
f(6) &= -467.997827\boxed{9}018488\ldots\\
f(7) &= -276.7336246\boxed{1}00834\ldots\\
f(8) &= 795.46213042\boxed{5}3476\ldots\\
&\vdots \qquad\qquad \ddots
\end{aligned}$$

Figure IV.6: Cantor's diagonalization argument.

$0.d_1d_2d_3\ldots$ (where d_i is the i-th digit after the decimal point) built as follows:

- Take the first digit after the decimal point of $f(1)$, and add 1 to it (truncating 10 to 0): this is d_1.
- Take the second digit after the decimal point of $f(2)$, and add 1 to it (truncating 10 to 0): this is d_2.

  ⋮

- Take the i-th digit after the decimal point of $f(i)$, and add 1 to it (truncating 10 to 0): this is d_i.

  ⋮

- Continue for infinitely many steps.

An illustration of this process that uses the digits of the diagonal (hence the name) is given Figure IV.6.
For any $i \in \mathbb{N}$, z differs from $f(i)$ at least in digit i, so $z \neq f(i)$. So z has no pre-image by f, and f is not surjective, and that contradicts the assumption that f is bijective. $\lightning$
Therefore $|\mathbb{R}| \neq |\mathbb{N}|$, and since $|\mathbb{R}| \geq |\mathbb{N}|$ that means $|\mathbb{R}| > |\mathbb{N}|$. □

Chapter V
Equivalence and Order Relations

Chapter contents

V.A Equivalence relations **74**
V.A.1 Definition and notation 74
V.A.1.i Formal and informal definitions 74
V.A.1.ii Proving equivalence 74
V.A.1.iii Notations 75
V.A.2 Equivalence classes 76
V.A.2.i Definition 76
V.A.2.ii Partition using equivalence classes 77
V.A.2.iii Quotient by an equivalence relation 77
V.A.2.iv Equivalence classes in mathematics 78
V.B Order relations **80**
V.B.1 Definition and notation 80
V.B.1.i Formal and informal definition 80
V.B.1.ii Proving a relation is an order 81
V.B.1.iii Notations for orders 82
V.B.1.iv Strict orders 82
V.B.2 Partial *v* total orders 83
V.B.3 Lexicographic orders 84
V.B.3.i Definition 84
V.B.3.ii Lexicographic orders are orders 85
V.B.3.iii Lexicographic orders and totality 86

V.A Equivalence relations

Equivalence relations are a special kind of relations between elements of the same set or domain. In English, two objects are equivalent if they are the same *for a given purpose.* And if the saying states that “if it talks like a duck and it walks like a duck, then it must be a duck”, the mathematician would be more prudent and only state that ”it is equivalent to a duck regarding the talking and walking”.

V.A.1 Definition and notation

V.A.1.i Formal and informal definitions

The criteria on which the equivalence is performed can be about anything. The only condition is that it follows the three conditions of the following definition:

Definition: Equivalence relation

A binary relation $R \subseteq A \times A$ is an *equivalence relation* if it is:

Reflexive $\forall x \in A, (x, x) \in R$

Symmetric $\forall x, y \in A, (x, y) \in R \rightarrow (y, x) \in R$

Transitive $\forall x, y, z \in A, (x, y) \in R \wedge (y, z) \in R \rightarrow (x, z) \in R$

Reflexivity states that every element is equivalent to itself. This makes intuitive sense: an object is equivalent to itself for every purpose. So a relation that does not see as equivalent an object with itself cannot be an equivalence relation.

Symmetry means that if x is equivalent to y, then y is equivalent to x. The order does not matter; in English we actually say that “two objects are equivalent”, and the grammar does not indicate any order either.

Transitivity requires than if x is equivalent to y and y equivalent to z, then x is equivalent to z as well. In this the math departs a bit from the intuition. In English, “equivalent” can be understood as ”almost the same”, and transitivity may not be assumed when speaking of almost the same objects. This ambiguity is at the heart of *Sorites paradoxes*: if a heap of 10000 rice grains is equivalent to a heap of 9999 rice grains, which is equivalent to a heap of 9998 rice grains, $\ldots$, which is equivalent to a heap of 1 grain; if there is transitivity, a heap of 10000 grains is equivalent to a heap of 1 grain, which sounds absurd. Transitivity is therefore required in the mathematical definition in order to prevent this kind of situation: a heap of 1 grain has to actually be equivalent to a heap of 10000 grains for the “equivalence” to be well defined (this equivalence can be for example “being a heap containing rice”).

When all the criteria are met, the relation is said to be an equivalence relation *over* A, indicating the domain by the same sentence. In that case, to elements in relation are said to be “equivalent *up to* the relation”.

V.A.1.ii Proving equivalence

Whenever a relation is claimed to be an equivalence relation, this fact must be proved. The proof follows the definition: it is in three parts to deal separately with reflexivity, symmetry, and transitivity (labeling the parts of the proof with R, S, T is advised). Each part is a universally quantified statement, so variables of the set have to be taken: 1 for reflexivity, 2 for symmetry, 3 for transitivity. Then, for symmetry and transitivity, the premise of the implication has to be assumed in order to prove the conclusion. The assumption of the premise combined with the definition of the relation usually provides enough to prove the conclusion rather easily. As a result, all proofs that a relation is an equivalence relation look alike.

V.A.1.ii.a Example: Identity relation

On any set A, the identity relation $Id = \{(x, x) \mid x \in A\}$ is an equivalence relation.

Although this proof is rather trivial, it still follows the aforementioned plan:

$\boxed{\textbf{R}}$ Let $x \in A$. By definition of Id, $(x, x) \in Id$, so Id is reflexive.

$\boxed{\textbf{S}}$ Let $x, y \in A$. Assume $(x, y) \in Id$. Then $y = x$, so $(y, x) = (x, x) \in Id$. So Id is symmetric.

$\boxed{\textbf{T}}$ Let $x, y, z \in A$. Assume $(x, y) \in Id$ and $(y, z) \in Id$. Then $x = y$ and $y = z$. Therefore $x = z$ and $(x, z) \in Id$. As a result Id is transitive.

Since, Id is reflexive, symmetric, and transitive, it is an equivalence relation. □

V.A.1.ii.b Example: Integral part

On the set of reals, define $F = \{(x, y) \in \mathbb{R} \times \mathbb{R} \mid \lfloor x \rfloor = \lfloor y \rfloor\}$.

Intuitively, this relation deems relates all numbers that have the same integral part: 2 is related to 2.39, to $\frac{5}{2}$, to e $(2.718\ldots)$, to $\sqrt{5}$ $(2.236\ldots)$, or any number that is 2 point something. Of course that means e is related to 2, and to $\sqrt{5}$ as well, so symmetry and transitivity do seem to hold.

It can be proved that F is an equivalence relation:

$\boxed{\textbf{R}}$ Let $x \in \mathbb{R}$. We have $\lfloor x \rfloor = \lfloor x \rfloor$ so $(x, x) \in F$, so F is reflexive.

$\boxed{\textbf{S}}$ Let $x, y \in \mathbb{R}$. Assume $(x, y) \in F$. That means, by definition of F, that $\lfloor x \rfloor = \lfloor y \rfloor$. Then $\lfloor y \rfloor = \lfloor x \rfloor$ and $(y, x) \in F$. So F is symmetric.

$\boxed{\textbf{T}}$ Let $x, y, z \in \mathbb{R}$. Assume $(x, y) \in F$ and $(y, z) \in F$. That means, by definition of F, that $\lfloor x \rfloor = \lfloor y \rfloor$ and $\lfloor y \rfloor = \lfloor z \rfloor$. As a result $\lfloor x \rfloor = \lfloor z \rfloor$ and $(x, z) \in F$, so F is transitive.

Since, F is reflexive, symmetric, and transitive, it is an equivalence relation. □

V.A.1.iii Notations

Since equivalence relations are binary, it is common to use an infix symbol to denote them: a symbol written in between the two variables or values.

And to highlight the symmetric nature of these relations, symmetrical symbols are often used: $\equiv$, $\sim$, $\approx$, $\simeq$, $\bowtie$, $\cong$. The equality symbol $=$ is, despite being symmetrical, being "reserved" for the identity relation, and is therefore not used for any other.

Definitions are often written using the infix symbol, for example: "Let $\sim = \{(n,p) \in \mathbb{Z}^2 \mid n^2 = p^2\}$". Note that the set-builder notation is not always used to define these relations. More often, an English statement of the same fact is used instead. For example the above relation can also be introduced as such: "Let $\sim$ be the relation over $\mathbb{Z}$ defined as $n \sim p$ if and only if $n^2 = p^2$".

Remark that using a symmetrical symbol does not make the relation symmetrical or even an equivalence relation, that must be proved separately!

Exercise V.1

Prove that the following relations are equivalence relations.

1. Relation $\equiv$ over $\mathbb{N} \times \mathbb{N}$ defined as $(n_1, p_1) \equiv (n_2, p_2)$ iff $n_1 + p_1 = n_2 + p_2$.
2. $\approx = \{(x,y) \in \mathbb{R} \times \mathbb{R} \mid \cos(x) = \cos(y) \text{ and } \sin(x) = \sin(y)\}$.
3. Relation $\simeq$ over $\mathbb{Z} \times (\mathbb{N} \setminus \{0\})$ defined as $(p_1, q_1) \simeq (p_2, q_2)$ iff $p_1 \cdot q_2 = p_2 \cdot q_1$.
4. $\sim = \{(n,p) \in \mathbb{Z} \times \mathbb{Z} \mid \exists k \in \mathbb{Z}, n = p + 5k\}$.

V.A.2 Equivalence classes

V.A.2.i Definition

Elements that are equivalent to each other are all equivalent together, by transitivity. In the example above of relation $F = \{(x,y) \in \mathbb{R} \times \mathbb{R} \mid \lfloor x \rfloor = \lfloor y \rfloor\}$, elements 2,2.39, $\frac{5}{2}$, e, are $\sqrt{5}$ all equivalent together. They can therefore be grouped together into an *equivalence class*.

Definition: Equivalence class

Let $\approx$ be an equivalence relation over a set A. The equivalence class of x for is the set

$$[x]_\approx = \{y \in A \mid x \approx y\}$$

Sometimes the relation is omitted and the equivalence class is simply denoted $[x]$ when the relation is clear from the context.

An immediate consequence of this definition is that being equivalent and being in the equivalence class is the same:

Lemma

Let $\approx$ be an equivalence relation over A. Then $x \approx y$ if and only if $[x]_\approx = [y]_\approx$.

Proof

This is an equivalence, which we can prove by proving implication in both directions: $x \approx y \Rightarrow [x]_\approx = [y]_\approx$ and $x \approx y \Leftarrow [x]_\approx = [y]_\approx$.

$\boxed{\Rightarrow}$ Assume $x \approx y$. To prove a set equality we prove set inclusion in both directions:

$[x]_\approx \subseteq [y]_\approx$ and $[x]_\approx \supseteq [y]_\approx$.

$\boxed{\subseteq}$ Let $t \in [x]_\approx$, by definition $t \approx x$, and by transitivity $t \approx y$. By symmetry $y \approx t$ so $t \in [y]_\approx$. As a result $[x]_\approx \subseteq [y]_\approx$.

$\boxed{\supseteq}$ The converse inclusion is proved in a similar way (only the point where symmetry is used changes); it could be omitted but is given here for completeness' sake. Let $t \in [y]_\approx$, by definition $t \approx y$. By symmetry $y \approx t$ and by transitivity $x \approx t$ so $t \in [x]_\approx$. As a result $[y]_\approx \subseteq [x]_\approx$.

Therefore $[x]_\approx = [y]_\approx$.

$\boxed{\Leftarrow}$ Assume $[x]_\approx = [y]_\approx$. We have in particular $y \in [y]_\approx$ (by reflexivity) so $y \in [x]_\approx$ hence $y \approx x$ and by symmetry $x \approx y$. □

For example, the set $[2]_F$ contains all real numbers that have integral part of 2: $2 \in [2]_F$, $\frac{5}{2} \in [2]_F$, $e \in [2]_F$, $\sqrt{5} \in [2]_F$, $\dots$. Note that $[2]_F = [\sqrt{5}]_F$, and any of the equivalent value could have been chosen as a representative.

V.A.2.ii Partition using equivalence classes

The second consequence of the definition of equivalence classes is that it not only groups equivalent elements together: it puts every element in a single equivalence class. Formally:

Theorem

Let $\approx$ be an equivalence relation over a set A. Equivalence classes for $\approx$ over A form a partition of A.

Proof

First, by reflexivity, every element $x \in A$ is in $[x]_\approx$. So every classes is non-empty ($[x]_\approx$ contains at least x) and together they cover A: $\bigcup_{x\in A}[x]_\approx = A$.
What remains to be proved is that equivalence classes are pairwise disjoint: two equivalence classes that are not identical are disjoint. We prove the contrapositive: two equivalence classes that are not disjoint are identical.
Let $[x]_\approx$ and $[y]_\approx$ be equivalence classes. Assume that $[x]_\approx$ and $[y]_\approx$ are not disjoint. That means $[x]_\approx \cap [y]_\approx \neq \emptyset$ so there exists an element $t \in [x]_\approx \cap [y]_\approx$, i.e. $t \in [x]_\approx$ and $t \in [y]_\approx$. By definition of $t \in [x]_\approx$, we have $t \approx x$. By definition of $t \in [y]_\approx$, we have $t \approx y$. By symmetry and transitivity, we have $x \approx y$, which means $[x]_\approx = [y]_\approx$ by the above Lemma. □

V.A.2.iii Quotient by an equivalence relation

Since equivalence classes partitions A, this set can be seen as “zoomed out” through the prism of an equivalence relation $\approx$: two elements that are equivalent for $\approx$ will be seen as the same.

This is called the *quotient of A by* $\approx$. It is defined as follows:

Definition: Quotient by $\approx$

Let $\approx$ be an equivalence relation over A. The quotient of A by $\approx$ is $A/_{\approx} = \{[x]_{\approx} \mid x \in A\}$, the set of equivalence relations of $\approx$.

V.A.2.iii.a Example: Quotient by integral part

For example over the reals, the relation $F = \{(x, y) \in \mathbb{R} \times \mathbb{R} \mid \lfloor x \rfloor = \lfloor y \rfloor\}$ groups into equivalence classes all numbers that start with the same integer (this is only completely true for positive numbers; for negative numbers it is a slight approximation since $\lfloor -4 \rfloor = \lfloor -3.14 \rfloor = -4$).

So an equivalence class can be represented by that integer, so the quotient of $\mathbb{R}$ by F is $\{[n]_F \mid n \in \mathbb{Z}\}$. That means that $\mathbb{R}/_F$ is just like $\mathbb{Z}$: we say that $\mathbb{R}/_F$ is isomorphic to $\mathbb{Z}$, since these sets behave the same for all mathematical purposes.

For sets, being isomorphic is also an equivalence relation!

V.A.2.iii.b Example: Quotient by equivalent fraction

Consider another example. Let $\simeq$ be the relation over $\mathbb{Z} \times (\mathbb{N} \setminus \{0\})$ defined as $(p_1, q_1) \simeq (p_2, q_2)$ iff $p_1 \cdot q_2 = p_2 \cdot q_1$. This relation can be seen as marking as equivalent fractions that represent the same number. Proving it is an equivalence relation was done in Exercise V.1, Question 1.

The quotient of $\mathbb{Z} \times (\mathbb{N} \setminus \{0\})$ by $\simeq$ has one element per rational number; it is actually a way to construct the set of rationals:

$$\mathbb{Q} = (\mathbb{Z} \times (\mathbb{N} \setminus \{0\}))/_{\simeq}$$

V.A.2.iv Equivalence classes in mathematics

V.A.2.iv.a Logical equivalence

See Section I.B.2.ii.

As the name indicates, logical equivalence is an equivalence relation over propositional formulas. Remember that logical equivalence is defined as follows: $\varphi \equiv \psi$ iff for any valuation of atomic proposition, φ is true if and only if ψ is true. Proving it is an equivalence is pretty straightforward and is left to the reader as an exercise.

The more interesting question here is: what do equivalence classes look like? What would be a good representative? As logically equivalent formulas share the same truth table (the last column), a truth table represents all formulas that are true exactly for these combinations of truth values for atomic propositions.

We can evan count how many classes there are. For n atomic propositions, there are 2^n lines in a truth table. Each can be either $\top$ or $\bot$, so 2 choices. As a result there are 2^{2^n} classes.

For example with 2 variables there are $2^{2^2} = 16$ formulas that are sufficient to describe all formulas *up to logical equivalence*. For 3 propositional variables, there are $2^{2^3} = 256$ possible truth tables.

This is for your personal culture more than to be applied in this course.

Remark that the truth table itself is not a formula, and sometimes it is better to use an actual formula as the representative. There are several options for that, which are called *normal forms*, among which the conjunctive normal form and the disjunctive normal form. The disjunctive normal form is the one that resembles the most to the truth table: it is built as the disjunction of all lines where the formula is true, each line being the conjunction of atomic propositions (when $\top$) or their negation (when $\bot$).

p	q	r	φ
$\top$	$\top$	$\top$	$\bot$
$\top$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\top$	$\bot$
$\top$	$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$

Figure V.1: A truth table: an equivalence class for the logical equivalence relation.

For example for formula φ whose truth table is given in Figure V.1 is equivalent to

$$(p \land q \land \neg r) \lor (p \land \neg q \land \neg r) \lor (\neg p \land q \land r) \lor (\neg p \land \neg q \land r).$$

In this disjunction, the first term $p \land q \land \neg r$ corresponds to the second line, where p is $\top$, q is $\top$, and r is $\bot$ so $\neg r$ is $\top$. This line was selected because φ is $\top$, and so were the fourth, fifth, and seventh lines, corresponding to the other terms in the disjunction.

Conjunctive normal form is obtained as the negation of the disjunctive normal form of the negation. In practice, that means building a disjunctive normal form using lines where the formula is $\bot$, then negating this formula and applying De Morgan's laws and double negation elimination in order to obtain a conjunction of disjunctions.

In the example of Figure V.1, the conjunctive normal form is obtained as:

$$\begin{gathered}
\neg\left((p \land q \land r) \lor (p \land \neg q \land r) \lor (\neg p \land q \land \neg r) \lor (\neg p \land \neg q \land \neg r)\right) \qquad \equiv \\
\neg(p \land q \land r) \land \neg(p \land \neg q \land r) \land \neg(\neg p \land q \land \neg r) \land \neg(\neg p \land \neg q \land \neg r) \qquad \equiv \\
(\neg p \lor \neg q \lor \neg r) \land (\neg p \lor \neg\neg q \lor \neg r) \land (\neg\neg p \lor \neg q \lor \neg\neg r) \land (\neg\neg p \lor \neg\neg q \lor \neg\neg r) \qquad \equiv \\
(\neg p \lor \neg q \lor \neg r) \land (\neg p \lor q \lor \neg r) \land (p \lor \neg q \lor r) \land (p \lor q \lor r)
\end{gathered}$$

V.A.2.iv.b Similar triangles

In geometry, similar triangles are triangle that bear the same proportions. It is formally defined over triplets of points in the plane (each being two coordinates), as $ABC \sim A'B'C'$ iff $\frac{AB}{A'B'} = \frac{BC}{B'C'} = \frac{AC}{A'C'}$.

In geometrical terms, similar triangles are triangles that are the same up to scaling, rotation, and mirror image.

Technically, this relation is a subset of $((\mathbb{R} \times \mathbb{R}) \times (\mathbb{R} \times \mathbb{R}) \times (\mathbb{R} \times \mathbb{R})) \times ((\mathbb{R} \times \mathbb{R}) \times (\mathbb{R} \times \mathbb{R}) \times (\mathbb{R} \times \mathbb{R})) = ((\mathbb{R}^2)^3)^2$, but no one writes it that convoluted way!

Exercise V.2

1. Let $\approx\ = \{(x, y) \in \mathbb{R} \times \mathbb{R} \mid \cos(x) = \cos(y) \text{ and } \sin(x) = \sin(y)\}$. It was proved to be an equivalence relation in Exercise V.1, Question 2. What are the equivalence classes of $\approx$? Describe the classes in English or formally.
2. Let $\sim\ = \{(n, p) \in \mathbb{Z} \times \mathbb{Z} \mid \exists k \in \mathbb{Z}, n = p + 5k\}$. It was proved to be an equivalence relation in Exercise V.1, Question 4. What are the equivalence classes of $\sim$ Describe the classes in English or formally; at least give how many there are.

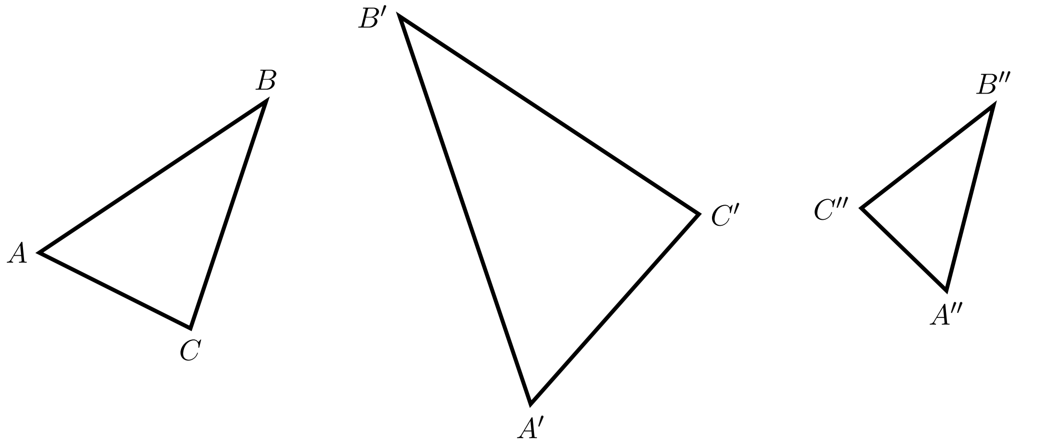


Figure V.2: Similar triangles ABC, $A'B'C'$, and $A''B''C''$.

V.B Order relations

V.B.1 Definition and notation

V.B.1.i Formal and informal definition

An order relation, or order for short, allows to compare elements. The choice of the comparison criterion is given in the relation's definition. This is defined as follows:

Definition: Order relation

A binary relation $R \subseteq A \times A$ is an *order relation over* A if it is:

Reflexive $\forall x \in A, (x,x) \in R$

Antisymmetric $\forall x,y \in A, ((x,y) \in R \wedge (y,x) \in R) \rightarrow x = y$

Transitive $\forall x,y,z \in A, (x,y) \in R \wedge (y,z) \in R \rightarrow (x,z) \in R$

Reflexivity requires that elements compare to themselves.

Antisymmetry requires states that the only way for elements to compare in both direction is to be equal. This property ensures that the relation is one-direction only.

Transitivity is conform to intuition: if a is smaller than b and b is smaller than c, then a is smaller than c.

Symmetry *v* Antisymmetry

Remark than antisymmetry is not the negation of symmetry. It is possible for a relation to be both symmetric and antisymmetric. In this case, whenever $(x,y) \in R$, $(y,x) \in R$ by symmetry, then $x = y$ by antisymmetry. That means the only relations that are both symmetric and antisymmetric are included in the identity relation.

In addition, a relation can be neither symmetric nor antisymmetric. For example over the set $\{-1,0,1\}$, consider relation $\multimap = \{(-1,-1),(-1,0),(-1,1),(0,0),(1,-1),(1,0),(1,1)\}$.

We don't have symmetry because $1 \multimap 0$ but not $0 \multimap 1$. We don't have antisymmetry either because $-1 \multimap 1$ and $1 \multimap -1$, but $-1 \neq 1$.

V.B.1.ii Proving a relation is an order

The definition being in three parts, proofs that a relation is an order relation will follow the same plan; it is useful to clearly mark these parts using the letter R, A, T. Then for reflexivity take one element, two for antisymmetry, and three for reflexivity. This plan is actually very similar to the one used for equivalence relations (see Section V.A.1.ii).

V.B.1.ii.a Example: Identity relation

It was proved in Section V.A.1.ii.a that the identity relation $Id = \{(x, x) \mid x \in A\}$ is reflexive and transitive. As noted above, it is also antisymmetric: if $(x, y) \in Id$ and $(y, x) \in Id$, then by definition $x = y$.

V.B.1.ii.b Example: Less than or equal to

Relation $Leq = \{(n, p) \in \mathbb{Z}^2 \mid \exists k \in \mathbb{N}, n + k = p\}$, which corresponds to $\leq$, is an order. Although that seems obvious since $\leq$ is "the" very first order relation that is introduced on numbers, it ought to be proved formally. In this case, two numbers are related when *there exists* a natural number satisfying a property. When $(n, p) \in Leq$ is assumed, then we will take such a number k (existential instantiation). When it must be proved that $(n, p) \in Leq$, we must exhibit such a value (existential generalization).

$\boxed{\textbf{R}}$ Let $n \in \mathbb{Z}$. We have $n + 0 = n$, so by choosing $k = 0$ we have $(n, n) \in Leq$. So Leq is reflexive.

$\boxed{\textbf{A}}$ Let $n, p \in \mathbb{Z}$. Assume $(n, p) \in Leq$ and $(p, n) \in Leq$. Then there exists $k, k' \in \mathbb{N}$ such that $n + k = p$ and $p + k' = n$. So $n + k + k' = n$, so $k + k' = 0$. That means $k = k' = 0$, so $n = p$. As a result, Leq is antisymmetric.

$\boxed{\textbf{T}}$ Let $n, p, m \in \mathbb{Z}$. Assume $(n, p) \in Leq$ and $(p, m) \in Leq$. Then there exists $k, k' \in \mathbb{N}$ such that $n + k = p$ and $p + k' = m$. So $n + k + k' = m$. Let $k'' = k + k'$. We have $n + k'' = m$, so $(n, m) \in Leq$. As a result Leq is transitive.

Relation Leq is reflexive, antisymmetric, and transitive. Therefore it is an order relation. □

V.B.1.ii.c Example: Divisibility

On natural numbers, $D = \{(n, p) \in \mathbb{N}^2 \mid \exists k \in \mathbb{N}, n = k \cdot p\}$ is called the *divisibility* relation: when $(n, p) \in D$, we say that p divides n (n is a multiple of p).

We can prove it is an order relation:

Section VII.A.1 is devoted to the concept of divisibility.

$\boxed{\mathbf{R}}$ Let $n \in \mathbb{N}$. We have $n = 1 \cdot n$. So by choosing $k = 1$ we have $(n, n) \in D$, hence D is reflexive.

$\boxed{\mathbf{A}}$ Let $n, p \in \mathbb{N}$. Assume $(n, p) \in D$ and $(p, n) \in D$. Then there are $k, k' \in \mathbb{N}$ such that $n = k \cdot p$ and $p = k' \cdot n$, and $n = k \cdot k' \cdot n$. So $k \cdot k' = 1$, hence $k = m = 1$ and $n = p$. As a result D is antisymmetric.

