Undergraduate Topics in Computer Science

David Makinson

Sets, Logic and Maths for Computing

Second Edition

UTiCS

Springer

Undergraduate Topics in Computer Science

Undergraduate Topics in Computer Science (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems. Many include fully worked solutions.

For further volumes:
http://www.springer.com/series/7592

David Makinson

Sets, Logic and Maths for Computing

Second edition

Springer

David Makinson
London School of Economics
Department of Philosophy
Houghton Street
WC2A 2AE London
United Kingdom
david.makinson@gmail.com

ISSN 1863-7310
ISBN 978-1-4471-2499-3   e-ISBN 978-1-4471-2500-6
DOI 10.1007/978-1-4471-2500-6
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2012930652

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The first part of this preface is for the student, the second for the instructor. Some readers may find it helpful to look at both. Whoever you are, welcome!

For the Student

You have finished secondary school and are about to begin at a university or technical college. You want to study computing. The course includes some mathematics – and that was not necessarily your favourite subject. But there is no escape: a certain amount of finite mathematics is a required part of the first-year curriculum, because it is a necessary toolkit for the subject itself.

What Is in This Book?

That is where this book comes in. Its purpose is to provide the basic mathematical language that you need to enter the world of the information and computer sciences.

It does not contain all the mathematics that you will need through the several years of your undergraduate career. There are other very good, quite massive, volumes that do that. At some stage, you will find it useful to get one and keep it on your shelf for reference. But experience has convinced this author that no matter how good the compendia are, beginning students tend to feel intimidated, lost and unclear about what parts to focus on. This short book, in contrast, offers just the basics that you need to know from the beginning, on which you can build further as needed.

It also recognizes that you may not have done much mathematics at school, may not have understood very well what was going on and may even have grown to detest it. No matter: you can learn the essentials here and perhaps even have fun doing so.

So what is the book about? It is about certain tools that we need to apply over and over again when thinking about computations. They include:

1. *Collecting* things together. In the jargon of mathematics, this is *set theory*.
2. *Comparing* things. This is the theory of *relations*.
3. *Associating* one item with another. Introduces the key notion of a *function*.
4. *Recycling outputs as inputs*. We introduce the ideas of *recursion* and *induction*.

5. *Counting*. The mathematician calls it *combinatorics*.
6. *Weighing the odds*. This is done with *probability*.
7. *Squirrel math*. Here we make use of *trees*.
8. *Yea and Nay*. Just two truth-values underlie *propositional logic*.
9. *Everything and nothing*. That is what *quantificational logic* is about.
10. *Just supposing*. How *complex proofs* are built out of simple ones.

Without an understanding of these basic concepts, large portions of computer science remain behind closed doors. As you begin to grasp the ideas and integrate them into your thought, you will also find that their application extends far beyond computing into many other areas.

How Should You Use It?

The good news is that there is not all that much to commit to memory. Your sister studying medicine, or brother going for law, will envy you terribly for this. In our subject, the essential things are to *understand* and be able to *apply*.

But that is a much more subtle affair than one might imagine, as the two are interdependent. Application without understanding is blind and quickly leads to errors – often trivial, but sometimes ghastly. On the other hand, comprehension remains poor without the practice given by applications. In particular, you do not fully understand a definition until you have seen how it takes effect in specific situations: positive examples reveal its scope, negative ones show its limits. It also takes some experience to be able to recognize *when* you have really understood something, compared to having done no more than recite the words or call upon them in hope of blessing.

For this reason, exercises have been included as an indispensable part of the learning process. Skip them at your peril: no matter how simple and straightforward a concept seems, you will not fully assimilate it unless you apply it. That is part of what is meant by the old proverb ‘there is no royal road in mathematics’. Even when a problem is accompanied by a sample solution, you will benefit much more if you first hide the answer and try to work it out for yourself. That requires self-discipline, but it brings real rewards. Moreover, the exercises have been chosen so that in many instances the result is just what is needed to make a step somewhere later in the book. They are thus integral to the development of the general theory.

By the same token, do not get into the habit of skipping proofs when you read the text. Postpone, yes, but omit, no. In mathematics, you have never fully understood a fact unless you have also grasped *why* it is true, that is, have assimilated at least one proof of it. The well-meaning idea that mathematics can be democratized by teaching the ‘facts’ and forgetting about the proofs wrought disaster in some secondary and university education systems.

In practice, the mathematical tools that we bulleted above are rarely applied in isolation from each other. They gain their real power when used in concert, setting up a crossfire that can bring tough problems to the ground. For example, the concept of a set, once explained in the first chapter, is used everywhere in what follows. Relations reappear in graphs, trees and logic. Functions are omnipresent.

For the Instructor

Any book of this kind needs to find delicate balances between competing virtues and shortcomings of modes of presentation and choices of material. We explain the vision behind the choices made here.

Manner of Presentation

Mathematically, the most elegant and coherent way to proceed is to begin with the most general concepts and gradually unfold them so that the more specific and familiar ones appear as special cases. Pedagogically, this sometimes works, but it can also be disastrous. There are situations where the reverse is often required: begin with some of the more familiar examples and special cases, and then show how they may naturally be broadened.

There is no perfect solution to this problem; we have tried to find a least imperfect one. Insofar as we begin the book with sets, relations and functions in that order, we are following the first path. But, in some chapters, we have followed the second one. For example, when explaining induction and recursion, we begin with the most familiar special case, simple induction/recursion over the positive integers; then pass to their cumulative forms over the same domain; broaden to their qualitatively formulated structural versions; and finally present the most general forms on arbitrary well-founded sets. Again, in the chapter on trees, we have taken the rather unusual step of beginning with rooted trees, where intuition is strongest and applications abound, then abstracting to unrooted trees.

In the chapters on counting and probability, we have had to strike another balance – between traditional terminology and notation antedating the modern era and its translation into the language of sets, relations and functions. Most textbook presentations do it all in the traditional way, which has its drawbacks. It leaves the student in the dark about the relation of this material to what was taught in earlier chapters on sets and functions. And, frankly, it is not always very rigorous or transparent. Our policy is to familiarize the reader with *both* kinds of presentation – using the language of sets and functions for a clear understanding of the material itself and the traditional languages of combinatorics and probability to permit communication in the local dialect.

In those two chapters, yet another balance had to be found. One can easily supply counting formulae and probability equalities to be committed to memory and applied in drills. It is more difficult to provide reader-friendly explanations and proofs that permit students to understand what they are doing and why. This book tries to do both, with a rather deeper commitment to the latter than is usual. In particular, it is emphasized that whenever we wish to count the number of selections of k items from a pool of n, a definite answer is possible only when it is clearly understood whether the selections admit repetition and whether they discriminate between orders, giving four options and thus four different counting formulae for the toolbox. The student must learn which tool to choose, and why, as much as how to use it.

The place of logic in the story is delicate. We have left its systematic exposition to the end – a decision that may seem rather strange, as one uses logic whenever reasoning mathematically, even about the most elementary things discussed in the first chapters. Do we not need a chapter on logic at the very beginning of the book? The author's experience in the classroom tells him that in practice that does not work well. Despite its simplicity – perhaps indeed because of it – logic can be elusive for beginning students. It acquires intuitive content only as examples of its employment are revealed. Moreover, it turns out that a really clear explanation of the basic concepts of logic requires some familiarity with the mathematical notions of sets, relations, functions and trees.

For these reasons, the book takes a different tack. In its early chapters, notions of logic are identified briefly as they arise in the discussion of more 'concrete' material. This is done in 'logic boxes'. Each box introduces just enough to get on with the task in hand. Much later, in the last three chapters, all this is brought together and extended. By then, the student should have little trouble appreciating what the subject is all about and how natural it all is, and will be ready to use other basic mathematical tools when studying it.

From time to time, there are boxes of a different nature – 'Alice boxes'. This little troublemaker comes from the pages of Lewis Carroll to ask embarrassing questions in all innocence. Often they are on points that students find puzzling, but which they have trouble articulating clearly or are too shy to pose. It is hoped that the Mad Hatter's replies are of assistance to them as well as to Alice.

The house of mathematics can be built in many different ways, and students often have difficulty reconciling the formulations and constructions of one text with those of another. From time to time, we comment on such differences. Two examples, which always give trouble if not dealt with explicitly, arise in quantificational (first-order) logic. They concern different, although ultimately equivalent, ways of reading the quantifiers and different ways of using the terms 'true' and 'false' for formulae of containing free variables.

Choice of Topics

Overall, our choice of topics is fairly standard, as the chapter titles indicate. If strapped for class time, an instructor could omit some of the starred sections of Chaps. 4, 5, 6, 7, 8, 9, 10. But it is urged that Chaps. 1, 2, 3 be taught entire, as everything in them is useful to the computer science student and is also needed in following chapters. When sufficient classroom time is available, some teachers may, on the other hand, wish to deepen topics or add further ones.

We have not included a chapter on the theory of graphs. That was a difficult call to make, and the reasons were as follows. Although trees are a particular kind of graph, there is no difficulty in covering everything we want to say about trees without entering into general graph theory. Moreover, an adequate treatment of graphs, even if squeezed into one chapter of about the same length as the others, takes a good 2 weeks of additional class time to cover properly with enough examples and exercises to make it sink in. The basic theory of graphs is a rather messy topic, with a rather high definition/theorem ratio and multiple options about how wide to cast the net (directed/undirected graphs, with or without loops, multiple edges and so on). The author's experience is that students gain little from a high-speed run through a series of distinctions and definitions, memorized for the examinations and then promptly forgotten.

On the other hand, recursion and induction are here developed in more detail than is usual in texts of this kind, where it is common to neglect recursive definition in favour of inductive proof and to restrict attention to the natural numbers. Although Chap. 4 begins with the natural numbers, it goes on to explain number-free forms, in particular the often neglected structural ones that are so important for computer science and logic. We also explain the general versions of induction and recursion for arbitrary well-founded relations. Throughout the presentation, the interplay between recursive definition and inductive proof is brought out, with the latter piggybacking on the former. This chapter ends up being the longest in the book.

Finally, a decision had to be made whether to include specific algorithms and, if so, in what form: ordinary English, pseudo-code outline or a real-life programming language in full detail? Most first-year students of computing will be taking, in parallel, courses on principles of programming and some specific programming language; but the languages chosen differ from one institution to another and change over time. The policy in this text is to sketch the essential idea of basic algorithms in plain but carefully formulated English. In a few cases (particularly the chapter on trees), we give optional exercises in expressing them in pseudo-code. Instructors wishing to make more systematic use of pseudo-code, or to link material with specific programming languages, should feel free to do so.

Courses Outside Computer Science

Computer science students are not the only ones who need to learn about these topics. Students of mathematics, philosophy as well as the more theoretical sides of linguistics, economics and political science, all need a course in basic formal methods covering more or less the same territory. This text can be used, with some omissions and additions, for a formal methods course in any of those disciplines.

In the particular case of philosophy, in the second half of the twentieth century, there was an unfortunate tendency to teach only elementary logic, leaving aside instruction on sets, relations, functions, recursion/induction, probability and trees. Even within logic, the focus was rather narrow, often almost entirely on so-called natural deduction. But as already remarked, it is difficult for the student to get a clear idea of what is going on in logic, and difficult for the instructor to say anything interesting about it, without having those other concepts available in the toolkit. The few philosophy students going on to more advanced courses in logic are usually exposed to such tools in bits and pieces but without a systematic grounding. It is the author's belief that all of the subjects dealt with in this book (with the exception of Chap. 5 on counting and the last two sections of Chap. 7 on trees) are also vital for an adequate course on formal methods for students of philosophy. With some additional practice on the ins and outs of approximating statements of ordinary language in the notation of propositional and predicate logic, the book can also be used as a text for such a course.

The Second Edition

For those already familiar with the first edition, we mention the modifications made for the second one. On a general level, affecting the entire book, the main developments are as follows:

- Formulations have been reviewed and retouched to improve clarity and simplicity. Sometimes the order of development within a chapter has been changed slightly. In the end, hardly a paragraph remains untouched.
- Further exercises have been added at points where classroom experience suggested that they would be useful.
- Additional sample answers have also been provided. But, resisting classroom pressures, not all exercises are supplied with answers! After students have learned to swim in shallow water, they need to acquire confidence and self-reliance by venturing out where their feet no longer touch bottom. Ultimately that leads to the satisfaction of getting things right where no crib is possible. Instructors can also use these exercises in tests and examinations.
- To facilitate cross-reference, the numbering of exercises has been aligned with that of the sections and subsections. Thus, Exercise *x.y.z* (*k*) is the *k*th exercise in Subsection *x.y.z* of Chapter *x*, with the (*k*) omitted when there is only one exercise for the subsection. When a section has no subsections, the numbering is reduced to *x.y* (*k*), and end-of-chapter exercises are listed simply as *x* (*k*), where *x* is the chapter number.

On the level of specific chapters, there have been significant revisions in the content and presentation of Chaps. 5, 6, 7:

- In Chap. 5 on counting, the section on rearrangements and configured partitions has been rewritten to make it clearer and more reader-friendly, and some material of lesser interest has been removed.
- In Chap. 6 on probability, the last sections have been refashioned, and some material of little interest in the finite context has been eliminated.
- In Chap. 7 on trees, we have clarified the presence and role of the empty tree, which had been left rather vague in the first edition.

The chapters on propositional and first-order logic have been very much reorganized, emerging as three rather than two chapters:

- *Order of presentation*. The general notion of a consequence (or closure) relation has been taken from its place at very beginning of Chap. 8 and relocated in the new Chap. 10. The previous position was appropriate from a theoretical point of view, but from a pedagogical one it just did not work. So the abstract concept has been kept under wraps until readers have become thoroughly familiar with the particular cases of classical propositional and first-order consequence.
- *Motivation*. Moreover, the concept of a consequence relation is now motivated by showing how it supports the validity of 'elementary derivations', understood as finite sequences or trees of propositions made up of individually valid steps. It is shown that the conditions defining consequence relations are just what are needed for elementary proofs to do their job.
- *Content*. The sections of Chaps. 8 and 9 that in the first edition were devoted to the skill of constructing 'natural deductions' in the symbols of propositional and quantificational logic have been suppressed. In their place, the new Chap. 10 includes an *informal explanation* of the main strategies of mathematical proof, *rigorous statements* of the higher-level rules on which those strategies are based and an explanation on how the rules support traditional practice. Roughly speaking, a formal drill of dubious value has given way to greater attention to a good understanding of the recursive structure of proofs.
- *Other*. The notation used in explaining the semantics of quantificational logic in Chap. 9 has been considerably streamlined, to keep it as simple as possible without loss of clarity or rigour.

A section of the author's webpage http://sites.google.com/site/davidcmakinson is set aside for material relating to this edition: residual errata, comments, further exercises, etc. Readers are invited to send their observations to david.makinson@gmail.com.

Acknowledgements

The author would like to thank Anatoli Degtyarev, Franz Dietrich, Valentin Goranko and George Kourousias for helpful discussions when writing the first edition. George also helped prepare the diagrams. Thanks to the London School of Economics (LSE) and successive department heads Colin Howson, Richard Bradley and Luc Bovens for the wonderful working environment that they provided while both editions were being written. Herbert Huber's emails helped find first-edition typos. For the second edition, Abhaya Nayak and Xavier Parent made valuable comments on a draft of the new tenth chapter.

Saint Martin de Hinx

Contents

List of Figures

Collecting Things Together: Sets 1

Abstract

In this chapter, we introduce the student to the world of sets. Actually, only a little bit of it, the part that is needed to get going.

After giving a rough intuitive idea of what sets are, we present the basic relations between them: *inclusion*, *identity*, *proper inclusion* and *exclusion*. We describe two common ways of identifying sets and pause to look more closely at the *empty set*. We then define some basic operations for forming new sets out of old ones: *intersection*, *union*, *difference* and *complement*. These are often called Boolean operations, after George Boole, who first studied them systematically in the middle of the nineteenth century.

Up to this point, the material is all 'flat' set theory, in the sense that it does not look at what we can do when the elements of sets are themselves sets. However, we need to go a little beyond flatland. In particular, we need to generalize the notions of intersection and union to cover arbitrary *collections of sets* and introduce the very important concept of the *power set* of a set, that is, the set of all its subsets.

1.1 The Intuitive Concept of a Set

Every day you need to consider things more than one at a time. As well as thinking about a particular individual, such as the young man or woman sitting on your left in the classroom, you may focus on some collection of people – say, all those students who come from the same school as you do, or all those with red hair. A *set* is just such a collection, and the individuals that make it up are called its *elements*. For example, each student with green eyes is an element of the set of all students with green eyes.

What could be simpler? But be careful! There might be many students with green eyes, or none or maybe just one, but there is always exactly one set of them. It is a single item, even when it has many elements. Moreover, whereas the students themselves are flesh-and-blood persons, the set is an abstract object, thus different

D. Makinson, *Sets, Logic and Maths for Computing*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-2500-6_1,

from those elements. Even when the set has just one element, it is not the same thing as that unique element. For example, even if Achilles is the only person in the class with green eyes, the corresponding set is distinct from Achilles; it is an abstract item and not a person. To anticipate later terminology, the set whose only element is Achilles is often called it the *singleton* for that person, and it is written as {Achilles}.

The elements of a set need not be people. They need not even be physical objects; they may in turn be abstract items. For example, they can be numbers, geometric figures, items of computer code, colours, concepts or whatever you like – and even other sets, which in turn may have sets as elements, and so on. The further reaches of set theory deal with such higher-level sets, but we will not need to go so far. We will only need to consider sets of items that are not themselves sets (*flat* sets, or sets of degree zero) and sets of them (sets of degree one). To avoid the unsettling repetition that occurs in the term 'set of sets', we will also speak synonymously of a *collection*, or *class* of sets.

We need a notation to represent the idea of elementhood. We write $x \in A$ for x is an element of A, and $x \notin A$ for x is *not* an element of A. Here, A is always a set, while x may or may not be a set; in the simple examples, it will not be one. The sign $\in$ is derived from one of the forms of the Greek letter epsilon.

1.2 Basic Relations Between Sets

Sets can stand in various relations to each other, notably inclusion, identity, proper inclusion, and these relationships can be made graphic with diagrams, as explained in this section.

1.2.1 Inclusion

One basic relation is that of *inclusion*. When A, B are sets, we say that A is *included in* B (or A is a *subset of* B) and write $A \subseteq B$ iff every element of A is an element of B. In other words, iff for all x, if $x \in A$, then $x \in B$. Put in another way that is sometimes useful: iff there is no element of A that is not an element of B. Looking at the same relation from the other direction, when this holds, we also say that B *includesA* (B is a *superset* of A) and write $B \supseteq A$.

Warning: Avoid saying that A is 'contained' in B, as this is rather ambiguous. It can mean that $A \subseteq B$, but it can also mean that $A \in B$. *These are not the same and should never be confused*. For example, the integer 2 is an element of the set $\mathbf{N}^{+}$ of all positive integers, but it is not a subset of $\mathbf{N}^{+}$. Conversely, the set E of all even integers is a subset of $\mathbf{N}^{+}$, that is, each of its elements 2, 4, ... is an element of $\mathbf{N}^{+}$, but E is not an element of $\mathbf{N}^{+}$. Neglect of this distinction quickly leads to serious confusion.

Alice Box: Iff

Alice: Hold on, what's this '*iff*'? It's not in my dictionary.

Hatter: Too bad for your dictionary. The expression was introduced around the middle of the last century by the mathematician Paul Halmos, as a handy shorthand for 'if and only if', and soon became standard among mathematicians.

Alice: OK, but aren't we doing some *logic* here? I see words like 'if', 'only if', 'every', 'not' and perhaps more. Shouldn't we begin by explaining what they mean?

Hatter: Life will be easier if for the moment we just use these particles intuitively. We will get back to their exact logical analysis later.

Exercise 1.2.1 (with solution) Which of the following sets are included in which? Use the notation above and express yourself as succinctly and clearly as you can. Recall that a prime number is a positive integer greater than 1 that is not divisible by any positive integer other than itself and 1.
A: the set of all positive integers less than 10
B: the set of all prime numbers less than 11
C: the set of all odd numbers greater than 1 and less than 6
D: the set whose only elements are 1 and 2
E: the set whose only element is 1
F: the set of all prime numbers less than 8

Solution Each of these sets is included in itself, and each of them is included in A. In addition, we have $C \subseteq B$, $E \subseteq D$, $F \subseteq B$, $B \subseteq F$.

Comments Note that none of the other converses hold. For example, we do not have $B \subseteq C$, since $7 \in B$ but $7 \notin C$. Note also that we do not have $E \subseteq B$ since we are not taking 1 to be a prime number.

1.2.2 Identity and Proper Inclusion

The notion of inclusion leads us to the concept of *identity* (alias *equality*) between sets. Clearly, if both $A \subseteq B$ and $B \subseteq A$, then A and B have exactly the same elements – by the definition of inclusion, every element of each is an element of the other; in other words, there is no element of one that is not an element of the other. A basic principle of set theory, called the axiom (or postulate) of *extensionality*, says something more: *When both $A \subseteq B$ and $B \subseteq A$, then the sets A, B are in fact identical*. They are one and the same set, and we write $A = B$.

On the other hand, when $A \subseteq B$ but $A \neq B$, then we say that A is *properly included* in B, and write $A \subset B$. Sometimes $\subset$ is written with a small $\neq$ underneath. That should not cause any confusion, but another notational dialect is more dangerous: a few older texts use $A \subset B$ for plain inclusion. Be wary when you read.

Exercise 1.2.2 (1) (with solution) In Exercise 1.2.1, which of the sets are identical to which?

Solution $B = F$ (so that also $F = B$). And of course $A = A$, $B = B$, etc. since each of the listed sets is identical to itself.

Comment The fact that we defined B and F in different ways makes no difference: the two sets have exactly the same elements, and so by the axiom of extensionality, they are identical.

Exercise 1.2.2 (2) (with solution) In Exercise 1.2.1, which of the sets are properly included in which? In each case give a 'witness' to the proper nature of the inclusion, that is, identify an element of the right one that is not an element of the left one.

Solution $C \subset B$, witnesses 2, 7; $E \subset D$, sole witness 2.

Comment $F \not\subset B$, since B and F have exactly the same elements.

Exercise 1.2.2 (3) (with solution) Which of the following claimed identities are correct? Read the curly braces as framing the elements of the set so that, for example, $\{1, 2, 3\}$ is the set whose elements are just 1, 2, 3. Be careful with the answers.
(a) $\{1, 2, 3\} = \{3, 2, 1\}$, (b) $\{9, 5\} = \{9, 5, 9\}$, (c) $\{0, 2, 8\} = \{|\sqrt{4}|, 0/5, 2^3\}$, (d) $\{7\} = 7$, (e) $\{8\} = \{\{8\}\}$, (f) {London, Beijing} = {Londres, Pekin}, (g) $\{+\} = \{\text{'}+\text{'}\}$.

Solution (a) yes, (b) yes, (c) yes, (d) no, (e) no, (f) yes, (g) no.

Comments
(a) The order of enumeration makes no difference – the sets still have the same elements.
(b) Repeating an element in the enumeration is inelegant, but it makes no substantive difference – the sets still have the same elements.
(c) The elements have been named differently, as well as being written in a different order, but they are the same.
(d) 7 is a number, not a set, while $\{7\}$ is a set with the number 7 as its only element, that is, its singleton.
(e) Both are sets, and both have just one element, but these elements are not the same. The unique element of the left set is the number 8, while the unique element of the right set is the set $\{8\}$. The right set is thus the singleton of the left one.

(f) London is the same city as Londres, although it is named in different languages (English, French). So the sets have exactly the same elements and thus are identical.

(g) The left set has the operation as its sole element, while, under a standard convention for naming by inverted commas, the right set has the *name* of that operation as its sole element, so the sets are not identical. This distinction may seem quite pedantic, but it turns out to be necessary to avoid confusion when dealing with sets of symbols, as is often the case in computer science, logic and linguistics.

Exercise 1.2.2 (3) (with partial solution) True or false? In each case, use your intuition to make a guess and establish it by either proving the point from the definitions (if you guessed positively) or giving a simple counterexample (if you guessed negatively). Make sure that you don't confuse $\subseteq$ with $\subset$.

(a) Whenever $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$.

(b) Whenever $A \subseteq B$ and $C \subseteq B$, then $A \subseteq C$.

(c) Whenever $A_1 \subseteq A_2 \subseteq \ldots \subseteq A_n$ and also $A_n \subseteq A_1$, then $A_i = A_j$ for all $i,j \leq n$.

(d) $A \subset B$ iff $A \subseteq B$ and not $B \subseteq A$.

(e) $A \subseteq B$ iff $A \subset B$ or $A = B$.

(f) $A = B$ iff neither $A \subset B$ nor $B \subset A$.

(g) Whenever $A \subseteq B$ and $B \subset C$, then $A \subset C$.

(h) Whenever $A \subset B$ and $B \subseteq C$, then $A \subset C$.

Solutions to (a), (b), (f), (h)

(a) True. Take any sets A, B, C. Suppose $A \subseteq B$ and $B \subseteq C$; it suffices to show $A \subseteq C$. Take any x, and suppose $x \in A$; by the definition of inclusion, it is enough to show $x \in C$. But since $x \in A$ and $A \subseteq B$, we have by the definition of inclusion that $x \in B$. So since also $B \subseteq C$, we have again by the definition of inclusion that $x \in C$, as desired.

(b) False. Counterexample: $A = \{1\}$, $B = \{1,2\}$, $C = \{2\}$.

(f) False. The left to right implication is correct, but the right to left one is false, so the entire co-implication (the 'iff') is also false. Counterexample: $A = \{1\}$, $B = \{2\}$.

(h) True. Take any sets A, B, C. Suppose $A \subset B$ and $B \subseteq C$. From the former by the definition of proper inclusion, we have $A \subseteq B$. So by exercise (a), we have $A \subseteq C$. It remains to show that $A \neq C$. Since $A \subset B$, we have by exercise (d) that not $B \subseteq A$, so by the definition of inclusion, there is an x with $x \in B$ but $x \notin A$. Thus, since $B \subseteq C$, we have $x \in C$ while $x \notin A$, so C is not included in A, and thus $A \neq C$, as desired.

Comment The only false ones are (b) and (f). All the others are true.

Logic Box: Proving Conditional Statements

In the preceding exercise, we needed to show several statements of the form 'if this then that', and we followed the most straightforward way of doing so: we *supposed that the first is true*, and on that basis *showed that the second is true*. This is known as *conditional proof.*

We carried it out, for example, in the verification of part (a) of the exercise. Actually, we did it there twice: First, we supposed that $A \subseteq B$ and $B \subseteq C$, and set our goal as showing $A \subseteq C$. That means that whenever $x \in A$, then $x \in C$, so we then supposed that $x \in A$ and aimed to get $x \in C$. There are other lines of attack for establishing 'if . . . then . . . ' statements, but we will come to them later.

Alice Box: Whenever

Alice: Aren't we missing something there? For (a), we were required to show that *whenever* $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$. To me, that says: *for all* sets A, B, C, if $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$. You have explained how we tackle the 'iffy' part, but say nothing about the 'every' part. There must be something underneath that too.

Hatter: One thing at a time! That's also important, and we will get to it in due course.

Alice: It's a promise?

Hatter: It's a promise!

Exercise 1.2.2 (3) is the first in which you are asked to show something, but it is certainly not the last. As well as the logic of such proofs, there are some general points of common sense that you need to bear in mind when trying to prove, or disprove, something.

First, *always be clear in your mind (and on paper) what you are trying to show*. If you don't know what you are attempting to prove, it is unlikely that you will succeed in proving it, and if by chance you do, the proof will probably be a real mess.

Following this rule is not as easy as may appear, for as a proof develops, the goal changes! For example, looking again at part (a) of the preceding exercise, we began by trying to show (a) itself. After choosing A, B, C arbitrarily, our goal was the conditional statement: If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$. Then, after supposing both $A \subseteq B$ and $B \subseteq C$, we switched our goal to $A \subseteq C$. We then chose an arbitrary x, supposed $x \in A$, and aimed to show $x \in C$. In half a dozen lines, four different goals! At each stage, we have to be aware of which one we are driving at. When we start using more sophisticated tools for building proofs, notably *reductio ad absurdum*, the goal-shifts become even more striking.

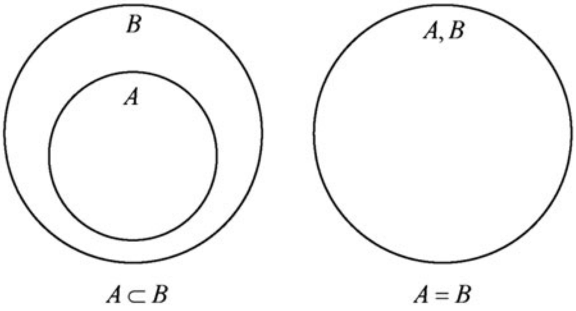

Fig. 1.1 Euler diagrams for proper inclusion and identity

A second rule of common sense when proving things: *As far as possible, be aware of what you are allowed to use and don't hesitate to use it*. If you neglect available information, you may not have enough to do the job.

So what are you allowed to use? To begin with, you may use the *definitions* of terms in the problem (in our example, the notions of subset and proper subset). Too many students come to mathematics with the idea that a definition is just something for decoration – something you can hang on the wall like a picture or diploma. A definition is for use. Indeed, in a very simple verification, most of the steps can consist of 'unpacking' the definitions and then, after just a little reasoning, packing them together again. Next, you may use whatever basic *axioms* (also known as *postulates*) that you have been supplied with. In parts (e) and (f) of the last exercise, that was just the principle of extensionality, stated at the beginning of the section. Finally, you can use anything that you have *already proven*. For example, we did this while proving part (h) of the exercise.