$\boxed{\mathbf{T}}$ Let $n, p, m \in \mathbb{N}$. Assume $(n, p) \in D$ and $(p, m) \in D$. Then there are $k, k' \in \mathbb{N}$ such that $n = k \cdot p$ and $p = k' \cdot m$, and $n = k \cdot k' \cdot m$. Let $k'' = k \cdot k'$. We have $n = k'' \cdot m$ so $(n, m) \in D$ and D is transitive.

Relation D is reflexive, antisymmetric, and transitive. Therefore it is an order relation.

□

V.B.1.ii.d Example: Inclusion

On any domain, the set inclusion relation $\mathit{In} = \{(A, B) \in \mathcal{P}(\mathbb{D})^2 \mid A \subseteq B\}$ is an order relation:

$\boxed{\mathbf{R}}$ Let A be a set. We have $A \subseteq A$ so $(A, A) \in \mathit{In}$ and In is reflexive.

$\boxed{\mathbf{A}}$ Let A, B be sets. Assume $(A, B) \in \mathit{In}$ and $(B, A) \in \mathit{In}$. Then $A \subseteq B$ and $B \subseteq A$ so $A = B$. Hence In is antisymmetric.

$\boxed{\mathbf{T}}$ Let A, B, C be sets. Assume $(A, B) \in \mathit{In}$ and $(B, C) \in \mathit{In}$. Let $x \in \mathbb{D}$. Assume $x \in A$. Since $A \subseteq B$, $x \in B$. Since $B \subseteq C$, $x \in C$. Hence $A \subseteq C$ and $(A, C) \in \mathit{In}$. Therefore In is transitive.

Relation In is reflexive, antisymmetric, and transitive. Therefore it is an order relation.

□

V.B.1.iii Notations for orders

To highlight the asymmetry of the relation, orders are often written with an *infix asymmetrical* symbol: $\preccurlyeq$, $\sqsubseteq$, $\trianglelefteq$, $\rtimes$, $\Subset$, $\lessdot$, $\ll$. The $\leq$ symbol is being "reserved" for the "less than or equal to" order on numbers, and $\subseteq$ is being "reserved" for the "inclusion" order on sets.

As for equivalence relations, definitions can be written in a set-builder notation or through a sentence. For example one can define $\preccurlyeq = \{((n_1, p_1), (n_2, p_2)) \mid n_1 \leq n_2 \wedge p_1 \leq p_2\}$ or by the sentence "Let $\preccurlyeq$ be the relation over $\mathbb{Z}^2$ defined as $(n_1, p_1) \preccurlyeq (n_2, p_2)$ if and only if $n_1 \leq n_2$ and $p_1 \leq p_2$". Then it remains to be proved that it actually is an order relation!

V.B.1.iv Strict orders

From any order R one can define a *strict* version R' of the order. The strict version is the same except it does not relate elements with themselves: $R' = \{(x, y) \mid (x, y) \in R \wedge x \neq y\}$

One example is the strict version $<$ of order $\leq$.

It is common to use a resembling symbol for the strict version of the order. Usually, this is done either by barring or removing the part that resembles the equality symbol:

- $x < y$ iff $x \leq y \land x \neq y$.
- $x \prec y$ iff $x \preccurlyeq y \land x \neq y$.
- $A \subsetneq B$ iff $A \subseteq B \land A \neq B$.
- $A \sqsubsetneq B$ iff $A \sqsubseteq B \land A \neq B$.

Remark the difference between $\subsetneq$, which is strict inclusion, and $\nsubseteq$, which is the negation of inclusion: $A \nsubseteq B$ iff $\neg(A \subseteq B)$.

Note that because the strict version is not reflexive, it is **not** an order relation! We can however prove that the strict order is transitive:

Lemma: Strict orders are transitive

Let $\preccurlyeq$ be an order over A and let $\prec$ be the strict version of this order. Then for any $x, y, z \in A$, if $x \prec y$ and $y \prec z$, then $x \prec z$.

Proof

Let $x, y, z \in A$. Assume $x \prec y$ and $y \prec z$. Therefore in particular $x \preccurlyeq y$ and $y \preccurlyeq z$, so by transitivity of $\preccurlyeq$, $x \preccurlyeq z$.
It remains to be proved that $x \neq z$. Assume, by contradiction, that $x = z$. Then we have $y \preccurlyeq x$, and by antisymmetry of $\preccurlyeq$ $x = y$, which is a contradiction with $x \prec y$.↯Therefore $x \neq z$ and $x \prec z$. □

Also remark that it is impossible to have both $x \prec y$ and $y \prec x$: that would mean in particular $x \preccurlyeq y$ and $y \preccurlyeq x$, so $x = y$ by antisymmetry of $\preccurlyeq$, which is a contradiction with $x \neq y$. As a result, $\prec$ is only antisymmetric because the premise of the implication never holds.

Exercise V.3

Prove that each of the following relation is an order.

1. $\Subset$ defined over $\mathbb{R} \times \mathbb{R}$ as $(a_1, b_1) \Subset (a_2, b_2)$ iff $a_2 \leq a_1$ and $b_1 \leq b_2$.
2. $\trianglelefteq$ defined over $\mathbb{R} \times \mathbb{N}$ as $(x_1, n_1) \trianglelefteq (x_2, n_2)$ iff either $x_1 < x_2$ or: $x_1 = x_2$ and there exists $k \in \mathbb{N}$ such that $n_1 = n_2 \times k$.

Exercise V.4

Prove that relation $R = \{(x_1, y_1), (x_2, y_2) \in \mathbb{R}^2 \times \mathbb{R}^2 \mid x_1 \leq x_2 \lor y_1 \leq y_2\}$ is **not** an order.

V.B.2 Partial *v* total orders

Assume $\preccurlyeq$ is an order relation. Nothing in the definition of an order (Reflexivity, Antisymmetry, Transitivity) requires that for any x, y, $x \preccurlyeq y$ or $y \preccurlyeq x$.

For example, for sets of natural numbers, we have neither $\{1, 2, 3\} \subseteq \{2, 3, 5\}$ nor $\{2, 3, 5\} \subseteq \{1, 2, 3\}$: these elements are *incomparable*. That does not contradict the fact that $\subseteq$ is an order.

The additional requirement that any two elements can be compared makes the order *total*:

Definition: Total order

An order $\preccurlyeq$ over A is total iff $\forall x, y \in A, x \preccurlyeq y \vee y \preccurlyeq x$.

An order that is not total is called a *partial order*.

Proving totality of an order

To prove that an order is total, one has to show that for any two elements, they are related one way or another. This can be done using a proof by cases, because the statement is a disjunction. It can also be done by assuming $x \not\preccurlyeq y$ and proving $y \preccurlyeq x$ (which is a proof by cases in hiding).

To prove an order is not total, on the other hand, one has to exhibit incomparable elements x and y such that $x \not\preccurlyeq y$ and $y \not\preccurlyeq x$.

Exercise V.5

Determine whether the following relation is a total or partial order. Justify your answer by a proof.

1. $D = \{(n, p) \in \mathbb{N}^2 \mid \exists k \in \mathbb{N}, n = k \cdot p\}$ (divisibility).
2. $\Subset$ defined over $\mathbb{R} \times \mathbb{R}$ as $(a_1, b_1) \Subset (a_2, b_2)$ iff $a_2 \leq a_1$ and $b_1 \leq b_2$ (this is from Exercise V.3, Question 1).
3. $\trianglelefteq$ defined over $\mathbb{R} \times \mathbb{N}$ as $(x_1, n_1) \trianglelefteq (x_2, n_2)$ iff $x_1 < x_2$ or $x_1 = x_2$ and there exists $k \in \mathbb{N}$ such that $n_1 = n_2 \times k$ (this is from Exercise V.3, Question 2).
4. $Leq = \{(n, p) \in \mathbb{Z}^2 \mid \exists k \in \mathbb{N}, n + k = p\}$. Note: you cannot use the fact that if $n \not\leq p$ then $p < n$ that would be using the result you are trying to prove!

V.B.3 Lexicographic orders

V.B.3.i Definition

A *lexicographic order* is a way to create orders on Cartesian products, using orders on the sets in the product. It works as follows: compare the first components; if they are the same, compare the second component, and so on until one component can be strictly compared or the tuples are completely identical.

This process is used in the way we sort words out in a dictionary: a word is a tuple of letters, we have an underlying order on letters, the alphabetical order $\preccurlyeq_\alpha$. To compare apple and pear, we compare their first letter a and p, and conclude that since a $\prec_\alpha$ p apple comes before pear. Now when comparing pear and *peach*, we first compare the first letters, which are both p, so move on to the second letter, both e, move on to the third letter, both a, move on to the fourth letter, and since c $\prec_\alpha$ r, we conclude that peach comes before pear.

This is formalized as follows; in the definition below we give the case of two sets and the general case of n sets.

Definition: Lexicographic order

- Let $\preccurlyeq_A$ be an order over A and $\preccurlyeq_B$ be an order over B. The *lexicographic order* $\preccurlyeq_{lex}$ over $A \times B$ is defined as: $(x_1, y_1) \preccurlyeq_{lex} (x_2, y_2)$ iff
  - either $x_1 \neq x_2$ and $x_1 \preccurlyeq_A x_2$ (i.e. $x_1 \prec_A x_2$);
  - or $x_1 = x_2$ and $y_1 \preccurlyeq_B y_2$.
- Let $\preccurlyeq_1, \dots, \preccurlyeq_n$ be n orders over $A_1, \dots, A_n$, respectively. The *lexicographic order* over $A_1 \times \cdots \times A_n$ is defined as: $(x_1, \dots, x_n) \preccurlyeq_{lex} (y_1, \dots, y_n)$ iff
  - either $(x_1, \dots, x_n) = (y_1, \dots, y_n)$;
  - or $\exists i \in \{1, \dots n\}, x_i \prec_i y_i$ and $\forall j < i, x_j = y_j$.

The definitions seem to differ a bit for two sets and the general case. They actually match for $n = 2$, but they are presented in a slightly different way. This is because for the general case it is important to be able to determine at which index the comparison is strict (if any). In the case of two sets, the case of equality is encompassed in the case $y_1 \preccurlyeq_B y_2$, whereas for the general case this case is further divided into the remaining components: there can be strict inequality at the second component, or equality and then we need to compare further, but we cannot yet conclude to equality.

V.B.3.ii Lexicographic orders are orders

As the name claims, but which remains to be proved, the lexicographic order thus defined is actually an order:

Theorem: Lexicographic order is an order

Let $\preccurlyeq_1, \dots, \preccurlyeq_n$ be n orders over $A_1, \dots, A_n$, respectively. The lexicographic order $\preccurlyeq_{lex}$ is an order over the Cartesian product $A_1 \times \cdots \times A_n$.

Proof

$\boxed{\textbf{R}}$ $(x_1, \dots, x_n) \preccurlyeq_{lex} (x_1, \dots, x_n)$ because $(x_1, \dots, x_n) = (x_1, \dots, x_n)$.

$\boxed{\textbf{A}}$ Assume $(x_1, \dots, x_n) \preccurlyeq_{lex} (y_1, \dots, y_n)$ and $(y_1, \dots, y_n) \preccurlyeq_{lex} (x_1, \dots, x_n)$. Assume, by contradiction, that $(x_1, \dots, x_n) \neq (y_1, \dots, y_n)$. That means there exist $i, k \in \{1, \dots, n\}$ such that:

$$x_i \prec_i y_i \text{ and } \forall j < i, x_j = y_j \tag{V.1}$$
$$y_k \prec_k x_k \text{ and } \forall j < k, x_j = y_j. \tag{V.2}$$

Now we have two cases: either the indices k and i match or they don't.

- Assume $i = k$, then we have $x_i \prec_i y_i$ and $y_i \prec_i x_i$, so $x_i \preccurlyeq_i y_i$, $y_i \preccurlyeq_i x_i$, and $x_i \neq y_i$. By antisymmetry of $\preccurlyeq_i$ we have $x_i = y_i$, which is a contradiction with $x_i \neq y_i$.
- Without loss of generality, we can now assume that $i < k$. (The case $k < i$ is the same.) We have both $x_i \prec_i y_i$, by (V.1), and $x_i = y_i$, by (V.2) and since $i < k$, which is a contradiction.

So we have a contradiction in all cases, so we have a contradiction and $(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$.

$\boxed{\mathbf{T}}$ Assume $(x_1, \ldots, x_n) \preccurlyeq_{lex} (y_1, \ldots, y_n)$ and $(y_1, \ldots, y_n) \preccurlyeq_{lex} (z_1, \ldots, z_n)$.

- If $(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ or $(y_1, \ldots, y_n) = (z_1, \ldots, z_n)$ we directly have $(x_1, \ldots, x_n) \preccurlyeq_{lex} (z_1, \ldots, z_n)$.
- Assume $(x_1, \ldots, x_n) \neq (y_1, \ldots, y_n)$ and $(y_1, \ldots, y_n) \neq (z_1, \ldots, z_n)$. Then we have $i \in \{1, \ldots n\}, x_i \prec_i y_i$ and $\forall j < i, x_j = y_j$ and $k \in \{1, \ldots, n\}$ such that $y_k \prec_k z_k$ and $\forall j < k, y_j = z_j$.

  We prove it by cases on the order between indices i and k:

  - Assume $i < k$. We have $\forall j < i, x_j = y_j = z_j$, and $x_i \prec_i y_i$, with $y_i = z_i$. So $x_i \prec_i z_i$, so $(x_1, \ldots, x_n) \preccurlyeq_{lex} (z_1, \ldots, z_n)$.
  - Assume $i = k$. We have $\forall j < i, x_j = y_j = z_j$, with $x_i \prec_i y_i$ and $y_i \prec_i z_i$. By the above Lemma (Section V.B.1.iv), $x_i \prec_i z_i$, and $(x_1, \ldots, x_n) \preccurlyeq_{lex} (z_1, \ldots, z_n)$.
  - Assume $k < i$. We have $\forall j < k, x_j = y_j = z_j$, and $x_k = y_k$, with $y_k \prec_k z_k$. So $x_k \prec_k z_k$ and $(x_1, \ldots, x_n) \preccurlyeq_{lex} (z_1, \ldots, z_n)$.

V.B.3.iii Lexicographic orders and totality

Theorem

Let $\preccurlyeq_1, \ldots, \preccurlyeq_n$ be n *total* orders over $A_1, \ldots, A_n$, respectively. The lexicographic order $\preccurlyeq_{lex}$ over $A_1 \times \cdots \times A_n$ is total.

Proof

Let $(x_1, \ldots, x_n), (y_1, \ldots, y_n) \in A_1 \times \cdots \times A_n$.
First, consider the case where these tuples are equal: assume $(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$. Then by definition of $\preccurlyeq_{lex}$ we have $(x_1, \ldots, x_n) \preccurlyeq_{lex} (y_1, \ldots, y_n)$.
Now assume $(x_1, \ldots, x_n) \neq (y_1, \ldots, y_n)$. So for at least an index i, $x_i \neq y_i$. Let i be the smallest of those, i.e. for $j < i$, $x_j = y_j$.
Since $\preccurlyeq_i$ is total, we have one of the two cases:

- Either $x_i \preccurlyeq_i y_i$, which means $x_i \prec_i y_i$ since $x_i \neq y_i$. Therefore $(x_1, \ldots, x_n) \preccurlyeq_{lex} (y_1, \ldots, y_n)$.

- Or $y_i \preccurlyeq_i x_i$, which means $y_i \prec_i x_i$ since $x_i \neq y_i$. Therefore$(y_1, \ldots, y_n) \preccurlyeq_{lex} (x_1, \ldots, x_n)$.

As a result $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ are comparable and $\preccurlyeq_{lex}$ is total. □

Chapter VI
Sequences and Recurrence

Chapter contents

VI.A Sequences **88**

VI.A.1 Definition, notation 88

VI.A.1.i Explicit definition 88

VI.A.1.ii Recursive definition 88

VI.A.2 Particular sequences 89

VI.A.2.i Arithmetic and geometric growth 89

VI.A.2.ii Sum of terms of a sequence 90

VI.A.2.iii Arithmetico-geometric sequences 92

VI.B Proofs by induction **93**

VI.B.1 Principle of induction 93

VI.B.2 Variations on induction 94

VI.B.2.i Finding the right predicate 94

VI.B.2.ii The proof by contradiction version of induction 94

VI.B.2.iii Strong induction 94

VI.B.2.iv Structural induction 96

VI.C Application: Growth and complexity **97**

VI.C.1 What is complexity? 97

VI.C.1.i Big-O notation 98

VI.C.1.ii Worst-case complexity 98

VI.C.2 Complexity of recursive programs 98

VI.C.2.i Example: The Towers of Hanoi 99

VI.C.2.ii Example: Merge sort 101

VI.A Sequences

VI.A.1 Definition, notation

Formally speaking, a sequence is a total function using natural numbers (sometimes excluding zero) as the domain. This particularity is reflected in the way sequences are denoted: instead of the usual notation $u(n)$, we write u_n for the element of index n, called the *nth term*. The whole sequence is denoted $(u_n)_{n\in\mathbb{N}}$ (rather than u as we do for functions). The choice of the letter u in itself is guided by tradition: by habit, letters u, v, w, are usually used in sequences (while f, g, h are more often used for functions in general).

Definition

A sequence is a total function with $\mathbb{N}$ (or $\mathbb{N}\backslash\{0\}$) as domain.

VI.A.1.i Explicit definition

We can define a sequence explicitly, as we do for functions, by providing an expression to calculate the nth term based on n. For example:

- Let $(u_n)_{n\in\mathbb{N}}$ the sequence of integers defined by $u_n = 3n - 5$.
- Let $(v_n)_{n\in\mathbb{N}}$ the sequence of reals defined by $v_n = \sqrt{n}$.
- Let $(w_n)_{n\in\mathbb{N}}$ the sequence of rationals defined by $w_n = \frac{2n+7}{n+1}$.

With an explicit definition, a sequence has little difference with a function, only a particular domain.

VI.A.1.ii Recursive definition

What makes sequences more specific, and therefore more interesting, is that is it possible to define a term using the previous one, as long as term 0 is provided. Calculating each term starting from 0 allows to calculate the nth term.

Definition: Recursive definition of a sequence

A recursive definition of a sequence $(u_n)_{n\in\mathbb{N}}$ is given by:

- A *base case* that gives the value u_0.
- A *recurrence relation* that expresses u_{n+1} using u_n for every n.

The above pattern can be generalized with more than a single base case. The base cases could be 0 and 1; the recurrence relation can then be defined by writing u_{n+2} as an expression of u_{n+1} and u_n. For example:

- $u_0 = 1$ and for $n \in \mathbb{N}$, $u_{n+1} = 2u_n+3$. Then we have $u_0 = 1$, $u_1 = 2u_0+3 = 2\cdot1+3 = 5$, $u_2 = 2u_1 + 3 = 2 \cdot 5 + 3 = 13\ldots$
- $v_0 = 3$, $v_1 = 2$ and for $n \in \mathbb{N}$, $v_{n+2} = 2v_{n+1} - v_n$. Then we have $v_0 = 3$, $v_1 = 2$, $v_2 = 2v_1 - v_0 = 2\cdot2-3 = 1$, $v_3 = 2v_2 - v_1 = 2\cdot1-2 = 0$, $v_4 = 2v_3 - v_2 = 2\cdot0-2 = -2\ldots$

Using more than one base case is not the only way to define a sequence recursively. The recursive relation can actually use any term as long as it is lower and all possible base cases are covered. This makes a recursive definition *well founded*. For example one can define the sequence$(w_n)_{n\in\mathbb{N}}$ as $w_0 = 1$ and $w_n = 2w_{\lfloor \frac{n}{2} \rfloor} + 3$. In this case, it is possible to calculate any term w_n using $w_{\lfloor \frac{n}{2} \rfloor}$, and since this number is strictly smaller the process will terminate when we eventually reach w_0, which is given.

Exercise VI.1

Give the first 6 terms (up to index 5) of the following sequences:

1. $(u_n)_{n\in\mathbb{N}}$ defined by $u_n = \frac{3n}{5}$.
2. $(v_n)_{n\in\mathbb{N}}$ defined by $v_{n+1} = \frac{3v_n}{5}$ and $v_0 = 125$.
3. $(w_n)_{n\in\mathbb{N}}$ defined by $w_{n+1} = 3w_n - 8$ and $w_0 = 5$.
4. $(w'_n)_{n\in\mathbb{N}}$ defined by $w'_{n+1} = 3w'_n - 8$ and $w'_0 = 3$.
5. $(F_n)_{n\in\mathbb{N}}$ defined by $F_{n+2} = F_{n+1} + F_n$, $F_0 = F_1 = 1$.

VI.A.2 Particular sequences

When given a recursive definition for a sequence, finding an explicit definition is the first goal: it allows for faster computation of any term. Each definition being different, there is no one recipe to find such an expression (if it is even possible). There are however some families of sequences for which there are formulas or methods to achieve an explicit definition.

VI.A.2.i Arithmetic and geometric growth

The simplest sequences are sequences that grow in the same manner between each term. If this growth is additive, the sequence is *arithmetic*; if the growth is multiplicative, the sequence is *geometric*.

Definition: Arithmetic sequence

An arithmetic sequence of increment d is defined:

- Recursively by $u_0 = a$ and $u_{n+1} = u_n + d$.
- Explicitly by $u_n = a + n \cdot d$.

Definition: Geometric sequence

A geometric sequence of ratio r is defined:

- Recursively by $u_0 = a$ and $u_{n+1} = r \cdot u_n$.
- Explicitly by $u_n = a \cdot r^n$.

The fact that the explicit definition of these sequences actually corresponds to the recursive definition will be proved in Section VI.B.1 and Exercise VI.3.

In practice, an arithmetic sequence can be recognized by looking at the difference between two consecutive terms: it is always the same. For a geometric sequence, the ratio of two consecutive terms is constant.

For example, let $(u_n)_{n\in\mathbb{N}}$ be defined by $u_n = 5n - 3$ for any $n \in \mathbb{N}$. For $n \in \mathbb{N}$, $u_{n+1} - u_n = (5(n+1) - 3) - (5n - 3) = 5n + 5 - 3 - 5n + 3 = 5$. So $(u_n)_{n\in\mathbb{N}}$ is an arithmetic sequence of increment 5.

Let's define $(v_n)_{n\in\mathbb{N}}$ by $v_0 = 2$, $v_{n+1} = 3v_n$. Then for any n, $\frac{v_{n+1}}{v_n} = \frac{3v_n}{v_n} = 3$, so $(v_n)_{n\in\mathbb{N}}$ is a geometric sequence of ratio 3.

Exercise VI.2

For the following starts of sequences:

- Conjecture based on the first terms whether it is an arithmetic, a geometric sequence (or none of these two kinds). Justify your answer.
- If so:
  - Find the increment or ratio.
  - Give a recursive definition.
  - Give an explicit definition.
  - Find the next term of the sequence.

1. $u_0 = 7, u_1 = 4, u_2 = 1, u_3 = -2, u_4 = -5, \dots$
2. $v_0 = 3, v_1 = -6, v_2 = 12, v_3 = -24, v_4 = 48, \dots$
3. $w_0 = 13, w_1 = 6.5, w_2 = 3.25, w_3 = 1.625, w_4 = 0.8125, \dots$
4. $s_0 = 2, s_1 = 3, s_2 = 8, s_3 = 33, s_4 = 158, \dots$

VI.A.2.ii Sum of terms of a sequence

It is quite common that the sum of the value is of more interest than the individual values. For example when the sequence represents a yearly value and the total value is of interest.

In mathematical notations, sums are written with the Σ symbol (which is the Greek letter for S).

$$\sum_{i=0}^{n} u_i = u_0 + u_1 + \cdots + u_n$$

Below the Σ is the index name (here i) and its starting value (0). Above is the ending value (here n). Next to the Σ is the expression that is being summed, using the index.

For those familiar with programming, this is akin to declaring a loop with index `i`, starting at `0` and ending at `n`: `sum=0; for (int i=0; i<=n; i++) {sum += u(i);}`

This notation is quite handy because it has some good properties:

- Linearity: when the terms being summed is a linear expression, it can be decomposed into a linear expression of sums:

$$\sum_{i=0}^{n}(x \cdot a_i + y \cdot b_i) = x \cdot \sum_{i=0}^{n} a_i + y \cdot \sum_{i=0}^{n} b_i$$

- Decomposition: a sum can be split into several sums at any index between the start and end: for $0 \le k \le n$,

$$\sum_{i=0}^{n} u_i = \sum_{i=0}^{k} u_i + \sum_{i=k+1}^{n} u_i$$

Summing sequences up to infinity (this is called a *series*) is also possible, although one has to be careful because the result may be infinite. These series show up as a good way to approximate real numbers and functions (such as the exponential, logarithm, sine, and cosine). For example, it can be proved that $\pi = 4\sum_{i=0}^{\infty} \dfrac{(-1)^i}{2i+1}$, so calculating this sum up to a certain point provides an approximation of π.

This is called *Leibniz formula.*

In the case of arithmetic and geometric sequences, the sum of their terms up to n can actually be expressed in a closed-form formula (i.e. a formula that does not use the terms of the sequence themselves).

VI.A.2.ii.a Sum of an arithmetic sequence

Theorem: Closed-form formula for an arithmetic series

If $u_n = a + n \cdot d$ then $\sum_{i=0}^{n} u_n = \frac{(n+1)(2a+n\cdot d)}{2}$.

Proof

This proof uses a technique that is very useful for sequences: we introduce symmetry by summing the terms twice, in order to match the first term with the last.
We write $S = \sum_{i=0}^{n} u_n$. So

$$\begin{aligned} 2S &= u_0 + u_1 + \cdots + u_n + u_0 + u_1 + \cdots + u_n \\ &= (u_0 + u_n) + (u_1 + u_{n-1}) + \cdots + (u_n + u_0) \\ &= (a + a + n \cdot d) + (a + d + a + (n-1) \cdot d) + \cdots + (a + n \cdot d + a) \\ &= (2a + n \cdot d) + (2a + n \cdot d) + \cdots + (2a + n \cdot d) \\ 2S &= (n+1)(2a + n \cdot d) \end{aligned}$$

So $S = \sum_{i=0}^{n} u_n = \frac{(n+1)(2a+n\cdot d)}{2}$. □

VI.A.2.ii.b Sum of a geometric sequence

Theorem: Closed-form formula for a geometric series

If $u_n = a \cdot r^n$ with $r \neq 1$ then $\sum_{i=0}^{n} u_n = \frac{a(r^{n+1}-1)}{r-1}$.

Proof

This proof uses another technique that is very useful for sequences: trying to find self-similarities. In this case, if the sum is multiplied by r it looks a lot like the original sum: Let $S = \sum_{i=0}^{n} u_n$. We have:

$$\begin{aligned} r \cdot S &= \phantom{a+{}} a \cdot r + \cdots + a \cdot r^n + a \cdot r^{n+1} \\ S &= a + a \cdot r + \cdots + a \cdot r^n \end{aligned}$$

So $rS - S = a \cdot r^{n+1} - a \Leftrightarrow S(r-1) = a(r^{n+1}-1)$ so for $r \neq 1$, $S = \sum_{i=0}^{n} u_n = \frac{a(r^{n+1}-1)}{r-1}$. □

Remark that in the case where $r = 1$, then all terms are equal and $\sum_{i=0}^{n} u_n = (n+1)a$.

VI.A.2.iii Arithmetico-geometric sequences

One step of difficulty higher than arithmetic and geometric sequences is a mixing of these two: arithmetico-geometric sequences, where the next term is calculated as a linear expression of the current one.

Definition: Arithmetico-geometric sequence

An arithmetico-geometric sequence is defined recursively by $u_0 = a$ and $u_{n+1} = r \cdot u_n + d$.

To find an explicit equivalent definition, we will use the *fixed-point* (sometimes written *fixpoint*) of this recurrence relation: a value x such that if $u_n = x$, then $u_{n+1} = x$. This can be calculated by solving the simple linear equation $x = rx + d \Leftrightarrow x(1-r) = d \Leftrightarrow x = \frac{d}{1-r}$. Note that we have assumed here that $r \neq 1$, otherwise it is an arithmetic sequence (with no fixed-point if $d \neq 0$) and we have $u_n = a + n \cdot d$.

Now the trick is to define an auxiliary sequence, that will be the original one shifted by the fixed-point: we define the shifted sequence $v_n = u_n - \frac{d}{1-r}$ (i.e. $u_n = v_n + \frac{d}{1-r}$). We can the rewrite the equality $u_{n+1} = r \cdot u_n + d$ as follows:

$$\begin{aligned}
u_{n+1} &= r \cdot u_n + d & \Leftrightarrow \\
v_{n+1} + \tfrac{d}{1-r} &= r \cdot (v_n + \tfrac{d}{1-r}) + d & \Leftrightarrow \\
v_{n+1} &= r \cdot (v_n + \tfrac{d}{1-r}) + d - \tfrac{d}{1-r} & \Leftrightarrow \\
v_{n+1} &= r \cdot v_n + \tfrac{rd}{1-r} + d - \tfrac{d}{1-r} & \Leftrightarrow \\
v_{n+1} &= r \cdot v_n + \tfrac{rd+d-rd-d}{1-r} & \Leftrightarrow \\
v_{n+1} &= r \cdot v_n
\end{aligned}$$

So v_n is a geometric sequence and $v_n = v_0 \cdot r^n = (a - \frac{d}{1-r})r^n$, hence $u_n = (a - \frac{d}{1-r})r^n + \frac{d}{1-r}$.

While it is not really interesting to memorize the formula for the explicit definition of an arithmetico-geometric sequence, it is good to memorize the method of using the auxiliary sequence obtained by shifting by a fixed-point; this is because this trick is not easily rediscovered, while the calculations that then lead to the result are easily deducted.

Example of arithmetico-geometric sequence

Consider the sequence recursively defined by $u_0 = 2$ and $u_{n+1} = 3u_n + 5$. The fixed point of the recurrence relation is x that satisfies the equation $x = 3x + 5 \Leftrightarrow x = -\frac{5}{2}$.

We can now define the shifted sequence $v_n = u_n + \frac{5}{2}$, so $u_n = v_n - \frac{5}{2}$. Then $v_{n+1} - \frac{5}{2} = 3(v_n - \frac{5}{2}) + 5 \Leftrightarrow v_{n+1} = 3v_n - \frac{15}{2} + 5 + \frac{5}{2} \Leftrightarrow v_{n+1} = 3v_n + \frac{-15+10+5}{2} \Leftrightarrow v_{n+1} = 3v_n$. Since $(v_n)_{n \in \mathbb{N}}$ is a geometric sequence of ratio 3, we can write $v_n = v_0 \cdot 3^n = \frac{9}{2} \cdot 3^n$. So for any n, $u_n = \frac{9}{2} \cdot 3^n - \frac{5}{2} = \frac{3^{n+2}-5}{2}$.

VI.B Proofs by induction

VI.B.1 Principle of induction

Universal properties over sequences defined recursively can be proved in a manner following the recursive definition, using a proof *by induction*.

Proof by induction rule

Let $P(n)$ be a predicate over integers.
If $P(0)$ and $\forall n \in \mathbb{N}, P(n) \Rightarrow P(n+1)$, then $\forall n \in \mathbb{N}, P(n)$.

The idea behind this proof rule is as follows. Assume $P(0)$ and $\forall n \in \mathbb{N}, P(n) \Rightarrow P(n+1)$. To prove that $\forall n \in \mathbb{N}, P(n)$, take an integer n. Starting at 0, we have $P(0)$ and the universal quantifier can be instantiated with $n = 0$: $P(0) \rightarrow P(1)$. By Modus Ponens we obtain $P(1)$.

Similarly, the universal quantifier can be instantiated with $n = 1$: $P(1) \rightarrow P(2)$. Since we have $P(1)$, by Modus Ponens we obtain $P(2)$. This continues until reaching the desired value n.

This reasoning is being abstracted away in the rule, and in practice it is applied as follows:

Proof by induction scheme

- State the property being proved by defining predicate P.
- Prove $P(0)$.
- Take $n \in \mathbb{N}$, assume $P(n)$ and prove $P(n+1)$.
- Conclude.

In this scheme, predicate $P(n)$ is called the *induction hypothesis*. The proof of $P(0)$ is called the *base case*, while proving $P(n+1)$ from $P(n)$ is the *induction case*. These steps are usually explicitly specified in the proof.

As this mimics exactly how recursive sequences are defined, it is well suited for any property on sequences.

Example: Proof by induction of explicit representation of arithmetic sequences

Let's prove by induction that if $(u_n)_{n\in\mathbb{N}}$ is defined ny $u_0 = a$ and $u_{n+1} = u_n + d$, then for any n, $u_n = a + n \cdot d$. Formally, let $P(n)$ be the predicate $u_n = a + n \cdot d$.

Base case: For $n = 0$. We have $u_0 = a = a + 0 \cdot d$ so the property holds for 0, i.e. $P(0)$ is true.

Induction case: Let $n \in \mathbb{N}$. Assume $P(n)$, meaning that the property holds for n, namely that $u_n = a+n\cdot d$. Then $u_{n+1} = u_n + d = a+n\cdot d+d = a+(n+1)\cdot d$, so the property holds for $n+1$: $P(n+1)$ is true.

Therefore, by induction the property holds for any n: $\forall n \in \mathbb{N}, P(n)$.

Exercise VI.3

Prove by induction that a geometric sequence of ratio r and initial term $u_0 = a$ can be expressed as $u_n = a \cdot r^n$ for any n.

Exercise VI.4

Let $u_0 = 4$ and $u_{n+1} = 3 \cdot u_n - 2$. Prove that for any $n \in \mathbb{N}$, $u_n = 3^{n+1} + 1$.

Exercise VI.5

1. Calculate a closed-form formula for $\sum_{i=0}^{n} i$.
2. Prove the same result using a proof by induction.

VI.B.2 Variations on induction

VI.B.2.i Finding the right predicate

One issue with proofs by induction is that they can only be written once you know exactly what you want to prove. And this may not be easy, for example for formulas it is sometimes hard to know in advance what is the closed form. This can be *conjectured* based on a couple of terms: calculate the first few terms and try to find a pattern. This provides a conjecture that remains to be proved (by induction).

In some cases, to prove predicate P it is easier to use a *stronger* predicate P' which is proved by induction. Predicate P' is deemed stronger is for any $n \in \mathbb{N}$, $P'(n) \Rightarrow P(n)$. While it is in usually harder to prove a stronger predicate, in the case of induction the strength of P' can play in your favor because in the inductive case P' is used as an hypothesis: a stronger hypothesis means there is more to rely on.

In these case, choosing the right predicate is not obvious. It is often found by trial and error: when failing to prove $P(n+1)$ from $P(n)$, try a stronger version P', if that fails try an even stronger one P''... One case where the stronger predicate is quite straightforward is the *strong induction* scheme, described in Section VI.B.2.iii.

VI.B.2.ii The proof by contradiction version of induction

To prove $\forall n \in \mathbb{N}, P(n)$ by contradiction, we assume its negation and show a contradiction. In this case the negation is $\exists n \in \mathbb{N}, \neg P(n)$. So we can assume that there is some value n that violates predicates P: $\neg P(n)$. Because we are working with natural numbers, we can assume n to be the *smallest* such value: it is possible because $\mathbb{N}$ has a smallest element ($\mathbb{N}$ is *Well ordered*). Then we prove first that $n \neq 0$, i.e. $P(0)$ holds (corresponding to the base case). Now that it can be assumed that $n > 0$, we show that $n-1$ also violates P: $\neg P(n-1)$ (corresponding to the induction case). That contradicts the fact that n was the smallest. Therefore the assumption that $\exists n \in \mathbb{N}, \neg P(n)$ does not hold so the property $\forall n \in \mathbb{N}, P(n)$ is true.

VI.B.2.iii Strong induction

While the induction scheme was presented starting from 0, it is possible it actually starts higher: often 1, sometimes 2, rarely but possibly higher. There can even be more than one base case, and therefore the induction case will assume the predicate holds for several values to prove it for the next one.

For example, when trying to prove a property P about the Fibonacci sequence defined by $F_0 = F_1 = 1$ and for any $n \in \mathbb{N}$, $F_{n+2} = F_{n+1} + F_n$, one will need:

- to prove that $P(0)$ and $P(1)$ hold (base cases);
- to assume $P(n)$ and $P(n+1)$ and prove $P(n+2)$ (induction case).

As a rule of thumb, proofs by induction on sequences follow the same structure as the recursive definition of said sequence.

A proof using two base cases will *in fine* be a proof that relies on two steps before. This principle can be generalized to write proofs that rely on *several* steps before, while not specifying exactly how many: the assumption used to prove $P(n)$ is that P holds for any value strictly smaller than n.

Proof by strong induction rule

Let $P(n)$ be a predicate over integers.
If $P(0)$ and $\forall n \in \mathbb{N}, (\forall i \leq n, P(i)) \to P(n+1)$, then $\forall n \in \mathbb{N}, P(n)$.

This is very useful when a recurrence relation is defined using terms that are smaller but not necessarily the previous one. For example for $u_n = u_{\lfloor \frac{n}{2} \rfloor} + 4$: the term used in the recurrence relation is not the previous one, but *a* previous one, thus requiring to assume any property holds for this term, regardless of its actual index (as long as it is smaller so that the induction terminates).

Proof by strong induction scheme

- Prove $P(0)$.
- Take n, assume $\forall i \leq n, P(i)$ and prove $P(n+1)$. (Sometimes notations are made easier by assuming $\forall i < n, P(i)$ and then proving $P(n)$.)

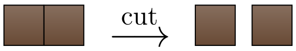
This scheme is actually a proof by induction on the stronger predicate P' defined as $P'(n) = \forall i \leq n, P(i)$.

Example of strong induction: The chocolate bar problem

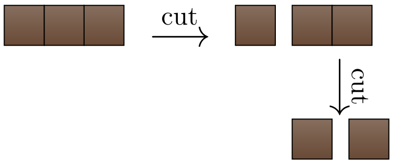
The problem is as follows: a chocolate bar is made of n squares in a single line:

How many cuts are needed to cut it in n individual squares?

First, let's conjecture what this number is by calculating it for a couple of small values:

- $n = 1$, no need to cut so 0 cuts needed:
- $n = 2$, cut once to get two pieces, so 1 cut needed: 

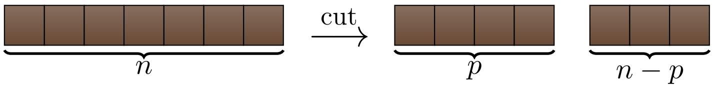
- $n = 3$, cut once to get 2 and 1 (or 1 and 2), then again to cut the 2-bar into single pieces, so 2 cuts:



It seems that $n - 1$ cuts are needed to cut a bar of length n: this will be our conjecture. Let's prove by strong induction that $n - 1$ cuts are needed for a bar of n squares.

Base case $n = 1$: no cut needed, so 0 cuts.

Inductive case Assume that for any $p < n$, $p - 1$ cuts are needed for a bar of p squares. Let's number the squares of the bar from 1 to n, and choose a number p between 1 and $n - 1$. When we cut in the bar of n squares right after square number p, we obtain two bars of lengths p and $n - p$, with $1 \leq p < n$.



We have both $p < n$ and $n - p < n$ so there are $p - 1$ cuts needed for one piece and $n - p - 1$ needed for the other. So a total of $1 + (p - 1) + (n - p - 1) = n - 1$ cuts. □

While this problem may look a bit artificial, more mathematical applications of strong induction will be done in Chapter VII when cutting integers into pieces as part of number theory.

Exercise VI.6

Let $(u_n)_{n \in \mathbb{N}}$ be the sequence defined by $u_0 = 0$, $u_1 = 1$, for any $n \in \mathbb{N}$, $u_{n+2} = \frac{1}{4} \cdot (u_{n+1}^2 + u_n + 2)$. Prove that for any n, $0 \leq u_n \leq 1$.

VI.B.2.iv Structural induction

This is for your personal culture more than to be applied in this course.

The induction may be over something else than an integer.

For example, one could use an induction proof scheme over a couple of integers ordered lexicographically (so that the order is total). In this case the induction is often a strong induction: prove $P(0, 0)$, then for $n, m \in \mathbb{N}$, assume for any couple $(n', m') \prec (n, m)$, $P(n', m')$ is true, and prove $P(n, m)$.

The same principle can be used for more than numbers, actually. Induction works because we use a proof of a predicate P on a *smaller* integer to prove $P(n)$. This can be extended to any *well-ordered set*: a set that has minimal elements. These minimal elements are what serves as base cases: the induction does not go on forever.

So we can use a proof by induction to prove properties about a set of objects where the proof of a predicate on *smaller* objects is used. The definition of what makes an object smaller may be built as a lexicographic order, but it can also be a more natural notion, such as being a sub-part of the bigger object. Once again, the form of the proof will follow the structure of the definition of the object. Since it relies exclusively on said structure, it is named *structural induction*.

For example, it is possible to write proofs about the set of formulas of propositional logic by structural induction on sub-formulas:

Base case Prove the property on atomic propositions.

Induction case Prove for every operator $*$ that $P(\varphi)$ and $P(\psi)$, then $P(\varphi * \psi)$.

We can use such a proof scheme to prove that every formula can be written using only $\to$ and $\bot$ (as was done in Section I.B.3.v.b): this holds for atomic propositions, and for any operator we prove that it can actually be replaced by combinations of $\to$ and $\bot$. In this case there will be more than one induction case, as there will be one per operator.

A structural induction can also be used to write proofs about proof trees in sequent calculus.

VI.C Application: Growth and complexity

VI.C.1 What is complexity?

In computer science, *complexity* is the number of basic steps a program must take to execute. Such basic steps include arithmetic operations, variable assignments, function calling... As such, complexity allows to measure of how long the program takes to execute, while abstracting away the differences that may arise by running the program on different hardware.

As most program execute on data, this measure depends on the value of the inputs to the program. For example, consider the following program that calculates $\sum_{i=1}^{n} i^2$:

- Take a value n as an argument (think input).
- Start with the result at 0.
- For each number i from 1 to n, add i^2 to the result
- Then return the result.

Executing this program requires to perform n times the following: multiply i by itself, then adding it to the current result, and finally assign the new value to the result. That means $3n$ basic operations. To which we must add the initialization of the result to 0. So in the end we can say that this program requires $3n + 1$ operations to execute on input n.

This calculation has neglected to count that there must be some operations performed to maintain the value of i, so that is a couple of operations n times, and maybe count the action of returning as a basic operation. But that actually does not really matter: we are not interested in a precise number but in the *growth* of the complexity when n increases. And in this case, what matters is that it is proportional to n: we don't need to know precisely how many operations are performed each time, as long as it is the same for all the values of i. And we don't need to know precisely how many operations are needed to initialize the program and have it return, because this does not change with n. Both these precise values could be made negligible by using a faster computer, but it cannot change the fact that doubling the value of n will (roughly) double the execution time.

VI.C.1.i Big-O notation

To write the complexity of programs while denoting that we are only interested in growth, we use the O ("big O" or "big Omicron") notation. Formally, it is defined as follows:

$$f(n) = O((g(n)) \text{ iff } \exists M \in \mathbb{R}, \limsup_{n \to +\infty} \left| \frac{f(n)}{g(n)} \right| \leq M$$

What the above definition states is that he limit of $\frac{f}{g}$ is bounded, meaning that f does not grow any order of magnitude faster than g.

So the function g used in practice is the leading factor of growth of f, stripped of any constant coefficient. Let's consider the following examples:

- $2021 = O(1)$: any constant is a $O(1)$; that is the complexity of a program that executes in the same time regardless of its arguments.
- $3n + 1 = O(n)$: this means the growth is (at most) linear, as the program described above.
- $7n^3 - 3n^2 + 4n - 9 = O(n^3)$: in the case of polynomials, the main growth factor is the leading coefficient. Sometimes knowing that the complexity is polynomial is enough (not considering the degree), this is what constitutes the P class.
- $2^{7n} + 42n^{79} = O(2^{7n}) = 2^{O(n)}$: since exponentials grow faster than polynomials, the leading growth factor is the exponential here.

See Section I.C.3.ii for more about the P class.

VI.C.1.ii Worst-case complexity

Programs do not always execute the same code: that is the purpose of conditionals (`if` statements). So the number of operations may be very different depending on the actual value. In these situations what matters is the *worst-case*.

Consider this (purposefully weird) program:

- Take argument n?
- If n is even return 42
- Otherwise (i.e. if n is odd) calculate the decomposition of $n! + 17!$ into prime factors.

When n is even, it executes in constant time, while it can take a very long time since $n!$ growth exponentially and there is not polynomial factorization known (and it is possible that none exist). In this case, what matters for complexity is only the case when n is odd.

Stirling's formula shows that $n!$ grows as fast as $n^n e^n \sqrt{2\pi n}$.

As a result, complexities are always stated with the understanding that the program executes in *less than* $O(f(n))$ steps.

VI.C.2 Complexity of recursive programs

One way to think about complex problems is to break them down into smaller ones, that ought to be more manageable. This idea is at the crux of the *divide and conquer* approach: break a big problem into smaller ones, solve these, then bring the solutions together This strategy was named after the method used by Julius Caesar to conquer Gauls (more or less equivalent to modern France): instead of fighting the whole Gallic peoples, fight each tribe separately; a legion of 1000 can easily beat a tribe of 100, and do so 20 times instead of facing a coalition of 2000 in one go.

Most common variations of the divide and conquer approach are as follows. In the first one, assume you can solve a smaller version of the same problem, and prove you can from that solve the current version. Otherwise said, to solve a problem for input n (or size n), assume you can solve the same problem for size $n-1$ and then show how to go from the solution for $n-1$ to a solution for n. In the second one, divide the problem in two, solve these sub problems, and combine. Otherwise said, to solve a problem for input n (or size n), assume you can solve the same problem for size $\lceil \frac{n}{2} \rceil$ and then show how to combine the solutions for $\lceil \frac{n}{2} \rceil$ to a solution for n. Of course, for this idea to work, one ought to be able to solve the problem for $n = 0$ or $n = 1$, i.e. the base case.

The complexity of the solving the whole problem is therefore expressed as a recurrence relation on smaller problems. The base case, usually size 0 or 1, is often solved with a constant number of operations. From this recurrence relation, to get a complexity expressed in the O notation, an explicit definition is needed. But finding this explicit definition from the recurrence relation may not be easy: the recurrence rarely is an arithmetico-geometric sequence for which we have a predefined strategy. Therefore the approach will be in two steps:

1. Calculate a couple of terms from n down ("unwinding"), until you are able to *conjecture* (i.e. guess) the explicit form.

2. Prove that the explicit form correspond to the recurrence relation using a proof by induction.

VI.C.2.i Example: The Towers of Hanoi

The problem of the Towers of Hanoi is one of the best known in recursive programming and complexity analysis. It is stated as follows:

- There are three poles A, B, C, and n discs of increasing size, labeled 1 to n.

- Initially, all the discs are stacked in a pyramid on the first pole.

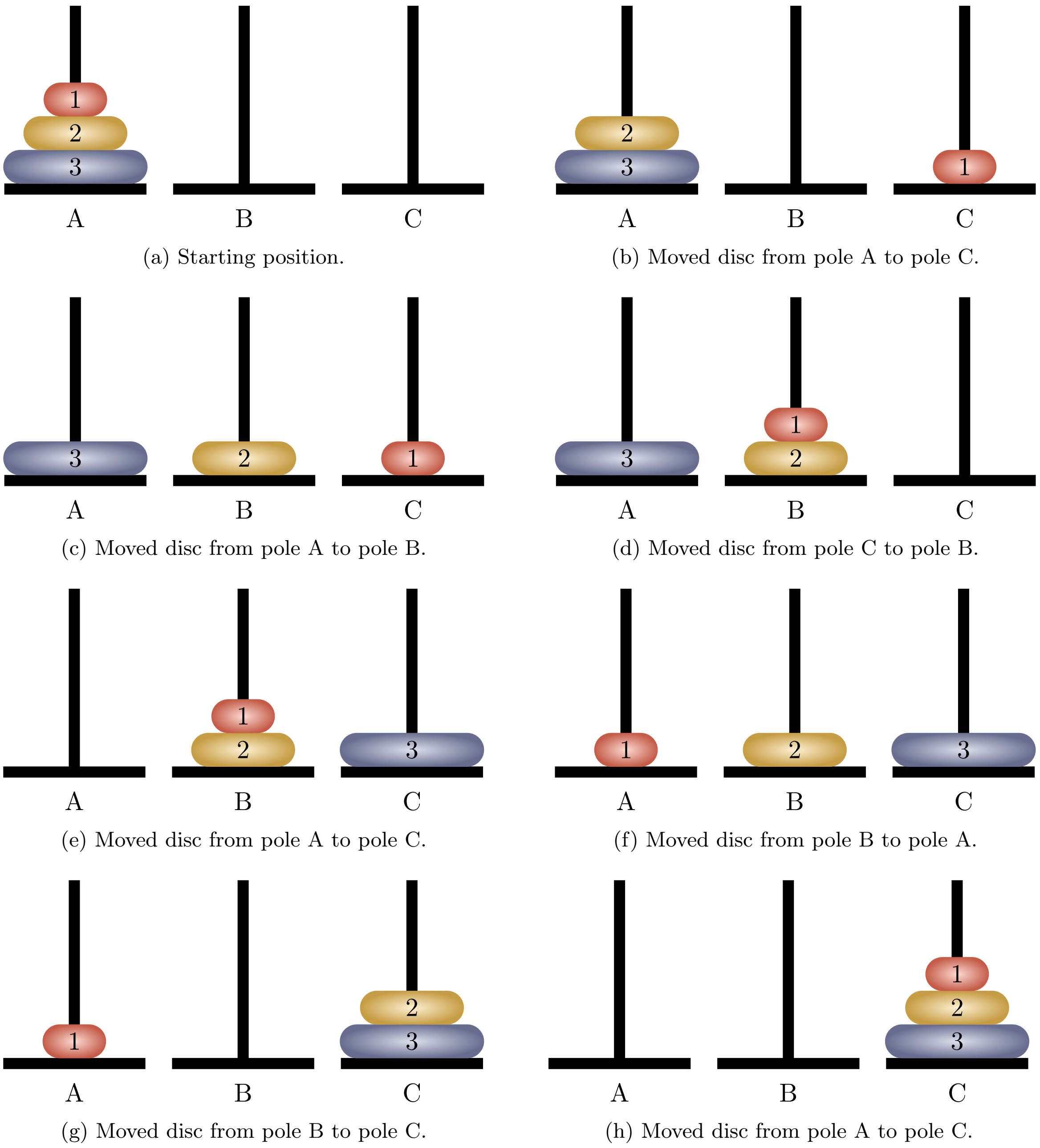
- You can move the topmost disc of a pole to any other pole, but a disc must always rest on a bigger disc.

The questions being: How do you move the whole stack from one pole to another? How many moves does that take?

This problem can be solved using the divide and conquer approach. Assuming I know how to move a stack of $n-1$ discs. To move n discs from A to C, I proceed as follows: I can move the first $n-1$ discs from A to B. Then move disc number n from A to C. Finally move the first $n-1$ discs from B to C. For this to be complete I must tell how to proceed when $n = 1$, which is quite trivial: move the single disc.

Note that the beauty of this description lies in the fact that the move of the $n-1$ discs is not described extensively. This kind of procedure can therefore be programmed in a very elegant code, that follows the structure of the reasoning above. Nonetheless, the details can be filled in by looking into what it means for $n-1$, and that yields an actual procedure to move the stack of n discs. For example, for $n = 3$, the set of moves is given in Figure VI.1[1]. Note that one can easily ignore the intermediate steps: only consider Figures VI.1(a,d,e,h), showing the moves of the stack of 2 discs from A to B in one step instead of 3.

[1] Adapted from `http://www.texample.net/tikz/examples/towers-of-hanoi/`.



(a) Starting position.

(b) Moved disc from pole A to pole C.

(c) Moved disc from pole A to pole B.

(d) Moved disc from pole C to pole B.

(e) Moved disc from pole A to pole C.

(f) Moved disc from pole B to pole A.

(g) Moved disc from pole B to pole C.

(h) Moved disc from pole A to pole C.

Figure VI.1: Tower of Hanoi – 3 Discs.

Let's now analyze the complexity of the problem, i.e. count the number of moves that are required to move a stack of n discs, which we will call $f(n)$.. Our procedure can be summarized as follows: to move n discs, we need to move $n-1$ discs, move one disc, move $n-1$ discs again. We can therefore deduct that the function f follows the following recurrence: $f(n) = f(n-1) + 1 + f(n-1) = 2 \cdot f(n-1) + 1$. The base case being a single move to move a single disc, we have the base case of the recurrence:$f(1) = 1$.

As it is not straightforward to see what is the explicit definition of f that corresponds to this recursive definition, we will unwind the recurrence and try to find a pattern.

$$\begin{array}{rcl}
f(n) & = & 2 \cdot f(n-1) + 1 \\
& = & 2 \cdot (2 \cdot f(n-2) + 1) + 1 \\
& = & 2^2 \cdot f(n-2) + 2 + 1 \\
& = & 2^2 \cdot (2 \cdot f(n-3) + 1) + 2 + 1 \\
& = & 2^3 \cdot f(n-3) + 2^2 + 2^1 + 2^0 \\
& = & 2^3 \cdot f(n-3) + (2^3 - 1) \\
& \vdots & \\
& = & 2^i \cdot f(n-i) + (2^i - 1) \\
& \vdots & \\
& = & 2^{n-1} \cdot f(n-(n-1)) + (2^{n-1} - 1) \\
& = & 2^{n-1} \cdot f(1) + (2^{n-1} - 1) \\
& = & 2^{n-1} \cdot 1 + (2^{n-1} - 1) \\
f(n) & = & 2^n - 1
\end{array}$$

We can conjecture from this that $f(n) = 2^n - 1$. It remains to be proved that this function matches the recursive definition:

Base case: For $n = 1$, $2^1 - 1 = 2 - 1 = 1 = f(1)$ so the base case is satisfied.