Third point of common sense: *Be flexible and ready to go into reverse*. If you can't prove that a statement is true, try looking for a counterexample in order to show that it is false. If you can't find a counterexample, try to prove that it is true. With some experience, you can often use the failure of your attempted proofs as a guide to finding a suitable counterexample, and the failures of your trial counterexamples to give a clue for constructing a proof. This is all part of the *art of proof and refutation*.

1.2.3 Diagrams

If we think of a set A as represented by all the points in a circle (or any other closed plane figure), then we can represent the notion of one set A being a proper subset of another B by putting a circle labelled A inside a circle labelled B. We can diagram equality, of course, by drawing just one circle and labelling it both A and B. Thus, we have *Euler diagrams* (so named after the eighteenth-century mathematician Euler who used them when teaching a princess by correspondence) (Fig. 1.1).

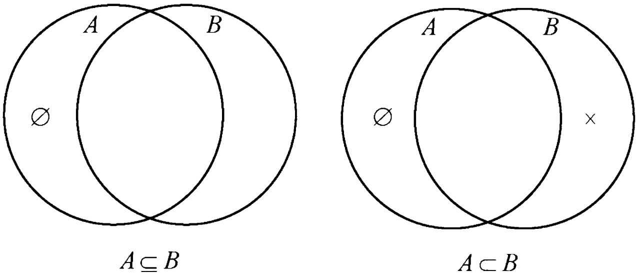

Fig. 1.2 Venn diagrams for inclusion and proper inclusion

How can we diagram inclusion in general? Here we must be careful. *There is no single Euler diagram that does the job*. When $A \subseteq B$, then we may have either of the above two configurations: If $A \subset B$, then the left diagram of Fig. 1.1 is appropriate; if on the other hand $A = B$, then the right diagram is the correct one.

Diagrams are a very valuable aid to intuition, and it would be pedantic and unproductive to try to do without them. But we must also be clearly aware of their limitations. If you want to visualize $A \subseteq B$ using Euler diagrams, and you don't know whether the inclusion is proper, you will need to consider two diagrams and see what happens in each.

However, there is another kind of diagram that can represent plain inclusion without ambiguity. It is called a *Venn diagram* (after the nineteenth-century logician John Venn). It consists of drawing two circles, one for A and one for B, always intersecting no matter what the relationship between A and B, and then putting a mark (e.g. ∅) in an area to indicate that it has no elements, and another kind of mark (e.g. a cross) to indicate that it does have at least one element. With these conventions, the left diagram below represents $A \subseteq B$, while the right diagram represents $A \subset B$ (Fig. 1.2).

Note that the disposition of the circles is always the same: what changes are the areas noted as empty or as non-empty. There is considerable variability in the signs used for this – dots, ticks, etc.

A great thing about Venn diagrams is that they can represent common relationships like $A \subseteq B$ by a single diagram, rather than by an alternation of different diagrams. Another advantage is that with some additions they can be used to represent basic operations on sets as well as relations between them. However, they also have their shortcomings. The bad news is that when you have more than two sets to consider, Venn diagrams can become very complicated and lose their intuitive clarity – which was, after all, their principal *raison d'être*.

Warning: *Do not confuse Euler diagrams with Venn diagrams*. They are constructed differently and read differently. Unfortunately, textbooks themselves are sometimes rather sloppy about this, using the terms interchangeably or even in reverse.

Exercise 1.2.3 (1) (with hints)

(a) Suppose we want to diagram the double proper inclusion $A \subset B \subset C$. We can, of course, do it with two separate Euler (or Venn) diagrams. Do it with a single diagram of each kind, with three circles.

(b) Suppose we want to represent the two proper inclusions $A \subset B$, $A \subset C$ in a single diagram with three circles. How many different Euler diagrams are compatible with this configuration? What kind of difficulty do you get into with placing crosses in the Venn diagram, and how might you resolve it?

Hints: For (a), Euler is straightforward. For Venn, put $\varnothing$ in the smallest areas that must be empty, then place crosses in the smallest areas that must be non-empty. For (b), when constructing Euler diagrams, think of the various possible relations between B and C, and when building the Venn diagram, try to proceed as you did for (a).

1.2.4 Ways of Defining a Set

The examples we have so far looked at already illustrate two important ways of defining or identifying a set. One way is to *enumerate all its elements individually* between curly brackets, as we did in Exercise 1.2.2 (3). Evidently, such an enumeration can be completed only when there are finitely many elements, and is convenient only when the set is fairly small. The order of enumeration makes no difference to what items are elements of the set, for example, $\{1,2,3\} = \{3,1,2\}$, but we usually write elements in some conventional order such as increasing size for smooth communication.

Another way of identifying a set is by *providing a common property*: The elements of the set are understood to be all (and only) the items that have that property. That is what we did for several of the sets in Exercise 1.2.1. There is a notation for this. For example, we write the first set as follows: $A = \{x \in \mathbf{N}^+ \colon x < 10\}$, where $\mathbf{N}^+$ stands for the set of all integers greater than zero (the positive integers). Some authors use a vertical bar in place of a colon.

Exercise 1.2.4 (1) with solution

(a) Identify the sets A, B, C, F of Exercise 1.2.1 by enumeration.

(b) Identify the sets D, E of the same exercise by properties, using the notation introduced.

Solution

(a) $A = \{1,2,3,4,5,6,7,8,9\}$; $B = \{2,3,5,7\}$; $C = \{3,5\}$, $F = \{2,3,5,7\}$.

(b) There are many ways of doing this, here are some. $D = \{x \in \mathbf{N}^+ \colon x$ divides all even integers$\}$; $E = \{x \in \mathbf{N}^+ \colon x$ is less than or equal to every positive integer$\}$.

When a set is infinite, we often use an incomplete 'suspension points' notation. Thus, we might write the set of all even positive integers and the set of all primes, respectively, as follows: $\{2,4,6,\ldots\}$, $\{2,3,5,7,11,\ldots\}$. But it should be emphasized that this is an informal way of writing, used when it is well understood between writer and reader what particular continuation is intended. Clearly, there are many ways of continuing each of these partial enumerations. We normally intend that the most familiar or simplest is the one that is meant. *Without a specific understanding of this kind, the notation is meaningless*.

These two methods of identifying a set – by enumeration and by a common property – are not the only ones. In a later chapter, we will be looking at another very important one, known as recursive definition, where one specifies certain initial elements of the set, and a rule for generating new elements out of old ones. It can be thought of as a mathematically precise way of expressing the idea hinted at by suspension points.

1.3 The Empty Set

Just as zero is a very important natural number, the empty set is basic to set theory. Just as mathematics took a very long time to come up with a clear conceptualization and standard notation for zero, so students can have some initial difficulties with emptiness. By the end of this section, such difficulties should be over.

1.3.1 Emptiness

What do the following two sets have in common?

$A = \{x \in \mathbf{N}^+ \colon x \text{ is both even and odd}\}$

$B = \{x \in \mathbf{N}^+ \colon x \text{ is prime and } 24 \le x \le 28\}$

Answer: Neither of them has any elements. From this it follows that they have exactly the same elements – neither has any element that is not in the other. So by the principle of extensionality given in Sect. 1.2.2, they are identical, that is, they are the same set. Thus, $A = B$, even though they are described differently. This leads to the following definition. *The empty set is defined to be the (unique) set that has no elements at all.*

This set is written $\varnothing$. We already used this notation informally in Venn diagrams when indicating that a certain region of the diagram has no elements. The fact that it has no elements does not mean that it has any less 'existence' than other sets, any more than zero has less existence than the positive numbers.

Exercise 1.3.1 (1) with solution Show that $\varnothing \subseteq A$ for every set A.

Solution We need to show that for all x, if $x \in \varnothing$, then $x \in A$. In other words, there is no x with $x \in \varnothing$ but $x \notin A$. But by the definition of $\varnothing$, there is no x with $x \in \varnothing$, so we are done.

Alice Box: If . . . then . . .

Alice: That's a short proof, but a strange one. You say 'in other words', but are the two formulations really equivalent?

Hatter: Indeed they are. This is because of the way in which we understand 'if . . . then . . . ' statements in mathematics. We could explain that in detail now, but it is probably better to come back to it a bit later.

Alice: Another promise?

Hatter: Indeed!

1.3.2 Disjoint Sets

We say that sets A, B are *disjoint* (aka *mutually exclusive*) iff they have no elements in common. That is, iff there is no x such that both $x \in A$ and $x \in B$. When they are not disjoint, that is, when they have at least one element in common, we can say that they *overlap*.

More generally, when $A_1, \ldots, A_n$ are sets, we say that they are *pairwise disjoint* iff for any $i,j \leq n$, if $i \neq j$, then A_i has no elements in common with A_j.

Exercise 1.3.2 (1)
(a) Of the sets in Exercise 1.2.1, which are disjoint from which?
(b) Draw Euler and Venn diagrams to express the disjointness of two sets.
(c) Draw a Venn diagram to express the overlapping of two sets.
(d) How many Euler diagrams would be needed to express overlapping?
(e) Give an example of three sets X, Y, Z such that X is disjoint from Y, and Y is disjoint from Z, but X is not disjoint from Z.
(f) Show that the empty set is disjoint from every set, including itself.

1.4 Boolean Operations on Sets

We now define some operations on sets, that is, ways of constructing new sets out of old. We begin with three basic ones: intersection, meet and relative complement, and then several others that can be defined in terms of them.

1.4.1 Intersection

If A and B are sets, we define their *intersection* $A \cap B$, also known as their *meet*, by the following rule: for all x,

$$x \in A \cap B \text{ iff } x \in A \text{ and } x \in B.$$

Exercise 1.4.1 (1) (with partial solution) Show the following:

(a) $A \cap B \subseteq A$ and $A \cap B \subseteq B$.
(b) Whenever $X \subseteq A$ and $X \subseteq B$, then $X \subseteq A \cap B$.
(c) $A \cap B = B \cap A$ (commutation).
(d) $A \cap (B \cap C) = (A \cap B) \cap C$ (association).
(e) $A \cap A = A$ (idempotence).
(f) $A \cap \varnothing = \varnothing$ (bottom).
(g) Reformulate the definition of disjoint sets using intersection.

Solutions to (b), (f), (g) For (b): Suppose $X \subseteq A$ and $X \subseteq B$; we want to show $X \subseteq A \cap B$. Take any x and suppose $x \in X$; we need to show that $x \in A \cap B$. Since $x \in X$ and $X \subseteq A$, we have by the definition of inclusion that $x \in A$, and similarly, since $x \in X$ and $X \subseteq B$, we have $x \in B$. So by the definition of intersection, $x \in A \cap B$, as desired.

For (f): We already have $A \cap \varnothing \subseteq \varnothing$ by (a) above. And we also have $\varnothing \subseteq A \cap \varnothing$ by Exercise 1.3.1 (1), so we are done.

For (g): Sets A, B are disjoint iff $A \cap B = \varnothing$.

Remark Property (a) may be expressed in words by saying that $A \cap B$ is a *lower bound* for A, B. Property (b) tells us that it is a *greatest lower bound* for A, B.

Intersection has been defined using the word ‘and’. But what does this mean? In mathematics, it is very simple – much simpler than in ordinary life. Consider any two statements α, β. Each can be true, or false, but not both. When is the statement ‘α and β’, called the *conjunction* of the two parts, true? The answer is intuitively clear: When each of α, β considered separately is true, the conjunction is true, but in all other cases, the conjunction is false. What are the other cases? There are three of them: α true with β false, α false with β true, α false with β false. All this can be put in the form of a table, called the truth table for conjunction.

Logic Box: Conjunction

α	β	$\alpha \wedge \beta$
1	1	1
1	0	0
0	1	0
0	0	0

In this table, each row represents a possible combination of truth-values of the parts α, β. For brevity, we write 1 for ‘true’ and 0 for ‘false’. The rightmost entry in the row gives us the resulting truth-value of the conjunction ‘α and β’, which we write as $\alpha \wedge \beta$. Clearly, the truth-value of the conjunction is fully determined by each combination of truth-values of the parts. For this reason, conjunction is called a *truth-functional logical connective*.

In a later chapter, we will study the properties of conjunction and other truth-functional connectives. As you may already have guessed, the behaviour of intersection (as an operation on sets) reflects that of conjunction (as a connective between statements). This is because the latter is used in the definition of the former. For example, the commutativity of intersection comes from the fact that ‘α and β’ has exactly the same truth-conditions as ‘β and α’.

For reflection: How do you square the truth table for conjunction with the difference in meaning between ‘they got married and had a baby’ and ‘they had a baby and got married’?

1.4.2 Union

Alongside intersection we have another operation called *union*. The two operations are known as *duals* of each other, in the sense that each is like the other ‘upside down’. For any sets A and B, we define their union $A \cup B$ by the following rule. For all x:

$$x \in A \cup B \text{ iff } x \in A \text{ or } x \in B,$$

where this is understood in the sense:

$$x \in A \cup B \text{ iff } x \in A \text{ or } x \in B \text{ (or both)},$$

in other words:

$$x \in A \cup B \text{ iff } x \text{ is an element of } \textit{at least one} \text{ of } A, B.$$

The contrast with intersection may be illustrated by Venn diagrams, but they differ a little from those used earlier. Now that we are representing operations rather than statements, we no longer pepper the diagram with ∅ and crosses to say that regions are empty or not. Instead, we shade regions to indicate that they are the ones given by the operation. Thus, in Fig. 1.3, two intersecting circles continue to represent the sets A, B, and the shaded area in the left image represents $A \cup B$, while in the right one, it represents $A \cap B$.

The properties of union are just like those of intersection but ‘upside down’. Evidently, this is a rather vague way of speaking; it can be made precise, but it is better to leave the idea on an intuitive level for the moment.

Exercise 1.4.2 (1) Show the following:

(a) $A \subseteq A \cup B$ and $B \subseteq A \cup B$ (upper bound)
(b) Whenever $A \subseteq X$ and $B \subseteq X$, then $A \cup B \subseteq X$ (least upper bound)
(c) $A \cup B = B \cup A$ (commutation principle)
(d) $A \cup (B \cup C) = (A \cup B) \cup C$ (association)

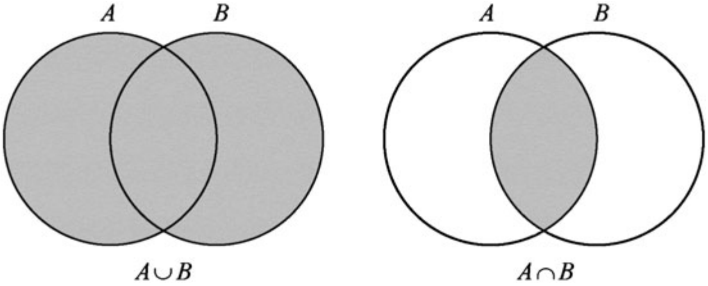

Fig. 1.3 Venn diagrams for union and intersection

(e) $A \cup A = A$ (idempotence)
(f) $A \cup \varnothing = A$ (bottom)

Exercise 1.4.2 (2) What would you naturally call principles (a) and (b) given the names of their counterparts for intersection?

When 'or' is understood in the sense used in the definition of union, it is known as *inclusive disjunction*, or briefly just *disjunction*. Statements 'α or β' are written as $\alpha \vee \beta$. Whereas there is just one way (out of four) of making a conjunction true, there is just one way of making a disjunction false. The truth table is as follows:

Logic Box: Disjunction

α	β	$\alpha \vee \beta$
1	1	1
1	0	1
0	1	1
0	0	0

Clearly, the truth-value of the disjunction is fully determined by each combination of truth-values of the parts. In other words, it is also a truth-functional logical connective. In a later chapter, we will see how the behaviour of union between sets reflects that of disjunction as a connective between statements.

Exercise 1.4.2 (3) In ordinary discourse, we often use 'α or β' to mean 'either α or β, but not both', in other words, 'exactly one of α, β is true'. This is called *exclusive disjunction*. What would its truth table look like?

We have seen some of the basic properties of intersection and of union, taken separately. But how do they relate to each other? The following exercise covers the most important interactions.

Exercise 1.4.2 (4) (with partial solution) Show the following:
(a) $A \cap B \subseteq A \cup B$
(b) $A \cap (A \cup B) = A = A \cup (A \cap B)$ (absorption)
(c) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ (distribution of intersection over union)
(d) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ (distribution of union over intersection)
(e) $A \subseteq B$ iff $A \cup B = B$, and likewise iff $A \cap B = A$

Solution to (c) Writing LHS, RHS for the left and right hand sides, respectively, we need to show that LHS $\subseteq$ RHS and conversely RHS $\subseteq$ LHS.

For LHS $\subseteq$ RHS, suppose that $x \in$ LHS. Then, $x \in A$ and $x \in B \cup C$. From the latter, we have that either $x \in B$ or $x \in C$. Consider the two cases separately. Suppose first that $x \in B$. Since also $x \in A$, we have $x \in A \cap B$, and so by Exercise 1.4.2 (1) part (a), we get $x \in (A \cap B) \cup (A \cap C) =$ RHS as desired. Suppose second that $x \in C$. Since also $x \in A$, we have $x \in A \cap C$ and so again $x \in (A \cap B) \cup (A \cap C) =$ RHS, as desired.

For RHS $\subseteq$ LHS, suppose that $x \in$ RHS. Then, $x \in A \cap B$ or $x \in A \cap C$. Consider the two cases separately. Suppose first that $x \in A \cap B$. Then, $x \in A$ and $x \in B$; from the latter, $x \in B \cup C$, and so combined with the former, $x \in A \cap (B \cup C) =$ LHS, as desired. Suppose second that $x \in A \cap C$. The argument is similar: $x \in A$ and $x \in C$; from the latter, $x \in B \cup C$, and so with the former, $x \in A \cap (B \cup C) =$ LHS, as desired.

Logic Box: Breaking into Cases

In the solution above, we used a technique known as *proof by cases*, or *disjunctive proof*. Suppose we know that either α is true or β is true, but we don't know which. It can be difficult to proceed with this rather weak information. So we break the argument into two parts. First, we suppose that α is true (one case), and with this stronger assumption, we head for whatever it was that we were trying to establish. Then, we suppose instead that β is true (the other case) and argue using this assumption to the same conclusion. If we succeed in reaching the desired conclusion in each case separately, then we know that it must hold irrespective of which case is true.

Unlike conditional proof, the goal remains unchanged through the proof. But the two cases must be treated quite separately: We cannot use the supposition for one case in the argument for the other case. The arguments carried out in the two separate cases may sometimes resemble each other closely (as in our exercise). But in more challenging problems, they may be very different.

Exercise 1.4.2 (5) For an example of proof by cases in arithmetic, show that every positive integer n has the same parity as n^2, that is, if n is even or odd then so is, respectively, n^2. The cases to choose suggest themselves naturally.

Alice Box: Overlapping Cases

Alice: What if *both* cases are true? For example, in the solution to the exercise, in the part for LHS $\subseteq$ RHS: what if both $x \in B$ and $x \in C$?

Hatter: No problem! This just means that we have covered that situation twice. For proof by cases to work, it is not required that the two cases be exclusive. In some examples (as in our exercise), it is easier to work with overlapping cases; sometimes, it is more elegant and economical to work with cases that exclude each other.

1.4.3 Difference and Complement

There is one more Boolean operation on sets that we wish to consider: *difference*. Let A, B be any sets. We define the *difference of B in A*, written $A\backslash B$ (alternatively as $A-B$) to be the set of all elements of A that are *not* elements of B. That is, $A\backslash B=\{x: x \in A \text{ but } x \notin B\}$.

Exercise 1.4.3 (1) (with partial solution)

(a) Draw a Venn diagram for difference.
(b) Give an example to show that we can have $A\backslash B \neq B\backslash A$.
(c) Show (i) $A\backslash A=\varnothing$, (ii) $A\backslash\varnothing=A$.
(d) Show that (i) if $A \subseteq A'$, then $A\backslash B \subseteq A'\backslash B$, and (ii) if $B \subseteq B'$, then $A\backslash B' \subseteq A\backslash B$.
(e) Show that (i) $A\backslash(B\cup C)=(A\backslash B)\cap(A\backslash C)$, (ii) $A\backslash(B\cap C)=(A\backslash B)\cup(A\backslash C)$.
(f) Show that $(A\backslash B)\backslash C=(A\backslash C)\backslash B$.
(g) Find a counterexample to $A\backslash(B\backslash C)=(A\backslash B)\backslash C$.
(h) Show that $A\backslash(B\backslash C)\subseteq(A\backslash B)\cup C$.

Sample solution to (g) Take A to be any non-empty set, for example, $\{1\}$ and put $C=B=A$. Then, LHS $=A\backslash(A\backslash A)=A\backslash\varnothing=A$, while RHS $=(A\backslash A)\backslash A=\varnothing\backslash A=\varnothing$.

The notion of difference acquires particular importance in a special context. Suppose that we are carrying out an extended investigation into the elements of some fairly large set, such as the set $\mathbf{N}^+$ of all positive integers, and that for the purposes of the investigation, the only sets that we need to consider are the subsets of this fixed set. Then, it is customary to refer to the large set as a *local universe*, writing it as U, and consider the differences $U\backslash B$ for subsets $B \subseteq U$. As the set U is fixed throughout the investigation, we then simplify notation and write $U\backslash B$ alias $U-B$ as $-_U B$, or even simply as $-B$ with U left as understood. This application of the difference operation is called *complementation* (within the given universe).

Many other notations are also used in the literature for this important operation, for example, B^-, B', B^c (where the index stands for 'complement'). To give it a Venn diagram, we simply take the diagram for relative complement $A\backslash B$ and blow the A circle up to coincide with the whole box containing the diagram, so as to serve as the local universe.

Exercise 1.4.3 (2) (with partial solution)

(a) Taking the case that A is a local universe U, rewrite equations (e) (i) and (ii) of the preceding exercise using the simple complementation notation described above.

(b) Show that (i) $-(-B) = B$, (ii) $-U = \varnothing$, (iii) $-\varnothing = U$.

Solutions to (a) and (b)(i)

(a) When $A = U$, then equation (i) becomes $U\backslash(B \cup C) = (U\backslash B) \cap (U\backslash C)$, which we can write as $-(B \cup C) = -B \cap - C$; while equation (ii) becomes $U\backslash(B \cap C) = (U\backslash B) \cup (U\backslash C)$, which we can write as $-(B \cap C) = -B \cup - C$.

(b) (i) We need to show that $-(-B) = B$, in other words that $U\backslash(U\backslash B) = B$ whenever $B \subseteq U$ (as assumed when U is taken to be a local universe). We show the two inclusions separately. First, to show $U\backslash(U\backslash B) \subseteq B$, suppose $x \in$ LHS. Then, $x \in U$ and $x \notin (U\backslash B)$. From the latter, either $x \notin U$ or $x \in B$, so $x \in B =$ RHS, as desired. For the converse, suppose $x \in B$. Then, $x \notin (U\backslash B)$. But by assumption $B \subseteq U$ so $x \in U$, and thus $x \in U\backslash(U\backslash B) =$ LHS, as desired.

The identities $-(B \cap C) = -B \cup - C$ and $-(B \cup C) = -B \cap - C$ are known as *de Morgan's laws*, after the nineteenth-century mathematician who drew attention to them. The identity $-(-B) = B$ is known as *double complementation*. Note how its proof made essential use of the hypothesis that $B \subseteq U$.

Exercise 1.4.3 (3) Show that whenever B is not a subset of U, then $U\backslash(U\backslash B) \neq B$.

Logic Box: Negation

You will have noticed that in our discussion of difference and complementation, there were a lot of *nots*. In other words, we made free use of the logical connective of *negation* in our reasoning. What is its logic? Like conjunction and disjunction, it has a truth table, which is a simple flip-flop.

α	$\neg\alpha$
1	0
0	1

The properties of difference and complementation stem, in effect, from the behaviour of negation used in defining them.

Alice Box: Relative Versus Absolute Complementation

Alice: There is something about this that I don't quite understand. As you define it, the complement $-B$ of a set B is always taken with respect to a given local universe U; it is $U - B$. But why not define it in absolute terms? Simply put the absolute complement of B to be the set of

(continued)

(continued)

all x that are not in B. In other words, take your U to be the set of everything whatsoever.

Hatter: A natural idea indeed, and that is more or less how things were understood in the early days of set theory. Unfortunately, it leads to unsuspected difficulties, indeed to contradiction, as was notoriously shown by Bertrand Russell at the beginning of the twentieth century.

Alice: What then?

Hatter: To avoid such contradictions, the standard approach as we know it today, called Zermelo-Fraenkel set theory (ZF for short), does not admit the existence of a universal set, that is, one containing as elements everything whatsoever. Nor a set of all sets. Nor does it admit the existence of the absolute complement of any set, containing as elements all those things that are not elements of a given set. For if it did, by union, it would also have to admit the universal set.

We should perhaps say a bit more about this than did the Hatter. In fact, there are some versions of set theory that do admit absolute complementation and the universal set. The best known, devised by the logician Quine, is called NF (short for 'New Foundations'). But to avoid contradiction, such systems must lose power in certain other respects, notably failing a ZF axiom scheme of 'separation'. This makes them rather annoying to use in practice, and they are not favoured by working mathematicians or computer scientists.

However, for the purposes of this book, none of that matters. For we are concerned with sets of items that need not themselves be regarded as sets, together with collections of those sets. In other words, all of our sets are of level 1 or 2. This may be continued to higher integer levels, which may in turn be made cumulative and even extended to levels represented by transfinite ordinal numbers (sorry, their explanation is beyond this text). It is only when one goes beyond all that, to sets without a specific ordinal level, that the problems of the universal set arise.

Thus, the absence of a universal set and of an operation of absolute complementation is not really troublesome for our work here. Whenever you feel that you need to have some elbow room, just look for a set that is sufficiently large to contain all the items that you are currently working on, and use it as your local universe for relative complementation.

1.5 Generalized Union and Intersection

It is time to go a little beyond the cosy world of 'flat' set theory, sets of level 1, and look at some constructions whose elements are themselves such sets, and thus of level 2. We begin with the operations of *generalized union and intersection*.

We know that when A_1, A_2 are sets, then we can form their union $A_1 \cup A_2$, whose elements are just those items that are in at least one of A_1, A_2. Evidently, we can repeat the operation, taking the union of that with another set A_3 giving us $(A_1 \cup A_2) \cup A_3$. We know from an exercise that this is independent of the order of assembly, that is, $(A_1 \cup A_2) \cup A_3 = A_1 \cup (A_2 \cup A_3)$. Thus, its elements are just those items that are elements of at least one of the three, and we might as well write it without brackets.

Clearly, we can do this any finite number of times, and so it is natural to consider doing it infinitely many times. In other words, if we have sets $A_1, A_2, A_3, \ldots$ we would like to consider a set $A_1 \cup A_2 \cup A_3 \cup \ldots$ whose elements are just those items in at least one of the A_i for $i \in \mathbf{N}^+$. To make the notation more explicit, we write this set as $\cup\{A_i: i \in \mathbf{N}^+\}$ or more compactly as $\cup\{A_i\}_{i \in \mathbf{N}+}$ or as $\cup_{i \in \mathbf{N}+} \{A_i\}$.

Quite generally, if we have a collection $\{A_i: i \in I\}$ of sets A_i, one for each element i of a fixed set I, we may consider the following two sets:

- $\cup_{i \in I}\{A_i\}$, whose elements are just those things that are elements of *at least one* of the A_i for $i \in I$. It is called the *union* of the sets A_i for $i \in I$.
- $\cap_{i \in I}\{A_i\}$, whose elements are just those things that are elements of *all* of the A_i for $i \in I$. It is called the *meet* (or *intersection*) of the sets A_i for $i \in I$.

When there is any possibility of confusing the signs for generalized union and intersection with those for the two-place versions, they are customarily written larger, as in the exercise below. They are natural generalizations, and their properties are similar. For example, we have de Morgan and distribution principles. These are the subject of the next exercise.

Alice Box: Sets, Collections, Families, Classes

Alice: One moment! Why do you refer to $\{A_i: i \in I\}$ as a *collection* while its elements A_i, and also its union $\cup_{i \in I}\{A_i\}$ and its intersection $\cap_{i \in I}\{A_i\}$, are called *sets*?

Hatter: The difference of words does not mark a difference of content. It is merely to make reading easier. The human mind has difficulty in processing phrases like 'set of sets', and even more difficulty with 'set of sets of sets . . . ', and the use of a word like 'collection' or 'class' helps keep us on track.

Alice: I think I have also seen the word 'family'.

Hatter: Here we should be a little careful. Sometimes the term 'family' is used rather loosely to refer to a set of sets. But strictly speaking, it is something different, a certain kind of function. So better not to use that term until it is explained in Chap. 3.

Exercise 1.5 (1) (with partial solution) From the definitions of general union and intersection, prove the following distribution and de Morgan principles. In the last two, complementation is understood to be relative to an arbitrary sufficiently large universe:

(a) $A \cap \cup_{i \in I}\{B_i\} = \cup_{i \in I}\,(A \cap B_i\}$ (distribution of intersection over general union)
(b) $A \cup \cap_{i \in I}\{B_i\} = \cap_{i \in I}\,(A \cup B_i\}$ (distribution of union over general intersection)
(c) $-\cup_{i \in I}\{A_i\} = \cap_{i \in I}\{-A_i\}$ (de Morgan)
(d) $-\cap_{i \in I}\{A_i\} = \cup_{i \in I}\{-A_i\}$ (de Morgan)

Solution to LHS ⊆ RHS Part of *(a)* This is a simple 'unpack, rearrange, repack' verification. Suppose $x \in$ LHS. Then, $x \in A$ and $x \in \cup_{i \in I}\{B_i\}$. From the latter, by the definition of general union, we know that $x \in B_i$ for some $i \in I$. So for this $i \in I$, we have $x \in A \cap B_i$, and thus again by the definition of general union, $x \in$ RHS, as desired.