Induction case: Let $n > 1$. Assume that $f(n-1) = 2^{n-1}$. Then let's use the recursive definition: $f(n) = 2 \cdot f(n-1) + 1 = 2 \cdot (2^{n-1} - 1) + 1 = 2 \cdot 2^{n-1} - 2 + 1 = 2^n - 1$ so the recurrence relation is satisfied

We can therefore conclude by induction that for any n, $f(n) = 2^n - 1$, so $2^n - 1$ steps are needed to move a stack of n discs. We can deduct an approximate complexity: $f(n) = O(2^n)$.

Note that in this case, it takes an exponential number of moves to move the discs: adding one disc doubles the number of moves. The legend associated to this problem stated that in a monastery in Hanoi (Vietnam), monks had the task of moving a stack of 64 discs according to these rules. Once they are done, the world would end; but that is not a very frightening threat: even if a move takes as little as one second, over 584 billion years would be needed for the whole procedure to be complete (while the sun will go out in "only" 5 billion years). It must also be noted that there is no such monastery, it is pure invention by French mathematician Édouard Lucas who created the problem in the 19th century.

VI.C.2.ii Example: Merge sort

Sorting data is a very common action in computer systems. It is also an interesting problem because there are lots of ways of sorting, and different algorithms may have different complexity. Note that in theory the sorting can be performed as long as the data is equipped with an order, but here we will assume the data is integers for simplicity.

Orders are defined in Section V.B.

The problem is formally stated as follows: the input is an array of size n containing integers. The returned array must be sorted in increasing order.

The solution used here applied the divide and conquer principle:

- Divide the array in two in the middle.
- Recursively sort the two halves, which are of sizes $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$.
- Merge them together as follows. Take an empty array. Start inserting elements from the two sorted arrays, always choosing the smallest one that has not been inserted. Note that because the arrays are sorted this procedure only requires traversing the array in one way.

The base cases occur when the size is either 1 or 0: the array is already sorted and there is nothing to do. Note that we never actually reach size 0 with this algorithm, and this case is mentioned for completeness' sake in case the original input is an array of size 0.

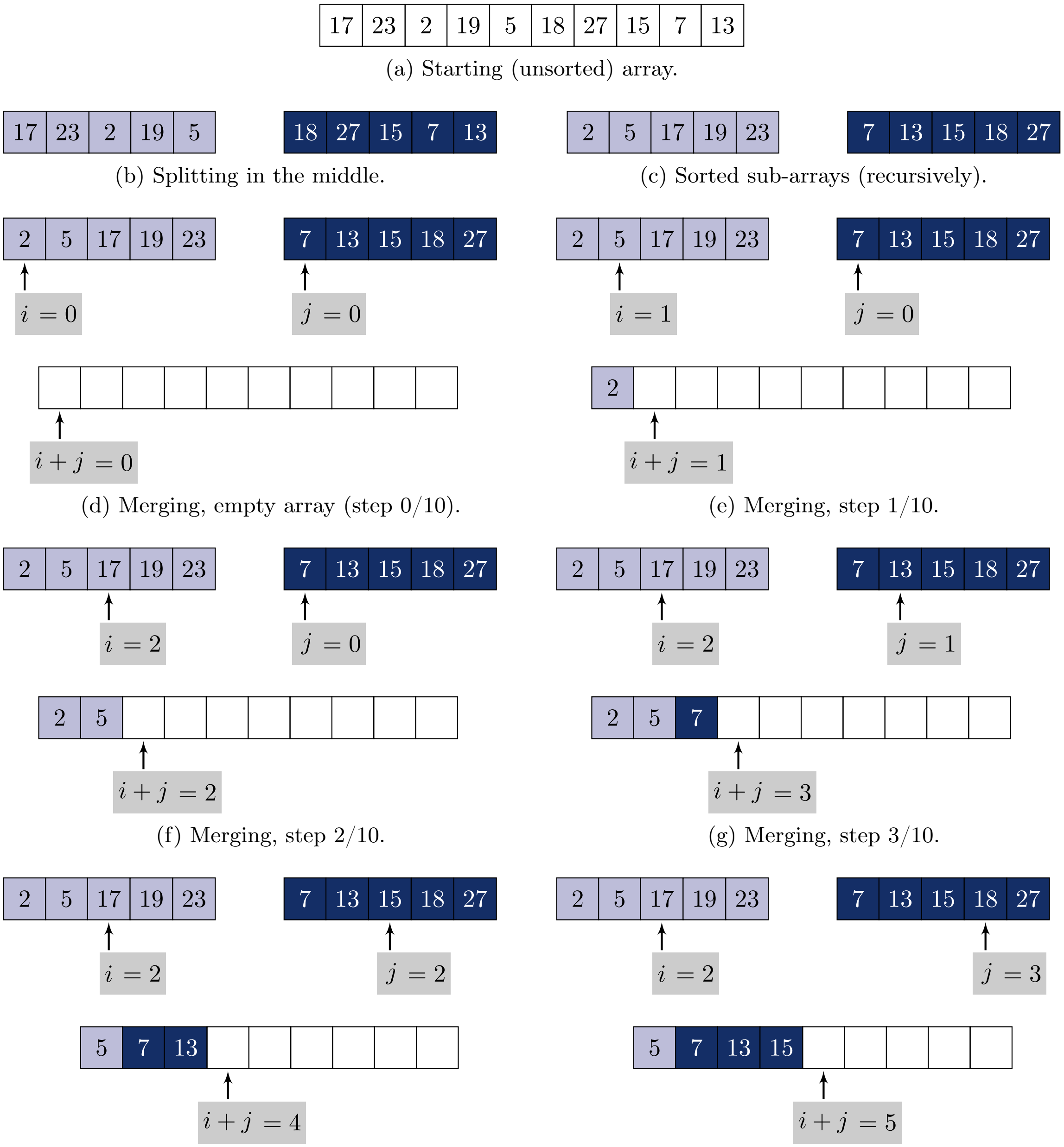
An example of this procedure is given in Figure VI.2 for an array of size 10. The recursive sorting of the sub arrays (of size 5) is not given.

To analyze the complexity of the Merge sort, let's assume there are $n = 2^k$ elements in the original array. Although this is rarely the case in practice that the array's size is precisely a power of two, it can be assumed that it is the size of the next power of two, padded with ∞. So in the case of the array of Figure VI.2, we could assume it is the following array of size 16:

17	23	2	19	5	18	27	15	7	13	∞	∞	∞	∞	∞	∞

Now let's consider the complexity in terms of k rather than n: we are looking for the complexity function $g(k)$ that gives the number of operations needed to sort an array of size 2^k with the merge sort algorithm. Splitting the array in the middle means copying all the elements in two new arrays. And the merging means, for every cell of the array, comparing the two current minimal elements of the sorted sub-arrays and writing the smallest one in the array. In both cases these operations take a time proportional to the number of elements, so a time $c \cdot 2^k$, with c being a constant. The sorting of each sub-arrays is done in time $g(k-1)$, because each array is half the length of the original one. As a result, the complexity function g obeys the following recurrence relation: $g(k) = 2 \cdot g(k-1) + c \cdot 2^k$. The base case, which means sorting an array of size 1, requires no action, besides recognizing that this case has been reached. This is done in constant time so $g(0) = c'$, for some constant c'.

In practice the splitting can be done without copy, but that does not change the overall complexity.



(a) Starting (unsorted) array.

(b) Splitting in the middle.

(c) Sorted sub-arrays (recursively).

(d) Merging, empty array (step 0/10).

(e) Merging, step 1/10.

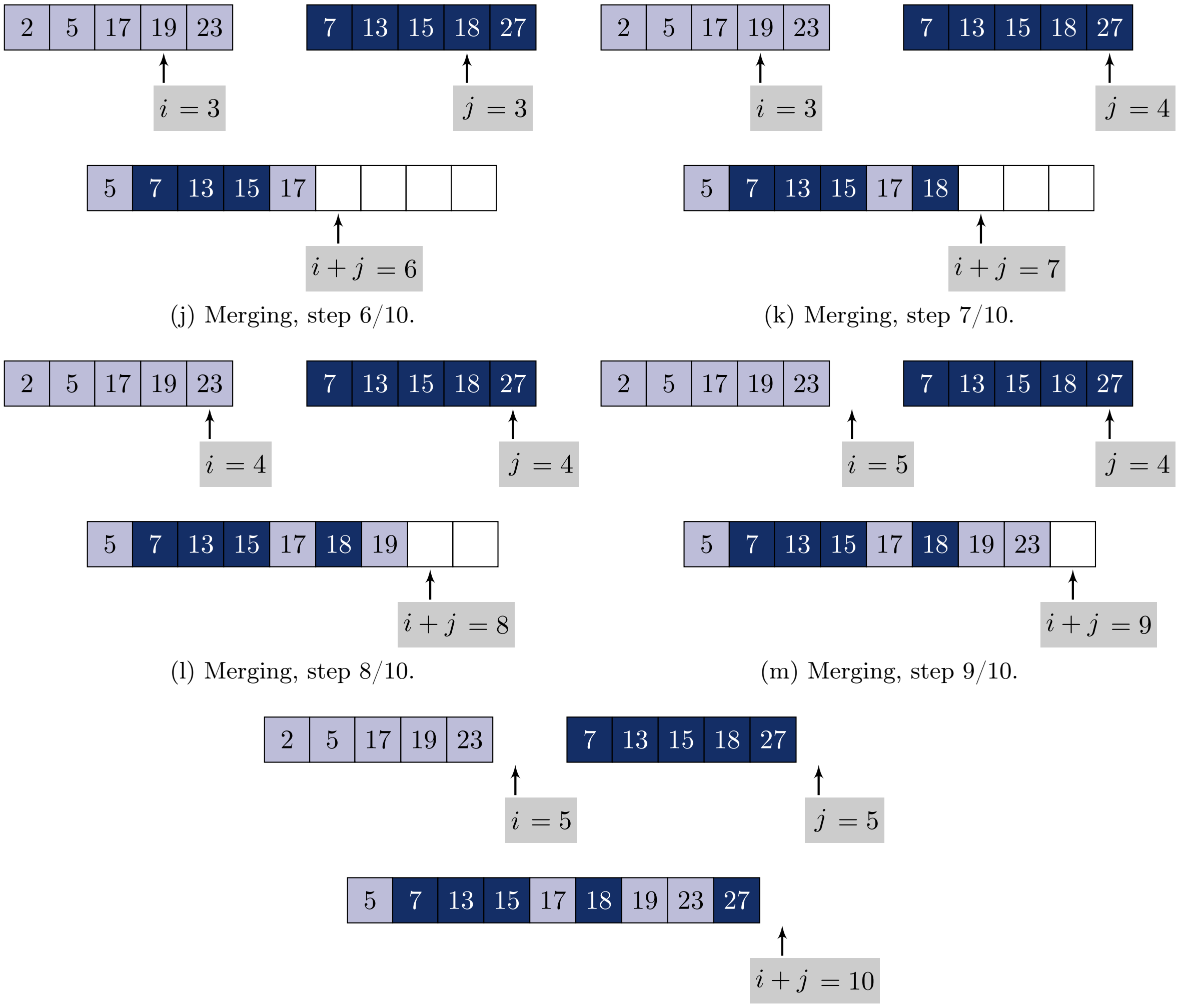
(f) Merging, step 2/10.

(g) Merging, step 3/10.

(h) Merging, step 4/10.

(i) Merging, step 5/10.

Figure VI.2: Example of merge sort; the recursive calls are not detailed. Continued next page.



(j) Merging, step 6/10.

(k) Merging, step 7/10.

(l) Merging, step 8/10.

(m) Merging, step 9/10.

(n) Merging, ended (step 10/10).

Figure VI.2: Example of merge sort; the recursive calls are not detailed. Continued from previous page.

We can now try to conjecture an explicit form by unwinding this recurrence relation:

$$\begin{array}{rcl}
g(k) & = & 2 \cdot g(k-1) + c \cdot 2^k \\
 & = & 2 \cdot (2 \cdot g(k-2) + c \cdot 2^{k-1}) + c \cdot 2^k \\
 & = & 4 \cdot g(k-2) + c \cdot 2^k + c \cdot 2^k \\
 & = & 2^2 \cdot g(k-2) + 2 \cdot c \cdot 2^k \\
 & = & 2^2 \cdot (2 \cdot g(k-3) + c \cdot 2^{k-2}) + 2 \cdot c \cdot 2^k \\
 & = & 2^3 \cdot g(k-3) + 3 \cdot c \cdot 2^k \\
 & \vdots & \\
 & = & 2^i \cdot g(k-i) + i \cdot c \cdot 2^k \\
 & \vdots & \\
 & = & 2^k \cdot g(k-k) + k \cdot c \cdot 2^k \\
 & = & 2^k \cdot g(0) + k \cdot c \cdot 2^k \\
g(k) & = & 2^k \cdot c' + k \cdot c \cdot 2^k
\end{array}$$

We can see the previous calculations as follows. Each time the array is divided into two, so there are k divisions before reaching arrays of size 1. After i divisions, let's call that level i, we have 2^i arrays, each of size 2^{k-i}. So all the merging done at level i means reading 2^i arrays of size $2^k - i$, so reading 2^k values. That means the time to merge is the same at every level. Since there are k levels, the global time needed for all the merging will be $k \cdot 2^k$. The $2^k \cdot c'$ is the time needed to treat all the base cases, but that is negligible compared to $k \cdot 2^k$ as k grows to infinity.

Therefore the conjecture on g is that for any $k \in \mathbb{N}$, $g(k) = 2^k \cdot c' + k \cdot c \cdot 2^k$. Let's prove it by induction.

Base case: For $k = 0$, we have $2^k \cdot c' + k \cdot c \cdot 2^k = 2^0 \cdot c' + 0 \cdot c \cdot 2^0 = c' = g(0)$.

Induction case: Let $k > 0$, assume the property holds for $k-1$, i.e. $g(k-1) = 2^{k-1} \cdot c' + (k-1) \cdot c \cdot 2^{k-1}$. Let's use the recursive definition:

$$\begin{array}{rcl}
g(k) & = & 2 \cdot g(k-1) + c \cdot 2^k \\
 & = & 2(2^{k-1} \cdot c' + (k-1) \cdot c \cdot 2^{k-1}) + c \cdot 2^k \\
g(k) & = & 2^k \cdot c' + k \cdot c \cdot 2^k
\end{array}$$

So the property holds for k.

As a result the property holds for any k, and the complexity of merge sort is $2^k \cdot c' + k \cdot c \cdot 2^k$. We can express it as a O notation: the term in $k \cdot 2^k$ is the leading one, so the complexity is $O(k \cdot 2^k)$. Remark that we can translate that back into a complexity for n: if $n = 2^k$, then $k = \log_2(n)$, so the complexity is $O(n \log(n))$.

The base of the logarithm does not matter in O notation, since the change of base is only a constant term.

Exercise VI.7

Consider the following algorithm (called *binary search*) to find a value v in a **sorted** array:

- If the array has size > 0, look at the middle element m:

  - If $m = v$ we have found the element, return the index of m.
  - If $m < v$ search for v in the right half of the array (m is excluded).
  - If $m > v$ search for v in the left half of the array (m is excluded).
- An array of size 0 does not contain v.

Assuming the array has a size $2^k - 1$:

1. What is the size of the left half of the array? And of the right half?
2. Find a recurrence relation that expresses $f(k)$, the number of steps it takes to find v in the worst case (which is when v is actually not in the array), based on $f(k-1)$. Assume the complexity for an empty array is a constant p.
3. Unwind this relation and conjecture an explicit formula for $f(k)$.
4. Prove your conjecture.
5. Conclude about the complexity of this algorithm.

Chapter VII
Number Theory

Chapter contents

VII.A Division of integers **108**
VII.A.1 Divisibility 108
VII.A.1.i Definition 108
VII.A.1.ii Properties 108
VII.A.2 Euclidean division (a.k.a Long division) 110
VII.B Prime numbers **113**
VII.B.1 Definition 113
VII.B.2 Euclid's Lemma 114
VII.B.3 Fundamental Theorem of Arithmetic 114
VII.B.4 Checking primality 116
VII.B.5 Decomposition into prime factors 116
VII.B.6 From the decomposition to the list of divisors 117
VII.B.7 Infinity of primes 118
VII.C Greatest Common Divisor and Least Common Multiple **119**
VII.C.1 Definitions 119
VII.C.2 Properties 120
VII.C.3 Euclid's Algorithm 121
VII.C.4 Applications: fractions 123
VII.D Diophantine Equations **123**
VII.D.1 Extension to $\mathbb{Z}$ 123
VII.D.2 Definition and Bézout's Theorem 124
VII.E Congruence **127**
VII.E.1 The congruence relation 127
VII.E.2 Equivalence classes and canonical representative 129
VII.E.3 Modulo Arithmetic 130
VII.E.3.i Addition of equivalence classes 130
VII.E.3.ii Multiplication with equivalence classes 130
VII.E.3.iii Modular exponentiation 131
VII.E.4 Applications 132
VII.E.4.i Toolbox (Lemmas) 132
VII.E.4.ii Divisibility criteria 133
VII.E.4.iii Chinese Remainder Theorem 133
VII.E.4.iv RSA cryptography 136

VII.A Division of integers

Number theory refers to the study of how integers are formed and relate to each other, especially through multiplication and division. Hence the central role that the concept of divisibility plays in this chapter. Note that most of the theory focuses on natural integers, but that it can be extended to deal with all integers, as is explained in Section VII.D.1.

VII.A.1 Divisibility

VII.A.1.i Definition

Definition: Divisibility

Let $n, m \in \mathbb{N}$. n *divides* p, written $n \mid p$, iff there exists $k \in \mathbb{N}$ such that $n \cdot k = p$.

So to prove that, $n \mid p$, one has to exhibit the value k such that $n \cdot k = p$. For example:

- $3 \mid 15$ because $3 \cdot 5 = 15$
- $7 \mid 42$ because $7 \cdot 6 = 42$
- $6 \mid 42$ because $6 \cdot 7 = 42$
- $1 \mid 17$ because $1 \cdot 17 = 17$
- $42 \mid 0$ because $42 \cdot 0 = 0$
- $23 \mid 23$ because $23 \cdot 1 = 23$

We write $n \nmid p$ when it is not the case that n divides p. This is proved by considering all values for k and proving they cannot work. For example:

- $7 \nmid 24$ because $7 \cdot k$ is either $0, 7, 14, 21, 28$ or > 28.
- $23 \nmid 1$ because $23 \cdot k$ is either 0 or > 23.

Note that there are other (better) ways to prove that $n \nmid m$, which will be explained in Section VII.A.2.

When $n \mid p$, $\frac{p}{n}$ denotes the integer k such that $n \cdot k = p$. In that case n and $\frac{p}{n}$ are called *divisors* or *factors* of p. If, however, $n \nmid p$, the $\frac{p}{n}$ notation has no real sense, since we are only considering integers, and from this point of view fractions that are not whole do not even exist.

VII.A.1.ii Properties

VII.A.1.ii.a Relation with addition (and subtraction)

Although divisibility deals with multiplication, it is interesting to consider how it behaves with respect to addition (and its reverse subtraction). Namely, we can only have results when both operands are divisible by the same number.

Proposition

Let $n, p, q \in \mathbb{N}$.

- If $n \mid p$ and $n \mid q$, then $n \mid p + q$.
- If $n \mid p$ and $n \mid p + q$, then $n \mid q$.

Proof

Let $n, p, q \in \mathbb{N}$.

- Assume $n \mid p$ and $n \mid q$. Then there exists k, m such that $n \cdot k = p$ and $n \cdot m = q$. So $n \cdot (k + m) = p + q$ and $n \mid p + q$.
- Assume $n \mid p$ and $n \mid p + q$. Then there exists k, m such that $n \cdot k = p$ and $n \cdot m = p + q$. Note: since $q \geq 0$, we have $k \leq m$. So $n \cdot m = n \cdot k + q$, and $n \cdot (m - k) = q$ So $n \mid q$. □

This can be adapted for subtraction:

Proposition

Let $n, p, q \in \mathbb{N}$.

- If $n \mid p$, $q \geq p$ and $n \mid q - p$, then $n \mid q$.
- If $n \mid p$, $q \geq p$ and $n \mid q$, then $n \mid q - p$.

Although a direct proof can be used (left as exercise to the reader), this is only a particular case of the previous Proposition (using $q - p$ instead of q).

VII.A.1.ii.b Divisibility Order

Another interesting aspect of divisibility is how it can propagate. In fact, we can show the property is transitive, and even better, that it is a partial order. This was already proved in Section V.B.1.ii.c, but the main elements of the proof will be reproduced here as well.

We can however note that 0 and 1 have a particular role with respect to this order. Since 1 divides any integer, it is a *minimal element* to the order. On the other hand, 0 does not divide any number (but itself), so it is a *maximal element*.

Theorem

- Divisibility is a partial order.
- 1 is a minimal element with respect to this order.
- 0 is a maximal element with respect to this order.

Proof

Let $n, p, m \in \mathbb{N}$.

- $\boxed{\mathbf{R}}$ We have $n \mid n$ by choosing $k = 1$ in the definition. So we have reflexivity.

  $\boxed{\mathbf{A}}$ Assume $n \mid p$ and $p \mid n$, then $n \cdot k = p$ and $p \cdot k' = n$ so $k \cdot k' = 1$ hence $k = k' = 1$ and $n = p$. So we have antisymmetry.

  $\boxed{\mathbf{T}}$ Assume $n \mid p$ and $p \mid m$ then $n \cdot k = p$ and $p \cdot k' = m$ so $n \cdot k \cdot k' = m$ hence $n \mid m$. So we have transitivity.
- The order is partial, because for example $7 \not| 24$ (see above) and $24 \not| 7$, as for

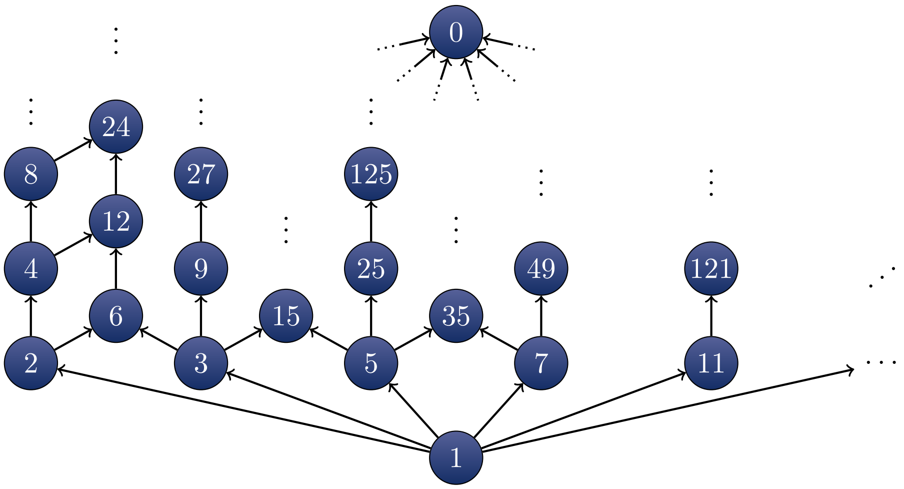


Figure VII.1: The divisibility lattice.

any k $24k$ is either 0 (if $k = 0$) or strictly greater than 7 (if $k \geq 1$).

- $1 \mid n$, by choosing $k = n$. If $n \neq 1$ then $n \nmid 1$: for any k, $n \cdot k$ is either 0 or strictly greater than 1.
- $n \mid 0$, by choosing $k = 0$. If $n \neq 0$ then $0 \nmid n$: for any k, $0 \cdot k = 0$.

It is possible to draw a *graph* of this order, as follows. Put each number in a vertex (a.k.a. node, a circle with the value inside), and draw an arrow between node containing numbers n to node containing number p if $n \mid p$. In graph vocabulary, we use $\mathbb{N}$ as the vertices and $\mid$ as the transition relation.

To clarify a bit this graph, we can omit self-loops and transitions that can be deducted from transitivity. And of course, to actually represent it, we must choose a finite subset of integers.

Part of this graph is represented on Figure VII.1. The shape that this graph has is particular, and is called a *lattice*, in reference to the criss-crossed pattern. It has several properties that will be detailed throughout the chapter. What we can already see is that 1, being the minimal element, is at the base of the lattice, while the maximal element 0 is on top, with arrows from every number pointing to it.

Several notions in this chapter bear the name of the ancient Greek mathematician *Euclid*, who introduced them in his book the *Elements*.

VII.A.2 Euclidean division (a.k.a Long division)

When m is a multiple of n, it is possible to *divide* m by n, which is exactly what is behind the concept of divisibility. This can be generalized to the case when n does not divide m, using the *Euclidean division*. This is not really a new concept, as this is also called *long division*. What may be new is that we are actually going to prove properties of this division, so what we are giving is not a definition, but rather a theorem that states these properties.

Theorem: Euclidean Division

Given two integers $n, m \in \mathbb{N}$, with $m > 0$, there exists two *unique* integers $q, r \in \mathbb{N}$ such that

$$n = m \cdot q + r \quad \text{and} \quad 0 \leq r < m$$

The very important fact is that there is only one pair (q, r) that satisfies the property: there can be no ambiguity when writing a Euclidean division. Because of this uniqueness, we call q *the quotient* (not "a") and r is *the remainder* (again, not "a").

The equality $n = m \cdot q + r$ itself is called *the Euclidean division of n by m*. And n is called the *dividend*, while m is the *divisor*. Note that "the divisor in a Euclidean division" is not the same as "a divisor", since in the former case divisibility is not assumed.

For example:

- The Euclidean division of 42 by 5 is $42 = 5 \cdot 8 + 2$: quotient is 8, remainder is 2.
- The Euclidean division of 132 by 3 is $123 = 3 \cdot 41 + 0$: quotient is 41, remainder is 0.
- The Euclidean division of 79 by 12 is $79 = 12 \cdot 6 + 7$: quotient is 6, remainder is 7.
- The Euclidean division of 79 by 6 is $79 = 6 \cdot 13 + 1$: quotient is 13, remainder is 1.

In the last two examples, one can see that dividing 79 by its quotient by 12 does not yield back 12 as the quotient. This is because the criterion on the remainder depends on the divisor.

Proof of the Euclidean Division

If we write down what we need to prove as a first order formula, it has the following form:

$$\forall n, m \in \mathbb{N}, m > 0 \to \exists! q, r \in \mathbb{N}, n = m \cdot q + r \wedge 0 \leq r < m.$$

We can actually change it a bit, using the rules of first order logic, in order to have the quantification on m at the outermost level. So we use the rule $\forall x, \varphi \to P(x) \equiv \varphi \to (\forall x, P(x))$ where x does not appear in φ. Here we have that n does not appear in "$m > 0$", so we obtain the equivalent formula

$$\forall m \in \mathbb{N}, m > 0 \to \forall n \in \mathbb{N}, \exists! q, r \in \mathbb{N}, n = m \cdot q + r \wedge 0 \leq r < m.$$

Let $m \in \mathbb{N}$ and assume that $m > 0$. We are going to prove by strong induction that $\forall n \in \mathbb{N}, \exists! q, r \in \mathbb{N}$ such that $n = m \cdot q + r$ and $0 \leq r < m$. Note that we have to prove two things: existence and uniqueness. For existence, we have to exhibit q and r. For uniqueness, we assume the existence of another pair $(q', r') \neq (q, r)$ and deduct a contradiction.