1.6 Power Sets

Our next construction is a little more challenging. Let A be any set. We may form a new set, called *the power set* of A, written as $\mathcal{P}(A)$ or 2^A, consisting of all (and only) the subsets of A. In other words, $\mathcal{P}(A) = \{B: B \subseteq A\}$. This may seem like a rather exotic construction, but we will need it as early as Chap. 2 when working with relations.

Exercise 1.6 (1) (with solution) Let $A = \{1,2,3\}$. List all the elements of $\mathcal{P}(A)$. Using the list, define $\mathcal{P}(A)$ itself by enumeration. How many elements does $\mathcal{P}(A)$ have?

Solution The elements of $\mathcal{P}(A)$ are (beginning with the smallest and working our way up): ∅, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}. Thus, $\mathcal{P}(A) = \{\varnothing,\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\}, \{1,2,3\}\}$. Counting, we see that $\mathcal{P}(A)$ has 8 elements.

Warning: Do not forget the two 'extreme' elements of $\mathcal{P}(A)$: the smallest subset A, namely, ∅, and the largest one, namely, A itself. Also be careful with the curly brackets. Thus, 1, 2, 3 are *not* elements of $\mathcal{P}(A)$, but their singletons {1}, {2}, {3} are. When defining $\mathcal{P}(A)$ by enumeration, don't forget the outer brackets enclosing the entire list of elements. These may seem like pedantic points of punctuation, but if they are missed, then you can easily get into a dreadful mess.

All of the elements of $\mathcal{P}(A)$ are subsets of A, but some of them are also subsets of others. For example, the empty set is a subset of all of them. This may be brought out clearly by the following *Hasse diagram*, called after the mathematician who introduced it (Fig. 1.4).

Exercise 1.6 (2) (with partial solution) Draw Hasse diagrams for the power sets of each of ∅, {1}, {1,2}, {1,2,3,4}. How many elements does each of these power sets have?

Partial solution They have 1, 2, 4 and 16 elements, respectively.

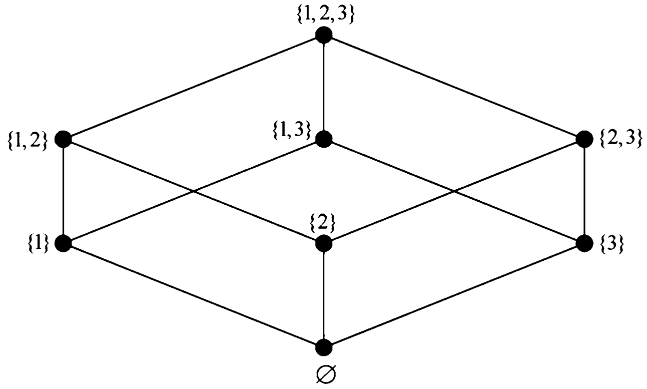

Fig. 1.4 Hasse diagram for $\mathscr{P}(A)$ when $A = \{1,2,3\}$

There is a pattern here. Quite generally, if A is a finite set with n elements, its power set $\mathscr{P}(A)$ has 2^n elements. Here is a rough but simple proof. Let $a_1, \ldots, a_n$ be the elements of A. Consider any subset $B \subseteq A$. For each a_i, there are two possibilities: either $a_i \in B$ or $a_i \notin B$. That gives us $2 \cdot 2 \cdot \ \ldots \ \cdot 2$ (n times) $= 2^n$ independent choices to determine which among $a_1, \ldots, a_n$ are elements of B, thus 2^n possible identities for B.

This fact is very important for computing. Suppose that we have a way of measuring the cost of a computation in terms of, say, time of calculation as a function of, say, the number of input items. It can happen that this measure increases in proportion to 2^n, that is, is of the form $k \cdot 2^n$ for some fixed k. This is known as *exponential growth*, and it is to be avoided like the plague, as it quickly leads to unfeasibly expensive calculations. For example, suppose that such a process is dealing with an input of 10 items. Now $2^{10} = 1{,}024$, which may seem reasonable. But if the input has 100 items to deal with, we have 2^{100} steps to be completed, which would take a very long time indeed (try writing it out in decimal notation).

In this chapter, we have frequently made use of *if* . . . *then* . . . (alias *conditional*) statements and also *iff* (alias *biconditional*) ones. It is time to fulfil a promise to Alice, to make their meanings clear. In mathematics, they are used in a very simple way. Like conjunction, disjunction and negation, they are truth-functional. The biconditional is perhaps the easier to grasp.

Logic Box: Truth Table for 'iff'

The biconditional α *if and only if* β, briefly written α *iff* β and in formal notation $\alpha \leftrightarrow \beta$, is true whenever α and β have the same truth-value, and false whenever they have opposite truth-values. Its table is as follows:

(continued)

(continued)

α	β	$\alpha \leftrightarrow \beta$
1	1	1
1	0	0
0	1	0
0	0	1

The conditional is rather more difficult for students to assimilate. It differs from the biconditional in its third row.

Logic Box: Truth Table for 'if'

The conditional *if* α *then* β, in formal notation $\alpha \rightarrow \beta$, is true in all cases except one – the 'disastrous combination': α true and β false.

α	β	$\alpha \rightarrow \beta$
1	1	1
1	0	0
0	1	1
0	0	1

From its table, it is clear that a statement $\alpha \rightarrow \beta$ is always true except in the second row. Comparing the two tables, it is clear that $\alpha \leftrightarrow \beta$ is true just when $\alpha \rightarrow \beta$ and its converse $\beta \rightarrow \alpha$ are both true.

Alice Box: The Truth Table for the Conditional

Alice: Well, at least you kept your promise! But I am not entirely satisfied. I see why the 'disastrous combination' makes *if* α *then* β false. But why do all the other combinations make it come out true? The two statements α and β may have nothing to do with each other, like 'London is in France' and 'kangaroos are fish'. These are both false, but it is strange to say that the statement 'if London is in France then kangaroos are fish' is true.

Hatter: Indeed, it *is* rather strange and, to be frank, in everyday life we do use *if . . . then . . .* in very subtle ways – much more so than any such table. But in mathematics, we use the conditional in the simplest possible manner, which is that given by the truth table. Moreover, it turns out to underlie all the other ones.

Once again, we should say a bit more than did the Hatter. Here is an example that may not entirely convince you but might at least make the truth table less strange. We know that all positive integers divisible by four are even. This is, in effect, a universally generalized conditional. It says: *For every positive integer n, if n is divisible by four*, *then it is even*. So for every particular choice that we make of a positive integer n, the statement, *if n is divisible by four then it is even*, comes out true. For example, the following three are all true:
If 8 is divisible by 4, then it is even.
If 2 is divisible by 4, then it is even.
If 9 is divisible by 4, then it is even.
But the first of these corresponds to the top row of our truth table (both components true), the second to the third row (α false, β true), and the last to the fourth row (both components false) – while in each of these three cases, the conditional is true. The following exercise, due to Dov Gabbay, will also help bring out the same point.

Exercise 1.6 (2) A shop hangs a sign in the window saying 'If you buy a computer, then you get a free printer'. The relevant government office suspects the shop of false advertising and sends agents disguised as customers to make purchases. Each agent reports on what happens. How many relevant configurations can emerge in these reports? Which would justify charging the manager with false advertising?

Finally, we recall the names of some familiar sets of numbers. We have already mentioned the set $\mathbf{N}^+ = \{1,2,3, \ldots\}$ of all positive integers. Some other number sets that we will need to refer to are the following:
$\mathbf{N} = \mathbf{N}^+ \cup \{0\}$, that is, the set consisting of zero and all the positive integers. This is called the set of the *natural numbers*. Warning: Some authors use this name to refer to the positive integers only. Be wary when you read.
$\mathbf{Z} = \{0, \pm 1, \pm 2, \pm 3, \ldots\}$, the set of all *integers* (positive, negative and zero).
$\mathbf{Z}^- = \{\ldots, -3, -2, -1\} = \mathbf{Z} \backslash \mathbf{N}$, which is the set of all *negative integers*.
$\mathbf{Q} = \{p/q \colon p,q \in \mathbf{Z}$ and $q \neq 0\}$, the set of all *rational numbers* (positive, negative and zero).
$\mathbf{R} =$ the set of all *real numbers*, also representable as the set of all numbers of the form $p + d_1 d_2 \ldots$ where $p \in \mathbf{Z}$ and $d_1 d_2 \ldots$ is an ending or unending decimal (series of digits from 0 to 9).

End-of-Chapter Exercises

Exercise 1.1: Boolean operations on sets We define the operation $A + B$ of *symmetric difference* (sometimes known as disjoint union) by putting $A + B = (A \backslash B) \cup (B \backslash A)$. Notations vary: Often $\oplus$ is used for this operation, sometimes Δ.

(a) Show that for any x, $x \in A + B$ iff x is an element of exactly one of A, B.
(b) Draw a Venn diagram for the operation (and use it to help intuition in what follows).

(c) Show that $A+B \subseteq A \cup B$.
(d) Show that $A+B$ is disjoint from $A \cap B$.
(e) Show that $A+B=(A \cup B) \backslash (A \cap B)$.
(f) For each of the following properties of $\cup$, check out whether or not it also holds for $+$, giving a proof or a counterexample as appropriate: (i) commutativity, (ii) associativity, (iii) distribution of $\cap$ over $+$, (iv) distribution of $+$ over $\cap$.
(g) Express $-(A+B)$ using union, intersection and complement.
(h) We have seen that each of intersection, union and difference corresponds to a truth-functional logical connective. To what connective mentioned in this chapter does symmetric difference correspond? Draw its truth table.

Exercise 1.2: Counting principles for Boolean operations When A is a finite set, we write $\#(A)$ for the number of elements that it contains. Use the definitions of the Boolean operations together with your knowledge of elementary arithmetic to verify the following for finite sets. They will all be used in the chapter on counting:
(a) $\#(A \cup B) \leq \#(A)+\#(B)$.
(b) Sometimes $\#(A \cup B) < \#(A)+\#(B)$ (give an example).
(c) When A, B are disjoint, then $\#(A \cup B)=\#(A)+\#(B)$. What does this tell you about $\#(A+B)$?
(d) When A, B are any finite sets (not necessarily disjoint), then $\#(A \cup B)=\#(A)+\#(B)-\#(A \cap B)$. This is known as the *rule of inclusion and exclusion*.
(e) $\#(A \cap B) \leq min(\#(A), \#(B))$. Here, $min(m,n)$ is whichever is the lesser of the integers m,n.
(f) Sometimes $\#(A \cap B) < min(\#(A), \#(B))$ (give an example).
(g) Formulate and verify a necessary and sufficient condition for the equality $\#(A \cap B) = min(\#(A), \#(B))$ to hold.
(h) $\#(\varnothing)=0$.
(i) $\#(A \backslash B)=\#(A)-\#(A \cap B)$.
(j) $\#(A+B)=\#(A)+\#(B)-2\#(A \cap B)$.
(k) When $B \subseteq A$, then $\#(A \backslash B)=\#(A)-\#(B)$.

Exercise 1.3: Generalized union and intersection
(a) Let $\{A_i\}_{i \in I}$ be any collection of sets. Show that for any set B, we have (i) $\cup\{A_i\}_{i \in I} \subseteq B$ iff $A_i \subseteq B$ for every $i \in I$, (ii) $B \subseteq \cap\{A_i\}_{i \in I}$ iff $B \subseteq A_i$ for every $i \in I$.
(b) Find a collection $\{A_i\}_{i \in \mathbf{N}}$ of non-empty sets with each $A_i \supset A_{i+1}$ but with $\cap\{A_i\}_{i \in I}$ empty.
(c) Why can't (b) be done for some *finite* collection $\{A_i\}_{i \in I}$ of non-empty sets?

Exercise 1.4: Power sets
(a) Show that whenever $A \subseteq B$, then $\mathcal{P}(A) \subseteq \mathcal{P}(B)$.
(b) True or false: $\mathcal{P}(A \cap B)=\mathcal{P}(A) \cap \mathcal{P}(B)$? Give a proof or a counterexample.
(c) True or false: $\mathcal{P}(A \cup B)=\mathcal{P}(A) \cup \mathcal{P}(B)$? Give a proof or counterexample.

Selected Reading

For a very gentle introduction to sets, which nevertheless takes the reader up to an outline of the Zermelo-Fraenkel axioms for set theory, see:

Bloch ED (2011) Proofs and fundamentals: a first course in abstract mathematics, 2nd edn. Springer, New York, chapter 3

A classic of beautiful exposition, but short on exercises (readers should instead verify the claims made in the text):

Halmos PR (2001) Naive set theory, new edn. Springer, New York, chapters 1–5, 9

The present material is covered with lots of exercises in:

Lipschutz S (1998) Set theory and related topics, Schaum's outline series. McGraw Hill, New York, chapters 1–2 and 5.1–5.3

2 Comparing Things: Relations

Abstract

Relations play an important role in computer science, both as tools of analysis and for representing computational structures such as databases. In this chapter, we introduce the basic concepts you need to master in order to work with them.

We begin with the notions of an *ordered pair* (and more generally, ordered *n-tuple*) and the *Cartesian product* of two or more sets. We then consider operations on relations, notably those of forming the *converse*, *join* and *composition* of relations, as well as some other operations that make relations interact with sets, notably the *image* and the *closure* of a set under a relation.

We also explore two of the main jobs that relations are asked to carry out: to classify and to order. For the former, we explain the notion of an *equivalence relation* (reflexive, transitive, symmetric) over a set and how it corresponds to the notion of a *partition* of the set. For the latter, we look first of all at several kinds of *reflexive order*, and then at their *strict parts*.

2.1 Ordered Tuples, Cartesian Products and Relations

What do the following have in common: one car overtaking another, a boy loving a girl, one tree being shadier than another, an integer dividing another, a point lying between two others and a student exchanging one book for another with a friend?

They are all examples of *relations* involving at least two items – in some instances three (one point between two others), four (the book exchange) or more. Often, they involve actions, intentions, the passage of time and causal connections; but in mathematics and computer science, we abstract from all those features and work with a very basic, stripped-down concept. To explain what it is, we begin with the notions of an ordered tuple and Cartesian product.

D. Makinson, *Sets, Logic and Maths for Computing*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-2500-6_2,

2.1.1 Ordered Tuples

Recall from the preceding chapter that when a set has exactly one element, it is called a *singleton*. When it has exactly two distinct elements, it is called a *pair*. For example, the set {7,9} is a pair, and it is the same as the pair {9,7}. We have $\{7,9\} = \{9,7\}$ because the order is irrelevant: the two sets have exactly the same elements.

An *ordered pair* is like a (plain, unordered) pair except that order matters. To highlight this, we use a different notation. The ordered pair whose first element is 7 and whose second element is 9 is written as (7,9) or, in older texts, as <7,9>. It is distinct from the ordered pair (9,7) although they have exactly the same elements: $(7,9) \neq (9,7)$, because the elements are considered in a different order.

Abstracting from this example, the *criterion for identity* of ordered pairs is as follows: $(x_1,x_2) = (y_1,y_2)$ iff both $x_1 = y_1$ and $x_2 = y_2$. This contrasts with the criterion for identity of plain sets: $\{x_1,x_2\} = \{y_1,y_2\}$ iff the left and right hand sets have exactly the same elements, which, it is not difficult to show, holds iff *either* ($x_1 = y_1$ and $x_2 = y_2$) *or* ($x_1 = y_2$ and $x_2 = y_1$).

More generally, the criterion for identity of two ordered n-tuples $(x_1,x_2, \ldots ,x_n)$ and $(y_1,y_2, \ldots ,y_n)$ is as you would expect: $(x_1,x_2, \ldots ,x_n) = (y_1,y_2, \ldots ,y_n)$ iff $x_i = y_i$ for all i from 1 to n.

Exercise 2.1.1 Check in detail the claim that $\{x_1,x_2\} = \{y_1,y_2\}$ iff either ($x_1 = y_1$ and $x_2 = y_2$) or ($x_1 = y_2$ and $x_2 = y_1$). *Hint*: It helps to break your argument into cases.

Alice Box: Ordered Pairs

Alice: Isn't there something circular in all this? You promised that relations will be used to build a theory of order, but here you are defining the concept of a relation by using the notion of an ordered pair, which already involves the concept of order!

Hatter: A subtle point, and a good one! But I would call it a spiral rather than a circle. We need just a *rock-bottom* kind of order – no more than the idea of one thing coming before another – in order to understand what an ordered pair is. From that we can build a very sophisticated theory of the various kinds of order that relations can create.

2.1.2 Cartesian Products

With this in hand, we can introduce the notion of the Cartesian product of two sets. If A, B are sets, then their *Cartesian product*, written $A \times B$ and pronounced 'A cross B' or 'A by B', is defined as follows: $A \times B = \{(a,b): a \in A \text{ and } b \in B\}$.

In English, $A \times B$ is the set of all ordered pairs whose first term is in A and whose second term is in B. When $B = A$, so that $A \times B = A \times A$, it is customary to write it as A^2, calling it 'A squared'. The operation takes its name from René Descartes who, in the seventeenth century, made use of the Cartesian product $\mathbf{R}^2$ of the set $\mathbf{R}$ of all real numbers. His seminal idea was to represent each point of a plane by an ordered pair (x,y) of real numbers and use this representation to solve geometric problems by algebraic methods. The set $\mathbf{R}^2$ is called the *Cartesian plane*.

A very simple concept – but be careful, it is also easy to trip up! Take note of the *and* in the definition, but don't confuse Cartesian products with intersections. For example, if A, B are sets of numbers, then $A \cap B$ is also a set of *numbers*; but $A \times B$ is a set of *ordered pairs of numbers*.

Exercise 2.1.2 (1) (with solution) Let $A = \{\text{John, Mary}\}$ and $B = \{1,2,3\}$, $C = \varnothing$. What are $A \times B$, $B \times A$, $A \times C$, $C \times A$, A^2, B^2? Are any of these identical with others? How many elements in each?

Solution
$A \times B = \{(\text{John},1), (\text{John},2), (\text{John},3), (\text{Mary},1), (\text{Mary},2), (\text{Mary},3)\}$
$B \times A = \{(1,\text{John}), (2,\text{John}), (3,\text{John}), (1,\text{Mary}), (2,\text{Mary}), (3,\text{Mary})\}$
$A^2 = \{(\text{John},\text{John}), (\text{John}, \text{Mary}), (\text{Mary},\text{John}), (\text{Mary}, \text{Mary})\}$
$B^2 = \{(1,1), (1, 2), (1,3), (2,1), (2, 2), (2,3), (3,1), (3, 2), (3,3)\}$

Of these, $A \times C = \varnothing = C \times A$, but that is all; in particular $A \times B \neq B \times A$.
Counting: $\#(A \times B) = 6 = \#(B \times A)$, $\#(A \times C) = 0 = \#(C \times A)$, $\#(A^2) = 4$, $\#(B^2) = 9$.

Comment Note that also $A \times B = \{(\text{John},1), (\text{Mary},1), (\text{John},2), (\text{Mary},2), (\text{John},3), (\text{Mary},3)\}$. Within the curly brackets, we are enumerating the elements of a *set*, so we can write them in any order we like; but within the round brackets, we are enumerating the terms of an *ordered pair* (or ordered n-tuple), so there the order is vital.

From the exercise, you may already have guessed a general counting principle for the Cartesian products of finite sets: $\#(A \times B) = \#(A) \cdot \#(B)$, where the dot stands for ordinary multiplication. Here is a proof. Let $\#(A) = m$ and $\#(B) = n$. Fix any element $a \in A$. Then there are n different pairs (a,b) with $b \in B$. And when we fix a different $a' \in A$, then the n pairs (a',b) will all be different from the pairs (a,b) since they differ on their first terms. Thus, there are $n + n + \cdots + n$ (m times), that is, $m \cdot n$ pairs altogether in $A \times B$.

Thus, although the operation of forming the Cartesian product of two sets is *not* commutative (i.e. we may have $A \times B \neq B \times A$), the operation of counting the elements of the Cartesian product *is* commutative: always $\#(A \times B) = \#(B \times A)$.

Exercise 2.1.2 (2) (with partial solution)
(a) Show that when $A \subseteq A'$ and $B \subseteq B'$, then $A \times B \subseteq A' \times B'$.
(b) Show that when both $A \neq \varnothing$ and $B \neq \varnothing$, then $A \times B = B \times A$ iff $A = B$.

Solution to (b) Suppose $A \neq \varnothing$ and $B \neq \varnothing$. We need to show that $A \times B = B \times A$ iff $A = B$. We do this in two parts. First, we show that if $A = B$, then $A \times B = B \times A$. Suppose $A = B$. By the supposition, $A \times B = A^2$ and also $B \times A = A^2$, so that $A \times B = B \times A$ as desired. Next, we show the converse, that if $A \times B = B \times A$, then $A = B$. The easiest way to do this is by showing the *contrapositive*: if $A \neq B$, then $A \times B \neq B \times A$. Suppose $A \neq B$. Then at least one of $A \subseteq B$, $B \subseteq A$ fails. We consider the former case; the latter is similar. Then there is an $a \in A$ with $a \notin B$. By supposition, $B \neq \varnothing$, so there is a $b \in B$. Thus, $(a,b) \in A \times B$, but since $a \notin B$, $(a,b) \notin B \times A$. Thus, $A \times B \neq B \times A$, as desired.

The solution to this exercise is instructive in two respects. In the first place, it uses a 'divide and rule' strategy, breaking the problem down into two component parts and tackling them one by one. In particular, in order to prove an *if and only if* statement (also known as a *biconditional*), it is often convenient to do it in two parts: first prove the *if* in one direction, and then prove it in the other. As we saw in the preceding chapter, these are not the same; they are called *converses* of each other.

In the second place, the exercise illustrates the tactic of proving by contraposition. Suppose we want to prove a statement *if* α *then* β. As mentioned earlier, the most straightforward way of tackling this is to suppose α and drive towards β. But that does not always give the most transparent proof. Sometimes it is better to suppose *not*-β and head for *not*-α. That is what we did in the example: to prove that if $A \times B = B \times A$, then $A = B$ for non-empty sets A,B, we supposed $A \neq B$ and showed $A \times B \neq B \times A$.

Why is proof by contraposition legitimate? It is because the two conditionals $\alpha \rightarrow \beta$ and $\neg\beta \rightarrow \neg\alpha$ are equivalent, as can be seen from their truth tables in the preceding chapter. How can the tactic help? Often, suppositions like $A \neq B$ give us something to 'grab hold of'. In our example, it tells us that there is an a with $a \in A$, $a \notin B$ (or conversely); we can then consider such an a, and start reasoning about it. This pattern occurs quite often.

Exercise 2.1.2 (3)

(a) An example from arithmetic of proving a biconditional via its two parts. Show that for any positive integers m,n, we have $n = m$ iff each of m,n divides the other.

(b) An example from arithmetic of proving via the contrapositive. Show that for any two real numbers x,y, if $x \cdot y$ is irrational, then at least one of x,y is irrational.

2.1.3 Relations

Let A,B be any sets. A *binary relation from A to B* is defined to be any subset of the Cartesian product $A \times B$. It is thus any set of ordered pairs (a,b) such as $a \in A$ and $b \in B$. A binary relation is thus fully determined by the ordered pairs that it covers. It does not matter how these pairs are presented or described. It is customary to use

R, S, ... as symbols standing for relations. As well as saying that the relation is from A to B, one also says that it is *over* $A \times B$.

From the definition, it follows that in the case that $A = B$, a binary relation from A to A is any set of ordered pairs (a,b) such as both $a,b \in A$. It is thus a relation over A^2, but informally we often abuse language a little and describe it as a relation *over* A.

Evidently, the notion may be generalized to any number of places. Let $A_1, \ldots ,A_n$ be sets ($n \geq 1$). An n-place relation over $A_1 \times \ldots \times A_n$ is defined to be any subset of $A_1 \times \ldots \times A_n$. In other words, it is any set of n-tuples $(a_1, \ldots ,a_n)$ with each $a_i \in A_i$. Binary relations are thus 2-place relations. As a limiting case of the more general notion, when $n = 1$ we have 1-place relations over A, which may be identified with subsets of A.

In this chapter, we will be concerned mainly with binary relations, and when we speak of a relation without specifying the number of places, we will mean a binary one.

Exercise 2.1.3 (1) (with solution) Let A, B be the sets given in Exercise 2.1.2 (1).

(a) Which of the following are (binary) relations from A to B: (i) {(John,1), (John,2)}, (ii) {(Mary,3), (John,Mary)}, (iii) {(Mary,2), (2,Mary)}, (iv) {(John,3), (Mary,4)}, (v) ({Mary,1}, {John,3})?
(b) What is the largest relation from A to B? What is the smallest?
(c) Identify (by enumeration) three more relations from A to B.
(d) How many relations are there from A to B?

Solution

(a) Only (i) is a relation from A to B. (ii) is not, because Mary is not in B. (iii) is not, because 2 is not in A. (iv) is not, because 4 is not in B. (v) is not, because it is a an ordered pair of sets, not a set of ordered pairs.
(b) $A \times B$ is the largest, $\varnothing$ is the smallest, in the sense that $\varnothing \subseteq R \subseteq A \times B$ for every relation R from A to B.
(c) For brevity, we can choose three *singleton relations*: {(John,1)}, {(John,2)}, {(John,3)}.
(d) Since $\#(A) = 2$ and $\#(B) = 3$, $\#(A \times B) = 2{\cdot}3 = 6$. From the definition of a relation from A to B it follows that the set of all relations from A to B is just $\mathcal{P}(A \times B)$, and by a counting principle in the chapter on sets, $\#(\mathcal{P}(A \times B)) = 2^{\#(A \times B)} = 2^6 = 64$.

When R is a relation from A to B, we call the set A a *source* of the relation, and B a *target*. Sounds simple enough, but care is advised. As already noted in an exercise, when $A \subseteq A'$ and $B \subseteq B'$, then $A \times B \subseteq A' \times B'$, so when R is a relation from A to B, then it is also a relation from A' to B'. Thus, the source and target of R, in the above sense, are not unique: a single relation will have indefinitely many sources and targets.

For this reason, we also need terms for the least possible source and target of R. We define the *domain* of R to be the set of all a such that $(a,b) \in R$ for some b, writing briefly $dom(R) = \{a$: there is a b with $(a,b) \in R\}$. Likewise we define $range(R) = \{b$: there is an a with $(a,b) \in R\}$. Clearly, whenever R is a relation from A to B, then $dom(R) \subseteq A$ and $range(R) \subseteq B$.

> *Warning*: You may occasionally see the term 'codomain' contrasting with 'domain'. But care is needed, as the term is sometimes used broadly for 'target', sometimes more specifically for 'range'. In this book, we will follow a fairly standard terminology, with *domain* and *range* defined as above, and *source*, *target* for any supersets of them.

Exercise 2.1.3 (2)

(a) Consider the relation $R = \{(1,7), (3,3), (13,11)\}$ and the relation $S = \{(1,1), (3,11), (13,12), (15,1)\}$. Identify $dom(R)$, $range(R)$, $dom(S)$, $range(S)$.

(b) The *identity relation* I_A over a set A is defined by putting $I = \{(a,a)$: $a \in A\}$. Identify $dom(I_A)$ and $range(I_A)$.

(c) Identify $dom(A \times B)$, $range(A \times B)$.

Since relations are sets (of ordered pairs or tuples), we can apply to them all the concepts that we developed for sets. In particular, it makes sense to speak of one relation R being *included* in another relation S: every tuple that is an element of R is an element of S. In this case, we also say that R is a *subrelation* of S. Likewise, it makes sense to speak of the *empty relation*: it is the relation that has no elements, and it is unique. It is thus the same as the empty set, and can be written $\varnothing$.

Exercise 2.1.3 (3)

(a) Use the definitions to show that the empty relation is a subrelation of every relation.

(b) Identify $dom(\varnothing)$, $range(\varnothing)$.

(c) What would it mean to say that two relations are disjoint? Give an example of two disjoint relations over a small finite set A.

2.2 Tables and Digraphs for Relations

In mathematics, rigour is important, but so is intuition. The two should go hand in hand. One way of strengthening one's intuition is to use graphic representations. This is particularly so in the case of binary relations. For sets in general, we used Euler and Venn diagrams; for the more specific case of relations, tables and arrow diagrams are helpful.

Table 2.1 Table for a relation

R	1	2	3
John	1	0	1
Mary	0	1	1

2.2.1 Tables

Let's go back to the sets $A = \{\text{John,Mary}\}$ and $B = \{1,2,3\}$ of earlier exercises. Consider the relation $R = \{(\text{John},1), (\text{John},3), (\text{Mary},2), (\text{Mary},3)\}$. How might we represent R by a table?

We need two rows and three columns for the elements of A and B, respectively. Each cell in this table is uniquely identified by its coordinate (a,b), where a is the element for the row and b is the element for the column. In the cell write 1 (for 'true') or 0 (for 'false') according as the ordered pair (a,b) is or is not an element of the relation. For the R chosen above, this gives us Table 2.1.

Tabular representations of relations are particularly useful when dealing with databases because good software is available for writing and manipulating them. Moreover, when a table has the same number of columns as rows, geometrical operations such as folding along the diagonal can also reveal interesting features.

Exercise 2.2.1

(a) Let $A = \{1,2,3,4\}$. Draw tables for each of the following relations over A^2: (i) $<$, (ii) $\leq$, (iii) $=$, (iv) $\varnothing$, (v) A^2.

(b) In each of the four tables, draw a line from the top left cell (1,1) to the bottom right cell (4,4). This is called the *diagonal* of a relation over A^2. Comment on the contents of the diagonal in each of the four tables.

(c) Imagine folding the table along the diagonal. Comment on any symmetries that become visible.

2.2.2 Digraphs

Another way of representing binary relations, less suitable for software implementation but more friendly to humans, is by means of arrow diagrams known as *directed graphs* or more briefly *digraphs*. The idea is simple, at least in the finite case. Given a relation from A to B, mark a point for each element of $A \cup B$, labelling it with a name if desired. Draw an arrow from one point to another just when the first stands in the relation to the second. When the source A and target B of the relation are not the same, it can be useful to add into the diagram circles for the sets A and B.

In the example considered for the table above, the digraph comes out as in Fig. 2.1.

The Hasse diagram between the subsets of $\{1, 2, 3\}$ that we presented in Chap. 1 can be seen as a digraph for the relation of *being immediately included in*. This holds from a set B to a set C iff $B \subset C$, but there is no X with $B \subset X \subset C$. However, the

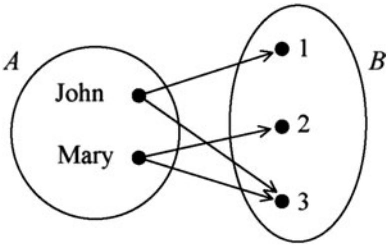

Fig. 2.1 Diagram for a relation

Hasse diagram uses plain lines rather than arrows, with the convention that these are always read with subset below and superset above.

Given a diagram for a relation, we can use it to read off its reflexive and transitive closure. In particular, given the Hasse diagram for the relation of B being *immediately included* in C, we can read off the relation of B being *included* in C: it holds iff there is an ascending path from B to C (including the one-element path). This is much more economical than drawing a digraph for the entire relation of inclusion.

Exercise 2.2.2 Take from Chap. 1 the Hasse diagram for immediate inclusion between subsets of the set $A = \{1,2,3\}$, and compare it with the digraph for the entire relation of inclusion. How many links in each?

Diagrams are valuable tools to illustrate situations and stimulate intuitions. They can often help us think up counterexamples to general claims, and they can sometimes be used to illustrate a proof, making it much easier to follow. However, they have their limitations. In particular, it should be remembered that *a diagram can never itself constitute a proof of a general claim.*

2.3 Operations on Relations

Since relations are sets, we can carry out on them all the Boolean operations for sets that we learned in the preceding chapter, provided we keep track of the sources and targets. Thus, if R and S are relations from the same source A to the same target B, their intersection $R \cap S$, being the set of all ordered pairs (x,y) that are simultaneously elements of R and of S, will also be a relation from A to B. Likewise for the union $R \cup S$ and also for complement $-R$ with respect to $A \times B$. Note that just as for sets, $-R$ is really a difference $(A \times B) - R$ and so depends implicitly on the source A and target B as well as on R itself.

As well as the Boolean operations, there are others that arise only for relations. We describe some of the most important ones.

2.3.1 Converse

The simplest is that of forming the *converse* (alias *inverse*) R^{-1} of a relation. Given a relation R, we define R^{-1} to be the set of all ordered pairs (b,a) such that $(a,b) \in R$.

> *Warning*: Do not confuse this with complementation! There is nothing negative about conversion despite the standard notation R^{-1}: we are simply *reversing the direction* of the relation. The complement of *loves* is *doesn't love*, but its converse is *is loved by*.

Exercise 2.3.1 (1) (with solutions)

(a) Let A be the set of natural numbers. What are the converses of the following relations over A: (i) less than, (ii) less than or equal to, (iii) equal to.

(b) Let A be a set of people. What are the converses of the following relations over A: (i) being a child of, (ii) being a descendant of, (iii) being a daughter of, (iv) being a brother of, (v) being a sibling of, (vi) being a husband of.

Solutions

(a) (i) greater than, (ii) greater than or equal to, (iii) equal to.

(b) (i) being a parent of, (ii) being an ancestor of, (iii) having as a daughter, (iv) having as a brother, (v) being a sibling of, (vi) being a wife of.

Comments In group (a) we already have examples of how the converse of a relation may be disjoint from it, overlap with it or be identical with it.

Some of the examples in group (b) are a little tricky. Note that the converse of being a brother of is *not* being a brother of: when a is a brother of b, b may be female and so a sister of a.

Sometimes, ordinary language has a single word for a relation and another single word for its converse. This is the case for (i) (child/parent) (ii) (ancestor/descendant) and (vi) (husband/wife). But it is not always so: witness (iii) and (iv) where we have to use special turns of phrase: daughter/having as daughter, brother/having as brother. The range of words available for family relations differs from one language to another.

Exercise 2.3.1 (2)

(a) What does the converse of a relation look like from the point of view of a digraph for the relation? And from the point of view of a table for it?

(b) Show that $dom(R^{-1}) = range(R)$ and $range(R^{-1}) = dom(R)$.

(c) Show that (i) $(R^{-1})^{-1} = R$, (ii) $(R \cap S)^{-1} = R^{-1} \cap S^{-1}$, (iii) $(R \cup S)^{-1} = R^{-1} \cup S^{-1}$. Compare these equalities with those for complementation.

Alice Box: Converse of an n-Place Relation

Alice: Does the notion of conversion make sense for relations with more than two places?

Hatter: It does, but there we have not just one operation but several. For simplicity, take the case of a three-place relation R, whose elements are ordered triples (a,b,c). Then, there is an operation that converts the first two, giving triples (b,a,c), and an operation converting the last two, giving triples (a,c,b).

Alice: And one switching the first and the last, giving triples (c,b,a)?

Hatter: Indeed. However, once we have some of these operations we can get others by iterating them. For example, your operation may be obtained by using the first two as follows: from (a,b,c) to (b,a,c) to (b,c,a) to (c,b,a), using the first, second and first operations. We could also get the first operation, say, by iterating the second and third . . .

Alice: Stop there, I've got the idea.

2.3.2 Join, Projection, Selection

Imagine that you are in the personnel division of a firm, and that you are in charge of a database recording the identity numbers and names of employees. Your colleague across the corridor is in charge of another database recording their names and telephone extensions. Each of these databases may be regarded as a binary relation. In effect, we have a large set A of allowable identity numbers (e.g. any six digit figure), a set B of allowable names (e.g. any string of at most twenty letters and spaces, with no space at beginning or end and no space immediately following another one), and a set C of allowable telephone extension numbers (e.g. any figure of exactly four digits). Your database is a subset R of $A \times B$, your colleague's database is a subset S of $B \times C$; they are thus both relations. These relations may have special properties. For example, it may be required that R cannot associate more than one name with any given identity number, although S may legitimately associate more than one telephone extension to a given name. We will not bother with these special properties at the moment, leaving them to the chapter on functions.

The manager may decide to merge these two databases into one, thus economizing on the staff needed to maintain them and liberating one of you for other tasks (or for redundancy). What would the natural merging be? A three-place relation over $A \times B \times C$ consisting of all those triples (a,b,c) such that $(a,b) \in R$ and $(b,c) \in S$. This is clearly an operation on relations, taking two binary relations with a common middle set (target of one, source of the other) to form a three-place relation. The resulting relation/database gives us all the data of the two components, with no loss of information.

Of course, in practice, databases make use of relations with more than just two places, and the operation that we have described may evidently be generalized to cover them. Let R be a relation over $A_1 \times \ldots \times A_m \times B_1 \times \ldots \times B_n$

and let S be a relation over $B_1 \times \ldots \times B_n \times C_1 \times \ldots \times C_p$. We define the *join* of R and S, written *join*(R,S), to be the set of all those $(m+n+p)$-tuples $(a_1,\ldots,a_m,b_1,\ldots,b_n,c_1,\ldots,c_p)$ such that $(a_1,\ldots,a_m,b_1,\ldots,b_n) \in R$ and $(b_1,\ldots,b_n,c_1,\ldots,c_p) \in S$.

Another terminological warning: while database theorists (and this book) call this operation 'join'; set theorists and algebraists sometimes use the same word as a synonym for the simple union of sets.

There are several further operations of database theory that cannot be obtained from conversion and join alone. One is *projection*. Suppose $R \subseteq A_1 \times \ldots \times A_m \times B_1 \times \ldots \times B_n$ is a database. We may *project R* onto its first m places, forming a relation whose elements are those m-tuples $(a_1,\ldots,a_m)$ such that there are $b_1,\ldots,b_n$ with $(a_1,\ldots,a_m,b_1,\ldots,b_n) \in R$.

Another database operation is *selection* (alias *restriction* in the language of set theorists and algebraists). Again, let $R \subseteq A_1 \times \ldots \times A_m \times B_1 \times \ldots \times B_n$ be a database and let $C_1,\ldots,C_m$ be subsets of $A_1,\ldots,A_m$, respectively (i.e. $C_i \subseteq A_i$ for each $i \leq m$). We may *select* (or restrict the relation to) the subsets by taking its elements to be just those $(m+n)$-tuples $(c_1,\ldots,c_m,b_1,\ldots,b_n)$ such that $(c_1,\ldots,c_m,b_1,\ldots,b_n) \in R$ and each $c_i \in C_i$. Here we have selected subsets from the first m arguments. We could equally well have selected from the last n, or from any others. In database contexts, the subsets C_i will often be singletons $\{c_i\}$.

Notice that conversion and selection both leave the *arity* (i.e. number of places) of a relation unchanged, while projection diminishes and join increases it. By combining them, one can do a lot of manipulation.

Exercise 2.3.2 (with partial solution)

(a) Formulate the definition of *join*(R,S) for the case that R is a two-place relation over $A \times B$ and S is a three-place relation over $B \times C \times D$.

(b) Formulate the definition of projection for the case of a two-place relation $R \subseteq A \times B$ and that we project to (i) A, (ii) B.

(c) Formulate the definition of selection (alias restriction) in the case that we restrict a two-place relation $R \subseteq A \times B$ to (i) a subset A' of A, (ii) a subset B' of B, (iii) both A' and B'.

(d) Let $A = \{2, 3, 13\}$, $B = \{12, 15, 17\}$, $C = \{11, 16, 21\}$. Put $R = \{(a,b)$: $a \in A$, $b \in B$, a divides $b\}$, $S = \{(b,c)$: $b \in B$, $c \in C$, b has the same parity as $c\}$. (i) Enumerate each of R and S. (ii) Enumerate *join*(R,S). (iii) Project R onto A, S onto C, and *join*(R,S) onto $A \times C$. (iv) Restrict R to the odd elements of B, S to the even elements of B, and *join*(R,S) to the prime elements of B.

Partial Solution

(a) *join*(R,S) is the set of all 4-tuples (a,b,c,d) such that $(a,b) \in R$, and $(b,c,d) \in S$.

(b) In part: the projection of R to A is the set of all $a \in A$ such that there is a $b \in B$ with $(a,b) \in R$.

(c) In part: the selection of A' as source for R is the set of all pairs $(a,b) \in R$ with $a \in A'$.

2.3.3 Composition

While the join operation is immensely useful for database manipulation, it does not occur very often in everyday language. But there is a rather similar operation that children learn at a very young age – as soon as they can recognize members of their extended family. It was first studied in the nineteenth century by Augustus de Morgan who called it *relative product*. Nowadays it is usually called *composition*. We consider it for the most commonly used case; the composition of two two-place relations.

Suppose we are given relations $R \subseteq A \times B$ and $S \subseteq B \times C$, with the target of the first the same as the source of the second. We have already defined their join as the set of all triples (a,b,c) such that $(a,b) \in R$ and $(b,c) \in S$. But we can also define their *composition* $S \circ R$ as the set of all ordered pairs (a,c) such *that there is some* x with both $(a,x) \in R$ and $(x,c) \in S$. Notice that the arity of the composition is one less than that of the join.

For example, if F is the relation of 'a father of' and P is the relation of 'a parent of', then $P \circ F$ is the relation consisting of all ordered pairs (a,c) such that there is some x with $(a,x) \in F$ and $(x,c) \in P$, that is, all ordered pairs (a,c) such that for some x, a is a father of x and x is a parent of c. It is thus the relation of a being a father of a parent of c, that is, of a being a grandfather of c.

Alice Box: Notation for Composition of Relations

Alice: That looks funny, the wrong way round. Wouldn't it be easier to write the composition, as you have defined it, with the letters F and P the other way round as $F \circ P$. Then, they would follow the same order as they occur in the phrase 'a father of a parent of', which is also the order of the predicates when we say '$(a,x) \in F$ and $(x,c) \in P$' in the definition?

Hatter: Indeed, it would correspond more directly with the way that ordinary language handles composition, and early authors working in the theory of relations did it that way.

Alice: Why the switch?

Hatter: To bring it into agreement with the standard way of doing things in the theory of functions. As we will see in the next chapter, a function can be defined as a special kind of relation, and notions such as composition for functions turn out to be the same as for relations; so it is best to use the same notation in the two contexts, and the notation for functions won out.

Alice: I'm afraid that I'll always mix them up!

Hatter: The best way to avoid that is to write the definition down again before each exercise involving composition of relations. To help remember the definition correctly, note that the arrangement of R and S is symmetric around the centre of the definition.

Logic Box: Existential Quantifier

Our definition of the composition of two relations made use of the phrase 'there is an x such that ...'. This is known as the *existential quantifier*, and is written $\exists x(\ldots)$. For example, 'There is an x with both $(a,x) \in R$ and $(x,c) \in S$ and is written as $\exists x((a,x){\in}R \wedge (x,c){\in}S)$, or more briefly as $\exists x(Rax \wedge Sxc)$'.

The existential quantifier $\exists$ has its own logic, which we will describe in a later chapter along with the logic of its companion, the universal quantifier $\forall$. In the meantime, we will simply adopt the notation $\exists x(\ldots)$ as a convenient shorthand for the longer phrase 'there is an x such that ...', and likewise use $\forall x(\ldots)$ to abbreviate 'for every x, ... holds'.

The existential and universal quantifiers always come with a variable attached. However, ordinary speech does not contain variables; even in written text variables are rare outside mathematical contexts. But colloquial language manages to convey the notion in other ways. For example, when we say 'some composers are poets', we are not using a variable, but we are in effect saying that there is an x such that x is both a composer and a poet, which the set-theorist would write as $\exists x(x \in C \wedge x \in P)$ and the logician as $\exists x(Cx \wedge Px)$. Pronouns can also be used like variables. For example, when we say 'if there is a free place, I will reserve it', the 'it' is doing the work of a variable. However, once we begin to formulate more complex statements involving several different quantifications, it can be difficult to be precise without using variables explicitly.

Exercise 2.3.3 (1) (with solution) Let A be the set of people, and P, F, M, S, B the relations over A of 'parent of', 'father of', 'mother of', 'sister of' and 'brother of', respectively. Describe *exactly* the following relative products: (a) $P{\circ}P$, (b) $M{\circ}F$, (c) $S{\circ}P$, (d) $B{\circ}B$.

Warnings: (1) Be careful about order. (2) In some cases there will be a handy word in English for just the relation, but in others it will have to be described in a more roundabout (but still precise) way.

Solution and Commentary

(a) $P{\circ}P =$ 'grandparent of'. *Reason*: a is a grandparent of c iff there is an x such that a is a parent of x and x is a parent of c.

(b) $M{\circ}F =$ 'maternal grandfather of'. *Reason*: a is a maternal grandfather of c iff there is an x such that a is a father of x and x is a mother of c.

Comments on (b)

There are two common errors here: (1) getting the order wrong and (2) rushing to the answer 'grandfather of', since a father of a mother is always a grandfather. But there is another way of being a grandfather, namely, by being a father of a father, so that relation is too broad.

(c) $S \circ P =$ ‘parent of a sister of’. *Reason*: a is a parent of a sister of c iff there is an x such that a is a parent of x and x is a sister of c.

Comments on (c)

This one is tricky. When there is an x such that a is a parent of x and x is a sister of c, then a is also a parent of c, which tempts one to rush into the answer ‘parent of’. But that relation is also too broad, because a may also be a parent of c when c has no sisters! English has no single word for the relation $S \circ P$; we can do no better than use a circumlocution such as ‘parent of a sister of’.

(d) $B \circ B =$ ‘brother of a brother of’.

Comments on (d)

Again, English has no single word for the relation. One is tempted to say that $B \circ B =$ ‘brother’. But that answer is both too broad (in one respect) and too narrow (in another respect). Too broad because a may be a brother of c without being a brother of a brother x of c: there may be no third brother to serve as the x. Too narrow because a may be a brother of x and x a brother of c, without a being a brother of c, since a may be the same person as c! In this example, again, we can do little better than use the phrase ‘brother of a brother of’.

Different languages categorize family relations in different ways, and translations are often only approximate. There may be a single term for a certain complex relation in one language but none in another. Languages of tribal communities are richest in this respect, and those of modern urban societies are poorest. Even within a single language, there are sometimes ambiguities. In English, for example, let P be the relation of ‘parent of’ and S the relation of ‘sister of’. Then, $P \circ S$ is the relation of ‘being a sister of a parent of’. This is certainly a *subrelation* of the relation of being an aunt of, but are the two relations identical? That depends on whether you include aunts by marriage, that is, whether you regard the wives of the brothers of your parents as your aunts. If you *do* include them under the term, then $P \circ S$ is a proper subrelation of the relation of aunt. If you *don't* include them, then it is the whole relation, that is, $P \circ S =$ ‘aunt’.

Exercise 2.3.3 (2)

(a) Show that $(S \cup R) \circ T = (S \circ T) \cup (R \circ T)$.

(b) Show that $(S \circ R)^{-1} = R^{-1} \circ S^{-1}$.

2.3.4 Image

The last operation that we consider in this section records what happens when we apply a relation to the elements of a set. Let R be any relation from set A to set B, and let $a \in A$. We define the *image of a under R*, written $R(a)$, to be the set of all $b \in B$ such that $(a,b) \in R$.

Exercise 2.3.4 (1) (with partial solution)

(a) Let $A = \{$John,Mary,Peter$\}$ and let $B = \{1,2,3\}$. Let R be the relation $\{$(John,1), (Mary,2), (Mary,3)$\}$. What are the images of John, Mary and Peter under R?
(b) Represent these three images in a natural way in the digraph for R.
(c) What is the image of 9 under the relation $\leq$ over the natural numbers?

Solution to (a) and (c)

(a) R(John) $= \{1\}$, R(Mary) $= \{2,3\}$, R(Peter) $= \varnothing$.
(c) $\leq(9) = \{n \in \mathbf{N}: 9 \leq n\}$.

It is useful to 'lift' the notion from of image from elements of A to subsets of A. If $X \subseteq A$, then we define the *image of X under R*, written $R(X)$, to be the set of all $b \in B$ such that $(x,b) \in R$ for some $x \in X$. In shorthand notation, $R(X) = \{b \in B: \exists x \in X, (x,b) \in R\}$.

Exercise 2.3.4 (2)

(a) Let A, B, R be as in the preceding exercise. Identify $R(X)$ for each one of the eight subsets X of A.
(b) What are the images of the following sets under the relation $\leq$ over the natural numbers? (i) $\{3, 12\}$, (ii) $\{0\}$, (iii) $\varnothing$, (iv) the set of all even natural numbers, (v) the set of all odds, (vi) $\mathbf{N}$.
(c) Let P (for predecessor) be the relation defined by putting $(a,x) \in P$ iff $x = a + 1$. What are the images of each of the above six sets under P?
(d) Identify the converse of the relations $\leq$ and P over the natural numbers, and the images of the above six sets under these converse relations.

It can happen that the notation $R(X)$ for the image of $X \subseteq A$ under R is ambiguous. This will be the case when the set A already has among its elements certain subsets X of itself. For this reason, some authors instead write R“(X) for the image of a subset. However, we will rarely if be dealing with such sets, and it will usually be safe to stick with the simpler and more common notation.

Exercise 2.3.4 (3)

(a) Construct a set A with just one element, which is also a subset of A.
(b) Do the same but with two elements in A, both of which are subsets of A.
(c) Indicate how this can be done for n elements, where n is any positive integer?

2.4 Reflexivity and Transitivity

In this section and the following two, we will look at special properties that a relation may have. Some of these properties are useful when we ask questions about how similar two items are, and when we want to *classify* objects. Some are needed when we look for some kind of *order* among items. Two of the properties, reflexivity and transitivity, are essential to both tasks, and so we begin with them.

2.4.1 Reflexivity

Let R be a (binary) relation. We say that it is *reflexive* over a set A iff $(a,a) \in R$ for all $a \in A$. For example, the relation $\leq$ is reflexive over the natural numbers, since always $n \leq n$; but $<$ is not. Indeed, $<$ has the opposite property of being *irreflexive* over the natural numbers: never $n < n$.

Clearly, reflexivity and irreflexivity are not the only possibilities: a relation R over a set may be neither one nor the other. For example, n is a prime divisor of n for some natural numbers (e.g. 2 is a prime divisor of itself) but not for some others (e.g. 4 is not a prime divisor of 4).

If we draw a digraph for a reflexive relation, every point will have an arrow going from it to itself; an irreflexive relation will have no such arrows; a relation that is neither reflexive nor irreflexive will have such arrows for some but not all of its points. However, when a relation is reflexive we sometimes reduce clutter by omitting these arrows from our diagrams and treating them as understood.

Exercise 2.4.1 (1) (with partial solution)

(a) Give other examples of relations over the natural numbers satisfying the three properties reflexive/irreflexive/neither.
(b) Can a relation R ever be both reflexive and irreflexive over a set A? If so, when?
(c) Identify the status of the following relations as reflexive/irreflexive/neither over the set of all people living in the UK: sibling of, shares at least one parent with, ancestor of, lives in the same city as, has listened to music played by.
(d) What does reflexivity mean in terms of a tabular representation of relations?

Solution to (c) The relation 'sibling of' as ordinarily understood is not reflexive since we do not regard a person as a sibling of himself or herself. In fact, it is irreflexive. By contrast, 'shares at least one parent with' is reflexive. It follows that these two relations are not quite the same. 'Ancestor of' is irreflexive. 'Lives in the same city as', under its most natural meaning, is neither reflexive nor irreflexive, since not everyone lives in a city. 'Has listened to music played by' is neither reflexive nor irreflexive for similar reasons.

A subtle point needs attention. Let R be a relation over a set A, that is, $R \subseteq A^2$. Then as we observed earlier, R is also a relation over any superset B of A, since then $R \subseteq A^2 \subseteq B^2$. It can happen that R is reflexive over A but not over its superset B. This is because we may have $(a,a) \in R$ for all $a \in A$ but $(b,b) \notin R$ for some $b \in B - A$. For this reason, whenever we say that a relation is reflexive or irreflexive, we should always specify what set A we have in mind. Doing this explicitly can be rather laborious, and so in practice, the identification of A is quite often left as understood. But you should always be sure *how* it is to be understood.

Exercise 2.4.1 (2)

(a) Give a small finite example of the dependence described in the text.
(b) Show that when R is reflexive over a set then it is reflexive over all of its subsets.

(c) Show that when each of two relations is reflexive over a set, so too are their intersection and their union.
(d) Show that the converse of a relation that is reflexive over a set is also reflexive over that set.

2.4.2 Transitivity

Another important property for relations is transitivity. We say that R is *transitive* iff whenever $(a,b) \in R$ and $(b,c) \in R$, then $(a,c) \in R$.

For example, the relation $\leq$ over the natural numbers is transitive: whenever $a \leq b$ and $b \leq c$, then $a \leq c$. Likewise for the relation $<$. But the relation of having some common prime factor is not transitive: 4 and 6 have a common prime factor (namely, 2), and 6 and 9 have a common prime factor (namely, 3), but 4 and 9 do not have any common prime factor. By the same token, the relation between people of being first cousins (that is, sharing some grandparent, but not sharing any parent) is not transitive.

Another relation over the natural numbers failing transitivity is that of being an immediate predecessor: 1 is an immediate predecessor of 2, which is an immediate predecessor of 3, but 1 does not stand in that relation to 3. In this example, the relation is indeed *intransitive*, in the sense that whenever $(a,b) \in R$ and $(b,c) \in R$, then $(a,c) \notin R$. In contrast, the relation of sharing a common prime factor, and that of being a first cousin, are clearly neither transitive nor intransitive.

In a digraph for a transitive relation, whenever there is an arrow from one point to a second, and another arrow from the second point to a third, there should also be an arrow from the first to the third. Evidently, this tends to clutter the picture considerably, even more so than for reflexivity. So, often when a relation is known to be transitive we adopt the convention of omitting the 'third arrows' from the digraph, treating them as understood. However, one must be clear about whether such a convention is being used.

Exercise 2.4.2
(a) Can a relation be both transitive and intransitive? If so, when?
(b) For each of the following relations over persons, state whether it is transitive, intransitive or neither: sister of, sibling of, parent of, ancestor of.
(c) Show that the intersection of any two transitive relations is transitive.
(d) Give an example to show that the union of two transitive relations need not be transitive.
(e) Is the composition of two transitive relations always transitive? Give a verification or a counterexample.

2.5 Equivalence Relations and Partitions

We now look at properties that are of particular interest when we want to express a notion of equivalence or similarity between items.

2.5.1 Symmetry

We say that R is *symmetric* iff whenever $(a,b) \in R$, then $(b,a) \in R$. For example, the relation of identity (alias equality) over the natural numbers is symmetric: whenever $a = b$ then $b = a$. So is the relation of sharing a common prime factor: if a has some prime factor in common with b, then b has a prime factor (indeed, the same one) in common with a. On the other hand, neither $\leq$ nor $<$ over the natural numbers is symmetric.

The way in which $<$ fails symmetry is not the same as the way in which $\leq$ fails it! When $n < m$ then we *never* have $m < n$, and the relation is said to *asymmetric*. In contrast, when $n \leq m$ we sometimes have $m \leq n$ (when $m = n$) but sometimes $m \nleq n$ (when $m \neq n$), and so this relation is neither symmetric nor asymmetric.

In a digraph for a symmetric relation, whenever there is an arrow from one point to a second, then there is an arrow going back again. A natural convention reduces clutter in the diagram, without possible ambiguity: use double-headed arrows, or links without heads.

Exercise 2.5.1

(a) What does symmetry mean in terms of the tabular representation of a relation?

(b) Can a relation R ever be both symmetric and asymmetric? If so, when?

(c) For each of the following relations over persons, state whether it is symmetric, asymmetric or neither: sister of, sibling of, parent of, ancestor of.

(d) Show that the converse of any symmetric relation is symmetric, as are also the intersection and union of any symmetric relations. How does this compare with the situation for transitivity? Why the difference?

(e) Is the composition of two symmetric relations always symmetric? Give a verification or a counterexample.

2.5.2 Equivalence Relations

When a relation is both reflexive and symmetric, it is sometimes called a *similarity relation*. When it has all three properties – transitivity, symmetry and reflexivity – it is called an *equivalence relation*.

Equivalence relations are often written using a symbol such as $\approx$ to bring out the idea that they behave rather like identity. And like identity, they are usually written by *infixing*, that is, as $a \approx b$, rather than in *basic set notation* as in $(a,b) \in R$ or by *prefixing* as in Rab.

Exercise 2.5.1 (with partial solution)

(a) Give another example of an equivalence relation over the natural numbers, and one over the set of all polygons in geometry.

(b) (i) Check that the identity relation over a set is an equivalence relation over that set. (ii) Show also that it is the *least* equivalence over that set, in the sense that it is included in every equivalence relation over the set. (iii) What is the largest equivalence relation over a set?

(c) Over a set of people, is the relation of having the same nationality an equivalence relation?

(d) What does the digraph of an equivalence relation look like?

Solutions to (a), (b), (c)

(a) For example, the relation of having the same parity (that is, both even, or else both odd) is an equivalence relation. For polygons, the relation of having the same number of sides is an equivalence relation. Of course, there are many others in both domains.

(b) (i) Identity is reflexive over any set because always $a = a$. It is symmetric because whenever $a = b$ then $b = a$. It is transitive because whenever $a = b$ and $b = c$ then $a = c$. (ii) It is included in every other equivalence relation $\approx$ over the same set. This is because every equivalence relation $\approx$ is reflexive, that is, $a \approx a$ for every $a \in A$, so that whenever $a = b$ we have $a \approx b$. (iii) The largest equivalence relation over A is A^2. Indeed it is the largest relation over A^2, and it is easily checked to be an equivalence relation.

(c) In the case that everyone in A has exactly one nationality, then this is an equivalence relation. But if someone in A has no nationality, then reflexivity fails, and if someone has more than one nationality, transitivity may fail. In contrast, the relation of 'having the same *set of* nationalities' is always an equivalence relation.

Alice Box: Identity or Equality?

Alice: You speak of people being *identical* to each other, but for numbers and other mathematical objects, I more often hear my instructors talk of *equality*. Are these the same?

Hatter: Yes, they are. Number theorists talk of equality, while logicians tend to speak of identity. But they are the same: identity is equal to equality, equality is identical with identity. However, you should be careful.

Alice: Why?

Hatter: In informal talk, both terms are sometimes used, rather loosely, to indicate that the items concerned agree on all properties that are considered important in the context under discussion. Strictly speaking, this is not full identity, only a 'tight' equivalence relation, perhaps also satisfying certain 'congruence' conditions. But more on this later.