Base case $n = 0$.

Existence We have $0 = m \cdot 0 + 0$ and $q = r = 0$ satisfies the property.

Uniqueness Assume there exists some other couple $(q', r') \neq (0, 0)$ that satisfies the property.

- Assume $q' > 0$: then $m \cdot q' > 0$, so $n = m \cdot q' + r' > 0$, which is a contradiction with $n = 0$.

- Otherwise, $q' = 0$, so $r' > 0$, so $n = m \cdot q' + r' > 0$, which is again a contradiction with $n = 0$.

$\boxed{\textbf{Inductive case}}$ Assume that for any $p < n$, the property holds, i.e., there exists a unique couple (q_0, r_0) that satisfies the conditions: $p = m \cdot q_0 + r_0$ and $0 \leq r_0 < m$. We have two cases to consider, that actually match the basic algorithm used to compute this division: if you cannot fit m into n, then you are done. If you can, then subtract m from n and increment the quotient, then start again.

- Assume $n < m$.

  $\boxed{\textbf{Existence}}$ Write $n = m \cdot 0 + n$: we have $q = 0$ and $r = n < m$ by assumption.

  $\boxed{\textbf{Uniqueness}}$ Assume there exists some other values for q' and r': another couple $(q', r') \neq (0, n)$ that satisfies the property.

  - If $q' > 0$ that means $q' \geq 1$, therefore $m \cdot q' \geq m$. So $n = m \cdot q' + r' \geq m + r' \geq m$, which is a contradiction with $n < m$.
  - Otherwise, $q' = 0$ so it must be the case that $r \neq n$. Writing the division, we have $n = m \cdot q' + r' = m \cdot 0 + r' = r' \neq n$ which is a contradiction.

- Otherwise, i.e. $n \geq m$. Write $p = n - m < n$. By induction hypothesis, we have a unique couple (q_0, r_0) that satisfies the conditions: $n - m = m \cdot q_0 + r_0$ and $0 \leq r_0 < m$.

  $\boxed{\textbf{Existence}}$ From the previous equation we get $n = m \cdot q_0 + r_0 + m = m \cdot (q_0 + 1) + r_0$. We choose $q = q_0 + 1$ and $r = r_0$.

  $\boxed{\textbf{Uniqueness}}$ Assume by contradiction that there is another couple (q', r'), with $n = m \cdot q' + r'$ and $0 \leq r' < m$. If $q' = 0$ then $n = r' < m$, which contradicts $n \geq m$, so it must be the case that $q' \geq 1$. We can therefore subtract m from both sides of the division: $n - m = m \cdot (q' - 1) + r'$. If $q' - 1 \neq q_0$ or $r' \neq r_0$, that contradicts the uniqueness of q_0 and r_0.

Exercise VII.1

For the following numbers n and m, write the Euclidean division of n by m.

1. $n = 37$, $m = 4$
2. $n = 43$, $m = 8$
3. $n = 39$, $m = 3$

As we remarked, Euclidean division is a generalization of the division when $n \mid m$. The special case can actually be detected because then the remainder is 0. That provides an easier proof of $n \not| m$ than trying any value for k and showing that $n \cdot k \neq m$: just show that the remainder is not 0.

Proposition

$m \mid n$ iff the remainder of the Euclidean division of n by m is 0.

Proof

We can prove this equivalence by proving two implications.

$\boxed{\Rightarrow}$ Assume $m \mid n$. Then there exists q such that $m \cdot q = n$. This is the Euclidean division with $r = 0$: $n = m \cdot q + 0$.

$\boxed{\Leftarrow}$ Assume $r = 0$ in the Euclidean division of n by m. Then $n = m \cdot q$ so $m \mid n$.

Exercise VII.2

For the following numbers n and m, determine whether $n \mid m$ or $n \nmid m$. Justify your answer.

1. $n = 10$, $m = 50$
2. $n = 50$, $m = 10$
3. $n = 13$, $m = 63$
4. $n = 37$, $m = 37$
5. $n = 37$, $m = 0$
6. $n = 1$, $m = 19$

Exercise VII.3

List *all* the divisors of 28 by trial and error.

VII.B Prime numbers

Prime numbers are the basic atoms (in the literal sense: “which cannot be cut”) of numbers, as they cannot be decomposed as a product of other numbers:

VII.B.1 Definition

Definition: Prime number

A number $n \in \mathbb{N}$ is *prime* iff it has exactly two divisors: 1 and n.

Not that by this definition, 0 and 1 are not prime numbers, even though they cannot really be decomposed either. They have a special role in arithmetic, as 1 is the neutral element while 0 is absorbing for multiplication, so it is convenient to exclude them from this definition.

The first prime numbers are $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37$. We can see in Figure VII.1 that they constitute the “line” at the base of the lattice of divisibility: they are not divisible by anything but 1.

A number that is not prime is *composite*, so for example 21 is composite because its divisors are $1, 3, 7, 21$. As a result it can be written as $21 = 3 \cdot 7$, which is a *decomposition* into the product 3 and 7. For a prime number, the only possible “decomposition” being the trivial one $p = 1 \cdot p$, which actually is not a real decomposition because p appears as the component.

VII.B.2 Euclid's Lemma

Because of their atomicity, prime numbers have good properties regarding divisibility. This result is best known as Euclid's Lemma:

Euclid's Lemma

Let $n, m, p \in \mathbb{N}$. If is p prime and $p \mid n \cdot m$, then $p \mid n$ or $p \mid m$.

Before even attempting to prove this Lemma, we can remark that the condition of p being prime is crucial here. For example $6 \mid 8 \cdot 3$, because $8 \cdot 3 = 24$ and $6 \cdot 4 = 24$. But $6 \nmid 8$ and $6 \nmid 3$. What is happening here which cannot happen with a prime number is that $6 = 2 \cdot 3$, so one can write 24 in different ways: $24 = 6 \cdot 4 = 2 \cdot 3 \cdot 4 = 8 \cdot 3$. As a prime number cannot be decomposed, any way of writing the product $n \cdot m$ has to feature p or a multiple of p.

Proof of Euclid's Lemma

Let $n, m, p \in \mathbb{N}$. Assume that p is prime, and $p \mid n \cdot m$. First, we can treat the case when either n or m is 1: the result is trivial. For example if $n = 1$, then $n \cdot m = m$, so the assumption $p \mid n \cdot m$ yields directly $p \mid m$; the case for $m = 1$ being similar. So we can assume that both n and m are not 1 in the sequel of the proof.

We will prove the result by contradiction: assume $p \nmid n$ and $p \nmid m$. We can choose m to be minimal: for given p and n, m is the smallest such value such that $p \mid n \cdot m$, $p \nmid n$, and $p \nmid m$.

We write the Euclidean division of m by p: $m = p \cdot q + r$, with $0 \leq r < p$. Since $p \nmid m$, the case $r = 0$ is impossible, so we actually have $0 < r < p$.

- Assume $q = 0$. Then $m = r$, so we have $0 < m < p$. We now write the Euclidean division of p by m: $p = m \cdot q' + r'$ with $0 < q'$, $0 < r' < m < p$; the case $r' = 0$ being impossible since $m \nmid p$ as p is prime. Therefore we can rewrite the division as $r' = p - m \cdot q'$. Multiplying both sides by n we obtain $n \cdot r' = n \cdot p - n \cdot m \cdot q'$. Since $p \mid n \cdot m$, there exists k such that $p \cdot k = n \cdot m$, so $n \cdot r' = n \cdot p - p \cdot k \cdot q' = p \cdot (n - k \cdot q')$. As a result $p \mid n \cdot r'$. We also have $p \nmid r'$ because $0 < r' < p$. So $p \mid n \cdot r'$, $p \nmid n$, and $p \nmid r'$ with $r' < m$, which contradicts the minimality of m.

- Otherwise, $q > 0$. Multiplying the Euclidean division by n on both sides, we get $n \cdot m = p \cdot q \cdot n + n \cdot r$. Since $p \mid n \cdot m$ and $p \mid p \cdot q \cdot n$, $p \mid n \cdot r$. Because $0 < r < p$, we have $p \nmid r$. Since $q > 0$, we have $r < m$. So $p \mid n \cdot r$, $p \nmid n$, and $p \nmid r$ with $r < m$, which contradicts the minimality of m. □

VII.B.3 Fundamental Theorem of Arithmetic

As prime numbers cannot be decomposed, they are the building blocks of all other numbers. This is what is stated in the *Fundamental Theorem of Arithmetic* (FTA):

Fundamental Theorem of Arithmetic

Every number $n > 1$ can be written in a unique way as a product of prime factors.

Here the uniqueness of the decomposition is what makes the result beautiful. Indeed, it makes intuitive sense that if a number can be decomposed, and said components themselves are decomposed, at one point the decomposition will only have elements that cannot be broken down any further, in our cases prime numbers. What is less intuitive is that this decomposition is unique: regardless of how the decomposition goes, in the end the same basic elements will always be there. Note that the uniqueness is understood up to commutativity: $21 = 7 \cdot 3 = 3 \cdot 7$ is the decomposition of 21 into 3 and 7.

Also note that once again 0 and 1 are excluded from the Theorem. Number 0 can be written in infinitely many ways that all require the use of 0 itself, which is not prime: $0 = 0 \cdot 5 \cdot 13 = 0 \cdot 3 \cdot 29 \cdot 101 = \ldots$. Number 1 can be written in a unique way, but it requires the use of the non-prime number 1: $1 = 1$.

Proof of the Fundamental Theorem of Arithmetic

As a decomposition breaks down a number into smaller ones but not necessarily the previous one, the proof uses strong induction. Note that both existence and uniqueness have to be proved.

Base case $n = 2$.

Existence $n = 2$ is a product of primes.

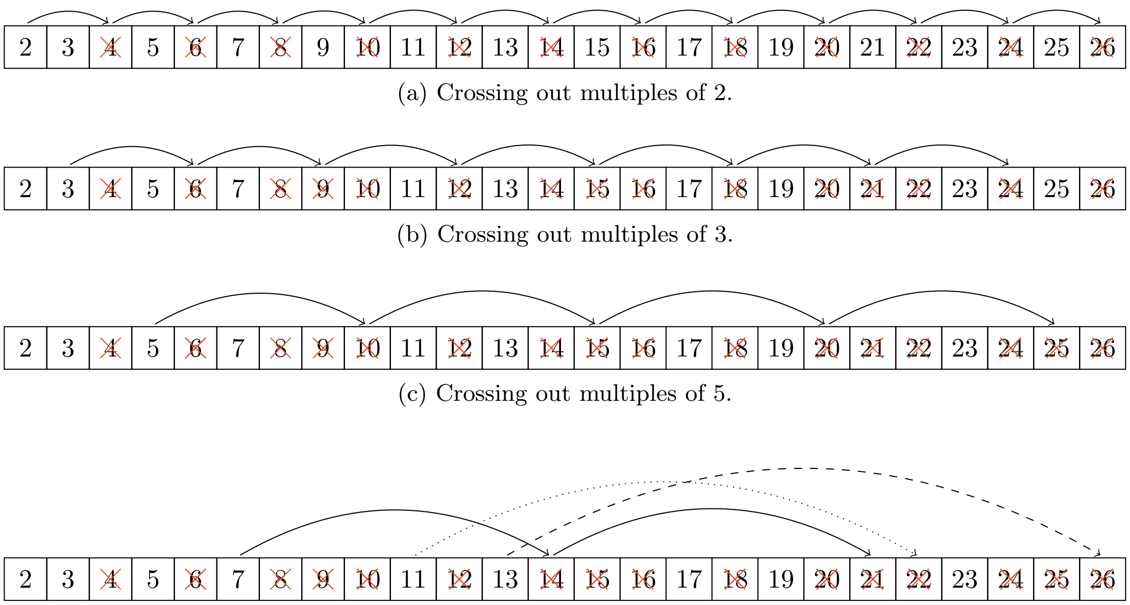
Uniqueness Any other decomposition $2 = p_1 \cdots p_n$ would violate the primality of 2, hence it is unique.

Inductive case Assume that for any $m < n$, m can be written uniquely as a product of primes.

- If n is prime, it is a product of primes. Any other decomposition $n = p_1 \cdots p_n$ contradicts the primality of n, hence it is unique.
- If n is not prime:

  Existence Let m be a divisor of n with $1 < m < n$. Then there is a k such that $m \cdot k = n$, and we also have $1 < k < n$. By induction hypothesis, $m = p_1 \cdot p_2 \cdots p_i$ is a product of primes and so is $k = p'_1 \cdot p'_2 \cdots p'_j$. As a result $n = m \cdot k = p_1 \cdot p_2 \cdots p_i \cdot p'_1 \cdot p'_2 \cdots p'_j$ is a product of primes.

  Uniqueness Assume that there are distinct decomposition of primes for n: $q_1 \cdot q_2 \cdots q_i = q'_1 \cdot q'_2 \cdots q'_j$. So $q_1 \mid q'_1 \cdot q'_2 \cdots q'_j$. By repeatedly applying Euclid's Lemma, we can show that q_1 divides one of the q'_ℓ: either $q_1 \mid q'_1$, or $q_1 \mid q'_2 \cdots q'_j$; in the latter case either either $q_1 \mid q'_2$, or $q_1 \mid q'_3 \cdots q'_j$; and so on until we find an index ℓ such that $q_1 \mid q'_\ell$. Since they are both primes, that means $q_1 = q'_\ell$. So there are two distinct decompositions for $\frac{n}{q_1} = q_2 \cdots q_i = q'_1 \cdots q'_{\ell-1} \cdot q'_{\ell+1} \cdots q'_j$. Since $\frac{n}{q_1} < n$, this contradict the induction hypothesis. □



(a) Crossing out multiples of 2.

(b) Crossing out multiples of 3.

(c) Crossing out multiples of 5.

(d) Crossing out multiples of 7, 11, 13.

Figure VII.2: Sieve of Eratosthenes up to 26.

VII.B.4 Checking primality

A consequence of the FTA is that every number is either prime or a product of primes. So in order to know whether a number n is prime, it is sufficient to try to divide by all prime numbers smaller than n, which reduces the number of divisions to try.

This idea is at the core of the *Sieve of Eratosthenes*, which works as follows:

1. Write all the numbers in a table (up to a certain limit).
2. Start at 2, strikeout all multiples of 2, as they won't be primes.
3. Move to the next non-stricken number p: as it has not been stricken out, it is not a multiple of any other prime so p is a prime number.
4. Strikeout all multiples of p, as they won't be primes.
5. Repeat from point 3 until the end of the table is reached.
6. All non-stricken numbers are the primes.

An example up to 26 is given in Figure VII.2.

Note that there are other (faster) methods to check if a number is prime, but they are beyond the scope of this course.

VII.B.5 Decomposition into prime factors

As the order of the prime factors do not matter, we usually write the decomposition starting from the lowest prime number to the highest. If a prime number appears more than one, then all the occurrence are to be gathered in a single power of a prime. For example $24 = 2 \cdot 2 \cdot 2 \cdot 3 = 2^3 \cdot 3$.

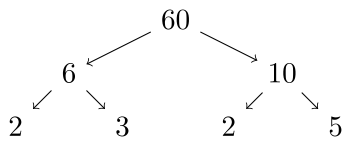


Figure VII.3: Decomposition of 60 into prime factors.

The proof of the FTA gives a procedure to decompose a number n into prime factors, based on the two sub-cases of the inductive case:

- If the number is prime, the decomposition is $n = n$.
- Otherwise: find a divisor d of n and decompose $\frac{n}{d}$ and d independently.

For example: $60 = 6 \cdot 10 = (2 \cdot 3) \cdot (2 \cdot 5) = 2^2 \cdot 3 \cdot 5$. The uniqueness of the decomposition means that it does not really matter what divisor is chosen, the end result will be the same. When calculating as a human, it is often more practical to use the factors that spring to mind; in the case of 60, splitting it as $6 \cdot 10$ was quite natural. On paper, it is not uncommon to write these decompositions in a tree-like pattern ,as is done in Figure VII.3.

Another method which is a bit more tedious but more systematic and hence easier to program in a computer is to always trying prime numbers until they do not divide in anymore: $60 = 2 \cdot 30 = 2 \cdot 2 \cdot 15 = 2 \cdot 2 \cdot 3 \cdot 5 = 2^2 \cdot 3 \cdot 5$. It is more systematic in part because one does not actually need to know when a number is prime: a non-prime number d will never be found as a factor since the prime factors of d would have been factored out already. So it does not hurt (for a computer, a human may be a bit smarter than that when prime numbers are known) to try all the numbers. In the example above, 12 is not found as a factor because all the factors 2 and 3 have been factored out already. This method has the added advantage of already providing the factors in the usual order.

To decompose by hand, a couple of basic criteria of divisibility allow for a fast check of whether a prime number is a factor:

$\boxed{\mathbf{2}}$ Last digit is even (0, 2, 4, 6, 8).

$\boxed{\mathbf{3}}$ The sum of the digits is a multiple of 3: 3, 6, 9, or a number with more digits that can also be summed in the same fashion.

This fact is proved in Section VII.E.4.ii.

$\boxed{\mathbf{5}}$ Last digit is either 0 or 5.

Exercise VII.4

Decompose the following numbers into prime factors:

1. 100
2. 198
3. 102
4. 105
5. 43
6. 1125

VII.B.6 From the decomposition to the list of divisors

As prime factors are a particular kind of divisors, they are quite useful in order to find all the divisors. Indeed, the prime factor decomposition of a number and any of its divisor are related.

Proposition

Let $n, m \in \mathbb{N}$. Assume that the decomposition of n into prime factors is $n = p_1^{k_1} \cdot p_2^{k_2} \cdots p_i^{k_i}$. If $m \mid n$, then the decomposition of m into prime factors is $m = p_1^{k'_1} \cdot p_2^{k'_2} \cdots p_i^{k'_i}$ where for any $i, k'_i \leq k_i$.

Proof

Take a number $n \in \mathbb{N}$ and its decomposition into prime factors: $n = p_1^{k_1} \cdot p_2^{k_2} \cdots p_i^{k_i}$. Let $m \in \mathbb{N}$ and assume that $m \mid n$. Then for any prime factor p of m, it is also a prime factor of n. This can be proved by repeatedly applying Euclid's Lemma, as was done in the proof of the FTA. Since this process can be repeated on $\frac{m}{p}$ which divides $\frac{n}{p}$, in the end every prime factor of m is also a prime factor of n. As a result we can write $m = p_1^{k'_1} \cdot p_2^{k'_2} \cdots p_i^{k'_i}$ where for any $i, k'_i \leq k_i$.

Remark that in this proposition, some powers in m may be 0, that only means that this prime factor of n is not a factor of m. Also note that the factors (or powers of factors) appearing in n but not in m allow to calculate $\frac{n}{m}$: $\frac{n}{m} = p_1^{k_1 - k'_1} \cdot p_2^{k_2 - k'_2} \cdots p_i^{k_i - k'_i}$.

Since a divisor of n share the same prime factores with lower powers, listing all the possible power combination for each prime yields the list of all the divisors of n: $p_1^0 \cdot p_2^0 \cdots p_i^0 = 1$, $p_1^1 \cdot p_2^0 \cdots p_i^0$, ..., $p_1^{k_1} \cdot p_2^0 \cdots p_i^0$, $p_1^{k_1} \cdot p_2^1 \cdots p_i^0$,..., $p_1^{k_1} \cdot p_2^{k_2} \cdots p_i^0$, ...$p_1^{k_1} \cdot p_2^{k_2} \cdots p_i^{k_i} = n$.

For example: $60 = 2^2 \cdot 3^1 \cdot 5^1$ so its divisors are:

- $2^0 \cdot 3^0 \cdot 5^0 = 1$
- $2^1 \cdot 3^0 \cdot 5^0 = 2$
- $2^2 \cdot 3^0 \cdot 5^0 = 4$
- $2^0 \cdot 3^1 \cdot 5^0 = 3$
- $2^1 \cdot 3^1 \cdot 5^0 = 6$
- $2^2 \cdot 3^1 \cdot 5^0 = 12$
- $2^0 \cdot 3^0 \cdot 5^1 = 5$
- $2^1 \cdot 3^0 \cdot 5^1 = 10$
- $2^2 \cdot 3^0 \cdot 5^1 = 20$
- $2^0 \cdot 3^1 \cdot 5^1 = 15$
- $2^1 \cdot 3^1 \cdot 5^1 = 30$
- $2^2 \cdot 3^1 \cdot 5^1 = 60$

Exercise VII.5

1. Decompose 1575 into prime factors.
2. List all its divisors.

VII.B.7 Infinity of primes

As prime numbers are the building blocks of integers, getting to know them a bit better is necessary to better understand arithmetics. One of the first question to answer is: how many primes are they? The answer is:

Theorem

There are infinitely many prime numbers.

Proof

We will use a proof by contradiction. Assume, by contradiction, that there is only finitely many prime numbers. They can be listed as: $p_1, p_2, \ldots, p_n$.
Let $m = 1 + \prod_{i=1}^{n} p_i = 1 + p_1 \cdot p_2 \cdots p_n$. That is m is the product of all the primes, plus 1.
For any $i \in \{1, \ldots, p\}$, consider the Euclidean division of m by p_i: we have $m = p_i \cdot (p_1 \cdot p_2 \cdots p_{i-1} \cdot p_{i+1} \cdots p_n) + 1$. So the remainder of the division is 1, hence $p_i \not| m$. So m cannot be decomposed into a product of primes, which contradicts the Fundamental Theorem of Arithmetic. □

Now that we know that there are infinitely many primes, there are more questions pertaining to "how many" there are that can be asked. For example, one can ask how many prime numbers are smaller than or equal to a given integer n. It has been shown (but will not be proved here), that as n grows, the number of primes smaller than n grows as $\frac{n}{\ln(n)}$. Otherwise stated, if a number is randomly selected between 1 and n (for a big enough n), the probability of selecting a prime number is $\frac{1}{\ln(n)}$. So the bigger the number n, the less likely a prime number is to be selected: prime numbers become rarer as n grows. This is known as the Prime Number Theorem, proved by Hadamard and Poussin in the late 19th century.

ℹ This is for your personal culture more than to be applied in this course.

VII.C Greatest Common Divisor and Least Common Multiple

VII.C.1 Definitions

Based on the building blocks that are the prime numbers, one can compare other numbers to one another, according to how they are built. So two numbers are alike up to the prime factors that they share, which is akin to saying that they are alike up to the largest divisor they share:

Definition: Greatest Common Divisor (GCD)

The Greatest Common Divisor of n and m (both > 0) is the largest number p such that $p \mid n$ and $p \mid m$. It is written $\gcd(n, m)$.

For example: let's calculate $\gcd(12, 27)$. The divisors of 12 are: $1, 2, 3, 4, 6, 12$. the divisors of 27 are: $1, 3, 9, 27$. The common divisors are therefore 1, 3, and 3 is the largest, so $\gcd(12, 27) = 3$.

For 24 and 35, the divisors are 1, 2, 3, 4, 6, 8, 12, 24 and 1, 5, 7, 35, respectively. The only common divisor is 1, so $\gcd(24, 35) = 1$.

For 4 and 36, the divisors are 1, 2, 4 and 1, 2, 3, 4, 6, 9, 12, 18, 36, so 1, 2, 4 are common and $\gcd(4, 36) = 4$.

Remark that 1 is always a common divisor, although not always the greatest one. One consequence is that the GCD is always defined for any two numbers. When $\gcd(n, m) = 1$, n and m are *co-prime*.

The GCD can be read from the lattice of divisibility (see Figure VII.1 page 110) as follows: from n and m, take all the arrows backwards (that corresponds to divisors), and the first time these explorations meet is the GCD. For example to calculate $\gcd(12, 27)$ from the lattice: from 24 explore 8 and 12, then 4 and 6 then 2 and 3, then 1. From 27 explore 9, then 3, then 1. The first time they meet is 3, so $\gcd(12, 27) = 3$.

The meeting point (this is lattice terminology), has a dual notion: the join. Namely, for the meet we explored downward in the lattice, while for the join we explore upward. Upward exploration means exploring multiples, so the first time these explorations join, the least common multiple is found:

Definition: Least Common Multiple (LCM)

The Least Common Multiple of n and m (both > 0) is the smallest number $p > 0$ such that $n \mid p$ and $m \mid p$. It is written $\mathrm{lcm}(n, m)$.

For examples, let's calculate $\mathrm{lcm}(6, 8)$. Non-zero multiples of 6 include 6, 12, 18, 24, 30, 36, 42, 48,... Non-zero multiples of 8 include 8, 16, 24, 32, 40, 48,... The common one include $24, 48, \ldots$, so the smallest is 24 and $\mathrm{lcm}(6, 8) = 24$. Note that on the lattice of Figure VII.1, the explorations went: 6, 12, 24 and 8, 16, 24 (16 is not displayed) so they join at 24.

In the case of 9 and 6, multiples are 9, 18, 27, 36, 45, 54,... and 6, 12, 18, 24, 30, 36, 42, 48, 54, ..., respectively. The common multiples being 18, 54,... we have $\mathrm{lcm}(9, 6) = 18$.

For 14 and 15, the multiples are 14, 28, 42, 56, 70, 84, 98, 112, 126, 140, 154, 168, 182, 196, 210,... and 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210,... So $\mathrm{lcm}(15, 14) = 210$. Note that here $210 = 14 \cdot 15$.

We can remark that 0 is always a common multiple, but as it is not interesting, it is excluded from the definition. Another easy common multiple of n and m is $n \cdot m$. While it may not be the smallest, its existence ensures that the LCM is always defined.

VII.C.2 Properties

As the definitions of GCD and LCM are symmetric for n and m, these notions are symmetrical:

Proposition: Symmetry

Let $n, m \in \mathbb{N}$ and assume $n, m > 0$.

- $\gcd(n, m) = \gcd(m, n)$.
- $\mathrm{lcm}(n, m) = \mathrm{lcm}(m, n)$.

As noted, the GCD is related to how prime factors are shared between n and m. This gives a method to compute GCD, and LCM, based on the decompositions:

Proposition: GCD and LCM from prime factor decomposition

Let $n, m \in \mathbb{N}$ and assume $n, m > 0$. Let $n = p_1^{k_1} \cdot p_2^{k_2} \cdots p_i^{k_i}$ and $m = p_1^{j_1} \cdot p_2^{j_2} \cdots p_i^{j_i}$ be the decompositions into prime factors of n and m. Then:

- $\gcd(n, m) = p_1^{\min(k_1, j_1)} \cdot p_2^{\min(k_2, j_2)} \cdots p_i^{\min(k_i, j_i)}$.
- $\mathrm{lcm}(n, m) = p_1^{\max(k_1, j_1)} \cdot p_2^{\max(k_2, j_2)} \cdots p_i^{\max(k_i, j_i)}$.