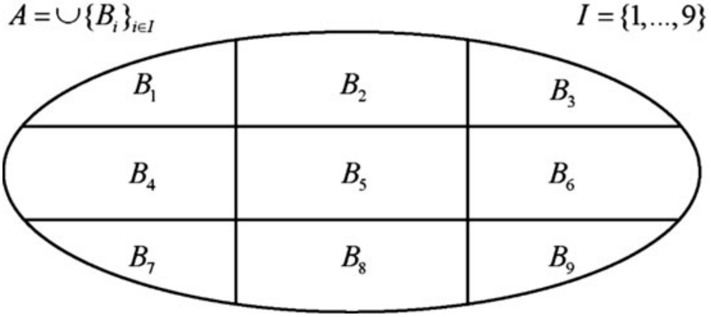

Fig. 2.2 Diagram for a partition

There is a very important difference between identity and other equivalence relations, which is a consequence of being the smallest. When $a = b$, the items a and b are elements of exactly the same sets and have exactly the same properties. Any mathematical statement that is true of one is true of the other. So *in any mathematical statement we may replace one by the other without loss of truth*.

This is one of the fundamental logical properties of identity. To illustrate it, return to the last exercise, part (b) (ii). There we said that if $a = b$ then whenever $a \approx a$ we have $a \approx b$. Here we were taking the statement $a \approx a$ *and* replacing the second a in it by b. In school you may have learned this under the rather vague title of ‘substitution of equals gives equals’.

2.5.3 Partitions

Your paper correspondence is in a mess – one big heap. You need to classify it, putting items into mutually exclusive but together exhaustive categories, but you don’t want to go into subcategories. Mathematically speaking, this means that you want to create a *partition* of the set of all the items in the heap.

We now define this concept. Let A be any non-empty set, and let $\{B_i\}_{i\in I}$ be a collection of subsets of A.

- We say that the collection *exhausts* A iff $\cup\{B_i\}_{i\in I} = A$, that is, iff every element of A is in at least one of the B_i. Using the notation that we introduced earlier for the universal and existential quantifiers, iff $\forall a \in A\ \exists i\in I\ (a\in B_i)$.
- We say that the collection is *pairwise disjoint* iff every two distinct sets B_i in the collection are disjoint. That is, for all $i,j \in I$, if $B_i \neq B_j$, then $B_i \cap B_j = \varnothing$. Using logical notation, iff $\forall i, j \in I\ (B_i \neq B_j \rightarrow B_i \cap B_j = \varnothing)$.

A *partition of* A is defined to be any collection $\{B_i\}_{i\in I}$ of non-empty subsets of A that are pairwise disjoint and together exhaust A. The sets B_i are called the *cells* (or sometimes, *blocks*) of the partition. We can diagram a partition in the manner of Fig. 2.2.

Exercise 2.5.3 (with solution)

(a) Which of the following are partitions of $A = \{1,2,3\ 4\}$? (i) $\{\{1,2\}, \{3\}\}$, (ii) $\{\{1,2\}, \{2,3\}, \{4\}\}$, (iii) $\{\{1,2\}, \{3,4\}, \varnothing\}$, (iv) $\{\{1,4\}, \{3,2\}\}$.

(b) We say that one partition of a set is *at least as fine* as another, iff every cell of the former is a subset of a cell of the latter. What is the finest partition of $A = \{1,2,3,4\}$? What is the least fine partition? In general, if A has $n \geq 1$ elements, how many cells in its finest and least fine partitions?

Solution

(a) Only (iv) is a partition of A. In (i) the cells do not exhaust A*;* in (ii) they are not pairwise disjoint; in (iii) one cell is the empty set, which is not allowed in a partition.

(b) The finest partition of $A = \{1, 2, 3, 4\}$ is $\{\{1\},\{2\},\{3\},\{4\}\}$, and the least fine is $\{\{1,2,3,4\}\}$. In general, if A has $n \geq 1$ elements, then its finest partition has n cells, while its least fine partition has only 1 cell.

2.5.4 The Partition/Equivalence Correspondence

It turns out that partitions and equivalence relations are two sides of the same coin. On the one hand, every partition of a set A determines an equivalence relation over A in a natural manner; on the other hand, every equivalence relation over a non-empty set A determines, in an equally natural way, a partition over A. The verification of this is rather abstract, and hence challenging, but you should be able to follow it.

We begin with the left-to-right direction. Let A be any set, and let $\{B_i\}_{i \in I}$ be a partition of A. We define the relation R *associated with* this partition by putting $(a,b) \in R$ iff a and b are in the same cell, that is, iff $\exists i \in I$ with $a,b \in B_i$. We need to show that it is an equivalence relation:

- By the exhaustion condition, R is *reflexive* over A. Since the partition exhausts A, we have $\forall a \in A$, $\exists i \in I$ with $a \in B_i$ and so immediately $\exists i \in I$ with both of a, $a \in B_i$.
- By the definition, R is *symmetric*. When $(a,b) \in R$, then $\exists i \in I$ with $a,b \in B_i$, so immediately $b,a \in B_i$, and thus $(b,a) \in R$.
- Finally, by the disjointness condition, R is *transitive*. Suppose $(a,b) \in R$ and $(b,c) \in R$; we want to show $(a,c) \in R$. Since $(a,b) \in R$, $\exists i \in I$ with both $a,b \in B_i$, and since $(b,c) \in R$, $\exists j \in I$ with $b,c \in B_j$. But since the cells of the partition are pairwise disjoint, either $B_i = B_j$ or $B_i \cap B_j = \varnothing$. Since $b \in B_i \cap B_j$ the latter is not possible, so $B_i = B_j$. Hence both $a,c \in B_i$, which gives us $(a,c) \in R$ as desired.

Now for the right-to-left direction, let $\approx$ be an equivalence relation over a non-empty set A. For each $a \in A$ we consider its image $\approx(a) = \{x \in A\colon a \approx x\}$ under the relation $\approx$. This set is usually written as $|a|_{\approx}$ or simply as $|a|$ when the equivalence relation $\approx$ is understood, and that is the notation we will use here. We consider the collection of these sets $|a|$ for $a \in A$ and verify that it has the properties needed to be a partition:

- *Non-emptiness*. Since $\approx$ is a relation over A, each set $|a|$ is a subset of A, and it is non-empty because $\approx$ is reflexive over A.

- *Exhaustion*. The sets $|a|$ together exhaust A, that is, $\cup\{|a|\}_{a\in A}=A$, again because $\approx$ is reflexive over A.
- *Pairwise disjointedness*. We argue contrapositively. Let $a,a' \in A$ and suppose $|a|\cap|a'| \neq \varnothing$. We need to show that $|a|=|a'|$. For that, it suffices to show that $|a| \subseteq |a'|$ and conversely $|a'| \subseteq |a|$. We do the former; the latter is similar. Let $x \in |a|$, that is, $a \approx x$; we need to show that $x \in |a'|$, that is, that $a' \approx x$. By the initial supposition, there is a y with $y \in |a|\cap|a'|$, so $y \in |a|$ and $y \in |a'|$, so $a \approx y$ and $a' \approx y$. Since $a \approx y$ symmetry gives us $y \approx a$, and so we may apply transitivity twice: first to $a' \approx y$ and $y \approx a$ to get $a' \approx a$, and then to that and $a \approx x$ to get $a' \approx x$ as desired.

Note that in this verification we appealed to all three of the conditions reflexivity, symmetry and transitivity. You can't get away with less.

Alice Box: Similar Verifications

Alice: In the verification, you said 'We do the former; the latter is similar'. But how do I know that it is really similar if I don't spell it out?

Hatter: Strictly speaking, you don't! But when two cases under consideration have the same structure, and the arguments are exactly the same except for, say, a permutation of variables (in this case, we permuted a and a'), then we may take the liberty of omitting the second so as to focus on essentials.

Alice: Isn't it dangerous?

Hatter: Of course, there can be small but crucial differences, so care is needed. Writers should check all cases before omitting any, and should present things in such a way that the reader can always reconstruct missing parts in a routine way.

When $\approx$ is an equivalence relation over A, the sets $|a|=\{x \in A\colon a \approx x\}$ for $a \in A$ (i.e. the cells of the corresponding partition) are called the *equivalence classes* of $\approx$.

Exercise 2.5.4

(a) Let A be the set of all positive integers from 1 to 10. Consider the partition into evens and odds. Write this partition by enumeration as a collection of sets, then describe the corresponding equivalence relation in words, and write it by enumeration as a set of ordered pairs.

(b) Let A be the set of all positive integers from 2 to 16. Consider the relation of having exactly the same prime factors (so e.g. $6 \approx 12$ since they have the same prime factors 2 and 3). Identify the associated partition by enumerating it as a collection of subsets of A.

(c) How would you describe the equivalence relation associated with the finest (respectively: least fine) partition of A?

2.6 Relations for Ordering

Suppose that we want to use a relation to order things. It is natural to require it to be transitive. We can choose it to be reflexive, in which case we get an *inclusive order* (like $\leq$ over the natural numbers or $\subseteq$ between sets); or to be irreflexive, in which case we get what is usually known as a *strict order* (like $<$ or $\subset$). In this section, we examine some special kinds of inclusive order, and then look at strict orders.

2.6.1 Partial Order

Consider a relation that is both reflexive and transitive. What other properties do we want it to have if it is to serve as an ordering?

Not symmetry, for that would make it an equivalence relation. Asymmetry? Not quite, for a reflexive relation over a non-empty set can never be asymmetric. What we need is the closest possible thing to asymmetry: whenever $(a,b) \in R$, then $(b,a) \notin R$ provided that $a \neq b$. This property is known as *antisymmetry*, and it is usually formulated in a contraposed (and thus equivalent) manner: whenever $(a,b) \in R$ and $(b,a) \in R$ then $a = b$.

A relation R over a set A that is reflexive (over A), transitive, and also antisymmetric is called a *partial order* (or partial ordering) of A, and the pair (A,R) is called for short a *poset*. It is customary to write a partial ordering as $\leq$ (or some square or curly variant of the same sign) even though, as we will soon see, the familiar relation 'less than or equal to' over the natural numbers (or any other number system) has a further property that not all partial orderings share. Two examples of partial order:

- For sets, the relation $\subseteq$ of inclusion is a partial order. As we already know, it is reflexive and transitive. We also have antisymmetry since whenever $A \subseteq B$ and $B \subseteq A$, then $A = B$.
- In arithmetic, an important example is the relation of being a divisor of, over the positive integers, that is, the relation R over $\mathbf{N}^+$ defined by $(a,b) \in R$ iff $b = ka$ for some $k \in \mathbf{N}^+$.

Exercise 2.6.1

(a) Check that the relation of being divisor of, over the positive integers, is indeed a partial order.

(b) What about the corresponding relation over $\mathbf{Z}$, that is, the relation R over $\mathbf{Z}$ defined by $(a,b) \in R$ iff $b = ka$ for some $k \in \mathbf{Z}$?

(c) And the corresponding relation over $\mathbf{Q}^+$, that is, the relation R over $\mathbf{Q}^+$ defined by $(a,b) \in R$ iff $b = ka$ for some $k \in \mathbf{Q}^+$?

(d) Consider the relation R over people defined by $(a,b) \in R$ iff either $b = a$ or b is descended from a. Is it a poset?

(e) Show that the relation of 'being at least as fine as', between partitions of a given set A, is a partial order.

2.6.2 Linear Orderings

A reflexive relation R over a set A is said to be *complete* over A iff for all $a,b \in A$, either $(a,b) \in R$ or $(b,a) \in R$. A poset that is also complete is often called a *linear* (or *total*) *ordering*.

Clearly the relation $\leq$ over $\mathbf{N}$ is complete and so a linear ordering, since for all $m,n \in \mathbf{N}$ either $m \leq n$ or $n \leq m$. So is the usual lexicographic ordering of words in a dictionary. On the other hand, whenever a set A has more than one element, then the relation $\subseteq$ over $\mathcal{P}(A)$ is not complete. Reason: Take any two distinct $a,b \in A$, and consider the singletons $\{a\},\{b\}$; they are both elements of $\mathcal{P}(A)$, but neither $\{a\} \subseteq \{b\}$ nor $\{b\} \subseteq a\}$ because $a \neq b$.

Exercise 2.6.2 (with solution)

(a) Give two more linear orderings of **N**.

(b) Which of the following are linear orderings over an arbitrary set of people: (i) is at least as old as, (ii) is identical to or a descendent of?

Solution

(a) There are plenty, but here are two: (i) the relation $\geq$, that is, the converse of $\leq$, (ii) the relation that puts all odd positive integers first, and then all the even ones, each of these blocks being ordered separately as usual.

(b) (i) Yes, it meets all the requirements, (ii) no, since it is not complete.

2.6.3 Strict Orderings

Whenever we have a reflexive relation $\leq$, we can always look at what is known as its *strict part*. It is usually written as $<$ (or a square or curly variant of this) even though it need not have all the properties of 'less than' over the usual number systems. The definition is as follows: $a < b$ iff $a \leq b$ but $a \neq b$. In language that looks like gibberish but makes perfect sense if you read it properly: $(<) = (\leq \cap \neq)$.

Exercise 2.6.3 (1) (with solution)

(a) Show that every asymmetric relation over a set A is irreflexive.

(b) Show that when $\leq$ is a partial ordering over a set A, then its strict part $<$ is (i) asymmetric and (ii) transitive.

Solution

(a) We argue contrapositively. Suppose that $<$ is not irreflexive. Then, there is an $a \in A$ with $a < a$. Thus, both $a < a$ and $a < a$ so that asymmetry fails.

(b)(i) For asymmetry, let $\leq$ be a partial ordering and suppose $a < b$ and $b < a$ where $<$ is the positive part of $\leq$. We get a contradiction. By the suppositions, $a \leq b$, $b \leq a$ and $a \neq b$, which is impossible by the antisymmetry of $\leq$.

(b)(ii) For transitivity, suppose $a < b$ and $b < c$; we want to show that $a < c$. By the suppositions, $a \leq b$ and $b \leq c$ but $a \neq b$ and $b \neq c$. Transitivity of $\leq$ thus gives

$a \le c$; it remains to check that $a \neq c$. Suppose $a = c$; we get a contradiction. Since $b \le c$ and $a = c$ we have $b \le a$, so by the antisymmetry of $\le$ using also $a \le b$ we have $a = b$, contradicting $a \neq b$.

Alice Box: Proof by Contradiction (Reductio ad Absurdum)

Alice: There is something in the solution to this exercise that worries me. We supposed the *opposite* of what we wanted to show. For example, in (b)(i), to show that the relation < there is asymmetric, we supposed that both $a < b$ and $b < a$, which is the opposite of what we are trying to prove.

Hatter: Indeed we did, and the goal of the argument changed when we made the supposition: it became one of deriving a contradiction. In the exercise, we got our contradictions very quickly; sometimes it takes more argument.

Alice: Will any contradiction do?

Hatter: Any contradiction you like. Any pair of propositions α and $\neg\alpha$. Such pairs are as bad as each other, and thus just as good for the purposes of the proof.

Alice: Is this procedure some new invention of modern logicians?

Hatter: Not at all. It was well known to the ancient Greeks, and can be found in Euclid. In the Middle Ages it was taught under its Latin name reductio ad absurdum, sometimes abbreviated to RAA, and this name is still often used.

Reductio ad absurdum is a universal proof method, in the sense that it is always possible to use it. Some people like to use it whenever they can; others do so only when they get stuck without it. Most mathematicians are somewhere in the middle: they use it when they see that it can make the argument shorter or more transparent.

Exercise 2.6.3 (2) Do part (b) of the preceding exercise again, without using proof by contradiction, for example, by proving the contrapositive. Compare the solutions, and decide which you prefer.

In the preceding chapter we observed that the relation of immediate inclusion between finite sets may be represented by a Hasse diagram. We can do the same for any partial order $\le$ over a finite set. The links of the diagram, read from below to above, represent the relation of *being an immediate predecessor of*. This is the relation that holds from a to b iff $a < b$ but there is no x with $a < x < b$, where < is the strict part of $\le$.

Once we have a Hasse diagram for the immediate predecessor part of a partial ordering, we can read off from it the entire relation: $a \le b$ iff there is an ascending path (with at least one element) from a to b. Evidently, this is a much more economical representation than drawing the digraph for the entire partial ordering.

For this reason, we loosely call it the Hasse diagram of the partial ordering itself, and similarly for its strict part.

Exercise 2.6.3 (3) Draw a Hasse diagram for (the immediate predecessor part of) the relation of being an exact divisor of, over the set of positive integers up to 13.

We have seen in an exercise that when $\leq$ is a partial ordering over a set A, then its strict part $<$ is asymmetric and transitive. We also have a converse: whenever a relation $<$ is both asymmetric and transitive, then the relation $\leq$ defined by putting $a \leq b$ iff either $a < b$ or $a = b$ is a partial order. Given this two-way connection, asymmetric transitive relations are often called *strict partial orders*.

Exercise 2.6.3 (4)

(a) Show, as claimed in the text, that when $<$ is a transitive asymmetric relation, then the relation $\leq$ defined by putting $a \leq b$ iff either $a < b$ or $a = b$ is a partial order.

(b) Suppose we start with a partial order $\leq$, take its strict part $<$, and then define from it a partial order $\leq'$ by the rule in (a). Show that $\leq'$ is the same relation as $\leq$.

(c) Suppose we start with a strict partial order $<$, define from it a partial order $\leq$ by the rule in (a) and then take its strict part $<'$. Show that $<'$ is the same relation as $<$.

Hint: In part (b), first show that $\leq'$ is a subrelation of $\leq$, then show the converse. Use a similar procedure for the strict relations in (c).

2.7 Closing with Relations

In this section, we explain two very useful concepts, the transitive closure of an arbitrary relation, and the closure of a set under a relation.

2.7.1 Transitive Closure of a Relation

Suppose you are given a relation R that is not transitive, but which you want to 'make' transitive. Of course, you cannot change the status of R itself, but you can expand it to a larger relation that satisfies transitivity. In general, there will be many of these. For example, whenever $R \subseteq A^2$, then A^2 itself is a transitive relation that includes R. But there will be much smaller ones, and it is not difficult to see that there must always be a *unique smallest* transitive relation that includes R; it is called the *transitive closure* of R and is written as R^*.

There are several equivalent ways of defining R^*. We give three, which we might call the *finite path* formulation (using finite sequences), the *bottom-up* definition (using union of sets), and the *top-down* one (using intersection of sets).

The *finite path* definition puts $(a,b) \in R^*$ iff there is a finite sequence $x_1, \ldots x_n$ $(n \geq 2)$ with $a = x_1$, $x_n = b$, and $x_i R x_{i+1}$ for all i with $1 \leq i < n$.

For the '*bottom-up*' definition, we begin by defining the relations R_0, $R_1, \ldots$ recursively as follows:

$$R_0 = R$$
$$R_{n+1} = R_n \cup \{(a, c) : \exists x \text{ with } (a, x) \in R_n \text{ and } (x, c) \in R\}.$$

We then put R^* to be their union: $R^* = \cup\{R_n\colon n \in \mathbf{N}\}$. In this way, R^* is built up by successively adding pairs (a,c) whenever (a,x) is in the relation constructed so far and $(x,c) \in R$. In effect, the definition tells us to keep on doing this until there are no such pairs (a,c) still needing to be put in or indefinitely if there are always still such pairs.

The 'top-down' account defines R^* as the intersection of the collection of all transitive relations that include R. That is: when R is a relation over A^2 then $R^* = \cap\{S\colon R \subseteq S \subseteq A^2$ and S is transitive$\}$.

Why bother with three definitions of the same thing? They provide three quite different ways of looking at R^*, and three distinct ways of proving facts about it. Sometimes it is more convenient to use one definition, sometimes another, and you are encouraged to have all three in your toolkit.

Exercise 2.7.1 (1) (with partial solution)

(a) Identify the transitive closures of the following relations: (i) parent of, (ii) mother of, (iii) descendant of.

(b) Show that when R^* is defined as the intersection of the collection of all transitive relations that include R, it is indeed the least transitive relation that includes R, where 'least' means' least under set inclusion. *Hint*: It suffices to check that (i) $R \subseteq R^*$, (ii) R^* is transitive, and (iii) $R^* \subseteq S$ for every transitive relation S with $R \subseteq S$.

(c) Write the following assertions in the notation for converse (see Sect. 2.3.1) and transitive closure, and determine whether they are true or false: (i) the converse of the transitive closure of a relation equals the transitive closure of the converse of that relation, (ii) the intersection of the transitive closures of two relations equals the transitive closure of their intersection.

Solution to (a) (i) Ancestor of. (ii) Ancestor of in the female line (there is no single word for this in English). (iii) Descendant of (as this relation is already transitive, it coincides with its transitive closure).

Exercise 2.7.1 (2)

(a) Draw an arrow diagram to illustrate the definition of R_{n+1} out of R_n and R.

(b) Can you show that the three definitions of transitive closure are equivalent? This is quite a challenging exercise, and if you run into difficulty now you may wish to return after the chapter on recursion and induction.

2.7.2 Closure of a Set Under a Relation

Transitive closure is in fact a particular instance of a more general construction of great importance – the closure of an arbitrary set under an arbitrary relation.

To introduce it, we recall the definition of image from earlier in this chapter (Sect. 2.3.4). The *image* $R(X)$ of a set X under a relation R is the set of all b such that $(x,b) \in R$ for some $x \in X$; briefly $R(X) = \{b: \exists x \in X, (x,b) \in R\}$.

Now, suppose we are given a relation R (not necessarily transitive) over a set B, and a subset $A \subseteq B$. We define the *closure* $R[A]$ *of* A *under* R to be the least subset of B that includes A and also includes $R(X)$ whenever it includes X.

Again, this is a top-down definition, with 'least' understood as the result of intersection. It can also be expressed bottom-up, as follows:

$$\begin{aligned} &A_0 = A \\ &A_{n+1} = A_n \cup R(A_n) \text{ for each natural number } n \\ &R[A] = \cup\{A_n : n \in \mathbf{N}\}. \end{aligned}$$

The closure of A under R is thus constructed by beginning with A itself, and at each stage adding in the elements of the image of the set so far constructed. Closure thus differs from image in two respects: the closure of a set under a relation always includes that set, and we apply the relation not just once but over and over again.

Exercise 2.7.2 (with solution) In the set **N** of natural numbers, what is the closure $R[A]$ of the set $A = \{2,5\}$ under the following relations over $\mathbf{N}^2$: (a) $(m,n) \in R$ iff $n = m + 1$, (b) $(m,n) \in R$ iff $n = m - 1$, (c) $(m,n) \in R$ iff $n = 2m$, (d) $(m,n) \in R$ iff $n = m/2$, (e) $\leq$, (f) $<$.

Solution
(a) $\{n \in \mathbf{N}: n \geq 2\}$, (b) $\{n \in \mathbf{N}: n \leq 5\} = \{0,1,2,3,4,5\}$, (c) $\{2,4,8,\ldots; 5,10, 20,\ldots\}$, (d) $\{1,2,5\}$, (e) $\{n \in \mathbf{N}: n \geq 2\}$, (f) $\{n \in \mathbf{N}: n \geq 2\}$.

Comment For part (d), if you wrote $\{1\}$ you forgot that $A \subseteq R[A]$ for every relation R. Similarly if you wrote $\{n \in \mathbf{N}: n > 2\}$ for (f).

Notation and terminology: When the relation R is understood, we sometimes write the closure $R[A]$ more briefly as A^+, and say that R *generates* the closure from A.

End-of-Chapter Exercises

Exercise 2.1: Cartesian Products

(a) Show that $A \times (B \cap C) = (A \times B) \cap (A \times C)$.
(b) Show that $A \times (B \cup C) = (A \times B) \cup (A \times C)$.

Exercise 2.2: Domain, Range, Join, Composition, Image

(a) Consider the relations $R = \{(1,7), (3,3), (13,11)\}$ and $S = \{(1,1), (1,7), (3,11), (13,12), (15,1)\}$ over the positive integers. Identify $dom(R \cap S)$, $range(R \cap S)$, $dom(R \cup S)$, $range(R \cup S)$.

(b) In the same example, identify $join(R,S)$, $join(S,R)$, $S \circ R$, $R \circ S$, $R \circ R$, $S \circ S$.

(c) In the same example, identify $R(X)$ and $S(X)$ for $X = \{1,3,11\}$ and $X = \varnothing$.

(d) Explain how to carry out composition by means of join and projection.

Exercise 2.3: Reflexivity and Transitivity

(a) Show that R is reflexive over A iff $I_A \subseteq R$. Here I_A is the identity relation over A, defined in an exercise in Sect. 2.1.3.

(b) Show that the converse of a relation R that is reflexive over a set A is also reflexive over A.

(c) Show that R is transitive iff $R \circ R \subseteq R$.

(d) In a previous exercise, we showed that the intersection of any two transitive relations is transitive but their union may not be. (i) Show that the converse and composition of transitive relations are also transitive. (ii) Give an example to show that the complement of a transitive relation need not be transitive.

(e) A relation R is said the be *acyclic* iff there are no $a_1, \ldots, a_n$ ($n \geq 2$) such that each $(a_i, a_{i+1}) \in R$ and also $a_n = a_1$. (i) Show that a transitive irreflexive relation is always acyclic. (ii) Show that every acyclic relation is irreflexive. (iii) Give an example of an acyclic relation that is not transitive.

Exercise 2.4: Symmetry, Equivalence Relations and Partitions

(a) Show that the following three conditions are equivalent: (i) R is symmetric, (ii) $R \subseteq R^{-1}$, $R = R^{-1}$.

(b) Show that a relation that is reflexive over a non-empty set can never be intransitive.

(c) Show that if R is reflexive over A and also transitive, then the relation S defined by $(a,b) \in S$ iff both $(a,b) \in R$ and $(b,a) \in R$ is an equivalence relation.

(d) Show that the intersection of two equivalence relations is an equivalence relation, but that this is not the case for unions. *Hint*: Apply the results of exercises on reflexivity, transitivity, and symmetry in this chapter.

(e) Enumerate all the partitions of $A = \{1,2,3\}$ and draw a Hasse diagram for them under fineness.

(f) Show that one partition of a set is at least as fine as another iff the equivalence relation associated with the former is a subrelation of the equivalence relation associated with the latter.

Exercise 2.5: Antisymmetry, Partial Order, Linear Order

(a) Let R be any transitive relation over a set A. Define S over A by putting $(a,b) \in S$ iff either $a = b$ or both $(a,b) \in R$ and $(b,a) \notin R$. Show that S partially orders A.

(b) Show that the converse of a linear order is linear but that the intersection and composition of two linear orders need not be linear.

(c) Show that the identity relation over a set A is the unique partial order of A that is also an equivalence relation.

(d) Let A be a set and $\leq$ a partial ordering of A. An element $a \in A$ is said to be a *minimal* element of A (under $\leq$) iff there is no $b \in A$ with $b < a$. On the other hand, an element $a \in A$ is said to be a *least* element of A (under $\leq$) iff $a \leq b$ for every $b \in A$. Show the following for sets A and partial orderings $\leq$: (i) whenever a is a least element of A then it is a minimal element of A. (ii) The converse can fail (give a simple example). (iii) A can have zero, one or more than one minimal elements (give an example of each). (iv) A can have at most one least element, that is, if a least element exists, then it is unique. (v) For linear orderings, A can have at most one minimal element, and if it exists then it is the least element of A. (vi) If A is finite then it must have at least one minimal element. (vii) There are infinite linearly ordered sets without any minimal element.

Exercise 2.6: Strict Orderings

(a) Give examples of relations that are (i) transitive but not asymmetric, and (ii) asymmetric but not transitive.

(b) Show that a relation is antisymmetric iff its strict part is asymmetric.

(c) Show that every transitive acyclic relation is asymmetric.

Exercise 2.7: Closure

(a) Show that always $X \cup R(X) \subseteq R[X]$.

(b) Show that $R[A] = R(A)$ if R is both reflexive over A and transitive.

Selected Reading

The three books listed at the end of Chap. 1 for sets also have good chapters on relations. In detail:

Bloch ED (2011) Proofs and fundamentals: a first course in abstract mathematics, 2nd edn. Springer, New York, chapter 5

Halmos PR (2001) Naive set theory, new edn. Springer, New York, chapters 6–7

Lipschutz S (1998) Set theory and related topics, Schaum's outline series. McGraw Hill, New York, chapter 3

A gentle introductory account which, like that of Bloch, pays careful attention to underlying logical and heuristic matters, is:

Velleman DJ (2006) How to prove it: a structured approach, 2nd edn. Cambridge University Press, Cambridge, chapter 4

Finally, we mention a chapter from the following text on discrete mathematics written specifically for computer science students:

Hein JL (2002) Discrete structures, logic and computability, 2nd edn. Jones and Bartlett, Sudbury, chapter 4

Associating One Item with Another: Functions

3

Abstract

Functions occur everywhere in mathematics and computer science. In this chapter, we introduce the basic concepts needed in order to work with them.

We begin with the intuitive idea of a function and its mathematical definition as a special kind of relation. We then see how the general concepts for relations that were studied in the previous chapter play out in this particular case (*domain*, *range*, *restriction*, *image*, *closure*, *composition*, *inverse*) and distinguish some important kinds of function (*injective*, *surjective*, *bijective*) with special behaviour.

These concepts permit us to link functions with counting, via the *equinumerosity*, *comparison* and surprisingly versatile *pigeonhole* principles. Finally, we identify some very simple kinds of function that appear over and again (*identity*, *constant*, *projection* and *characteristic* functions) and explain the deployment of functions to represent *sequences* and *families*.

3.1 What Is a Function?

Traditionally, a function was seen as a rule, often written as an equation, which associates any number (called the *argument* of the function) with another number, called the *value* of the function. The concept has for long been used in physics and other sciences to describe processes whereby one quantity (such as the temperature of a gas, the speed of a car) affects another (its volume, its braking distance). In accord with such applications, the argument of the function was sometimes called the 'independent variable', and the value of the function termed the 'dependent variable'. The idea was that each choice for the argument or independent variable causally determines a value for the dependent variable.

Over the last 200 years, the concept of a function has evolved in the direction of greater abstraction and generality. The argument and value of the function need not be numbers – they can be items of any kind whatsoever. The function need

D. Makinson, *Sets, Logic and Maths for Computing*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-2500-6_3,

not represent a causal relationship, nor indeed any physical process, although these remain important applications. The function may not even be expressible by any linguistic rule, although all of the ones that we will encounter in this book are. Taken to the limit of abstraction, a function is no more than a set of ordered pairs, that is, a relation that satisfies a certain condition.

What is that condition? We explain with functions of one argument. A *one-place function* from a set A into a set B is any binary relation R from A to B satisfying the condition that *for all* $a \in A$ there is *exactly one* $b \in B$ with $(a,b) \in R$. The italicized parts of the definition deserve special attention:

- *Exactly one*...: This implies that there is always *at least one* $b \in B$ with $(a,b) \in R$, and *never more* than one.
- *For all* $a \in A$...: This implies that the specified source A of the relation is in fact its *domain*: there is no element a of A that fails to be the first term in some pair $(a,b) \in R$.

In terms of tables, a function from A to B is a relation whose table has exactly one 1 in each row. In terms of digraphs, every point in the A circle has exactly one arrow going out from it to the B circle.

Exercise 3.1 (1) (with solution) For each of the following relations from $A = \{a,b,c,d\}$ to $B = \{1,2,3,4,5\}$, determine whether or not it is a function from A to B. Whenever your answer is negative, give your reason.

(i) $\{(a,1), (b,2), (c,3)\}$
(ii) $\{(a,1), (b,2), (c,3), (d,4), (d,5)\}$
(iii) $\{(a,1), (b,2), (c,3), (d,5)\}$
(iv) $\{(a,1), (b,2), (c,2), (d,1)\}$
(v) $\{(a,5), (b,5), (c,5), (d,5)\}$

Solution (i) No, since $d \in A$, but there is no pair of the form (d,x) in the relation, so that the 'at least one' condition fails. (ii) No, since both $(d,4)$ and $(d,5)$ are in the relation and evidently $4 \neq 5$ (the 'at most one' condition fails). (iii) Yes, each element of the source is related to exactly one element of the target; it does not matter that the element 4 of the target is 'left out'. (iv) Yes, it does not matter that the element 2 of the target is 'hit twice'. (v) Yes, it does not matter that the only element of the target that is hit is 5. This is called the *constant function* with value 5 from A to B.

Functions are usually referred to with lower case letters $f,g,h,\ldots$. Since for all $a \in A$ there is a unique $b \in B$ with $(a,b) \in f$, we may use some familiar terminology and notation. We call the unique b the *value* of the function f for argument a and write it as $f(a)$. Computer scientists sometimes say that $f(a)$ is the *output* of the function f for *input* a. We also write $f\colon A \to B$ to mean that f is a function from A into B. In the case that $B = A$, we have a function $f\colon A \to A$ from A into itself, and we usually say briefly that f is a function *on* A.

Strictly speaking, we have only defined *one-place* functions, from a *single set* A into a set B. However, functions can have more than one argument; for example, addition and multiplication in arithmetic each have two. We can generalize the definition accordingly. If $A_1, \ldots, A_n$ are sets, then an *n-place function* from $A_1, \ldots, A_n$ into B is an $(n+1)$-place relation R such that for all $a_1, \ldots, a_n$ with each $a_i \in A_\mathrm{i}$, there is exactly one $b \in B$ with $(a_1, \ldots, a_n, b) \in R$. One-place functions are then covered as the case where $n = 1$.

However, the additional generality in the notion of n-place functions may also be seen as merely apparent. We can treat an n-place function from $A_1, \ldots, A_n$ into B as a one-place function from the Cartesian product $A_1 \times \ldots \times A_n$ into B. For example, addition and multiplication on the natural numbers are usually thought of a two-place functions f and g with $f(x,y) = x + y$ and $g(x,y) = x \cdot y$, but they may also be thought of as one-place functions on $\mathbf{N} \times \mathbf{N}$ into $\mathbf{N}$ with $f((x,y)) = x + y$ and $g((x,y)) = x \cdot y$. So there will be no real loss in generality when we formulate principles in terms of one-place functions.

Alice Box: Bracket-Chopping

Alice: Typo! Too many brackets there.

Hatter: No! Strictly speaking, the double parentheses are required. We need the outer brackets as part of the notation for functions in general, and the inner ones because the argument is an ordered pair (x,y).

Alice: But surely we can drop the inner ones if we read addition or multiplication as two-place, rather than one-place functions? We still need the outer brackets as part of the notation for functions, but instead of a single-ordered pair (x,y) as argument, we have two arguments x,y which are numbers. So in that respect, using n-place functions can actually simplify notation.

Hatter: Nice point. And in computing, we do have to keep our eyes open for brackets. But let's get back to more substantial things ...

There are many occasions like this in mathematics, where a concept or construction may be seen in either of two ways, each with its costs and benefits. One perspective can be taken as more general or basic than another, but also vice versa. We have just seen this for the specific concepts of one-place and many-place functions, but it also comes up for the wider concepts of relation and function themselves. In this book, we have taken the former as basic, defining the latter as a special case. It is also possible to do the reverse, and some texts do; each procedure has its advantages and disadvantages.

To end the section, we draw attention to an important generalization of the notion of a function, which relaxes the definition in one respect. A *partial function* from a set A to a set B is a binary relation R from A to B such that for all $a \in A$, there is *at most one* $b \in B$ with $(a,b) \in R$. The difference is that there may be elements of A that do not have the relation R to any element of B. Partial functions are also useful in computer science, and we employ them occasionally later, but in this chapter, we focus most of our attention on functions in the full sense of the term. When we speak of functions without qualification, we mean full ones.

Exercise 3.1 (2) (with solution)

(a) Is either of the non-functions in Exercise 3.1.1 nevertheless a partial function from A to B?

(b) Which of the following relations from **Z** into **Z** (see Chap. 1 Sect. 1.6 to recall its definition) is a function on **Z**? Which of those that fail to be a function from **Z** to **Z** is nevertheless a partial function on **Z**?

(i) $\{(a, |a|)\colon a \in \mathbf{Z}\}$, (ii) $\{(|a|, a)\colon a \in \mathbf{Z}\}$, (iii) $\{(a, a^2)\colon a \in \mathbf{Z}\}$, (iv) $\{(a^2, a)\colon a \in \mathbf{Z}\}$, (v) $\{(a, a+1)\colon a \in \mathbf{Z}\}$, (vi) $\{(a+1, a)\colon a \in \mathbf{Z}\}$, (vii) $\{(2a, a)\colon a \in \mathbf{Z}\}$.

Solution

(a) The relation (i) is a partial function from A to B.

(b) (i) Yes, for every $a \in \mathbf{Z}$ there is a unique $b \in \mathbf{Z}$ with $b = |a|$.

(ii) No, for two reasons. First, $dom(R) = \mathbf{N} \subset \mathbf{Z}$. Second, even within this domain, the 'at most one' condition is not satisfied, since a can be positive or negative. So this is not even a partial function.

(iii) Yes, for every $a \in \mathbf{Z}$ there is a unique $b \in \mathbf{Z}$ with $b = a^2$.

(iv) No, not even a partial function: same reasons as for (ii).

(v) Yes, for every $a \in \mathbf{Z}$ there is a unique $b \in \mathbf{Z}$ with $b = a + 1$.

(vi) Yes, every $x \in \mathbf{Z}$ is of the form $a + 1$ for a unique $a \in \mathbf{Z}$, namely, for $a = x - 1$.

(vii) No, $dom(R)$ is the set of all even (positive or negative) integers only. But it is a partial function.

3.2 Operations on Functions

As functions are relations, all the operations that we introduced in the theory of relations apply to them. However, in this context, some of those operations are more useful than others, or may be expressed in a simpler way or with different terminology, so we begin by reviewing them.

3.2.1 Domain and Range

These two concepts carry over without change. Recall that when R is a relation from A to B, then $dom(R) = \{a \in A\colon \exists\, b \in B\ ((a,b) \in R)\}$. When R is in fact a function f, this is expressed as $dom(f) = \{a \in A\colon \exists b \in B\ (f(a) = b)\}$. Thus when $f\colon A \to B$, then $dom(f) = A$.

Likewise, $range(R) = \{b \in B\colon \exists\, a \in A\ ((a,b) \in R)\}$, which for functions comes to $range(f) = \{b \in B\colon \exists a \in A\ (f(a) = b)\}$. When $f\colon A \to B$ then $range(f)$ may be B itself or one of its subsets.

Exercise 3.2.1 (with solution)

(a) Identify the domain and range of the three functions in Exercise 3.1 (1), calling them f, g, h.

(b) Can the domain of a function ever be empty? And the range?

Solutions

(a) *dom*(f) = *dom*(g) = *dom*(h) = A; *range*(f) = {1,2,3,5}, *range*(g) = {1,2}, *range*(h) = {5}.

(b) There is just one function with empty domain, namely, the empty function, which is also the unique function with empty range.

3.2.2 Restriction, Image, Closure

The concept of restriction is quite straightforward. Using the general definition from the theory of relations, the *restriction* of f: $A \to B$ to $X \subseteq A$ is the unique function on X into B that agrees with f over X. Notations vary, but one is f_X. Thus f_X: $X \to B$ with $f_X(a) = f(a)$ for all $a \in X$. Because this is such a trivial operation it is rarely at the centre of attention, and it is very common to omit the distracting subscript, leaving it to context to make it clear that the domain has been reduced from A to its subset X.

> *Warning*: When f: $A \to A$ and $X \subseteq A$, then the restriction of f to X will always be a function from X to A, but it will not always be a function from X to X, because there may be an $a \in X \subseteq A$ with $f(a) \in A \backslash X$. We will see a simple example of this in the next exercise.

Let f: $A \to B$, that is, let f be a function from A to B, and let $X \subseteq A$. The *image under f of* $X \subseteq A$ is the set $\{b \in B$: $\exists a \in X, b = f(a)\}$, which can also be written more briefly as $\{f(a)$: $a \in A\}$. Thus, to take limiting cases as examples, the image $f(A)$ of A itself is *range*(f), and the image $f(\varnothing)$ of $\varnothing$ is $\varnothing$, while the image of a singleton subset $\{a\} \subseteq A$ is the singleton $\{f(a)\}$. Thus, image is not quite the same thing as value: the *value* of $a \in A$ under f is $f(a)$. However, some texts also use the term 'image' rather loosely as a synonym of 'value'.

When f: $A \to B$ and $X \subseteq A$, the image of X under f is usually written as $f(X)$, and we will follow this standard notation. Just as for relations, there are contexts in which it can be ambiguous. Suppose that A is of *mixed type*, in the sense that some element X of A is also a subset of A, that is, both $X \in A$ and $X \subseteq A$. Then the expression $f(X)$ becomes ambiguous: in a basic sense it denotes the *value* of X under f, while in a derivative sense it stands for the *image* $\{f(x)$: $x \in X\}$ of X under f. In this book, we will rarely be discussing sets of mixed type, and so do not have to worry much about the problem, but in contexts where mathematicians deal with them, they sometimes avoid ambiguity by using the alternative notation f“(X) for the image.

Image should not be confused with closure, which is a concept arising for functions f on a set into itself. Let f: $A \to A$ and let $X \subseteq A$. Then, in a 'top down' version, the *closure* $f[X]$ of X under f is the least subset of A that includes X and also includes $f(Y)$ whenever it includes Y. Be careful: the word 'includes' here means 'is a superset of' and not 'contains as an element'. Equivalently, in a 'bottom up' or recursive form, we may define $A_0 = X$ and $A_{n+1} = A_n \cup f(A_n)$ for each natural number n, and put $f[X] = \cup\{A_n: n \in \mathbf{N}\}$. We will make use of this notion in the chapter on recursion and induction.

Exercise 3.2.2 (1) (with solution) In Exercise 3.1 (2), we saw that the relations (i) $\{(a, |a|): a \in \mathbf{Z}\}$, (iii) $\{(a, a^2): a \in \mathbf{Z}\}$, (v) $\{(a, a+1): a \in \mathbf{Z}\}$, (vi) $\{(a+1, a): a \in \mathbf{Z}\}$ are functions from $\mathbf{Z}$ to $\mathbf{Z}$. Call them *mod*, *square*, *successor* and *predecessor*, respectively. Now recall the set $\mathbf{N} = \{0,1,2,3,\ldots\}$ of the natural numbers.

(a) Determine the image of $\mathbf{N}$ under each of these four functions.
(b) Restrict the four functions to $\mathbf{N}$. Which of them are functions into $\mathbf{N}$?
(c) Determine the images and closures of the set $A = \{-1,0,1,2\}$ under each of the four functions.

Solution

(a) *mod*$(\mathbf{N}) = \{|n|: n \in \mathbf{N}\} = \{n: n \in \mathbf{N}\} = \mathbf{N}$; *square*$(\mathbf{N}) = \{n^2: n \in \mathbf{N}\} = \{0,1,4, 9,\ldots\}$; *successor*$(\mathbf{N}) = \{n+1: n \in \mathbf{N}\} = \mathbf{N}^+$; *predecessor*$(\mathbf{N}) = \{n-1: n \in \mathbf{N}\} = \{-1\} \cup \mathbf{N}$.
(b) The only one that is not into $\mathbf{N}$ is the predecessor function, since $0-1=-1 \notin \mathbf{N}$. It is thus a function on $\mathbf{N}$ into $\{-1\} \cup \mathbf{N}$.
(c) *Mod*$(A) = \{0,1,2\}$ but *mod*$[A] = A$; *square*$(A) = \{0,1,4\}$ but *square*$[A] = \{0,1,2, 4,16,32,\ldots\}$; *successor*$(A) = \{0,1,2,3\}$ but *successor*$[A] = \{-1,0,1,2,\ldots\} = \{-1\}\cup\mathbf{N}$; *predecessor*$(A) = \{-2,-1,0,1\}$ but *predecessor*$[A] = \{\ldots,-2,-1,0, 1,2\} = \mathbf{Z}^-\cup\{0,1,2\}$.

Comments (1) The restriction of *mod*: $\mathbf{Z} \to \mathbf{Z}$ to $\mathbf{N}$ is clearly the set of all pairs $(n,n) \in \mathbf{N}$, that is, the function f: $\mathbf{N} \to \mathbf{N}$ that puts $f(n) = n$ for all $n \in \mathbf{N}$. This is known as the identity function over $\mathbf{N}$. We will have more to say about identity functions later in the chapter (2). In part (c), remember that although A is not always included in its image $f(A)$, it is always included in its closure $f[A]$.

3.2.3 Composition

Perhaps the most basic operation to apply to functions is *composition*. Suppose we are given two relations $R \subseteq A \times B$ and $S \subseteq B \times C$, so that the target of R is the source of S. Recall that their composition $S \circ R$ is the set of all ordered pairs (a,c) such that there is some x with both $(a,x) \in R$ and $(x,c) \in S$. Now consider the case that f is in fact a function from A to B and g is a function from B to C, briefly f: $A \to B$ and g: $B \to C$. Then $g \circ f$ is the set of all ordered pairs (a,c) such that there is some x with both $x = f(a)$ and $c = g(x)$; in other words, such that $c = g(f(a))$. It follows immediately that $g \circ f$ is a function from A to C, since for every $a \in A$ there is a unique $c \in C$ with $c = g(f(a))$.

To sum up, composition is an operation on relations which, when applied to functions f: $A \to B$ and g: $B \to C$ where the domain of g is the same as the target of f, gives us the function $g \circ f$: $A \to C$ defined by putting $g \circ f(a) = g(f(a))$.

Composition is associative. Let f: $A \to B$, g: $B \to C$, h: $C \to D$. Then $(h{\circ}(g{\circ}f)) = ((h{\circ}g){\circ}f)$ since for all $a \in A$, $(h{\circ}(g{\circ}f))(a) = h(g(f(a))) = ((h{\circ}g){\circ}f)(a)$. In rather tedious detail: on the one hand, $(g{\circ}f)$: $A \to C$ with $g{\circ}f(a) = g(f(a))$, so $h{\circ}(g{\circ}f)$: $A \to D$ with $(h{\circ}(g{\circ}f))(a) = h(g(f(a)))$. On the other hand, $(h{\circ}g)$: $B \to C$ with $(h{\circ}g)(b) = h(g(b))$, so $(h{\circ}g){\circ}f$: $A \to D$ with also $((h{\circ}g){\circ}f)(a) = h(g(f(a)))$.

However, composition of functions is not in general commutative. Let f: $A \to B$, g: $B \to C$. Then the composition $g{\circ}f$ is a function, but $f{\circ}g$ is not a function unless also $C = A$. Even in the case that $A = B = C$, so that $g{\circ}f$ and $f{\circ}g$ are both functions, they may not be the same function. For example, suppose that $A = B = C = \{1,2\}$, with $f(1) = f(2) = 1$ while $g(1) = 2$ and $g(2) = 1$. Then $g{\circ}f(1) = g(f(1)) = g(1) = 2$, while $f{\circ}g(1) = f(g(1)) = f(2) = 1$.

Incidentally, we can now see the answer to a question that was bothering Alice in the preceding chapter: why write the composition of relations R and S as $S{\circ}R$ rather than in the reverse order? When R, S are functions f: $A \to B$, g: $B \to C$ then $S{\circ}R$ becomes $g{\circ}f$, which corresponds nicely to the order on the right hand side in the definition of $g{\circ}f(a) = g(f(a))$.

Exercise 3.2.3

(a) Draw a diagram to illustrate the above example of non-commutativity of composition.

(b) Give an example of familiar functions f,g:$\mathbf{N} \to \mathbf{N}$ such that $f{\circ}g \neq g{\circ}f$.

(c) Show that we can generalize the definition of composition a little, in the sense that the composition $g{\circ}f$: $A \to C$ is a function whenever f: $A \to B$ and g: $B' \to C$ and $B \subseteq B'$. Give an example to show that $g{\circ}f$ may not be a function when, conversely, $B' \subseteq B$.

3.2.4 Inverse

We recall the notion of the *converse* (alias *inverse*) R^{-1} of a relation: it is the set of all ordered pairs (b,a) such that $(a,b) \in R$. The converse of a relation R is thus always a relation. But when R is a function f: $A \to B$, its converse is sometimes a function, sometimes not, and when it is a function it need not have the whole of B as its domain.

Exercise 3.2.4 (with partial solution)

(a) Let $A = \{1,2,3\}$ and $B = \{a,b,c,d\}$. Let $f = \{(1,a), (2,a), (3,b)\}$ and $g = \{(1,a), (2,b), (3,c)\}$. (i) Explain why neither $f^{-1} = \{(a,1), (a,2), (b,3)\}$ nor $g^{-1} = \{(a,1), (b,2), (c,3)\}$ is a function from B to A. (ii) Draw a diagram of the example.

(b) Give examples of (i) a function f: $\mathbf{Z} \to \mathbf{Z}$ whose inverse is not a function, (ii) a function g: $\mathbf{N} \to \mathbf{N}$ whose inverse is not a function.

Solution to (a)(i) f^{-1} is not a function at all, because we have both $(a,1)$, $(a,2)$ in it. On the other hand, g^{-1} is a function, but not from B to A, but rather from the proper subset $B' = \{a,b,c\} \subset B$ to A. It is thus a partial function from B to A.

Alice Box: Converse or Inverse?

Alice: In the preceding chapter, you talked on converses, now of inverses. Why is that?

Hatter: The general notions of a relation and a function were developed rather separately before being linked up by defining functions as special relations (as here) or relations as functions into the two-element set {0,1} (as in some other presentations). The term ‘converse’ grew up in the former community, ‘inverse’ in the latter.

3.3 Injections, Surjections, Bijections

The status of the inverse of a function, as just a relation or itself a function, is closely linked with two further concepts, injectivity and surjectivity, together constituting bijectivity, which we now examine.

3.3.1 Injectivity

Let f: $A \to B$. We say that f is *injective* (alias one-one) iff whenever $a \neq a'$, then $f(a) \neq f(a')$. In words, iff it takes distinct arguments to distinct values (distinct inputs to distinct outputs). Contrapositively, iff whenever $f(a) = f(a')$, then $a = a'$. In yet other words, iff for each $b \in B$, there is at most one $a \in A$ with $f(a) = b$.

Of the two functions f, g described in Exercise 3.2.4, f is not injective, since $f(1) = f(2)$ although $1 \neq 2$. However, g is injective since it takes distinct arguments to distinct values.

Exercise 3.3.1 (1) (with partial solution)

(a) In Exercise 3.1 (2) we saw that the relations (i) $\{(a, |a|): a \in \mathbf{Z}\}$, (iii) $\{(a, a^2): a \in \mathbf{Z}\}$, (v) $\{(a, a+1): a \in \mathbf{Z}\}$, (vi) $\{(a+1, a): a \in \mathbf{Z}\}$ are functions over **Z**, that is, from **Z** to **Z**, called *mod*, *square*, *successor* and *predecessor*, respectively. Which of them are injective?

(b) What does injectivity mean in terms of a digraph for the function?

(c) What does injectivity mean in terms of a table for the function?

(d) Show that the composition of two injective functions is injective, and give examples to show that the failure of injectivity in either of the two components can lead to failure of injectivity for the composition.

Solution to (a) Over **Z**, *mod* is not injective, since, for example, $|1| = |-1|$, likewise for *square* since, for example, $3^2 = (-3)^2$. On the other hand, *successor* is injective, since $a + 1 = a' + 1$ implies $a = a'$ for all $a,a' \in \mathbf{Z}$. Similarly, *predecessor* is injective, since $a = a'$ implies $a + 1 = a' + 1$.

Comment This exercise brings out the importance of always keeping clearly in mind the intended domain of the function when checking whether it is injective. Take for example the function of squaring. As we have just seen, when understood with domain **Z**, it is not injective. But when understood with domain **N** (i.e. as the restriction of the former function to **N**) then it is injective, since for all natural numbers a,b, $a^2 = b^2$ implies $a = b$.

Note that the injectivity of a function from A to B is not enough to guarantee that its inverse is a function from B to A. For example, as noted in the comments on the last exercise, the function of squaring on **N** (rather than on **Z**) is injective. But its inverse is not a function on **N**, since there are elements of **N**, for example, 5, which are not in its domain, not being the square of any natural number.

Nevertheless, injectivity gets us part of the way: it suffices to ensure that the inverse relation f^{-1} of a function from A to B is a function from $range(f) \subseteq B$ to A, and so is a *partial function* from B to A. Reason: From our work on relations, we know that f^{-1} must be a *relation* from $range(f)$ to A, so we need only show that for all $b \in B$ there is at most one $a \in A$ such that $(b,a) \in f^{-1}$, that is, at most one $a \in A$ with $(a,b) \in f$. But this is exactly what is given by the injectivity of f. Indeed, we can also argue in the converse direction, with the result that we have the following equivalence: *a function f: $A \rightarrow B$ is injective iff its inverse relation f^{-1} is a function from range(f) to A.*

Exercise 3.3.1 (2) What would be the natural way of defining injectivity for a function of two arguments?

Even when a function f: $A \rightarrow B$ is not injective, so that its inverse relation f^{-1} is not a function, there is still a closely related function from B into the *power set of* A. With a little abuse of notation and considerable risk of confusion for novices, it is also often written as f^{-1} or more completely as f^{-1}: $B \rightarrow \mathcal{P}(A)$, and is defined by putting $f^{-1}(b) = \{a \in A\colon f(a) = b\}$ for each $b \in B$. Complicated when you spell it out, but the underlying idea is simple and natural: you go back from b to the *set* of all a with $f(a) = b$. Perhaps surprisingly, the notion is often useful.

Exercise 3.3.1 (3) Draw an arrow-style diagram on small finite sets A, B (say, four elements in each) to illustrate the notion of the 'abstract inverse' function f^{-1}: $B \rightarrow \mathcal{P}(A)$.

3.3.2 Surjectivity

Let f: $A \rightarrow B$. We say that f is *onto* B or *surjective* (with respect to B) iff for all $b \in B$ there is some $a \in A$ with $f(a) = b$. In other words, iff every element of B is the value of some element of A under f. Equivalently, iff $range(f) = B$.

For example, if $A = \{1,2,3\}$ and $B = \{7,8,9\}$, then the function f that puts $f(1) = 9, f(2) = 8, f(3) = 7$ is onto B but is not onto its superset $B' = \{6,7,8,9\}$, since it 'misses out' the element $6 \in B'$.

Exercise 3.3.2 (1)

(a) What does surjectivity mean in terms of a digraph for the function?
(b) What does it mean in terms of a table for the function?

For some more substantive examples, we look to the number systems. Over **Z** both of the functions $f(a) = a + 1$ and $g(a) = 2a$ are injective, but only f is onto **Z**, since the odd integers are not in the range of g. However, if we restrict these two functions to the set **N** of natural numbers, then not even f is onto **N**, since $0 \in \mathbf{N}$ is not the successor of any natural number.

Exercise 3.3.2 (2) (with partial solution)

(a) Consider the function $f(a) = a^2$ over **N**, $\mathbf{N}^+$, **Z, Q**, **R,** $\mathbf{R}^+$, respectively, and determine in each case whether it is surjective.
(b) Show that the composition of two surjective functions is surjective, and give examples to show that the failure of surjectivity in either of the two components can lead to its failure for the composition.
(c) What would be the natural definition of surjectivity for a function of two arguments?

Solution to (a) For **N**, $\mathbf{N}^+$, **Z and Q**, the answer is negative, since, for example, 2 is in each of these sets, but there is no number a in any of these sets with $a^2 = 2$. In common parlance, 2 does not have a rational square root. For **R**, the answer is negative, since, for example, -1 does not have a square root, but for $\mathbf{R}^+$ the answer is positive: any $b \in \mathbf{R}^+$ there is an $a \in \mathbf{R}^+$ with $a^2 = b$.

Both of the terms 'onto' and 'surjective' are in common use, but the former is more explicit in that it makes it easier for us to say onto *what*. We can say simply 'onto B', whereas 'surjective with respect to B' is rather more longwinded. Whichever of the two terms one uses, it is important to be clear what set B is intended, since quite trivially *every* function is onto *some* set – namely, its range!

Alice Box: What Is f(x)?

Alice: I'm a bit puzzled by your notation in the exercise. When f is a function, then $f(a)$ is supposed to be the value of that function when the argument is a. Is that right?

Hatter: Yes.

Alice: But in part (a) of the exercise, you used $f(a)$ to stand for the function itself. What's going on?

Hatter: To be honest, it is a bit sloppy. When doing the theory, we follow the official notation and use f for the function itself and $f(a)$ for its value when the argument is a, and that is an element of the range of f. But when we talk about specific examples, it can be convenient to use an expression like $f(x)$ or $f(a)$ to stand for the function itself, with the variable just reminding us that it is one-place.

Alice: Shouldn't you get your act together and be consistent?

(continued)

(continued)

Hatter: I guess so . . . But in mitigation, it's not just this book, and it is not just perversity. It's done everywhere, in all presentations, because it really is so convenient, and everyone gets used to resolving the ambiguity unconsciously. So let's just apologize on behalf of the community, and carry on.

An apology is really needed! Half the trouble that students have with simple mathematics is due to quirks in the language in which it is conveyed to them. Usually, the quirks have a good reason for being there – usually one of convenience – but they do need to be explained. This is just one example; we will see others as we go on.

3.3.3 Bijective Functions

A function $f: A \rightarrow B$ that is both injective and onto B is said to be a *bijection* between A and B. An older term sometimes used is *one-one correspondence*. An important fact about bijectivity is that it is equivalent to the inverse relation being a function from B into A. That is, *a function $f: A \rightarrow B$ is a bijection between A and B iff its inverse f^{-1} is a function from B to A.*

Proof: By definition, f is a bijection between A and B iff f is injective and onto B. As we saw earlier in the chapter, $f: A \rightarrow B$ is injective iff f^{-1} is a partial function from B to A; and moreover f is onto B iff $range(f) = B$, that is, iff $dom(f^{-1}) = B$. Putting this together, f is a bijection between A and B iff f^{-1} is a partial function from B to A with $dom(f^{-1}) = B$, that is, iff f^{-1} is a function from B to A.

Alice Box: Chains of Iffs

Alice: Why didn't you prove this equivalence in the same way as before, breaking it into two parts, one the converse of the other, and proving each separately by conditional proof?

Hatter: We could perfectly well have done that. But in this particular case, the argument going in one direction would have turned out to be essentially the same as its converse, run backwards. So in this case, we can economize by doing it all as a chain of *iffs*. However, such an economy is not always available.

Exercise 3.3.3

(a) What is a bijection in terms of (i) a digraph, (ii) a table for the function?

(b) Show that the inverse of a bijection from A to B is a bijection from B to A. Hint: you know that it is a function from B to A, so you only need check out injectivity and surjectivity.

(c) Show that any injective function on a finite set into itself is a bijection.

Our explanations of injection, surjection and bijection have been formulated for one-place functions $f: A \to B$. However, as remarked in Sect. 3.1, this already covers functions of two or more places. Thus, a function $f: A \times B \to C$ is injective iff whenever $(a,b) \neq (a',b')$, $f(a,b) \neq f(a',b')$. It is surjective iff for all $c \in C$ there are $a \in A$, $b \in B$ with $c = f(a,b)$.