Note that in this proposition we assume that the prime number appearing in the decompositions are identical because some powers might be 0. For example we write $12 = 2^2 \cdot 3^1 \cdot 5^0$ and $15 = 2^0 \cdot 3^1 \cdot 5^1$. Based on this $\gcd(12, 15) = 2^0 \cdot 3^1 \cdot 5^0 = 3$ and $\operatorname{lcm}(12, 15) = 2^2 \cdot 3^1 \cdot 5^1 = 60$.

Proof

Let d be a common divisor of m and n. As we remarked in Section VII.B.6 d can be written $p_1^{r_1} \cdot p_2^{r_2} \cdots p_i^{r_i}$ with for any $\ell \in \{1, \dots, i\}$, $r_\ell \leq k_\ell$ (since $d \mid n$) and $r_\ell \leq j_\ell$ (since $d \mid m$). So $r_\ell \leq \min(k_\ell, j_\ell)$. Now consider $g = p_1^{\min(k_1,j_1)} \cdot p_2^{\min(k_2,j_2)} \cdots p_i^{\min(k_i,j_i)}$. We just proved that g is a common divisor to n and m. Now for any divisor d, since the power associated to every prime number p_ℓ in d is $r_\ell \leq \min(k_\ell, j_\ell)$, $d \mid g$, so $d \leq g$, as a result g is the greatest common divisor.
The reasoning is similar for the LCM: any common multiple d can be written $p_1^{r_1} \cdot p_2^{r_2} \cdots p_i^{r_i}$ with for any $\ell \in \{1, \dots, i\}$, $r_\ell \geq k_\ell$ (since $n \mid d$) and $r_\ell \geq j_\ell$ (since $m \mid d$). So $r_\ell \geq \max(k_\ell, j_\ell)$. Therefore $q = p_1^{\max(k_1,j_1)} \cdot p_2^{\max(k_2,j_2)} \cdots p_i^{\max(k_i,j_i)}$ is indeed a common multiple of n and m. In addition, any other common multiple is also a multiple of q since the power associated with prime p_ℓ in d is at least $\max(k_\ell, j_\ell)$. □

Corollary

$\gcd(n, m) \cdot \operatorname{lcm}(n, m) = n \cdot m.$

Proof

In $n \cdot m$ every prime factor p_ℓ has power $k_\ell + j_\ell = \min(k_\ell, j_\ell) + \max(k_\ell, j_\ell)$. □

Indeed we have $3 \cdot 60 = 12 \cdot 15$:

$$\begin{array}{rcl} 3 \cdot 60 = (2^0 \cdot 3^1 \cdot 5^0) \cdot (2^2 \cdot 3^1 \cdot 5^1) & = & 2^2 \cdot 3^2 \cdot 5^1 = 180 \\ 12 \cdot 15 = (2^2 \cdot 3^1 \cdot 5^0) \cdot (2^0 \cdot 3^1 \cdot 5^1) & = & 2^2 \cdot 3^2 \cdot 5^1 = 180. \end{array}$$

Exercise VII.6

Calculate using the decomposition into prime factors:

1. $\gcd(36, 21)$ and $\operatorname{lcm}(36, 21)$.
2. $\gcd(78, 24)$
3. $\operatorname{lcm}(45, 33)$

VII.C.3 Euclid's Algorithm

Although the above provides a procedure to calculate the GCD and LCM of two numbers, it is highly inefficient as it requires to first decompose into prime factors. There is a more efficient way to calculate the GCD, called Euclid's Algorithm. Then from the GCD, calculating the LCM can be done using the formula $\operatorname{lcm}(n, m) = \frac{n \cdot m}{\gcd(n,m)}$.

Proposition

Let $n, m \in \mathbb{N}$ and assume $n, m > 0$. Let $n = m \cdot q + r$ be the Euclidean division of n by m.

- If $r = 0$, then $\gcd(n, m) = m$.
- If $r > 0$ (i.e. $m \not| n$), then $\gcd(n, m) = \gcd(m, r)$.

Proof

- If $r = 0$ then $m \mid n$ so m is a divisor of n; as m is also a divisor of m, it is a common divisor. There cannot be a larger one as it would not divide m.
- Let d be a common divisor of n and m. Then $d \mid m \cdot q$ and $d \mid m \cdot q + r$, so $d \mid r$. Reciprocally, a divisor of m and r is a divisor of $n = m \cdot q + r$. This is true in particular for the greatest divisor. □

This property gives rise to a recursive procedure to calculate the GCD of two numbers:

Euclid's Algorithm

If $m \mid n$, then $\gcd(n, m) = m$, otherwise $\gcd(n, m) = \gcd(m, r)$ where r is the remainder in the Euclidean division of n by m.

So it is possible to calculate $\gcd(n, m)$ by writing successive divisions. Each time m replaces n and r replaces m. The last non-zero remainder is the GCD of n and m.

Note that while it is in practice easier to assume that $n \geq m$, it is not required by the algorithm. When $n < m$,as the division of n by m will have a remainder of n, and the next division is m divided by n. For example, let's calculate $\gcd(24, 136)$:

$$24 = 136 \cdot 0 + 24 \quad \rightsquigarrow \quad 136 = 24 \cdot 5 + 16 \quad \rightsquigarrow \quad 24 = 16 \cdot 1 + \mathbf{8} \quad \rightsquigarrow \quad 16 = 8 \cdot 2 + 0$$

So $\gcd(24, 136) = 8$.

Exercise VII.7

Use Euclid's Algorithm to calculate:

1. $\gcd(364, 122)$
2. $\gcd(121, 374)$
3. $\gcd(764, 224)$

Exercise VII.8

1. Use Euclid's Algorithm to calculate $\gcd(21, 13)$.
2. Fibonacci's sequence is defined recursively as $F_{n+2} = F_{n+1} + F_n$, $F_0 = F_1 = 1$.
   a. Prove by strong induction that for $n > 0$, $0 < F_n < F_{n+1}$.
   b. Prove that for any $n \in \mathbb{N}$, $\gcd(F_{n+1}, F_n) = 1$.

☡ In the proofs above, be careful about the base case.

VII.C.4 Applications: fractions

While this is not really new, it is nice to see the operations done on fractions under the new light of arithmetics.

A fraction $\frac{n}{m}$ can be simplified by dividing both numerator and denominator by a common divisor of n and m. To be as efficient as possible, said divisor should be the largest, hence $\gcd(n, m)$:

$$\frac{n}{m} = \frac{\frac{n}{\gcd(n,m)}}{\frac{m}{\gcd(n,m)}}$$

A consequence is that $\frac{n}{m}$ is irreducible if and only if n and m are co-prime.

In order to add (or subtract) fractions $\frac{n}{m}$ and $\frac{p}{q}$, they must be put over the same denominator. As this denominator must be a multiple of m and q, but still be the smallest to preserve fractions as simplified as possible, $\operatorname{lcm}(m, q)$ must be used. What is actually used to make $\frac{n}{m}$ as an equivalent fraction with denominator $\operatorname{lcm}(m, q)$ is all the factors of q that are not factors of m, namely $\frac{q}{\gcd(m,q)}$. Indeed in the denominator $m \cdot \frac{q}{\gcd(m,q)} = \frac{m \cdot q}{\gcd(m,q)} = \operatorname{lcm} m, q$. A similar idea is used for fraction $\frac{p}{q}$, yielding the following formula:

$$\frac{n}{m} + \frac{p}{q} = \frac{n \cdot \frac{q}{\gcd(m,q)} + p \cdot \frac{m}{\gcd(m,q)}}{\operatorname{lcm}(m, q)}$$

Although it is not easy to read, it is easily implemented in a computer program. For fraction additions as a human, it is actually simpler to consider prime factor decomposition. For example:

$$\frac{154}{140} = \frac{2^1 \cdot 5^0 \cdot 7^1 \cdot 11^1}{2^2 \cdot 5^1 \cdot 7^1 \cdot 11^0} = \frac{11^1}{2^1 \cdot 5^1} = \frac{11}{10}$$

$$\frac{3}{14} + \frac{5}{12} = \frac{3}{2^1 \cdot 3^0 \cdot 7^1} + \frac{5}{2^2 \cdot 3^1 \cdot 7^0} = \frac{3 \cdot 2^1 \cdot 3^1 \cdot 7^0 + 5 \cdot 2^0 \cdot 3^0 \cdot 7^1}{2^2 \cdot 3^1 \cdot 7^1} = \frac{18 + 35}{2^2 \cdot 3^1 \cdot 7^1} = \frac{53}{84}$$

VII.D Diophantine Equations

VII.D.1 Extension to $\mathbb{Z}$

So far, this chapter treated only natural numbers. But what par tof that could actually works in $\mathbb{Z}$?

Actually, a lot, since the sign information is only one extra factor.

First, divisibility can still be defined as before. The only difference is that in $\mathbb{Z}$, it is no longer an order: $-n|n$ and $n|-n$, but $n \neq -n$. As there is still reflexivity and transitivity, but no antisymmetry, so this is called a *pre-order*.

Euclidean division can still be defined for $n, m \in \mathbb{Z}$: $n = m \cdot q + r$ with $0 \leq r < |m|$. Note that the remainder is still positive, even though that may be counter intuitive. For example, for ±17 divided by ±3, we have the following four cases:

- 17 by 3: $17 = 3 \cdot 5 + 2$
- -17 by 3: $-17 = 3 \cdot (-6) + 1$
- 17 by -3: $17 = (-3) \cdot (-5) + 2$

- -17 by -3: $-17 = (-3) \cdot 6 + 1$

In computer programming, the remainder operation (usually %) does not work in the mathematical way. In most programming languages the remainder is so that $-|m| \leq n\%m < |m|$. To find the mathematical remainder, one has to use $((n\%m) + m)\%m$.

Primes numbers are only positive, so -2 is not prime, for example. That allows the FTA to still hold, with the added sign information: a negative number $n < -2$ is decomposed into $n = -p_1^{k_1} \cdot p_2^{k_2} \cdots p_i^{k_i}$.

Similarly, GCD and LCM are defined to always be positive: $\gcd(n, m) = \gcd(|n|, |m|)$ and $\operatorname{lcm}(n, m) = \operatorname{lcm}(|n|, |m|)$.

VII.D.2 Definition and Bézout's Theorem

It is named after the ancient Greek mathematician Diophantus.

A Diophantine equation is an equation where solutions are only sought in $\mathbb{Z}$. We will restrict ourselves to *linear* Diophantine equations with 2 variables: equations of the form $ax + by = c$, where $a, b, c \in \mathbb{Z}$ are parameters and x and y the unknown variables, for example: $11x + 3y = 7$.

These equations are fairly easy to solve for reals (e.g. $(0, \frac{7}{3})$ is a real solution to $11x+3y = 7$), but when restricting to integers it becomes more difficult to find solutions. Thinking of this problem graphically, a linear Diophantine equation defines a line, but we are only interested on the points that are on the grid, and finding these special points make the problem harder. In the case of $11x + 3y = 7$, there are some integral solutions, for example $(-7, 28)$ or $(14, -49)$. Bézout's Theorem provides a way of determining whether such integral solutions exist, and the proof gives a method to actually find these solutions when they exist.

Bézout Theorem

The equation $ax + by = c$ has:

- Infinitely many integer solutions if $\gcd(a, b) \mid c$.
- No integer solution if $\gcd(a, b) \not\mid c$.

The proof consists in finding a solution (x_0, y_0) to $ax + by = \gcd(a, b)$, in a separate Lemma which proof contains the actual procedure to find a solution. Then it is possible to find all the solutions from (x_0, y_0). These solutions can then be transformed to solutions of $ax + by = c$ if $\gcd(a, b) \mid c$. Then we show the reciprocal: if $ax + by = c$ has a solution then $\gcd(a, b) \mid c$.

Lemma

Diophantine Equation $ax + by = \gcd(a, b)$ has a solution.

Proof of the Lemma

The idea behind this proof, and the practical method that can be used to find solutions is as follows. We write Euclid's algorithm, and then read it backwards. The last-but-one line is $r_{i-1} = r_i \cdot q_i + \gcd(a, b)$, which can be rewritten as $\gcd(a, b) = r_{i-1} - r_i \cdot q_i$. Then write the line before as $r_i = r_{i-2} - r_{i-1} \cdot q_{i-1}$ and so on until reaching a and b. To write the formal proof, we use the *Extended Euclid's Algorithm* which actually

calculates the solution as the algorithm unfolds. Think can be thought of as incorporating the backward reading of the algorithm as we go.
First, some notation: in this proof $n \div m$ is the quotient in the Euclidean division: $n = m \cdot (n \div m) + r$, $0 \leq r < m$. Then we define three sequences as follows:

- $r_0 = a$, $r_1 = b$, for any $n \in \mathbb{N}$, $r_{n+2} = r_n - (r_n \div r_{n+1}) \cdot r_{n+1}$
- $x_0 = 1$, $x_1 = 0$, for any $n \in \mathbb{N}$, $x_{n+2} = x_n - (r_n \div r_{n+1}) \cdot x_{n+1}$
- $y_0 = 0$, $y_1 = 1$, for any $n \in \mathbb{N}$, $y_{n+2} = y_n - (r_n \div r_{n+1}) \cdot y_{n+1}$
- The sequences are defined until r_n reaches 0.

Behind these definitions is the following idea. $(r_n)_{n\in\mathbb{N}}$ is the sequence of the remainders in Euclid's algorithm. So each time r_{n+2} is the remainder in the Euclidean division of r_n by r_{n+1}. As was proved in the proof of Euclid's Algorithm, we have that for $n \in \mathbb{N}$, $\gcd(a, b) = \gcd(r_n, r_{n+1})$.
Now sequences $(x_n)_{n\in\mathbb{N}}$ and $(y_n)_{n\in\mathbb{N}}$ are a bit harder to understand as is. What matters is that at any step, we maintain the equality $r_n = a \cdot x_n + b \cdot y_n$. Taking this equality just before the remainder sequence reaches 0 ($r_{n+1} = 0$) gives us $\gcd(a, b) = a \cdot x_n + b \cdot y_n$, and therefore integral solutions to the equation.
Let's prove by induction that for $n \in \mathbb{N}$, $r_n = a \cdot x_n + b \cdot y_n$.

Base case: Here there are two base case: 0 and 1.

$\boxed{n = 0}$ $a \cdot x_0 + b \cdot y_0 = a \cdot 1 + b \cdot 0 = a = r_0$.

$\boxed{n = 1}$ $a \cdot x_1 + b \cdot y_1 = a \cdot 0 + b \cdot 1 = b = r_1$.

Induction Case: Assume for n and $n+1$ the proposition holds, i.e. $r_n = a \cdot x_n + b \cdot y_n$ and $r_{n+1} = a \cdot x_{n+1} + b \cdot y_{n+1}$. Then:

$$\begin{array}{rcl}
a \cdot x_{n+2} + b \cdot y_{n+2} & = & a \cdot (x_n - (r_n \div r_{n+1}) \cdot x_{n+1}) + b \cdot (y_n - (r_n \div r_{n+1}) \cdot y_{n+1}) \\
 & = & a \cdot x_n + b \cdot y_n - (r_n \div r_{n+1})(a \cdot x_{n+1} + b \cdot y_{n+1}) \\
 & = & r_n - (r_n \div r_{n+1}) \cdot r_{n+1} \\
a \cdot x_{n+2} + b \cdot y_{n+2} & = & r_{n+2}
\end{array}$$

So for $n \in \mathbb{N}$, $r_n = a \cdot x_n + b \cdot y_n$, and when $r_{n+1} = 0$ we have $\gcd(a, b) = a \cdot x_n + b \cdot y_n$, so (x_n, y_n) is a solution. □

The calculation of Extended Euclid's Algorithm for 34 and 13 is given in Table VII.1. While any computer implementation should use this extended algorithm, when on paper, however, it is less tedious to just rewind the usual Euclid's algorithm. To facilitate the writing, each division is rewritten so that the remainder is the difference between the dividend and the product of the quotient and divisor: $n = m \cdot q + r \Leftrightarrow r = n - m \cdot q$. For 34 and 13,

n	Division	r_n	x_n	y_n	Check
0		34	1	0	$34 \cdot 1 + 13 \cdot 0 = 34$
1		13	0	1	$34 \cdot 0 + 13 \cdot 1 = 13$
2	$34 = 13 \cdot 2 + 8$	8	1	-2	$34 \cdot 1 + 13 \cdot (-2) = 34 - 26 = 8$
3	$13 = 8 \cdot 1 + 5$	5	-1	3	$34 \cdot (-1) + 13 \cdot 3 = -34 + 39 = 5$
4	$8 = 5 \cdot 1 + 3$	3	2	-5	$34 \cdot 2 + 13 \cdot (-5) = 68 - 65 = 3$
5	$5 = 3 \cdot 1 + 2$	2	-3	8	$34 \cdot (-3) + 13 \cdot 8 = -102 + 104 = 2$
6	$3 = 2 \cdot 1 + 1$	1	5	-13	$34 \cdot 5 + 13 \cdot (-13) = 170 - 169 = 1$
7	$2 = 1 \cdot 2 + 0$	0			

Details of calculations for x_n and y_n:

$$x_2 = x_0 - 2 \cdot x_1 = 1 \qquad y_2 = y_0 - 2 \cdot y_1 = -2 \qquad x_3 = x_1 - 1 \cdot x_2 = -1 \qquad y_3 = y_1 - 1 \cdot y_2 = 3$$

$$x_4 = x_2 - 1 \cdot x_3 = 2 \qquad y_4 = y_2 - 1 \cdot y_3 = -5 \qquad x_5 = x_3 - 1 \cdot x_4 = -3 \qquad y_5 = y_3 - 1 \cdot y_4 = 8$$

$$x_6 = x_4 - 1 \cdot x_5 = 5 \qquad y_6 = y_4 - 1 \cdot y_5 = -13$$

Table VII.1: Extended Euclid's Algorithm for 34 and 13.

it would be done as follows:

$$\begin{aligned}
34 = 13 \cdot 2 + 8 \quad &\Leftrightarrow \quad 8 = 34 - 13 \cdot 2 && \text{(VII.1)}\\
13 = 8 \cdot 1 + 5 \quad &\Leftrightarrow \quad 5 = 13 - 8 \cdot 1 && \text{(VII.2)}\\
8 = 5 \cdot 1 + 3 \quad &\Leftrightarrow \quad 3 = 8 - 5 \cdot 1 && \text{(VII.3)}\\
5 = 3 \cdot 1 + 2 \quad &\Leftrightarrow \quad 2 = 5 - 3 \cdot 1 && \text{(VII.4)}\\
3 = 2 \cdot 1 + 1 \quad &\Leftrightarrow \quad 1 = 3 - 2 \cdot 1 && \text{(VII.5)}\\
2 = 1 \cdot 2 + 0 \quad &\rightsquigarrow \quad \gcd(34, 13) = 1 && \text{(VII.6)}
\end{aligned}$$

Starting from $1 = 3 - 2 \cdot 1$ in Equation VII.5, we rewrite 2 using Equation VII.4, then 3 using Equation VII.3 until reaching an expression using our original numbers 34 and 13:

$$\begin{aligned}
1 &= 3 - 2 \cdot 1 && \text{[Rewriting 1 using Equation VII.5]}\\
&= 3 - (5 - 3 \cdot 1) = 3 \cdot 2 - 5 && \text{[Rewriting 2 using Equation VII.4]}\\
&= (8 - 5 \cdot 1) \cdot 2 - 5 = 8 \cdot 2 - 5 \cdot 3 && \text{[Rewriting 3 using Equation VII.3]}\\
&= 8 \cdot 2 - (13 - 8 \cdot 1) \cdot 3 = 8 \cdot 5 - 13 \cdot 3 && \text{[Rewriting 5 using Equation VII.2]}\\
&= (34 - 13 \cdot 2) \cdot 5 - 13 \cdot 3 = 34 \cdot 5 - 13 \cdot 13 && \text{[Rewriting 8 using Equation VII.1]}\\
1 &= 34 \cdot 5 + 13 \cdot (-13) && \text{[Integral solution found]}
\end{aligned}$$

Therefore the couple $(5, -13)$ is an integral solution.

Proof of Bézout's Theorem

First, assume that $\gcd(a, b) \mid c$. Be the above Lemma, equation $ax + by = \gcd(a, b)$ has a solution, which we name (x_0, y_0). Let $k \in \mathbb{Z}$, then let $x_k = x_0 + k \cdot \frac{b}{\gcd(a,b)}$ and

$y_k = y_0 - k \cdot \frac{a}{\gcd(a,b)}$. So

$$ax_k + by_k = a \cdot \left(x_0 + k \cdot \frac{b}{\gcd(a,b)}\right) + b \cdot \left(y_0 - k \cdot \frac{a}{\gcd(a,b)}\right) =$$

$$ax_0 + by_0 + k \cdot \left(\frac{ab - ba}{\gcd(a,b)}\right) = ax_0 + by_0 = \gcd(a,b)$$

Hence (x_k, y_k) is also a solution. Note that dividing by $\gcd(a,b)$ is not necessary to obtain *a* solution, it is necessary to obtain *all* solutions.
Since $\gcd(a,b) \mid c$, we can write $c = p \cdot \gcd(a,b)$. Let (x_k, y_k) be a solution of $ax + by = \gcd(a,b)$. Then $a(x_k \cdot p) + b(y_k \cdot p) = \gcd(a,b) \cdot p = c$, so $(x_k \cdot p, y_k \cdot p)$ is a solution of $ax + by = c$.

For the reciprocal, assume that $ax+by = c$ has a solution, let's prove that $\gcd(a,b) \mid c$. Let d be a common divisor to a and b. Then $d \mid ax + by = c$. In particular $\gcd(a,b)$ is a common divisor of a and b, so $\gcd(a,b) \mid c$. □

Corollary

When a and b are co-prime *all* equations $ax + by = c$ have solutions.

Proof

Assume a and b are co-prime. Therefore $\gcd(a,b) = 1$, so $\gcd(a,b) \mid c$, hence $ax + by = c$ have an infinity of integral solutions. □

Exercise VII.9

For the following equations, determine if they have integral solutions, and if so exhibit such a solution. Optionally: give the form of *all* solutions.

1. $75x + 40y = 5$
2. $12x + 35y = 2$
3. $86x + 36y = 7$
4. $23x + 42y = 1$
5. $268x + 28y = 6$
6. $124x + 36y = -8$

VII.E Congruence

VII.E.1 The congruence relation

The *congruence* relation *modulo* n relates numbers that are separated by a multiple of n:

Definition: Congruence modulo n

Let $n \in \mathbb{N}$, the *congruence modulo* n relation is

$$\mathrm{Con}_n = \{(m,p) \in \mathbb{Z} \times \mathbb{Z} \mid \exists k \in \mathbb{Z}, m = p + n \cdot k\}.$$

When $(m,p) \in \mathrm{Con}_n$, we write $m \equiv p \pmod{n}$.

Note that if $n = 0$, this relation boils down to identity, and is not very interesting. In the sequel, every time the congruence relation modulo n is mentioned, it can be assumed that $n > 0$.

This can be reformulated in terms of remainder in the Euclidean division by n:

Proposition

Let $n \in \mathbb{N}$ with $n > 0$, and $m, p \in \mathbb{Z}$. m and p are congruent modulo n iff they have the same remainder in the Euclidean division by n.

Proof

$\boxed{\Rightarrow}$ Assume $m = p + n \cdot k$. The Euclidean divisions of m and p by n are, respectively, $m = n \cdot q + r$ and $p = n \cdot q' + r'$ (with $0 \leq r, r' < n$). Then $n \cdot q + r = n \cdot q' + r' + n \cdot k$. So $n \cdot (q - q' - k) = r' - r$ and $n \mid r' - r$. But $-n < r', r < n$ so $r' - r = 0$, therefore $r = r'$.

$\boxed{\Leftarrow}$ Assume $m = n \cdot q + r$ and $p = n \cdot q' + r$ (with $0 \leq r < n$). Then setting $k = q - q'$, we have $m = p + n \cdot k$. □

Here are several examples of equivalences by congruence relations:

- $23 \equiv 58 \pmod{5}$ because $58 = 5 \cdot 11 + 3$ and $23 = 5 \cdot 4 + 3$.
- $23 \equiv 3 \pmod{5}$ because $23 = 5 \cdot 4 + 3$ and $3 = 5 \cdot 0 + 3$.
- $-17 \equiv 7 \pmod{8}$ because $-17 = 8 \cdot (-3) + 7$ and $7 = 7 \cdot 0 + 7$.
- $21 \equiv 51 \pmod{3}$ because $21 = 3 \cdot 7 + 0$ and $51 = 3 \cdot 17 + 0$.
- $21 \equiv 0 \pmod{5}$ because $21 = 3 \cdot 7 + 0$ and $0 = 3 \cdot 0 + 0$.
- $43 \equiv 16 \pmod{3}$ because $43 = 3 \cdot 14 + 1$ and $16 = 3 \cdot 5 + 1$.
- $43 \equiv 1 \pmod{5}$ because $43 = 3 \cdot 14 + 1$ and $1 = 3 \cdot 0 + 1$.
- $47 \equiv 35 \pmod{3}$ because $47 = 3 \cdot 15 + 2$ and $35 = 3 \cdot 11 + 2$.
- $47 \equiv 2 \pmod{5}$ because $47 = 3 \cdot 15 + 2$ and $2 = 3 \cdot 0 + 2$.

Theorem

Let $n \in \mathbb{N}$. Congruence modulo n is an equivalence relation.

Remark that this was proved in the case $n = 5$ in Exercise V.2. The general case is not much different, and is reproduced here for completeness' sake.

Proof

If $n = 0$, then $\text{Con}_n = Id$ which is an equivalence relation. Otherwise, we can use the above proposition to prove reflexivity, symmetry, and transitivity.

$\boxed{\textbf{R}}$ Let $m \in \mathbb{Z}$. m has the same remainder as m in the division by n.

$\boxed{\textbf{S}}$ Let $m, p \in \mathbb{Z}$. Assume $m \equiv p \pmod{n}$. Then m has the same remainder as p in the division by n, which can be rephrased as p has the same remainder as n in the division by n, so $p \equiv m \pmod{n}$.

$\boxed{\textbf{T}}$ Let $m, p, q \in \mathbb{Z}$. Assume $m \equiv p \pmod{n}$ and $p \equiv q \pmod{n}$. Then, in the division by n, m has the same remainder as p, and p has the same remainder as q, so m has the same remainder as q and $m \equiv q \pmod{n}$. □

VII.E.2 Equivalence classes and canonical representative

Let $n \in \mathbb{N}$, $n > 0$. The congruence relation modulo n being an equivalence relation, we can consider its equivalence classes. As congruence relates numbers that share the same remainder in the Euclidean division by n, there are n equivalence classes, which can be canonically represented by this remainder: $[0], [1], \ldots, [n-1]$. For any $p \in \{0, \ldots, n-1\}$, $[p]$ contains integers of the form $n \cdot k + p$. This is why we usually try to write $m \equiv r \pmod{n}$ with $0 \leq r < n$ being the remainder of the division by n. In that case we also sometimes write $m \bmod n = r$; this is a useful shorthand for "r is the remainder of m in the Euclidean division by n".