3.4 Using Functions to Compare Size

One of the many uses of functions is to compare the sizes of sets. In this section, we will consider only finite sets, although a more advanced treatment could also consider infinite ones. Recall from Chap. 1 that when A is a finite set, we write $\#(A)$ for the number of elements of A, also known as the *cardinality* of A. Another common notation for this is $|A|$. We will look at two principles, one for bijections and one for injections.

3.4.1 Equinumerosity

When do two finite sets A,B have the same number of elements? The *equinumerosity principle* gives us a necessary and sufficient condition: $\#(A) = \#(B)$ *iff there is some bijection* $f: A \to B$.

Proof. Suppose first that $\#(A) = \#(B)$. Since both sets are finite, there is some natural number n with $\#(A) = n = \#(B)$. Let $a_1, \ldots, a_n$ be the elements of A, and $b_1, \ldots, b_n$ those of B. Let $f: A \to B$ be the function that puts each $f(a_i) = b_i$. Clearly f is injective and onto B. For the converse, let $f: A \to B$ be a bijection. Suppose A has n elements $a_1, \ldots, a_n$. Then the list $f(a_1), \ldots, f(a_n)$ enumerates all the elements of B, counting none of them twice, so B also has n elements.

In general set theory, developed from its own axioms that do not mention anything from arithmetic, this condition is often used as a *definition* of when two sets have the same cardinality, covering the infinite case as well as the finite one. But in this book we consider only its application to finite sets.

Alice Box: Equinumerosity for Infinite Sets

Alice: OK, but just one question. Wouldn't such a definition give all infinite sets the same cardinality? Isn't there a bijection between any two infinite sets?

Hatter: At first sight it may seem so; and indeed the sets **N**, $\mathbf{N}^+$, **Z** and **Q** and even such sets as $\mathbf{Q}^n$ (for any positive integer n) are equinumerous in this sense: it is possible to find a bijection between any two of them. Any set that has a bijection with **N** is said to be *countable*, these sets are all countable. But one of Cantor's fundamental theorems was to show, surprisingly, that this is not the case for the set **R** of all real numbers. It is *uncountable*: there is no bijection between it and **N**. Further, he showed that there is no bijection between any set A and its power set $\mathcal{P}(A)$.

Alice: How can you prove that?

Hatter: The proof is ingenious and short. It uses what is known as the 'diagonal construction'. But it would take us out of our main path. You will find it in any good introduction to set theory for students of mathematics, for example, Paul Halmos' *Naïve Set Theory*.

A typical application of the equinumerosity principle, or more strictly of its right-to-left half, takes the following form. We have two sets A,B and want to show that they have the same cardinality, so we look for some function that is a bijection between the two; if we find one, then we are done. The following exercise gives two examples. In the first, finding a suitable bijection completes the proof; in the second, it serves to reset the problem in a more abstract but more amenable form.

Exercise 3.4.1 (with solution)

(a) Let A be the set of sides of a polygon, and B the set of its vertices. Show $\#(A) = \#(B)$.

(b) In a reception, everyone shakes hands with everyone else just once. If there are n people in the reception, how many handshakes are there?

Solution

(a) Let $f: A \to B$ be defined by associating each side with its right endpoint. Clearly this is both injective and is onto B. Hence $\#(A) = \#(B)$.

(b) Let A be the set of the handshakes, and let B be the set of all unordered pairs $\{x,y\}$ of distinct people (i.e. $x \neq y$) from the reception. We apply the equinumerosity principle by defining the function $f: A \to B$ that associates each handshake in A with the two distinct people in B involved in it. Clearly this gives us a bijection between A and B. The rest of the solution works on the set B. Since A and B have the same cardinality, we need only ask how many elements there are in B. It is not difficult to see that there must be $n.(n-1)$

ordered pairs of distinct people in the reception: we can choose any one of the n to fill the first place, then choose any one of the remaining $n-1$ to fill the second place. Thus, there are $n.(n-1)/2$ *unordered* pairs, since each unordered pair $\{x,y\}$ with $x \neq y$ corresponds to two distinct ordered pairs (x,y) and (y,x). Thus there are $n.(n-1)/2$ handshakes.

Don't feel bad if you can't work out the second example for yourself; we will be getting back to this sort of thing in Chap. 5 (on counting), and in Chap. 4 we will explain a quite different way of tackling the problem (by induction). But for the present, we note that the sought-for bijection can surprisingly often be a quite simple and obvious one. Sometimes, as in the second example, it associates the elements of a rather 'everyday' set (like the set of all handshakes) with a rather more 'abstract' one (like the set of all unordered pairs $\{x,y\}$ of distinct people from the reception), so that we can forget about irrelevant information concerning the former and apply counting rules to the latter.

3.4.2 Cardinal Comparison

When is one finite set at least as large as another? The *principle of comparison* tells us that for finite sets A,B: $\#(A) \leq \#(B)$ *iff there is some injective function* $f: A \rightarrow B$.

Proof. We use the same sort of argument as for the equinumerosity principle. Suppose first that $\#(A) \leq \#(B)$. Since both sets are finite, there is some $n \geq 0$ with $\#(A) = n$, so that $\#(B) = n + m$ for some $m \geq 0$. Let $a_1, \ldots, a_n$ be the elements of A, and $b_1, \ldots, b_n, \ldots, b_{n+m}$ those of B. Let $f: A \rightarrow B$ be the function that puts each $f(a_i) = b_i$. Clearly, f is injective (but not necessarily onto) B. For the converse, let $f: A \rightarrow B$ be injective. Suppose A has n elements $a_1, \ldots, a_n$. Then the list $f(a_1), \ldots, f(a_n)$ enumerates some (but not necessarily all) the elements of B, counting none of them twice, so B has at least n elements, that is, $\#(A) \leq \#(B)$.

Once again, general set theory turns this principle into a definition of the relation $\leq$ between cardinalities of sets, both finite and infinite. But that is beyond us here.

3.4.3 The Pigeonhole Principle

In applications of the principle of comparison, we typically use only the right-to-left half of it, formulating it contrapositively as follows. Let A,B be finite sets. If $\#(A) > \#(B)$, then no function $f: A \rightarrow B$ is injective. In other words, if $\#(A) > \#(B)$, *then for every function* $f: A \rightarrow B$ *there is some* $b \in B$ *such that* $b = f(a)$ *for two distinct* $a \in A$.

This simple rule is known as the *pigeonhole principle*, from the example in which A is a (large) set of memos to be delivered to people in an office, and B is the (small) set of their pigeonholes. It also has a more general form, as follows. *Let A,B be* *finite sets. If* $\#(A) > k{\cdot}\#(B)$ *then for every function* $f\colon A \to B$ *there is* a $b \in B$ *such that* $b = f(a)$ for at least $k + 1$ *distinct* $a \in A$.

The pigeonhole principle is surprisingly useful as a way of showing that, in suitable situations, at least two distinct items must have a certain property. The wealth of possible applications can only be appreciated by looking at examples. We begin with a very straightforward one.

Exercise 3.4.3 (1) (with solution) A village has 400 inhabitants. Show that at least two of them have the same birthday, and that at least 34 are born in the same month of the year.

Solution Let A be the set of people in the village, B the set of the days of the year, C the set of months of the year. Let $f\colon A \to B$ associate with each villager his or her birthday, while $g\colon A \to C$ associates each villager with the month of birth. Since $\#(A) = 400 > 366 \geq \#(B)$ the pigeonhole principle tells us that there is a $b \in B$ such that $b = f(a)$ for two distinct $a \in A$. This answers the first part of the question. Also, since $\#(A) = 400 > 396 = 33{\cdot}\#(C)$, the generalized pigeonhole principle tells us that there is a $c \in C$ such that $c = f(a)$ for at least $33 + 1 = 34$ distinct $a \in A$, which answers the second part of the question.

In this exercise, it was quite obvious what sets A,B and function $f\colon A \to B$ to choose. In other examples, this may need more reflection, and sometimes considerable ingenuity.

Exercise 3.4.3 (2) (with solution) In a club, everyone has just one given name and just one family name. It is decided to refer to everyone by two initials, the first initial being the first letter of the given name, the second initial being the first letter of the family name. How many members must the club have to make it inevitable that two distinct members are referred to by the same initials?

Solution It is pretty obvious that we should choose A to be the set of members of the club. Let B be the set of all ordered pairs of letters from the alphabet. The function $f\colon A \to B$ associates with each member a pair as specified in the club decision. Assuming that we are dealing with a standard English alphabet of 26 letters, $\#(B) = 26{\cdot}26 = 676$. So if the club has 677 members, the pigeonhole principle guarantees that $f(a) = f(a')$ for two distinct a, $a' \in A$.

When solving problems by the pigeonhole principle, always begin by choosing carefully the sets A,B and the function $f\colon A \to B$. Indeed, this will often be the key step in finding the solution. Sometimes, as in the above example, the elements of B will be rather abstract items, such as ordered n-tuples.

Exercise 3.4.3 (3) A multiple-choice exam has five questions, each with two possible answers. Assuming that each student enters a cross in exactly one box of each question (no unanswered, over-answered or spoiled questions), how many students need to be in the class to guarantee that at least four students submit the same answer paper?

3.5 Some Handy Functions

In this section, we look at some simple functions that appear over and again: the identity, constant, projection and characteristic functions. The student may find it difficult to assimilate all four in one sitting. No matter, they are here to be understood and then recalled whenever needed.

3.5.1 Identity Functions

Let A be any set. The identity function on A is the function $f: A \to A$ such that for all $a \in A$, $f(a) = a$. As simple as that! It is sometimes written as i_A, or with the Greek letter iota as ι_A. It is very important in abstract algebra, and comes up often in computer science, never at the centre of attention but (like the restriction of a function) as an everyday tool used almost without thinking.

Exercise 3.5.1

(a) Show that the identity function over any set is a bijection, and is its own inverse.
(b) Let $f: A \to B$. Show that $f \circ i_A = f = i_B \circ f$.
(c) Show that if $A \neq B$ then $i_A \neq i_B$.

Alice Box: One Identity Function or Many?

Alice: So, for every choice of set A you get a different identity function i_A?
Hatter: Strictly speaking, yes, and we showed it in the last exercise.
Alice: Why not define a great big identity function, for once and for all, by putting $i(a) = a$ for every a whatsoever?
Hatter: Attractive, but there is a technical problem. What would its domain and range be?
Alice: The set of all things whatsoever – call it the universal set.
Hatter: Unfortunately, as we saw when you asked about absolute complementation in Chap. 1, standard set theory does not admit such a set, on pain of contradiction. So we must relativize the concept to whatever 'local universe' set U we are working in. A little bit of a bother, but not too inconvenient in practice.

3.5.2 Constant Functions

Let A,B be non-empty sets. A *constant function* on A into B is any function $f: A \to B$ such that $f(a) = f(a')$ for all a,$a' \in A$. Equivalently, iff there is some $b \in B$ such that $b = f(a)$ for all $a \in A$.

The order of the quantifiers is vital in the second of these formulations. We require that $\exists b \in B\ \forall a \in A$: $b = f(a)$, in other words that all the elements of A

have the same value under f. This is much stronger than requiring merely that $\forall a \in A\ \exists b \in B$: $f(a) = b$; in fact, *that* condition holds for any function f whatsoever! In general, there is an immense difference between statements of the form $\forall x \exists\ y(\ldots)$ and corresponding ones of the form $\exists y \forall x(\ldots)$. In a later chapter on the logic of the quantifiers, we will set out systematically the relations between the various propositions $\mathrm{Q}x\mathrm{Q}'y(\ldots)$, where Q and Q′ are quantifiers (universal or existential), x and y are their attached variables, where the expression $(\ldots)$ is kept fixed.

Exercise 3.5.2

(a) Fix non-empty sets A,B with $\#(A) = m$ and $\#(B) = n$. How many constant functions f: $A \rightarrow B$ are there?

(b) Show that when $\#(A) > 1$, no constant function f: $A \rightarrow B$ is injective, and when $\#(B) > 1$ no constant function f: $A \rightarrow B$ is onto B.

3.5.3 Projection Functions

Let f: $A{\times}B \rightarrow C$ be a function of two arguments, and let $a \in A$. By the *right projection of f from a* we mean the one-argument function f_a: $B \rightarrow C$ defined by putting $f_a(b) = f(a,b)$ for each $b \in B$.

Likewise, letting $b \in B$, the *left projection of f from b* is the one-argument function f_b: $A \rightarrow C$ defined by setting $f_b(a) = f(a,b)$ for each $a \in A$.

In other words, to form the left or right projection of a (two-argument) function, we hold one of the arguments of f fixed at some value and consider the (one-argument) function obtained by allowing the other argument to vary.

Exercise 3.5.3

(a) What is the right projection of the two-argument function f: $\mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N}$ defined by putting $f(m.n) = m^n$, when m is chosen to be 2? And the left projection when n is chosen to be 3?

(b) Describe the right projection of the multiplication function $x \cdot y$ at the value $x = 3$. Also the left projection at the value $y = 3$. Are they the same function?

3.5.4 Characteristic Functions

Let U be any set fixed as a local universe. For each subset $A \subseteq U$ we can define a function f_A: $U \rightarrow \{1,0\}$ by putting $f_A(u) = 1$ when $u \in A$ and $f_A(u) = 0$ when $u \notin A$. This is known as the *characteristic function* of A (modulo U). Thus the characteristic function f_A specifies the truth-value of the statement that $u \in A$.

Conversely, when f: $U \rightarrow \{1,0\}$, we can define the *associated subset* of U by putting $A_f = \{u \in U: f(a) = 1\}$. Clearly, there is a bijection between the subsets of U and the functions f: $U \rightarrow \{1.0\}$, and in fact, we can make either do the work of the other. In some contexts, it can be conceptually or notationally more convenient to work with characteristic functions than with subsets.

Exercise 3.5.4

(a) What is the characteristic function of the set of all prime numbers, in the local universe $\mathbf{N}^+$ of positive integers? Of the composites (non-primes)? Of the empty set, and of $\mathbf{N}^+$ itself?

(b) Let U be a local universe, and let $A,B \subseteq U$. Express the characteristic function of $A \cap B$ in terms of those for A,B. Then do the same for $A \cup B$ and for $U \backslash A$. Do these remind you of anything from earlier logic boxes?

3.6 Families and Sequences

The uses of functions are endless. In fact, their role is so pervasive that some mathematicians prefer to see them, rather than sets, as providing the bedrock of their discipline. When that is done, sets are defined, roughly speaking, as their characteristic functions. We will not go further into that perspective, which belongs rather to the philosophy of mathematics. Instead, we illustrate the versatility of functions by seeing how they can clarify the notion of a *family of sets*, and a *sequence* of arbitrary items.

3.6.1 Families of Sets

In Sect. 1.5, we introduced sets of sets. They are usually written $\{A_i: i \in I\}$ where the A_i are sets and the set I, called an *index set*, helps us keep track of them. Because the phrase 'set of sets' tends to be difficult for the mind to process, we also speak of $\{A_i: i \in I\}$ as a *collection* of the sets A_i; but the term 'collection' does not mean anything new – it is merely to facilitate communication.

We now introduce the subtly different concept of a *family* of sets. This refers to any *function* on a domain I (called an *index set*) such that for each $i \in I, f(i)$ is a set. Writing $f(i)$ as A_i, it is thus the set of all ordered pairs $(i, f(i)) = (i, A_i)$ with $i \in I$. The range of this function is the collection $\{A_i: i \in I\}$.

The difference is subtle, and in some contexts sloppiness does not matter. The collection notation $\{A_i: i \in I\}$ is sometimes used also for the family. But in certain situations, notably applications of the pigeonhole principle and other counting rules that we will come to in a later chapter, the difference is very important. Suppose, for example, that the index set I has n elements, say $I = \{1, \ldots n\}$, but the function f is not injective, say $f(1) = f(2)$, that is, $A_1 = A_2$. In that case, the *family* containing all the pairs (i, A_i) with $i \leq n$, has n elements, but the *collection*, containing all the sets A_i with $i \leq n$, has fewer elements since $A_1 = A_2$.

Once more, the substance of mathematics is intricately entwined with the way that it uses language. The convention of subscripting items with a variable is often an implicit way of describing a function. For example, when we say 'let p_i be the ith prime number, for any $i \in \mathbf{N}^+$', then we are implicitly considering the function

p: $\mathbf{N}^+ \to \mathbf{N}^+$ such that each $p(i)$ is the ith prime number. However, the conventional notation with subscripts is often easier to read as it gets rid of brackets, and also focuses attention on the numbers themselves with the subscripts merely facilitating cross-reference. Moreover, in some contexts all we really need to know about the family function is its range, so *in those contexts* we can replace it by a simple collection of sets.

Exercise 3.6.1 Let I be a set with n elements, and f a constant function on I into a collection of sets. (i) How many elements are there in the collection $\{A_i: i \in I\} = \{f(i): i \in I\}$. How many elements are there in the family $\{(i, A_i)\}: i \in I\} = \{(i, f(i))\}: i \in I\}$? (ii) What if f is an injective function?

3.6.2 Sequences and Suchlike

In computer science as in mathematics itself, we often need to consider sequences $a_1,a_2,a_3,\ldots$ of items. The items a_i might be numbers, sets or other mathematical objects; in computer science they may be the instructions in a program or steps in its execution. The sequence itself may be finite, with just n terms $a_1,\ldots,a_n$ for some n, or infinite with a term a_i for each positive integer i, in which case we usually write it in an informal suspended dots notation, as $a_1,a_2,a_3,\ldots$. But what is such a sequence?

It is convenient to identify an infinite sequence $a_1,a_2,a_3,\ldots$ with a function f: $\mathbf{N}^+ \to A$ for some appropriately chosen set A, with $f(i) = a_i$ for each $i \in \mathbf{N}^+$. The ith term in the sequence is thus just the value of the function for argument (input) i. The sequence is thus just a family with index set $\mathbf{N}^+$.

When the sequence is finite, there are two ways to go. We can continue to identify it with a function f: $\mathbf{N}^+ \to A$ with $f(i) = a_i$ for each $i \le n \in \mathbf{N}^+$ and with $f(n+j) = f(n)$ for all $j \in \mathbf{N}^+$, so that the function becomes constant in its value from $f(n)$ upwards. Or, more intuitively, we can take it to be a function f: $\{i \le n \in \mathbf{N}^+\} \to A$, that is, as a partial function on $\mathbf{N}^+$ with domain $\{i \le n \in \mathbf{N}^+\}$. It doesn't really matter which way we do it; in either case we have made a rather vague notion sharper by explaining it in terms of functions.

Alice Box: n-Tuples, Sequences, Strings and Lists

Alice: I'm getting confused! So a finite *sequence* of elements of A is a function f: $\{i \le n \in \mathbf{N}^+\} \to A$ and we write it as $a_1,a_2,\ldots,a_n$. But what's the difference between this and the ordered *n-tuple* $(a_1,a_2,\ldots,a_n)$ that we saw back in Sect. 2.1.1? While we are at it, I have been reading around, and I also find computer scientists talking about finite *strings* and finite *lists*. Are they the same, or different?

Hatter: Well, er…

(continued)

(continued)

Alice: They come at me with different notations. Sequences are written as $a_1,a_2,\ldots,a_n$ and tuples with external brackets as in $(a_1,a_2,\ldots,a_n)$, but strings are written just as $a_1a_2\ldots a_n$ with neither commas nor external brackets. I see lists written with angular external brackets and sometimes with further internal brackets. And the symbols used for the empty tuple, sequence, string and list are all different. Help! What is going on?

Let's try to help Alice and get the Hatter off the hook, focussing on the finite case. This sort of thing is rarely explained, and it can be quite confusing to the beginning student.

The most abstract concept is that of a *tuple*. If we go back to the account in Sect. 2.1.1, we see that we don't care what it *is*. All we care about is how it *behaves*, and in particular the criterion for identity that we mentioned there: $(x_1,x_2,\ldots,x_n)=(y_1,y_2,\ldots,y_n)$ iff $x_i=y_i$ for all i from 1 to n. An n-tuple can be *anything that satisfies that condition*.

A *sequence* is a more specific kind of object. In this section, we have defined a finite sequence of n items to be a *function* f: $\{i \le n \in \mathbf{N}^+\} \to A$. Sequences thus satisfy the identity criterion just mentioned, and so we may regard them as a particular kind of tuple. Mathematicians like them because they are accustomed to working with functions on $\mathbf{N}^+$.

That leaves us with *strings* and *lists*. They are best thought of as tuples, usually of symbols, that come equipped with a tool for putting them together and taking them apart.

In the case of strings, this tool is the operation of *concatenation*, which consists of taking any strings s and t and forming a string $con(s,t)$. For symbols, this is the longer symbol formed by writing s and then immediately to its right the other symbol t. When we talk of strings, we are thinking of tuples where concatenation is the only available way, or the only permitted one, of building or dismantling them. Computer scientists like them, as concatenation is an operation that their machines can work with quickly.

Lists may also be thought of as tuples, likewise equipped with a single tool for their construction and decomposition. But this time the tool is different, and more limited in its power. It can be thought of as a restricted form of concatenation. We are allowed to take any list y and put in front of it an element a of the base set A of elementary symbols (often called the *alphabet* for constructing lists). This forms a slightly longer list $\langle a,y\rangle$. Here a is called the *head*, and y is the *tail* of the list $\langle a,y\rangle$. If y itself is compound, say $y=\langle b,x\rangle$ where $b \in A$ and x is a shorter list, then $\langle a,y\rangle = \langle a, \langle b,x\rangle\rangle$, and so on. Being a restricted form of concatenation, the operation is given the very similar name *cons*, so that $cons(a,y)=\langle a,y\rangle$. The important thing about the *cons* operation is that while it can take any list of any length as its second argument, it can take only elements of the basic alphabet A as its first argument. It is in effect a restriction of the first argument of the concatenation operation to the alphabet set A.

This restriction may at first seem rather odd, but there is a reason for making it. For the restricted kind of concatenation, lists satisfy a mathematical condition of *unique decomposability*, which concatenation in general does not satisfy. That permits us to carry out definitions by structural recursion, as we will explain in Chap. 4.

To sum up: For tuples we don't care what they are, so long as they satisfy the appropriate condition for identity. Sequences may be defined as functions on initial segments of the positive integers (or the natural numbers, if preferred) and are a particular kind of tuple. Strings and lists are tuples accompanied by a tool for their construction and decomposition – concatenation in one case and a restricted form of concatenation in the other.

End-of-Chapter Exercises

Exercise 3.1: Partial functions

(a) Characterize the notion of a partial function from A to B in terms of (i) its table as a relation and (ii) its digraph as a relation.

(b) Let R be a relation from A to B. Show that it is a partial function from A to B iff it is a function from $dom(R)$ to B.

Exercise 3.2: Image, closure

(a) The *floor function* from $\mathbf{R}$ into $\mathbf{N}$ is defined by putting $\lfloor x \rfloor$ to be the largest integer less than or equal to x. What are the images under the floor function of the sets $[0,1] = \{x \in \mathbf{R}\text{: } 0 \leq x \leq 1\}$, $[0,1) = \{x \in \mathbf{R}\text{: } 0 \leq x < 1\}$, $(0,1] = \{x \in \mathbf{R}\text{: } 0 < x \leq 1\}$, $(0,1) = \{x \in \mathbf{R}\text{: } 0 < x < 1\}$?

(b) Let $f\text{: } A \rightarrow A$ be a function from set A into itself. Show that for all $X \subseteq A$, $f(A) \subseteq f[A]$, and give a simple example of the failure of the converse inclusion.

(c) Show that when $f(A) \subseteq A$ then $f[A] = A$.

(d) Show that for any partition of A, the function f taking each element $a \in A$ to its cell is a function on A into the power set $\mathcal{P}(A)$ of A with the partition as its range.

(e) Let $f\text{: } A \rightarrow B$ be a function from set A into set B. Recall the 'abstract inverse' function $f^{-1}\text{: } B \rightarrow \mathcal{P}(A)$ defined at the end of Sect. 3.3.1 by putting $f^{-1}(b) = \{a \in A\text{: } f(a) = b\}$ for each $b \in B$. (i) Show that the collection of all sets $f^{-1}(b)$ for $b \in f(A) \subseteq B$ is a partition of A in the sense defined in Chap. 2. (ii) Is this still the case if we include in the collection the sets $f^{-1}(b)$ for $b \in B \backslash f(A)$?

Exercise 3.3: Injections, surjections, bijections

(a) Is the floor function from $\mathbf{R}$ into $\mathbf{N}$ injective? (ii) Is it onto $\mathbf{N}$?

(b) Show that the composition of two bijections is a bijection. You may make use of results of exercises in Sects. 3.3.1 and 3.3.2 on injectivity and surjectivity.

(c) Use the equinumerosity principle to show that there is never any bijection between a finite set and any of its proper subsets.

(d) Give an example to show that there can be a bijection between an infinite set and certain of its proper subsets.

(e) Use the principle of comparison to show that for finite sets A,B, if there are injective functions f: $A \rightarrow B$ and g: $B \rightarrow A$, then there is a bijection from A to B.

Exercise 3.4: Pigeonhole principle

(a) Use the general form of the pigeonhole principle to show that of any seven propositions, there are at least four with the same truth-value.

(b) If a set A is partitioned into n cells, how many distinct elements of A need to be selected to guarantee that at least two of them are in the same cell?

(c) Let $K = \{1,2,3,4,5,6,7,8\}$. How many distinct numbers must be selected from K to guarantee that there are two of them that sum to 9? *Hint*: Let A be the set of all unordered pairs $\{x,y\}$ with $x,y \in K$ and $x + y = 9$; check that this set forms a partition of K and apply the preceding part of the exercise.

Exercise 3.5: Handy functions

(a) Let f: $A \rightarrow B$ and g: $B \rightarrow C$. (i) Show that if at least one of f,g is a constant function, then $g{\circ}f$: $A \rightarrow C$ is a constant function. (ii) If $g{\circ}f$: $A \rightarrow C$ is a constant function, does it follow that at least one of f,g is a constant function (give a verification or a counterexample).

(b) What would be the natural way of defining the projection of an n-place function from its ith argument place, thus generalizing the notions of left and right projection.

(c) Let f: $A \times B \rightarrow C$ be a function of two arguments. Verify or give counterexamples to the following. (i) If f is injective, then every (left and right) projection of f is also injective. (ii) If f is surjective, then every projection of f is also surjective.

(d) What would be the natural way to define the characteristic function of a two-place relation $R \subseteq U \times U$ over a universe U?

Selected Reading

The texts listed at the end of Chap. 2 on relations also have useful chapters on functions. In detail:

Bloch ED (2011) Proofs and fundamentals: a first course in abstract mathematics, 2nd edn. Springer, New York, chapter 4

Halmos PR (2001) Naive set theory, New edn. Springer, New York, chapters 8–10

Hein JL (2002) Discrete structures, logic and computability, 2nd edn. Jones and Bartlett, Boston, chapter 2

Lipschutz S (1998) Set theory and related topics, Schaum's outline series. McGraw Hill, New York, chapters 4–5

Velleman DJ (2006) How to prove it: a structured approach, 2nd edn. Cambridge University Press, Cambridge, chapter 5

Recycling Outputs as Inputs: Induction and Recursion

4

Abstract

This chapter introduces induction and recursion, which are omnipresent in computer science. The simplest context in which they arise is in the domain of the positive integers, and that is where we begin. We explain induction as a method for *proving facts about the positive integers*, and recursion as a method for *defining functions* on the same domain. We will also describe two different methods for *evaluating* such functions.

From this familiar terrain, the basic concepts of recursion and induction can be extended to structures, processes and procedures of many kinds, not only numerical ones. Particularly useful for computer scientists are the forms known as *structural induction and recursion*, and we give them special attention. We will look at structural recursion as a way of *defining sets*, structural induction as a way of *proving* things about those sets, and then structural recursion once more as a way of *defining functions with recursively defined domains*. At this last point, special care is needed, as the definitions of such functions succeed only when a special condition of *unique decomposability* is satisfied which, happily, is the case in many computer science applications.

The broadest and most powerful kind of induction/recursion may be formulated for sets of any kind, provided they are equipped with a relation that is *well founded*, in a sense we explain. All other kinds may be seen as special cases of that one. In a final section, we look at the notion of a recursive *program* and see how the ideas that we have developed in the chapter are manifested there.

4.1 What Are Induction and Recursion?

The two words are used in different contexts. 'Induction' is the term more commonly applied when talking about *proofs*. 'Recursion' is the one used in connection with *constructions and definitions*. We will follow this tendency, speaking of

D. Makinson, *Sets, Logic and Maths for Computing*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-2500-6_4,

inductive proofs and recursive definitions. But it should not be thought that they answer to two fundamentally different concepts: the same basic idea is involved in each.

What is this basic idea? It will help if we look for a moment at the historical context. We are considering an insight that goes back to ancient Greece and India, but whose explicit articulation had difficulty breaking free from a longstanding rigidity. From the time of Aristotle onwards, it was a basic tenet of logic, and of science in general, that nothing should ever be defined in terms of itself on pain of making the definition circular. Nor should any proof assume what it is setting out to prove, for that too would create circularity.

Taken strictly, these precepts remain perfectly true. But it was realized that definitions, proofs and procedures may also 'call upon themselves', in the sense that later steps may systematically appeal to the outcome of earlier steps. The value of a function for a given value of its argument may be defined in terms of its value for smaller arguments; likewise, a proof of a fact about an item may assume that we have already proven the same fact about lesser items; and an instruction telling us how to carry out steps of a procedure or program may specify this in terms of what previous steps have already done. In each case, what we need to keep control is a *clear stepwise ordering of the domain we are working on, with a clearly specified starting point*.