Consider the equivalence classes reached when counting from 0 to infinity. We start at $[0]$, then $[1]$, then all the way to $[n-1]$. then the equivalence class of n is $[0]$, the class of $n+1$ is $[1]$, and so on: we have a cycle of classes appearing, which we can represent in a ring, as is done in Figure VII.4 for $n = 13$. Note that going backwards into negative numbers also yields a cycle: -1 is in the class $[n-1]$, -2 in $[n-2]$, etc until $-(n-1)$ is in the class $[1]$ (as $-(n-1) = -n+1$), $-n$ is in the class $[0]$, then $-(n+1)$ is in the class $[-1]$ again, as we cycle back to the first class.

Another way to understand this representation is to imagine that the line of integers (infinite on both sides) is coiled into a spring so that each ring contains n numbers . That way all numbers that are separated by n are on top of one another. Looking from the perspective of the equivalence relation means looking at the coil from the side, and only seeing a ring of n numbers.

The set of classes, which is the quotient of $\mathbb{Z}$ by equivalence classes modulo n, is named $\mathbb{Z}/n\mathbb{Z}$.

Exercise VII.10

Calculate:

1. 37 mod 8 (i.e. find x such that $37 \equiv x \pmod{8}$ and $0 \leq x < 8$).
2. -437 mod 7 (i.e. find x such that $-437 \equiv x \pmod{7}$ and $0 \leq x < 7$).

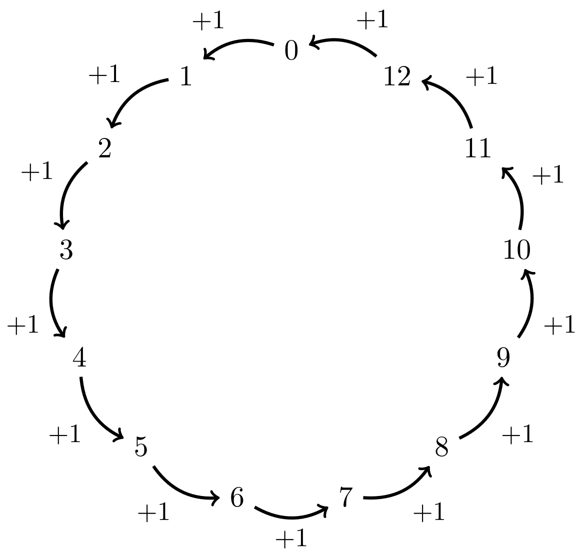


Figure VII.4: The ring of integers modulo 13.

VII.E.3 Modulo Arithmetic

What makes $\mathbb{Z}/n\mathbb{Z}$ is that we can define addition and multiplication of elements of classes directly. These calculations are called *modulo arithmetic.*

VII.E.3.i Addition of equivalence classes

The addition of two classes $[m]$ and $[p]$ is defined as $[m] + [p] = [m + p]$. For example with $n = 13$: $[2] + [5] = [7]$, $[9] + [11] = [20] = [7]$. One can check on Figure VII.4 that starting from [2] and incrementing 5 times yields 7. And starting from [9] and incrementing 11 times yields 7 as well, although crossing from 12 to 0.

This addition has good properties, which are actually similar to addition in $\mathbb{Z}$ (the usual addition), and actually are deducted directly from them. First, it is commutative: $[m] + [p] = [m + p] = [p + m] = [p] + [m]$. It is also associative: $([m] + [p]) + [q] = [(m + p) + q] = [m + (p + q)] = [m] + ([p] + [q]) = [m] + [p] + [q]$.

There is *neutral element* (a “zero”): [0] is such that for any $[m] \in \mathbb{Z}/n\mathbb{Z}$, $[0] + [m] = [m]$ Every class has an *additive inverse* (an “opposite”): for any $[m] \in \mathbb{Z}/n\mathbb{Z}$, the opposite of $[m]$ is $[n - m]$: $[m] + [n - m] = [n] = [0]$. For example with $n = 13$: the opposite of [2] is $[13 - 2] = [11]$ because $[2] + [11] = [13] = [0]$.

All these properties make $\mathbb{Z}/n\mathbb{Z}$ a commutative *group* with respect to this addition operation.

VII.E.3.ii Multiplication with equivalence classes

The multiplication of two classes $[m]$ and $[p]$ is defined as $[m] \times [p] = [m \cdot p]$.

In this case it is less obvious why the product of the remainders is the remainder of the product (in that particular ring of course). Let’s write the Euclidean divisions by n: $m = n \cdot q + r$ and $p = n \cdot q' + r'$.

$$m \cdot p = n^2 \cdot q \cdot q' + n \cdot q \cdot r' + n \cdot q' \cdot r + r \cdot r' = n \cdot (n \cdot q \cdot q' + q \cdot r' + q' \cdot r) + r \cdot r'$$

So modulo n, the term $n \cdot (n \cdot q \cdot q' + q \cdot r' + q' \cdot r)$ is equivalent to 0 (as a multiple of n), so $[m \cdot p] = [r \cdot r']$. Note that $[r \cdot r']$ mays not be a canonical representative, and would need to be divided by n: $r \cdot r' = n \cdot k + r''$ with $0 \leq r'' < n$.

For example with $n = 13$: $[2] \times [5] = [10]$, $[4] \times [7] = [28] = [2]$.

This multiplication also has somme good properties. It is commutative and associative (as is the underlying multiplication in $\mathbb{Z}$). There is a *neutral element* ("one"): [1] is such that for any $[m] \in \mathbb{Z}/n\mathbb{Z}$, $[1] \times [m] = [m]$. Furthermore, multiplication distributes over addition: $[m] \times ([p] + [q]) = [m] \times [p] + [m] \times [q]$. Again, this is proved by using distributivity in $\mathbb{Z}$.

All these properties make $\mathbb{Z}/n\mathbb{Z}$ a commutative *ring*. Note that this term is a mathematical term, and not all rings have the ring-shape that $\mathbb{Z}/n\mathbb{Z}$ has.

Modular inverse

In the case of addition, all classes have an opposite. We can try to find such a *multiplicative inverse* ("reciprocal'). For $[m] \in \mathbb{Z}/n\mathbb{Z}$, the reciprocal of $[m]$ would be a class $[p] \in \mathbb{Z}/n\mathbb{Z}$ such that $[m] \times [p] = [1]$. That would mean $[m \cdot p] = 1$. So finding $[p]$ is the same as finding p and k such that $m \cdot p = k \cdot n + 1$. This is the same as solving the Diophantine Equation $m \cdot x + n \cdot y = 1$. By Bézout's Theorem, there is a solution iff $\gcd(m, n) = 1$, i.e. n and m co-primes. So *some* classes have an inverse: the ones where $\gcd(m, n) = 1$.

For example with $n = 6$: 5 and 6 being co-primes, [5] has a reciprocal: it can be found by solving $5x + 6y = 1$. This can be done by Euclid's algorithm or by guessing a solution; here $(5, -4)$ is a solution as $5 \cdot 5 - 6 \cdot 4 = 25 - 24 = 1$. Therefore $[5] \times [5] = [25] = [1]$. On the other hand, 2 is not co-prime with 6, so 2 does not have a reciprocal.

If n is prime, then for any $0 < m < n$, $\gcd(m, n) = 1$, so in that case all classes except [0] have an inverse. For example with $n = 13$, which is prime, all classes (but [0]) have a reciprocal that can be calculated by solving a Diophantine equation. $[3] \times [9] = [27] = [1]$, $[7] \times [2] = [14] = [1]$. In that case, $\mathbb{Z}/n\mathbb{Z}$ is a commutative *field*.

VII.E.3.iii Modular exponentiation

As is the case in $\mathbb{Z}$, exponentiation in $\mathbb{Z}/n\mathbb{Z}$ is defined as the iterated multiplication:

$$[m]^p = \underbrace{[m] \times [m] \times \cdots \times [m]}_{p \text{ times}} = [\underbrace{m \times m \times \cdots \times m}_{p \text{ times}}] = [m^p]$$

In practice, whether by hand or for computers, it is easier to calculate $[m^p]$ by successively calculating $[m]$, $[m^2] = [m] \times [m]$, $[m^3] = [m^2] \times [m] \ldots$ as each time calculations involve numbers between 0 and n^2.

For example, to calculate $5^{10} \bmod 3$, the inefficient way would be to actually compute $5^{10} = 9765625$ then divide this big number by 3. A computer could handle that (because it is still relatively small), but by hand, it is much better to compute within the ring $\mathbb{Z}/3\mathbb{Z}$ to keep numbers low:

$$\begin{array}{rcl}
5^{10} \bmod 3 & = & (5 \bmod 3)^{10} \bmod 3 \\
& = & 2^{10} \bmod 3 = ((2 \bmod 3) \times (2^9 \bmod 3) \bmod 3 \\
& = & (((2 \times 2) \bmod 3) \times (2^8 \bmod 3)) \bmod 3 = 1 \times (2^8 \bmod 3) \\
& = & (((2 \times 2) \bmod 3) \times (2^6 \bmod 3)) \bmod 3 = 1 \times (2^6 \bmod 3) \\
& = & (((2 \times 2) \bmod 3) \times (2^4 \bmod 3)) \bmod 3 = 1 \times (2^4 \bmod 3) \\
& = & (((2 \times 2) \bmod 3) \times (2^2 \bmod 3)) \bmod 3 = 1 \times (2^2 \bmod 3) \\
5^{10} \bmod 3 & = & (2 \times 2) \bmod 3 = 1
\end{array}$$

This is for your personal culture more than to be applied in this course.

Note that here we used the most basic exponentiation algorithm that simply performs p multiplications to compute $[m]^p$. More efficient algorithms such as the fast exponentiation which uses $O(\log(p))$ multiplications can also be used in $\mathbb{Z}/n\mathbb{Z}$. For 5^{10} mod 3 that would go as follows:

$$\begin{array}{rcl}
5^{10} \bmod 3 & = & \Big(\big((5 \bmod 3)^2 \bmod 3\big)^2 \bmod 3 \times (5 \bmod 3)^2 \bmod 3\Big) \bmod 3 \\
 & = & \Big(\big(2^2 \bmod 3\big)^2 \bmod 3 \times 2^2 \bmod 3\Big) \bmod 3 \\
 & = & \big(1^2 \bmod 3 \times 1\big) \bmod 3 \\
 & = & (1 \times 1) \bmod 3 \\
 & = & 1 \bmod 3 \\
5^{10} \bmod 3 & = & = 1
\end{array}$$

Exercise VII.11

In $\mathbb{Z}/5\mathbb{Z}$ (i.e. the equivalence classes modulo 5).

1. Calculate $[3] + [4]$.
2. Calculate $[2] + [3]$.
3. Find the additive inverse of $[1]$.
4. Calculate $[4] \times [4]$.
5. Calculate the multiplicative inverse of $[3]$.
6. Calculate $[2]^7$.

VII.E.4 Applications

Calculation using congruences and modulo arithmetic have applications into everyday life. The most commonly used one is the RSA protocol that is at the heart of credit card security, so it is fair to say that it is used billions of times every day.

VII.E.4.i Toolbox (Lemmas)

We first state a couple of handy lemmas that will be used throughout.

Lemma: "Modulo-a-product lemma"

If n and m are co-prime numbers and $p \equiv q \pmod{n}$ and $p \equiv q \pmod{m}$, then $p \equiv q \pmod{n \cdot m}$.

Proof

By definition of the congruence relations modulo n and m, respectively: $p = k \cdot n + q$ and $p = h \cdot m + q$, for some $k, h \in \mathbb{Z}$. So $k \cdot n = h \cdot m$. Since $n \mid h \cdot m$ but shares no factor with m (as they are co-primes), we deduct that $n \mid h$. We can therefore write $h = n \cdot j$ for some $j \in \mathbb{Z}$. So $p = j \cdot (n \cdot m) + q$ and $p \equiv q \pmod{n \cdot m}$. □

Fermat's Little Theorem

If n is prime then for $0 < p < n$, $p^{n-1} \equiv 1 \pmod{n}$.

The proof of Fermat's Little Theorem requires a couple of tools of the next chapter, and therefore is proved in Section VIII.B.2.ii.

VII.E.4.ii Divisibility criteria

Being able to test divisibility at a glance helps a lot to factor a number. Some criteria, such as the ones for 2 and 5 are quite easy to prove without a real use of modulo arithmetic: since 10 is a multiple of 2 (resp. of 5), only the last digit matter.

For 3 and 9, the criteria can be proved using modulo arithmetic:

Proposition: Divisibility by 3

A number is a multiple of 3 iff the sum of the digits is a multiple of 3.

Proposition: Divisibility by 9

A number is a multiple of 9 iff the sum of the digits is a multiple of 9.

Proof (of both criteria)

A number in base 10 can be written as a sum of powers of 10 multiplied by the digit. Formally, let $x \in \mathbb{Z}$: $x = \sum_{i=0}^{n} d_i 10^i = d_0 \cdot 10^0 + d_1 \cdot 10^1 + d_2 \cdot 10^2 + \cdots + d_n \cdot 10^n$ (here d_0 is the least significant digit, the rightmost one).
Since $10 \equiv 1 \pmod 3$, so each $10^i \equiv 1 \bmod 3$. So

$$x \equiv \sum_{i=0}^{n} d_i \equiv d_0 + d_1 + d_2 + \cdots + d_n \pmod 3.$$

As a result x is a multiple of 3 iff $x \equiv 0 \pmod 3$ iff $\sum_{i=0}^{n} d_i \equiv 0 \pmod 3$. □
Similarly, $10 \equiv 1 \pmod 9$, so each $10^i \equiv 1 \bmod 9$. So

$$x \equiv \sum_{i=0}^{n} d_i \equiv d_0 + d_1 + d_2 + \cdots + d_n \pmod 9.$$

As a result x is a multiple of 9 iff $x \equiv 0 \pmod 9$ iff $\sum_{i=0}^{n} d_i \equiv 0 \pmod 9$. □

For example: is 354541 divisible by 3?

$354541 = 300000+50000+4000+500+40+1 = 3\cdot10^5+5\cdot10^4+4\cdot10^3+5\cdot10^2+4\cdot10^1+1\cdot10^0$.

In $\mathbb{Z}/3\mathbb{Z}$, $354541 \equiv 3+5+4+5+4+1 \equiv 22 \equiv 2+2 \equiv 4 \equiv 1 \pmod 3$. So 354541 is not divisible by 3; in fact we know the remainder in the division of 354541 by 3 is 1.

VII.E.4.iii Chinese Remainder Theorem

This Theorem is called "Chinese" because it is attributed to Chinese mathematician Sun-tzu.

Chinese Remainder Theorem

Let $k \in \mathbb{N}$. Let $n_1, n_2, \dots, n_k \in \mathbb{N}$ be pairwise co-primes: $\forall i, j, i \neq j \rightarrow \gcd(n_i, n_j) = 1$. Then for any numbers $a_1, a_2, \dots a_k \in \mathbb{N}$, there exists q such that for any i, $q \equiv a_i \pmod{n_i}$.
In addition, all solutions are equivalent modulo $n = n_1 \cdot n_2 \cdots n_k$.

As is often the case for of existential statements, the proof is constructing an element to exhibit. In that sense, the proof itself is more interesting than the result because it provides a procedure to find the solution.

Proof

Let $n_1, n_2, \dots, n_k \in \mathbb{N}$ be pairwise co-primes and $a_1, a_2, \dots a_n \in \mathbb{N}$. For $i \in \{1, \dots, k\}$, let $m_i = \frac{n}{n_i} = n_1 \cdots n_{i-1} \cdot n_{i+1} \cdots n_k$, the product of all the $(n_j)_{j \in \{1,\dots,k\}}$ except n_i. As n_i does not share any prime factor with any of the other $(n_j)_{j \in \{1,\dots,k\}}$, it does not share a factor with m_i either: m_i and n_i are co-primes. By Bézout's Theorem, there is a solution (x_i, y_i) to $n_i \cdot x_i + m_i \cdot y_i = 1$. Note that we can rephrase the above by considering that y_i is the inverse of m_i in $\mathbb{Z}/n_i\mathbb{Z}$. So we have $m_i \cdot y_i \equiv 1 \pmod{n_i}$. As m_i is the product of the $(n_j)_{j \in \{1,\dots,k\}, j \neq i}$, for $j \neq i$, $n_j \mid m_i$, therefore $m_i \cdot y_i \equiv 0 \pmod{n_j}$.
Define $q = \sum_{j=1}^{k} a_j \cdot m_j \cdot y_j$. For $i \in \{1, \dots, k\}$, consider $q \bmod n_i$, i.e. consider q in $\mathbb{Z}/n_i\mathbb{Z}$:

$$\begin{aligned}
q &\equiv \sum_{j=1}^{k} a_j \cdot m_j \cdot y_j \pmod{n_i} \\
&\equiv a_i \cdot m_i \cdot y_i + \sum_{j=1}^{i-1} a_j \cdot m_j \cdot y_j + \sum_{j=i+1}^{k} a_j \cdot m_j \cdot y_j \pmod{n_i} \\
&\equiv a_i \cdot 1 + \sum_{j=1}^{i-1} a_j \cdot 0 + \sum_{j=i+1}^{k} a_j \cdot 0 \pmod{n_i} \\
q &\equiv a_i \pmod{n_i}
\end{aligned}$$

So q is a solution to all the modulo equations.

Let $n = n_1 \cdot n_2 \cdots n_k$ Assume both q and q' are solutions. Then for $i \in \{1, \dots, k\}$, $q \equiv a_i \pmod{n_i}$ and $q' \equiv a_i \pmod{n_i}$, so by transitivity $q' \equiv q \pmod{n_i}$. By the "Modulo-a-product lemma", $q' \equiv q \pmod{n}$. Reciprocally, q is a solution and $q' \equiv q \pmod{n}$ then $q' = q + h \cdot n = q' + h \cdot n_i \cdot m_i$ so for any i, $q' \equiv q \pmod{n_i}$. As for any $i \in \{1, \dots, k\}$, $q \equiv a_i \pmod{n_i}$, by transitivity we have that $q' \equiv a_i \pmod{n_i}$ so q' is also a solution. □

For example, let's try to find the smallest positive integer q such that $q \equiv 4 \pmod 5$, $q \equiv 3 \pmod 6$, $q \equiv 5 \pmod 7$. First we see that since 5, 6 and 7 are co-primes (5 and 7 are primes and $6 = 2 \cdot 3$), so we can use the procedure laid out in the proof above.

We first need to calculate the multiplicative inverse of the product of two of our modulos modulo the third one; namely:

- 6×7 in $\mathbb{Z}/5\mathbb{Z}$
- 5×7 in $\mathbb{Z}/6\mathbb{Z}$
- 5×6 in $\mathbb{Z}/7\mathbb{Z}$

Inverse of 42 in $\mathbb{Z}/5\mathbb{Z}$: We can start by seeing that $42 \equiv 2 \pmod 5$, so we can look for the inverse of 2, instead of 42. That makes the Diophantine equation simpler, and usually one can guess a solution rather than compute Euclid's algorithm and revert it. In this case, the equation is $5x + 2y = 1$. And since $5 \cdot 1 = 5$ and $2 \cdot 2 = 4$ and their difference is 1, we can guess a solution: $(1, -2)$. Although -2 is not an element of $\mathbb{Z}/5\mathbb{Z}$, its representative is 3. And we can check that $2 \cdot 3 = 6 \equiv 1 \pmod 5$.

Inverse of 35 in $\mathbb{Z}/6\mathbb{Z}$: Again $35 \equiv 5 \pmod 6$, so we solve $6x + 5y = 1$, for which $(1, -1)$ is a solution. So we use 5, the representative of -1 in $\mathbb{Z}/6\mathbb{Z}$: $5 \cdot 5 = 25 \equiv 1 \pmod 6$.

Inverse of 30 in $\mathbb{Z}/7\mathbb{Z}$: Here we need to inverse 2 in $\mathbb{Z}/7\mathbb{Z}$ as $30 \equiv 2 \pmod 7$. Solving $7x + 2y = 1$ yields $(1, -3)$, so the inverse is 4: $2 \cdot 4 = 8 \equiv 1 \pmod 7$.

With these inverse, we define $q_0 = 4 \cdot 42 \cdot 3 + 3 \cdot 35 \cdot 5 + 5 \cdot 30 \cdot 4 = 1629$. By construction q_0 satisfies the conditions, but it can nonetheless be checked:

$$1629 = 5 \cdot 325 + 4 \qquad 1629 = 6 \cdot 271 + 3 \qquad 1629 = 7 \cdot 232 + 5$$

Or using modulo arithmetic:

$$\begin{aligned}
q_0 &\equiv 4 \cdot 42 \cdot 3 + 3 \cdot 35 \cdot 5 + 5 \cdot 30 \cdot 4 \pmod 5\\
&\equiv 4 \cdot 126 + (3 \cdot 7 \cdot 5) \cdot 5 + (5 \cdot 6 \cdot 4) \cdot 5 \pmod 5\\
&\equiv 4 \cdot 1 + (3 \cdot 7 \cdot 5) \cdot 0 + (5 \cdot 6 \cdot 4) \cdot 0 \pmod 5\\
q_0 &\equiv 4 \pmod 5
\end{aligned}$$

$$\begin{aligned}
q_0 &\equiv 4 \cdot 42 \cdot 3 + 3 \cdot 35 \cdot 5 + 5 \cdot 30 \cdot 4 \pmod 6\\
&\equiv (4 \cdot 7 \cdot 3) \cdot 6 + 3 \cdot 175 + (5 \cdot 5 \cdot 4) \cdot 6 \pmod 6\\
&\equiv (4 \cdot 7 \cdot 3) \cdot 0 + 3 \cdot 1 + (5 \cdot 5 \cdot 4) \cdot 0 \pmod 6\\
q_0 &\equiv 3 \pmod 6
\end{aligned}$$

$$\begin{aligned}
q_0 &\equiv 4 \cdot 42 \cdot 3 + 3 \cdot 35 \cdot 5 + 5 \cdot 30 \cdot 4 \pmod 7\\
&\equiv (4 \cdot 6 \cdot 3) \cdot 7 + (3 \cdot 5 \cdot 5) \cdot 7 + 5 \cdot 120 \pmod 7\\
&\equiv (4 \cdot 6 \cdot 3) \cdot 0 + (3 \cdot 5 \cdot 5) \cdot 0 + 5 \cdot 1 \pmod 7\\
q_0 &\equiv 5 \pmod 7
\end{aligned}$$

Note however that q_0 may not be the smallest such positive integer. Since all solutions are equivalent modulo $5 \cdot 6 \cdot 7 = 210$, we can actually compute the solution we are looking for by taking the remainder of q_0 in the division by 210: $q_0 = 210 \cdot 7 + 159$. So the number we are looking for is 159. Once again we know by construction that it is a solution, but we can check for good measure:

$$159 = 5 \cdot 31 + 4 \qquad 159 = 6 \cdot 26 + 3 \qquad 159 = 7 \cdot 22 + 5$$

Exercise VII.12

John has some flowers. He tries tried to group them by bunches of 9 but he had 5 flowers left. He then tries to group them by bunches of 10 and was left with 7 flowers. He tried to group them by bunches of 11 and was left with 3 flowers.

1. How many flowers does John have (at the minimum)?
2. Can you suggest a number of flowers per bunch that would divide evenly?

VII.E.4.iv RSA cryptography

This is for your personal culture more than to be applied in this course.

The RSA cryptography protocol, named after its three inventors Ron Rivest, Adi Shamir, and Len Adleman is the protocol behind secure communication. It is found in credit card payment protocols, online authentication (`https` protocol for example), and many more.

The principle is as follows. First, construct three integers e, d, and n such that for every integer m we have $(m^e)^d \equiv m \pmod{n}$. Then publish e and n (which are collectively known as the *public key*), while d is kept secret (this is the *private key*). Because there is a public and a private key, RSA is called an *asymmetric* protocol.

Given a message that needs to be transmitted, the first step is to translate the message into an integer $m \in \mathbb{N}$ such that $0 \leq m < n$. This translation algorithm is public, so from m the original message can be retrieved. In addition, the message can be cut into packets, each converted to an integer and glued back when translated back. In that case each integer is transmitted using the RSA protocol. For simplicity, we will focus on what happens when the integer m is given; it will be called the "message", even though it is only an integer.

To *encrypt* a message, calculate $m^e \bmod n$ using the recipient's public key. The rightful recipient can then *decrypt* by calculating $(m^e)^d \pmod{n}$, which is m and can be translated back into the message. Note that only using the right value d gives the original message, so only the intended recipient can retrieve the value m.

This protocol can also be used to *sign* a document. The key holder can provide a message m along with $m^d \bmod n$. Anyone with the public key is able to calculate $(m^d)^e \pmod{n}$ and find m. In this case, only the key holder would have been able to calculate $m^d \bmod n$, thus ensuring that he is the sender of the message.

VII.E.4.iv.a Constructing the keys

What remains to be shown is how to construct these values e, d, and n such that for every integer m we have $(m^e)^d \equiv m \pmod{n}$. In addition, d should be hard to find, otherwise the security of the protocol is weak.

To build a new RSA pair of public and private key, we start by choosing two (big) prime numbers p and q, with $p \neq q$. These integers, although not used in the protocol itself, must be kept secret since it would otherwise be easy to reconstruct the secret key. Then we set $n = p \cdot q$. Note that although n is part of the public key, it is hard to find p and q by factoring n, as there is no efficient algorithm to do so. In essence, to find p and q from n one would have to try all values smaller than $\sqrt{n}$. When p and q are big numbers, say 1024-bit integers, that means there is about $2^{1024} \simeq 10^{308}$ values to try as a factor of n.

Then calculate $k = \operatorname{lcm}(p-1, q-1)$ (kept secret as well). This can be done by performing Euclid's algorithm on $p-1$ and $q-1$ to find $\gcd(p-1, q-1)$; then $\operatorname{lcm}(p-1, q-1) = \frac{(p-1)(q-1)}{\gcd(p-1,q-1)}$. Choose $e < k$ and co-prime with k; usually the value chosen at this step

$e = 2^{16}+1$, which is a prime number. Finally, d can be calculated as the inverse of e in $\mathbb{Z}/k\mathbb{Z}$ by solving the Diophantine Equation $e \cdot d + k \cdot y = 1$.

So building the keys is a somewhat convoluted procedure, but it takes relatively little time on a computer. As we have seen, finding p and q from e and n is a hard problem since it involve factoring. The day an efficient (i.e. polynomial) algorithm is found for factoring is the day RSA stops being secure at all and the whole modern cryptographic world would collapse. It is, however, unlikely to happen any time soon.

VII.E.4.iv.b Proving correctness

What remains to be proved is that the keys e, d, and n that we built actually have the essential property required to encrypt/decrypt or sign messages using the RSA protocol.

Theorem: Correctness of RSA

Using e, d, and n as constructed above, for every integer m we have $(m^e)^d \equiv m \pmod{n}$.

Proof

Consider integers p, q, n, k, e, d build as described above. By construction we have $e \cdot d - 1 = k \cdot y$ for some y. Since $k = \operatorname{lcm}(p-1, q-1)$, we have $k = i_0 \cdot (p-1)$ and $k = j_0 \cdot (q-1)$ for some integers $i_0, j_0 \in \mathbb{N}$. By setting $i = i_0 \cdot y$ and $j = j_0 \cdot y$, we have $e \cdot d - 1 = (p-1) \cdot i = (q-1) \cdot j$, so $e \cdot d = 1 + (p-1) \cdot i = 1 + (q-1) \cdot j$.

Let $m \in \mathbb{N}$. Let's prove that $m^{e \cdot d} \equiv m \pmod{p}$. Using $e \cdot d = 1 + (p-1) \cdot i$, we can write $m^{e \cdot d} \equiv m^{1+(p-1) \cdot i} \equiv m \cdot m^{(p-1) \cdot i} \equiv m \cdot \left(m^{p-1}\right)^i \pmod{p}$. By Fermat's Little Theorem, $m^{p-1} \equiv 1 \pmod{p}$, so $m \cdot \left(m^{p-1}\right)^i \equiv m \cdot 1^i \equiv m \pmod{p}$. Hence $m^{e \cdot d} \equiv m \pmod{p}$.
We can do a similar proof to show that $m^{e \cdot d} \equiv m \pmod{q}$, using $e \cdot d = 1 + (q-1) \cdot j$. As p and q are distinct primes, they are co-primes, so by the "Modulo-a-product lemma" we have $m^{e \cdot d} \equiv m \pmod{p \cdot q}$, so $m^{e \cdot d} \equiv m \pmod{n}$. □

Chapter VIII
Introduction to Combinatorics

Chapter contents

VIII.A What is Combinatorics? **140**

VIII.A.1 Different approaches to combinatorics 140

VIII.A.2 Some basic tools 141

VIII.B Choosing k among n **142**

VIII.B.1 Definition, calculation, and closed form 142

VIII.B.1.i A recurrence relation 142

VIII.B.1.ii A closed form 143

VIII.B.2 Binomial coefficients 145

VIII.B.2.i Binomial expansion 145

VIII.B.2.ii Number Theory with binomial coefficients 148

VIII.A What is Combinatorics?

The term *combinatorics* can be though as a fancy word for *counting*. The main thing that is being counted it the number of elements in a set S from its definition or description. Usually the set being consider is built from some sets of know size, given by a parameter: we are looking for a function of these parameters. The set S is a combination of the underlying sets, hence the name *combinatorics*.

In computer science, combinatorics are used to evaluate the *complexity* of some algorithms, for example when all the elements of the set have to be considered: evaluating the size of the set allows to evaluate the overall complexity. In statistics and probability, when randomly selecting an element in a set S with uniform probability, each element has probability $\frac{1}{|S|}$ of being chosen; so knowing $|S|$ allows to compute these probabilities.

This chapter is a brief introduction to combinatorics. Some of its content is more a revisit on some notions described in earlier chapters. The idea is more to provide methods than formulas, as each combinatorics problem is different from the other. Some of the questions we will consider are:

- How many subsets are there in a set of size n?
- How many subsets of size k are there in a set of size n?
- How many propositional formulas (up to logical equivalence) exist when there are n propositional variables?

VIII.A.1 Different approaches to combinatorics

When faced with a combinatorics problem, there are three main ways to approach the problem.

Reformulation: Reformulate the description in order to have the number of elements appear. This is the most direct approach, which works well for relatively simple cases.

For example, to calculate the number of subsets of a set of size n, one can think of a subset as follows: for every element, it can be in the set or not, so there are 2^n sets. This was the approach taken in Section IV.B.2 when discussing the cardinality of the powerset.

Induction: Find a recurrence relation that relates the size of the set based on the underlying sets. This method works best when there is a single underlying set S_0 on which set S is built so $|S|$ is a single variable function of $|S_0|$: $|S| = f(|S_0|)$. The base case corresponds to $|S_0| = 0$, i.e. when the underlying set is empty. Then inductive case expresses the number of combinations when $|S_0| = n + 1$ using the number of combinations for $|S_0| = n$: express $f(n + 1)$ as an expression using $f(n)$. A closed form can be conjectured and proved by induction.

Using the same example as above: let $f(n)$ be the number of subsets of a set of size n.

Base case: $n = 0$: There is only one subset, the empty set, so $f(0) = 1$.

Inductive case: For a set S of size $n + 1$, take one element $e \in S$. Consider the subsets of $S \setminus \{e\}$. There are $f(n)$ subsets. Then a subset of S is either a subset of $S \setminus \{e\}$ or a subset of $S \setminus \{e\}$ to which e was added, therefore $f(n+1) = 2f(n)$.

In Section VI.C.2, several recurrence relations are transformed into a closed form using this method.

Proving that this function yields the closed form $f(n) = 2^n$ is left as exercise to the reader.

Indirect counting: This method is best described by this explanation: "To count the number of sheep, count the number of sheep legs and divide by 4". In practice it means that it is sometimes easier to count objects more than once, and then divide to account for this multiple counting.

An example of this situation is counting how many handshakes occurred in a meeting of n people. Everyone shook the hand of everyone else, so $n-1$ hands. In total that makes $n(n-1)$ handshakes But that means we counted the handshake between two people A and B twice: once as A shaking B's hand, and once as B shaking A's hand. Therefore the actual number of handshakes is $\frac{n(n-1)}{2}$.

VIII.A.2 Some basic tools

Sets can be built in a variety of ways from others. But some basic operations on sets are at the heart of most constructions. Therefore knowing how these operations affect cardinalities allows to calculate the size of the resulting set. Note that some of these tools already appear in Chapter IV, but are reproduced here for completeness' sake.

Proposition

- The number of tuples in the Cartesian product of p sets of sizes $n_1, n_2, \ldots, n_p$ is $\prod_{i=1}^{p} n_i = n_1 \cdot n_2 \cdots n_p$.
- The number of subsets a set of size n is 2^n.
- The number of ways to *arrange* (i.e. pick an order) of a set of size n is

$$n! = \prod_{i=1}^{n} i = 1 \cdot 2 \cdots n.$$

It is also known as the number of *permutations* of the set.

Proof

Only the last point remains to be proved. Two proofs will actually be done, each illustrating a different approach.

Reformulation: If we select elements from first to last, there are n ways to pick the first one, then $n-1$ way to take the second one, all the way down to only 1 way to pick the last element remaining. Therefore the number of ways to pick an order of a set of size n is $n \cdots 2 \cdot 2 = \prod_{i=1}^{n} i = n!$.

Induction: Let's find a recurrence relation for the number of permutations. Let $\pi(n)$ be the number of permutations in a set of size n. We'll prove that it corresponds to $n!$ at the same time, i.e. that $\pi(n) = n!$.

$\boxed{\textbf{Base case:}}$ If $n = 1$, there is only one way to arrange a singleton, so there is $\pi(1) = 1$ arrangement. As $1! = 1$, the base case matches what is to be proved. Technically $n = 0$ is the actual base case. There is also one way to arrange the empty set, so $\pi(0) = 1$, and that works with the conjecture as well because $0! = 1$.

Inductive case: To add the n+1th element to one of the order for n elements, it could be put before any of the n elements or at the last position, so there are $n + 1$ choices for that. Therefore $\pi(n + 1) = (n + 1) \cdot \pi(n)$. Assume $\pi(n) = n!$. Then $\pi(n + 1) = (n + 1) \cdot \pi(n) = (n + 1) \cdot (n!) = (n + 1)!$. So the induction holds.

Therefore there are $n!$ ways to arrange a set of size n. □

VIII.B Choosing k among n

VIII.B.1 Definition, calculation, and closed form

When “Choosing k among n”, we are counting how many ways are there to choose k elements among n, without any element repeating. In more formal terms, this amounts to count how many subsets of exactly k elements a set S of size n has:

$$\Big| \{A \subseteq S \big| |A| = k\} \Big|$$

There are several common ways to denote that number, although they are all read “from n choose k”. The one I’ll use in this book is $\binom{n}{k}$, also called Newton’s notation. You may sometimes see the notation C_n^k, the C standing for *combination*.

VIII.B.1.i A recurrence relation

To calculate the actual value of $\binom{n}{k}$, we can try to find a recurrence relation. Let S be a set with n elements. First assume $0 < k \leq n$. To choose k elements from S, consider one element $e \in S$. Then either choose:

- to exclude e and then choose k elements from $S \setminus \{e\}$.
- or to include e and then choose $k - 1$ elements from $S \setminus \{e\}$, and add e to the set.

That provides the recurrence relation:

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

Now the cases when we cannot assume that $0 < k \leq n$ are the base cases.

$\boxed{k = 0}$ There is only one way to pick 0 elements (the empty set), so $\binom{n}{0} = 1$ for $n \geq 0$.

$\boxed{n < k}$ There is no way to take more elements than there are in the set so $\binom{n}{k} = 0$.

$\boxed{k < 0}$ There is no way to take a negative number of elements so $\binom{n}{k} = 0$.

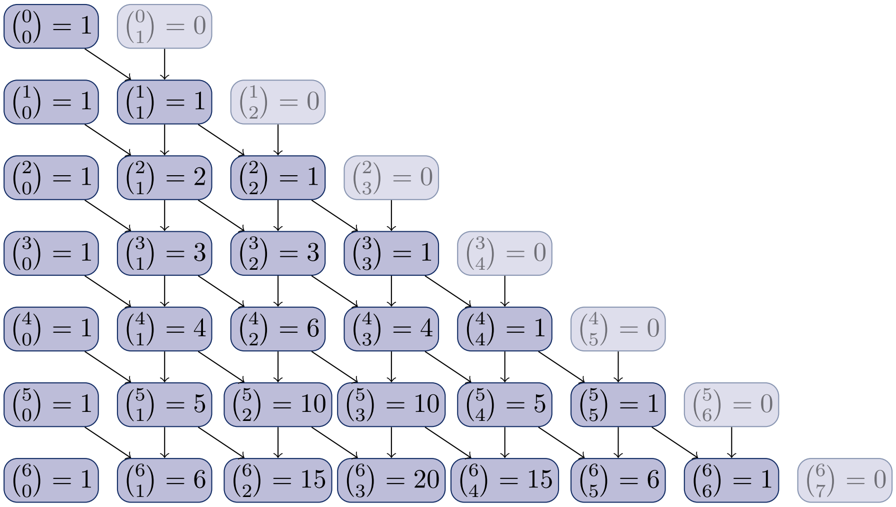

$\binom{0}{0} = 1$ $\binom{0}{1} = 0$
$\binom{1}{0} = 1$ $\binom{1}{1} = 1$ $\binom{1}{2} = 0$
$\binom{2}{0} = 1$ $\binom{2}{1} = 2$ $\binom{2}{2} = 1$ $\binom{2}{3} = 0$
$\binom{3}{0} = 1$ $\binom{3}{1} = 3$ $\binom{3}{2} = 3$ $\binom{3}{3} = 1$ $\binom{3}{4} = 0$
$\binom{4}{0} = 1$ $\binom{4}{1} = 4$ $\binom{4}{2} = 6$ $\binom{4}{3} = 4$ $\binom{4}{4} = 1$ $\binom{4}{5} = 0$
$\binom{5}{0} = 1$ $\binom{5}{1} = 5$ $\binom{5}{2} = 10$ $\binom{5}{3} = 10$ $\binom{5}{4} = 5$ $\binom{5}{5} = 1$ $\binom{5}{6} = 0$
$\binom{6}{0} = 1$ $\binom{6}{1} = 6$ $\binom{6}{2} = 15$ $\binom{6}{3} = 20$ $\binom{6}{4} = 15$ $\binom{6}{5} = 6$ $\binom{6}{6} = 1$ $\binom{6}{7} = 0$

Figure VIII.1: Calculating some $\binom{n}{k}$ from the recurrence relation.

Proposition: Recurrence relation for $\binom{n}{k}$

Let S be a set with n elements. Then

$$\binom{n}{k} = \begin{cases} 1 \text{ if } k = 0 \\ 0 \text{ if } n < k \text{ or } k < 0 \\ \binom{n-1}{k} + \binom{n-1}{k-1} \text{ otherwise} \end{cases}$$

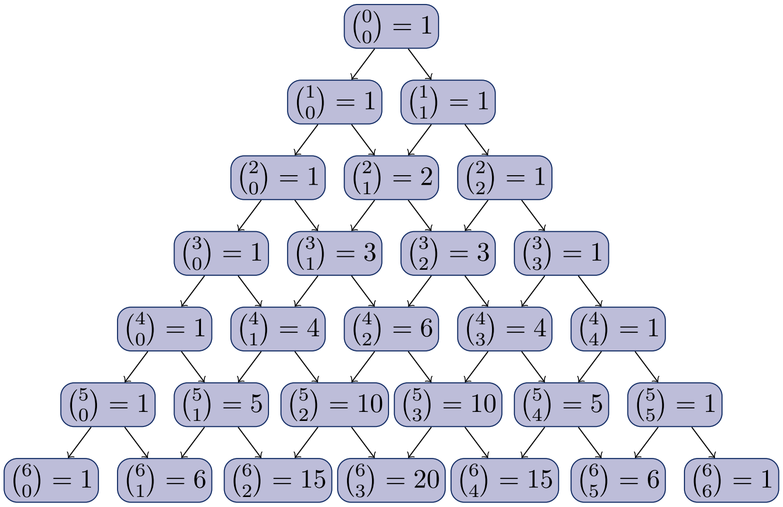
Before finding a closed form, we can calculate some of these values and lay them out in 2-dimensions. This is done in Figure VIII.1. In this figure, arrows represent the contribution of a coefficient in the other: so for example arrows from $\binom{4}{2}$ and $\binom{4}{3}$ to $\binom{5}{3}$ represent the fact that $\binom{4}{2} + \binom{4}{3} = \binom{5}{3}$. What we can remark (and which will be formally proved once we have a closed form), is that there are zeroes on all the fringes. These correspond to the base case $n < k$. As they don't really impact the calculation we can actually leave them out of the depiction (here they are semi-transparent). There should also be zeroes when $k < 0$ but this was left out of the picture.

We can also notice that there are ones on the edges: $\binom{n}{0} = \binom{n}{n} = 1$. That make sense because there is only one way to choose n from n: choosing the whole set. More generally, we can notice that there is symmetry: $\binom{n}{k} = \binom{n}{n-k}$. Again, intuitively that makes sense because choosing k elements to take is like choosing $n - k$ elements to leave. To highlight this symmetry, the $\binom{n}{k}$ are often displayed in a triangle, called *Pascal's Triangle*, depicted in Figure VIII.2[1].

VIII.B.1.ii A closed form

It is possible to do a direct proof from the problem to a closed-form formula, by indirect counting and reformulating the problem. First choose the first element: there n possibilities. Then there are $n-1$ possibilities for the second, as we don't want repetition. This continues

[1] Both Figures VIII.1 and VIII.2 were adapted from `https://tex.stackexchange.com/questions/17522/pascals-triangle-in-tikz`



$\binom{0}{0} = 1$
$\binom{1}{0} = 1$ $\binom{1}{1} = 1$
$\binom{2}{0} = 1$ $\binom{2}{1} = 2$ $\binom{2}{2} = 1$
$\binom{3}{0} = 1$ $\binom{3}{1} = 3$ $\binom{3}{2} = 3$ $\binom{3}{3} = 1$
$\binom{4}{0} = 1$ $\binom{4}{1} = 4$ $\binom{4}{2} = 6$ $\binom{4}{3} = 4$ $\binom{4}{4} = 1$
$\binom{5}{0} = 1$ $\binom{5}{1} = 5$ $\binom{5}{2} = 10$ $\binom{5}{3} = 10$ $\binom{5}{4} = 5$ $\binom{5}{5} = 1$
$\binom{6}{0} = 1$ $\binom{6}{1} = 6$ $\binom{6}{2} = 15$ $\binom{6}{3} = 20$ $\binom{6}{4} = 15$ $\binom{6}{5} = 6$ $\binom{6}{6} = 1$

Figure VIII.2: Pascal's Triangle.

until all k elements are chosen, so we have counted $n \cdot (n-1) \cdots (n-k+2) \cdot (n-k+1)$ options. Note that this can be rewritten as $\frac{n!}{(n-k)!} = \frac{n \cdot (n-1) \cdots (n-k+2) \cdot (n-k+1) \cdot (n-k) \cdots 1}{(n-k) \cdots 1} = n \cdot (n-1) \cdots (n-k+2) \cdot (n-k+1)$ But choosing so we have counted the same sets several times: picking a, then b, then c is the same as picking b then a, then c: we must divide by all the ways we had to pick the same set. Since there are $k!$ possible orderings of this set of size k, we obtain

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

Remark that this works for $0 \leq k \leq n$, otherwise the reasoning does not really work, and $\binom{n}{k}$ is 0.

As this kind of proof might seem a little informal, it is best to prove the result by induction, using the recurrence relation. Although to do so we had to have a closed form to state the induction hypothesis.

Theorem: Closed form for $\binom{n}{k}$

$$\binom{n}{k} = \begin{cases} \frac{n!}{(n-k)! \cdot k!} \text{ if } 0 \leq k \leq n \\ 0 \text{ otherwise} \end{cases}$$

Remark: this formula directly gives . If you are not convinced, or just want to prove that it matches the recurrence relation, we will do a proof by induction.

Proof

The proof will use strong induction. The property we want to prove is $\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!} \Leftrightarrow \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$.

Base cases:

$\boxed{k=0}$ $\frac{n!}{(n-k)!\cdot k!} = \frac{n!}{n!\cdot 0!} = \frac{n!}{n!\cdot 1} = 1$, so the property holds for $k=0$.

$\boxed{n<k,k<0}$ it is 0 in both formulas, so the property holds in these cases.

Inductive case: Assume the property holds for every $(k',n') \prec (k,n)$, where $\prec$ is the strict lexicographic ordering. In particular, it holds for $(k-1,n-1)$ and for $(k,n-1)$.

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

So the equality holds for (k,n).

Therefore the property holds for any k and n. □

Corollary

For $n,k \in \mathbb{N}$, $\binom{n}{k} = \binom{n}{n-k}$ and $\binom{n}{1} = \binom{n}{n-1} = n$.

Proof

- For the first point:

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

- $\binom{n}{1} = \binom{n}{n-1}$ by the above. In this particular case we have:

$$\binom{n}{1} = \frac{n!}{(n-1)!\cdot 1!} = \frac{n\cdot(n-1)\cdots 1}{(n-1)\cdots 1} = n.$$

VIII.B.2 Binomial coefficients

VIII.B.2.i Binomial expansion

Collectively, all the $\binom{n}{k}$ are known as the *binomial coefficients*, because they appear as the coefficients in the expansion of the *binomial* $(x+y)^n$. This is actually where they were introduced as first by Newton, rather than for counting.

Theorem

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

Proof

We prove the result by induction.

Base case: $n = 0$: $(x+y)^0 = 1 = 1 \cdot x^0 y^0$, so the property holds.

Inductive case: Assume the property holds for n, let's prove it holds for $n+1$.

$$(x+y)^{n+1} = (x+y)\cdot(x+y)^n = (x+y)\cdot\left(\sum_{k=0}^{n}\binom{n}{k}x^k y^{n-k}\right)$$

by induction hypothesis. We can now distribute:

$$(x+y)^{n+1} = x\cdot\left(\sum_{k=0}^{n}\binom{n}{k}x^k y^{n-k}\right) + y\left(\sum_{k=0}^{n}\binom{n}{k}x^k y^{n-k}\right) =$$

$$\left(\sum_{k=0}^{n}\binom{n}{k}x^{k+1} y^{n-k}\right) + \left(\sum_{k=0}^{n}\binom{n}{k}x^k y^{n-k+1}\right)$$

Now the idea is twofold: we'll need to combine like terms; and we'll need to have terms that match the result we want, namely terms using $n+1$ instead of n.

$$(x+y)^{n+1} = \left(\sum_{k=0}^{n}\binom{n}{k}x^{k+1} y^{(n+1)-(k+1)}\right) + \left(\sum_{k=0}^{n}\binom{n}{k}x^k y^{(n+1)-k}\right)$$

To make like terms appear, we remark that in the left sum we have $k+1$ while we have k in the right sum. But as k is only the summation variable, it can be renamed, as long as the sum bounds are changed to: so summing for k going from 0 to n is the same as summing for $k+1$ going from 1 to $n+1$. To more easily see what's going on, let's perform this change in two steps: first set $i = k+1$, so the sum in k from 0 to n becomes a sum in i from 1 to $n+1$. Then, because the summation variable is bound within the sum, it can be α-renamed into k:

See Section II.B.6 for details about variable scope and renaming.

$$(x+y)^{n+1} = \left(\sum_{i=1}^{n+1}\binom{n}{i-1}x^{i} y^{(n+1)-i}\right) + \left(\sum_{k=0}^{n}\binom{n}{k}x^k y^{(n+1)-k}\right) =$$

$$\left(\sum_{k=1}^{n+1}\binom{n}{k-1}x^{k} y^{(n+1)-k}\right) + \left(\sum_{k=0}^{n}\binom{n}{k}x^k y^{(n+1)-k}\right)$$

Now we do have like terms, but the summation bounds are not the same, so we cannot combine all the terms. We need to separate the terms that don't match to be able to actually perform the sum. Note that the terms that don't match are x^{n+1}, corresponding to $k = n+1$ in the first sum and y^{n+1} corresponding to $k = 0$ in the second. The resulting sums will both have k going from 1 to n.

$$(x+y)^{n+1} = \binom{n}{n}x^{n+1}y^0 + \left(\sum_{k=1}^{n}\binom{n}{k-1}x^k y^{(n+1)-k}\right) + \binom{n}{0}x^0y^{n+1} +$$

$$\left(\sum_{k=1}^{n}\binom{n}{k}x^k y^{(n+1)-k}\right) =$$

$$\binom{n}{n}x^{n+1}y^0 + \binom{n}{0}x^0y^{n+1} + \left(\sum_{k=1}^{n}\binom{n}{k-1}x^k y^{(n+1)-k} + \binom{n}{k}x^k y^{(n+1)-k}\right) =$$

$$\binom{n}{n}x^{n+1}y^0 + \binom{n}{0}x^0y^{n+1} + \left(\sum_{k=1}^{n}(\binom{n}{k-1} + \binom{n}{k})x^k y^{(n+1)-k}\right)$$

Using the recurrence property $\binom{n}{k-1} + \binom{n}{k} = \binom{n}{k}$:

$$(x+y)^{n+1} = \binom{n}{n}x^{n+1}y^0 + \binom{n}{0}x^0y^{n+1} + \left(\sum_{k=1}^{n}\binom{n+1}{k}x^k y^{(n+1)-k}\right)$$

Now to reintegrate x^{n+1} and y^{n+1}, we can remark that $\binom{n}{n} = \binom{n+1}{n+1} = 1$ and $\binom{n}{0} = \binom{n+1}{0} = 1$. So

$$(x+y)^{n+1} = \binom{n+1}{n+1}x^{n+1}y^0 + \binom{n+1}{0}x^0y^{n+1} + \left(\sum_{k=1}^{n}\binom{n+1}{k}x^k y^{(n+1)-k}\right) =$$

$$\sum_{k=0}^{n+1}\binom{n+1}{k}x^k y^{(n+1)-k}$$

So the property holds for $n+1$, therefore it holds for every $n \in \mathbb{N}$. □

For example we can expand $(x+y)^4 = x^4+4x^3y+6x^2y^2+4xy^3+y^4$. In practice, calculating this expansion can be done by using *Pascal's Triangle* (Figure VIII.2) and remembering that the powers of x and y always sum to n.

A consequence of this expansion is that by setting both x and y to 1, we have:

$$\sum_{k=0}^{n}\binom{n}{k} = 2^n$$

This also makes sense in terms of our original formulation in combinatorics. Adding for all the k how many subsets of k elements exist is just counting all the subsets, of which there are 2^n.

VIII.B.2.ii Number Theory with binomial coefficients

As the coefficients are integers, it makes sense to take a look in terms of number theory. Namely, how do $\binom{n}{k}$ behave modulo n?

Theorem: Freshman's dream

If n is prime, then $(x+y)^n \equiv x^n + y^n \pmod{n}$.

This theorem is called *Freshman's dream* as a joke to unexperienced mathematicians who try to expand a binomial by simply distributing the powers. Of courses that does not work in $\mathbb{R}$ (or even $\mathbb{Q}$, $\mathbb{Z}$, or $\mathbb{N}$), but is a particular property of $\mathbb{Z}/n\mathbb{Z}$ with n prime. The proof of the Theorem relies on the following Lemma:

Lemma: Binomial with n prime

If n is prime, then for $0 < k < n$, $n \mid \binom{n}{k}$.

Proof

The closed-form formula provides $\binom{n}{k} = \frac{n!}{(n-k)!\cdot k!}$. In this formula, all prime factors of $(n-k)!$ and $k!$ are smaller than n, therefore do not divide n (by primality) and cannot be divided by n. As a result n, which is a factor of the numerator, cannot be simplified in the fraction. Since $\binom{n}{k}$ is an integer (as the recurrence relation shows), then n is a factor of $\binom{n}{k}$. □

Proof of Freshman's Dream

Write the binomial expansion $(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$. Since for $0 < k < n$, $n \mid \binom{n}{k}$ by the above Lemma, $\binom{n}{k} \equiv 0 \pmod{n}$ and the term is canceled. What remains is $(x+y)^n \equiv \binom{n}{0} x^0 y^{n-0} + \binom{n}{n} x^n y^{n-n} \equiv x^n + y^n \pmod{n}$. □

Using *Freshman's dream*, we can now proof Fermat's Little Theorem (stated in Section VII.E.4.i and used to prove the correctness of the RSA protocol in Section VII.E.4.iv).

Fermat's Little Theorem

If n is prime then for $0 < p < n$, $p^{n-1} \equiv 1 \pmod{n}$.

Proof of Fermat's Little Theorem

We first prove that $p^n \equiv p \bmod n$ for $p \in \mathbb{N}$ by induction on p.

Base case: $p = 0$: $p^n \equiv 0^n \equiv 0 \pmod{n}$.

Induction case: Assume the Theorem holds for p.

By *Freshman's dream*, $(p+1)^n \equiv p^n + 1^n \pmod{n}$. But $p^n \equiv p \pmod{n}$ by induction hypothesis and $1^n \equiv 1 \pmod{n}$. Therefore we have $(p+1)^n \equiv p + 1 \pmod{n}$ and the property holds for $p+1$

Therefore for every $p \in \mathbb{N}$, $p^n \equiv p \bmod n$.

Now we assume that $0 < p < n$. Write $p^n \equiv p \pmod{n}$ as $p^n = p + kn$ for some k. Then $p(p^{n-1} - 1) = kn$. But $n \not| \, p$ (since $0 < p < n$) so by Euclid's Lemma $n \mid (p^{n-1} - 1)$ and $p^{n-1} - 1 = k'n$ for some k'. We can write $p^{n-1} = 1 + k'n$ therefore $p^{n-1} \equiv 1 \pmod{n}$. □