What do these two requirements mean? To clarify them, we begin by looking at the simplest context in which they are satisfied: the positive integers. There we have a definite starting point, 1. We also have a clear stepwise ordering, namely, the passage from any number n to its immediate successor $s(n)$, more commonly written $n + 1$. This order exhausts the domain in the sense that every positive integer may be obtained by applying the step finitely many times from the starting point.

Not all number systems are of this kind. For example, the set **Z** of all integers, negative as well as positive, has no starting point under its natural ordering. The set **Q** of rational numbers not only lacks a starting point, but it also has the wrong kind of order: no rational number has an immediate successor. Likewise for the set **R** of reals.

Conversely, the use of recursion and induction need not be confined to number systems. They can be carried out in any structure satisfying certain abstract conditions that make precise the italicized requirement above. Such non-numerical applications are very important for computer science but are often neglected in accounts written from a traditional mathematical perspective. We will come to them later in the chapter.

4.2 Proof by Simple Induction on the Positive Integers

We begin with a simple example familiar from high school, and then articulate the principle lying behind it.

4.2.1 An Example

Suppose that we want to find a formula that identifies explicitly the sum of the first n positive integers. We might calculate a few cases, seeing that $1 = 1$, $1 + 2 = 3$, $1 + 2 + 3 = 6$, $1 + 2 + 3 + 4 = 10$, etc. In other words, writing $\Sigma\{i\text{: } 1 \leq i \leq n\}$ or $\Sigma_i\{1 \leq i \leq n\}$ or just $f(n)$ for the sum of the first n even positive integers, we see by calculation that $f(1) = 1$, $f(2) = 3$, $f(3) = 6$, $f(4) = 10$, etc. After some experimenting, we may hazard the conjecture (or read somewhere) that quite generally $f(n) = n{\cdot}(n + 1)/2$. But how can we *prove* this?

If we continue calculating the sum for specific values of n without ever finding a counterexample to the conjecture, we may become more and more convinced that it is correct, but that will never give us a *proof* that it is so. For no matter how many specific instances we calculate, there will always be infinitely many still to come. We need another method – and that is supplied by simple induction. Two steps are needed:

- The first step is to note that the conjecture $f(n) = n{\cdot}(n + 1)/2$ holds for the initial case that $n = 1$, in other words, that $f(1) = 1{\cdot}(1 + 1)/2$. This is a matter of trivial calculation, since we have already noticed that $f(1) = 1$ and clearly also $1{\cdot}(1 + 1)/2 = 1$. This step is known as the *basis* of the inductive proof.
- The second step is to prove a general statement: whenever the conjecture $f(n) = n{\cdot}(n + 1)$ holds for $n = k$, then it holds for $n = k + 1$. In other words, we need to show that for all positive integers k, if $f(k) = k{\cdot}(k + 1)/2$, then $f(k + 1) = (k + 1){\cdot}(k + 2)/2$. This *general if-then* statement is known as the *induction step* of the proof. It is a universally quantified conditional statement. Note how the equality in its consequent is formulated by substituting $k + 1$ for k in the antecedent.

Taken together, these two are enough to establish our original conjecture. The first step shows that the conjecture holds for the number 1. The induction step may then be applied to that to conclude that it also holds for 2, but it may also be applied to *that* to conclude that the conjecture also holds for 3, and so on for any positive integer n. We don't actually have to perform all these applications one by one; indeed, we couldn't do so, for there are infinitely many of them. But we have a guarantee, from the induction step, that each of these applications could be made.

In the example, how do we go about proving the induction step? As it is a universally quantified conditional statement about all positive integers k, we proceed in the standard way described in an earlier logic box. We *let* k be an arbitrary positive integer, *suppose* the antecedent to be true, and *show* that the consequent must also be true.

In detail, *let* k be an arbitrary positive integer. *Suppose* $f(k) = k(k + 1)/2$. We need to *show* that $f(k + 1) = (k + 1)(k + 2)/2$. Now:

$$\begin{aligned}
f(k+1) &= 1+2+\cdots+k+(k+1) && \text{by the definition of the function } f\\
&= (1+2+\cdots+k)+(k+1) && \text{arranging brackets}\\
&= f(k)+(k+1) && \text{by the definition of the function } f \text{ again}\\
&= [k\cdot(k+1)/2]+(k+1) && \textit{by the supposition } f(k)=k\cdot(k+1)/2\\
&= [k\cdot(k+1)+2(k+1)]/2 && \text{by elementary arithmetic}\\
&= (k^2+3k+2)/2 && \text{ditto}\\
&= (k+1)\cdot(k+2)/2 && \text{ditto.}
\end{aligned}$$

This completes the proof of the induction step, and thus of the proof as a whole! The key link in the chain of equalities is the underlined one, where we apply the supposition.

4.2.2 The Principle Behind the Example

The rule used in this example is called the *simple principle of mathematical induction*, and may be stated as follows. Consider any property that is meaningful for positive integers. To prove that every positive integer has the property, it suffices to show:

Basis: The least positive integer 1 has the property.
Induction step: Whenever a positive integer k has the property, then so does $k+1$.

The same principle may be stated in terms of sets rather than properties. Consider any set $A \subseteq \mathbf{N}^+$. To establish that $A = \mathbf{N}^+$, it suffices to show two things:

Basis: $1 \in A$.
Induction step: Whenever a positive integer $k \in A$, then also $(k+1) \in A$.

As in our example, checking the basis is usually a matter of trivial calculation. Establishing the induction step is also carried out in the same way as in the example: we *let* k be an arbitrary positive integer, *suppose* that k has the property (that $k \in A$), and *show* that $k+1$ has the property (that $k+1 \in A$). In our example, that was easily done; tougher problems require more sweat, but still within the same general framework.

Important terminology: Within the induction step, the supposition that k has the property, is called the *induction hypothesis*. What we set out to show from that supposition, that is, that $k+1$ has the property, is called the *induction goal*.

Exercise 4.2.2 (1) (with solution) Use the principle of induction over the positive integers to show that for every positive integer n, the sum of the first n odd integers is n^2.

Solution Write $f(n)$ for the sum of the first n odd integers, that is, for $1+3+\cdots+(2n-1)$. We need to show that $f(n) = n^2$ for every positive integer n.

Basis: We need to show that $f(1) = 1^2$. But clearly, $f(1) = 1 = 1^2$, and we are done.

Induction step: Let k be any positive integer, and suppose (induction hypothesis) that the property holds when $n = k$, that is, suppose that $f(k) = k^2$. We need to show (induction goal) that it holds when $n = k + 1$, that is, that $f(k + 1) = (k + 1)^2$. Now:

$f(k+1) = 1 + 3 + \cdots + (2k-1) + (2(k+1) - 1)$	by definition of f
$= (1 + 3 + \cdots + (2k-1)) + (2(k+1) - 1)$	arranging brackets
$= f(k) + 2(k+1) - 1$	also by definition of f
$= k^2 + 2(k+1) - 1$	*by the induction hypothesis*
$= k^2 + 2k + 1$	by elementary arithmetic
$= (k+1)^2$	ditto

What if we wish to prove that every *natural number* has the property we are considering? We proceed in exactly the same way, except that we start with 0 instead of 1:

Basis: The least natural number 0 has the property.
Induction step: Whenever a natural number k has the property, then so does $k + 1$.

In the language of sets:

Basis: $0 \in A$.
Induction step: Whenever a natural number $k \in A$, then also $(k + 1) \in A$.

Sometimes it is more transparent to state the induction step in an equivalent way using subtraction by one rather than addition by one. For natural numbers, in the language of properties:

Induction step: For every natural number $k > 0$, if $k - 1$ has the property, then so does k.

In the language of sets:

Induction step: For every natural number $k > 0$, if $k - 1 \in A$, then also $k \in A$.

Note carefully the proviso $k > 0$: this is needed to ensure that $k - 1$ is a natural number when k is. If we are inducing over the positive integers only, then of course the proviso becomes $k > 1$.

When should you use induction over the positive integers, and when over the natural numbers? Quite simply, when you are trying to prove something for all the natural numbers, induce over them; if you are trying to show something for the positive integers only, normally you will induce over them. In what follows, we will pass from one to the other without further comment.

It is quite common to formulate the induction step in (equivalent) *contrapositive* form, particularly when it is stated with subtraction rather than addition. Thus, for example, the last version of the induction step becomes:

For every natural number $k > 0$, if $k \notin A$, then also $k - 1 \notin A$.

Exercise 4.2.2 (2) Reformulate each of the preceding five displayed formulations of the induction step for principles of induction over the positive integers or the natural numbers, in contrapositive form.

Exercise 4.2.2 (3) (with solution) In Chap. 3, we used the pigeonhole principle to show that in any reception attended by $n \geq 1$ people, if everybody shakes hands just once with everyone else, then there are $n(n-1)/2$ handshakes. Show this again, by using the simple principle of induction over the positive integers.

Solution

Basis: We show this for the case $n = 1$. In this case, there are 0 handshakes, and $1{\cdot}(1-1)/2 = 0$, so we are done.
Induction step: Let k be any positive integer, and suppose (induction hypothesis) that the property holds when $n = k$, that is, suppose that when there are k people, there are $k{\cdot}(k-1)/2$ handshakes. We need to show (induction goal) that it holds when $n = k+1$, in other words, that when there are just $k+1$ people, there are $(k+1){\cdot}((k+1){-}1)/2 = k{\cdot}(k+1)/2$ handshakes.

Consider any one of these $k+1$ people, and call this person a. Then the set of all handshakes is the union of two disjoint sets: the handshakes involving a and those not involving a. Hence (recall from Chap. 1), the total number of handshakes is equal to the number involving a plus the number not involving a. Clearly, there are just k handshakes of the former kind, since a shakes hands just once with everyone else. Since there are just k people in the reception other than a, we also know by the induction hypothesis that there are $k{\cdot}(k-1)/2$ handshakes of the latter kind. Thus, there is a total of $k + [k{\cdot}(k-1)/2]$ handshakes. It remains to check by elementary arithmetic that $k + [k{\cdot}(k-1)/2] = [2k + k{\cdot}(k-1)]/2 = (2k + k^2 - k)/2 = (k^2 + k)/2 = k{\cdot}(k+1)/2$, as desired.

This exercise illustrates the fact that a problem can sometimes be tackled in two quite different ways – either by induction or directly. The same phenomenon can be illustrated by the very first example of this chapter, where we showed by simple induction that the sum of the first n positive integers equals $n{\cdot}(n+1)/2$. A short, clever proof of the same fact is attributed to Gauss when still a schoolboy. It could be called a geometric proof or an argument by rearrangement. Let s be the sum of the first n positive integers. Clearly, $2s = (1+\cdots+n) + (n+\cdots+1) = n{\cdot}(n+1)$ by adding the corresponding terms n times, so $s = n{\cdot}(n+1)/2$.

We end this section with some general advice. When proving something by induction, it is essential to keep clearly in mind what the induction hypothesis and induction goal are. Unless they are made explicit, it is easy to become confused. Classroom experience shows that students rarely have difficulty with the basis, but often get into a mess with the induction step because they have not identified clearly what it is, and what its two components (induction hypothesis, induction goal) are. *Always write out the proposition expressing the induction step explicitly before trying to prove it* and, until it becomes second nature, *label the induction hypothesis and induction goal within it*. To keep a clear idea of what is going on, separate in your mind the general strategy of the induction from whatever numerical calculations may come up within its execution.

Students are sometimes puzzled why here (and in most presentations) use one variable n when stating the proposition to be proven by induction, but another variable k when proving the induction step. Does it make a difference? The short answer is No. The reason for using different variables is purely pedagogical. We could perfectly well use the same one (say n) in both, but experience again shows that doing so confuses irrevocably those who are already struggling. So, to emphasize that the proposition to be proven by induction and the proposition serving as induction step in the proof are two quite different things, we use different variables.

Alice Box: How to Cook Up the Right Property?

Alice: That is all very well, but I am still rather dissatisfied. If you ask me to prove by induction that, say, for every positive integer n, the sum of the first n even integers is $n(n+1)$, I am sure that I could do it now. But if you were to ask me to show by induction that we can express the sum of the first n even integers in a neat way, I am not sure that I could think up that expression $n(n+1)$ in order to begin the proof. Induction seems to be good for proving things that you already suspect are true, but not much use for imagining what might be true in the first place!

Hatter: Indeed, it is quite an art to guess the right equality (or more generally, property) to induce on; once you think it up, the induction itself is often plain sailing. Like all art, it needs practice and experience. But that's part of what makes it interesting . . .

Exercise 4.2.2 (4)

(a) Dowhat Alice is sure that she can do now.

(b) Do it again, but this time without a fresh induction. Instead, use what had already been shown by induction about the sum of the first n positive integers and the sum of the first n odd ones.

4.3 Definition by Simple Recursion on the Positive Integers

We now dig below the material of the preceding section. Roughly speaking, one can say that *underneath every inductive proof lurks a recursive definition*. In particular, when f is a function on the positive integers (or natural numbers) and we can prove inductively something about its behaviour, then f itself may be defined (or at least characterized) in a recursive manner.

For an example, consider again the function f used in the first example of this chapter. Informally, $f(n)$ was understood to be the sum $1+2+3+\cdots+n$ of the first n positive integers. The reason why induction could be applied to prove that $f(n)=n\cdot(n+1)/2$ is that the function f can itself be *defined recursively*.

What would such a definition look like? As one would expect, it consists of a *basis* giving the value of f for the least positive integer argument 1, and a *recursive* *(or inductive) step* giving the value of f for any argument $n + 1$ in terms of its value for argument n. Thus, in a certain sense, the function is being defined in terms of itself, but in a non-circular manner, the value $f(n + 1)$ is defined in terms of $f(n)$ with the lesser argument n. Specifically, in our example, the basis and recursive step are as follows:

Basis of definition: $f(1) = 1$.
Recursive step of definition: When $n \geq 1$, then $f(n + 1) = f(n) + (n + 1)$.

This way of expressing the recursive step, using addition by 1, is sometimes called recursion by *pattern-matching*. Another way of writing it uses subtraction by 1, in our example as follows:

Recursive step of definition: When $n > 1$, then $f(n) = f(n - 1) + n$.

Other ways of writing recursive definitions are also current among computer scientists. In particular, one can think of the basis and recursion step as being *limiting* and *principle cases*, respectively, writing in our example:

If $n = 1$, then $f(n) = 1$
If $n > 1$, then $f(n) = f(n - 1) + n$.

This can also be expressed in the popular *if-then-else* form:

If $n = 1$, then $f(n) = 1$
Else $f(n) = f(n - 1) + n$.

Some computer scientists like to abbreviate this further to the ungrammatical declaration:

$f(n) =$ if $n = 1$, then 1 else $f(n - 1) + n$,

which can look like mumbo-jumbo to the uninitiated. *All of these formulations are equivalent*. You will meet each of them in applications, and so should be able to recognize them. The choice is partly a matter of personal preference, partly a question of which one allows you to get on with the problem in hand with the least clutter.

Exercise 4.3 (with partial solution)

(a) Define recursively the following functions $f(n)$ on the positive integers:
  (i) The sum $2 + 4 + \cdots + 2n$ of the first n even integers
  (ii) The sum $1 + 3 + \cdots + (2n - 1)$ of the first n odd integers
(b) Define recursively the function that takes a natural number n to 2^n. This is called the *exponential function*.
(c) Define recursively the product of the first n positive integers. This is known as the *factorial function*, written $n!$, and is pronounced 'n factorial'.
(d) Use the recursive definitions of the relevant functions to show that for all $n \geq 4$, $n! > 2^n$. *Hint*: here the basis will concern the case that $n = 4$.

Solution to (b)–(d)

(b) Basis of definition: $2^0 = 1$. Recursive step of definition: $2^{n+1} = 2{\cdot}2^n$.

(c) Basis of definition: $1! = 1$. Recursive step of definition: $(n + 1)! = (n + 1){\cdot}n!$

(d) Basis of proof: We need to show that $4! > 2^4$. This is immediate by calculation, with $4! = 1{\cdot}2{\cdot}3{\cdot}4 = 24 > 16 = 2{\cdot}2{\cdot}2{\cdot}2 = 2^4$. Induction step of proof: Let k be any positive integer with $k \geq 4$. Suppose that $k! > 2^k$ (induction hypothesis). We need to show that $(k+1)! > 2^{k+1}$ (induction goal). This can be done as follows:

$$\begin{aligned}(k+1)! &= (k+1)\cdot k! \text{ by definition of the factorial function}\\ &> (k+1)\cdot 2^k \text{ by theinduction hypothesis}\\ &> 2\cdot 2^k \text{ since } k \geq 4\\ &= 2^{k+1} \text{ by definition of the exponential fuction.}\end{aligned}$$

The factorial and exponentiation functions are both very important in computer science, as indications of the alarming way in which many processes can grow in size (number of steps required, time taken, memory required, and so on) as inputs increase in size. In Chap. 1, we saw that exponentiation already gives unmanageable rates of growth. The exercise shows that factorial is worse.

4.4 Evaluating Functions Defined by Recursion

Go back yet again to the first function that we defined by recursion in the preceding section: the sum $f(n)$ of the first n positive integers. Suppose that we want to calculate the value of $f(7)$. There are basically two ways of doing it.

One very obvious way is *bottom-up* or *forwards*. We first calculate $f(1)$, use it to get $f(2)$, use it for $f(3)$, and so on. Thus, we obtain:

$$\begin{aligned}f(1) &= 1\\ f(2) &= f(1) + 2 = 1 + 2 = 3\\ f(3) &= f(2) + 3 = 3 + 3 = 6\\ f(4) &= f(3) + 4 = 6 + 4 = 10\\ f(5) &= f(4) + 5 = 10 + 5 = 15\\ f(6) &= f(5) + 6 = 15 + 6 = 21\\ f(7) &= f(6) + 7 = 21 + 7 = 28.\end{aligned}$$

Here we have made one application of the base clause and six of the recursion clause. Each application is followed by arithmetic simplification as needed. Each of the seven steps fully eliminates the function sign f and provides us with a specific numeral as value of $f(i)$ for some $i \leq 7$.

The other way of calculating, at first a little less obvious, is *top-down* or *backwards*. It is also known as *unfolding* the recursive definition or *tracing* the function, and is as follows:

$$\begin{aligned}
f(7) &= f(6) + 7 \\
&= (f(5) + 6) + 7 \\
&= ((f(4) + 5) + 6) + 7 \\
&= (((f(3) + 4) + 5) + 6) + 7 \\
&= ((((f(2) + 3) + 4) + 5) + 6) + 7 \\
&= (((((f(1) + 2) + 3) + 4) + 5) + 6) + 7 \\
&= (((((1 + 2) + 3) + 4) + 5) + 6) + 7 \\
&= ((((3 + 3) + 4) + 5) + 6) + 7 \\
&= (((6 + 4) + 5) + 6) + 7 \\
&= ((10 + 5) + 6) + 7 \\
&= (15 + 6) + 7 \\
&= 21 + 7 \\
&= 28.
\end{aligned}$$

In this calculation, we begin by writing the expression $f(7)$, make a substitution for it as authorized by the recursion clause of the definition, then in that expression substitute for $f(6)$, and so on until after six steps we can at last apply the basis of the definition to $f(1)$ to obtain in line 7 a numerical expression, not containing f. From that point, we simplify the numerical expression until, in another six steps, we emerge with a standard numeral.

Which is the better way to calculate? In this example, it does not make a significant difference. Indeed, quite generally, when the function is defined using simple recursion of the kind described in this section, the two modes of calculation will be of essentially the same length. The second one looks longer, but that is because we have left all arithmetic simplifications to the end (as is customary when working top-down) rather than doing them as we go. Humans often prefer the first mode, for the psychological reason that it gives us something ‘concrete’ at each step, making them feel safer; but a computer would not care.

Nevertheless, when the function is defined by more sophisticated forms of recursion that we will describe in the following sections, the situation may become quite different. It can turn out that one, or the other, of the two modes of evaluation is dramatically more economical in resources of memory or time.

Such economies are of little interest to the traditional mathematician, but they are of great importance for the computer scientist. They may make the difference between a feasible calculation procedure and one that, in a given state of technology, is quite unfeasible.

Exercise 4.4 Evaluate 6! bottom-up and then again top-down (unfolding).

4.5 Cumulative Induction and Recursion

We now turn to some rather more sophisticated forms of recursive definition and inductive proof. Ultimately, they can be derived from the simple form, but we will not do so here. We are interested in them as tools-for-use.

4.5.1 Cumulative Recursive Definitions

In definitions by simple recursion such as that of the factorial function, we reached back only one notch at a time. In other words, the recursion step defined $f(n)$ out of $f(n-1)$. But sometimes we want to reach back further. A famous example is the *Fibonacci function* on the natural numbers (i.e. beginning from 0 rather than from 1), named after an Italian mathematician who considered it in the year 1202. Since then, it has found surprisingly many applications in computer science, biology and elsewhere. To define it, we use recursion. The basis now has two components:

$F(0) = 0$
$F(1) = 1.$

So far, F behaves just like the identity function. But then, in the recursion step, Fibonacci takes off:

$F(n) = F(n-1) + F(n-2)$ whenever $n \geq 2$.

The Fibonacci function illustrates the way in which a top-down evaluation by unfolding can lead to great inefficiencies in computation. Beginning a top-down computation of $F(8)$, we have to make many calls, as represented in Table 4.1. To read the table, notice that each cell in a row is split into two in the row below, following the recursive rule for the Fibonacci function, so that value of each cell equals the sum of the values in the two cells immediately below.

The table is not yet complete – it hasn't even reached the point where all the letters F are eliminated and the arithmetic simplifications begin – but it is already

Table 4.1 Calls when computing F(8) top-down

<table>
<tr><td colspan="16">$F(8)$</td></tr>
<tr><td colspan="8">$F(7)$</td><td colspan="8">$F(6)$</td></tr>
<tr><td colspan="4">$F(6)$</td><td colspan="4">$F(5)$</td><td colspan="4">$F(5)$</td><td colspan="4">$F(4)$</td></tr>
<tr><td colspan="2">$F(5)$</td><td colspan="2">$F(4)$</td><td colspan="2">$F(4)$</td><td colspan="2">$F(3)$</td><td colspan="2">$F(4)$</td><td colspan="2">$F(3)$</td><td colspan="2">$F(3)$</td><td colspan="2">$F(2)$</td></tr>
<tr><td>$F(4)$</td><td>$F(3)$</td><td>$F(3)$</td><td>$F(2)$</td><td>$F(3)$</td><td>$F(2)$</td><td>$F(2)$</td><td>$F(1)$</td><td>$F(3)$</td><td>$F(2)$</td><td>$F(2)$</td><td>$F(1)$</td><td>$F(2)$</td><td>$F(1)$</td><td>$F(1)$</td><td>$F(0)$</td></tr>
<tr><td colspan="16">etc</td></tr>
</table>

clear that there is a great deal of repetition. The value of $F(6)$ is calculated twice, that of $F(5)$ three times, $F(4)$ five times, $F(3)$ eight times (including the one still to come in the next row). Indeed, when calculating $F(n)$, the number of times each $F(k)$ is calculated itself follows a Fibonacci function run backwards from $n+1$. For example, to calculate $F(8)$, each $F(k)$ for $k \leq 8+1 = 9$ is calculated $F(9-k)$ times. Unless partial calculations are saved and recycled in some manner, the inefficiency of the procedure is very high.

Exercise 4.5

(a) Express the recursion clause of the definition of the Fibonacci function in pattern-matching form, in other words, defining $F(n+2)$ in terms of $F(n+1)$ and $F(n)$. Be explicit about the bound of your quantification.

(b) Write the pattern-matching form of the definition in *if-then-else* language.

(c) Carry out a bottom-up evaluation of $F(8)$ and compare it with the (incomplete) top-down evaluation in the text.

In contrast, there are cases in which evaluation bottom-up can be less efficient. Here is a simple example. Define f as follows:

Basis: $f(0) = 2$.
Recursion step: $f(n) = f(n-1)^n$ for odd $n > 0$, $f(n) = f(n/2)$ for even $n > 0$.

To calculate $f(8)$ by unfolding is quick and easy: $f(8) = f(4) = f(2) = f(1)$ by three applications of the second case of the recursion step and finally $f(1) = f(0)^1 = 2^1 = 2$ by the first case of the recursion step and the basis. On the other hand, if we were to calculate $f(8)$ bottom-up without introducing shortcuts, we would pass through each of $f(0)$ to $f(8)$ doing a lot of unnecessary work on the odd values. We will see a more dramatic example of the same phenomenon shortly.

4.5.2 Proof by Cumulative Induction

There is no limit to how far a recursive definition may reach back when defining $f(n)$, and so it is useful to have a form of proof that permits us to do the same. It is called *cumulative induction*, sometimes also known as *course-of-values* induction.

Formulated for the natural numbers, in terms of properties, the *principle of cumulative induction* is as follows: To show that every natural number has a certain property, it suffices to show the basis and induction step:

Basis: 0 has the property.
Induction step: For every natural number k, if every natural number $j < k$ has the property, then k itself also has the property.

Exercise 4.5.2 (1) (with solution)

(a) Formulate the induction step of the principle of cumulative induction contrapositively.

(b) Use cumulative induction to show that every natural number $n \geq 2$ is the product of (one or more) prime numbers.

Solution

(a) For every natural number k, if k lacks the property, then there is a natural number $j < k$ that lacks the property.

(b) *Basis*: We need to show that 2 has the property. Since 2 is itself prime, we are done. *Induction step*: Let k be any natural number with $k \geq 2$. Suppose (induction hypothesis) that every natural number j with $2 \leq j < k$ has the property. We need to show (induction goal) that k also has it. There are two cases to consider. If k is prime, then we are done. On the other hand, if k is not prime, then by the definition of prime numbers $k = a{\cdot}b$ where a, b are positive integers ≥ 2. Hence, $a < k$ and $b < k$. So we may apply the induction hypothesis to get that each of a, b has the property, that is, is the product of (one or more) prime numbers. Hence, their product is too, and we are done.

Part (b) of this exercise prompts several comments. The first concerns presentation. In the solution, we kept things brief by speaking of ‘the property’ when we mean ‘the property of being the product of two prime numbers’. The example was simple enough for it to be immediately clear what property we are interested in; in more complex examples, you are advised to take the precaution of stating the property in full at the beginning of the proof.

Second, we began the induction at 2 rather than at 0 or even 1. This is in fact covered by the version that begins from 0, for when we say that every natural number $n \geq 2$ has a certain property, this is the same as saying that for every natural number n, *if* $n \geq 2$, *then* n has the property. But it is also convenient to formulate the principle in a separate form when the induction begins at a specified number $a > 0$, with the following basis and induction step:

Basis: a has the property.
Induction step: For every natural number $k \geq a$, if every natural number j with $a \leq j < k$ has the property, then k itself also has the property.

Third, for cumulative induction, the basis is, strictly speaking, covered by the induction step, and so is redundant! This contrasts with simple induction, where the basis is quite independent of the induction step and always needs to be established separately. The reason for this redundancy in the cumulative case is that there are no natural numbers less than zero. Hence, vacuously, every natural number $j < 0$ has whatever property is under consideration, so that applying the induction step, zero itself has the property and the basis holds. Nevertheless, even for cumulative induction, it is quite common to state the basis explicitly and even to check it out separately, to double-check that we did not carelessly overlook some peculiarity of zero when establishing the induction step in its generality. You may do likewise.

Finally, the principle of cumulative induction is in fact derivable from that of simple induction. If this were a text for students of mathematics, the proof should follow, but we omit it here. Although it is derivable from the simple form, it is nevertheless extremely useful to have available as a rule in its own right, ready for application.

Exercise 4.5.2 (2) Use cumulative induction to show that for every positive integer n, $F(n)$ is even iff n is divisible by 3, where F is the Fibonacci function.

4.5.3 Simultaneous Recursion and Induction

When a function has more than one argument, its definition will have to take account of both of them. If the definition is recursive, sometimes we can get away with recursion on just one of the arguments, holding the others as parameters. A simple example is the recursive definition of multiplication over the natural numbers, using addition, which can be expressed as follows:

Basis: $(m{\cdot}0) = 0$.
Recursion step: $m{\cdot}(n + 1) = (m{\cdot}n)+m$.

The equality in the recursion step is usually taught in school as a *fact* about multiplication, which is assumed to have been defined or understood in some other way. In axiomatizations of arithmetic in the language of first-order logic, the equality is treated as an *axiom* of the system. But here we are seeing it as the recursive part of a *definition* of multiplication, given addition. The same mathematical edifice can be built in many ways!

Exercise 4.5.3 (1)

(a) Give a recursive definition of the power function that takes a pair (m,n) to m^n, using multiplication, and with recursion on the second argument only.
(b) Comment on the pattern shared by this definition of the power function, and the preceding definition of multiplication.

When a two-argument function is defined in this way, then inductive proofs for it will tend to follow the same lines, with induction carried out on the argument that was subject to recursion in the definition. But sometimes we need to define functions of two (or more) arguments with recursions on each of them. This is called *definition by simultaneous recursion*. A famous example is the *Ackermann function*. It has two arguments and, in one of its variants (due to Rósza Péter), is defined as follows:

$A(0,n) = n + 1$
$A(m,0) = A(m{-}1,1)$ for $m > 0$
$A(m,n) = A(m - 1, A(m, n - 1))$ for $m,n > 0$.

The reason why this function is famous is its spectacular rate of growth. For $m < 4$, it remains leisurely, but when $m \geq 4$, it accelerates dramatically, much more so than either the exponential or the factorial function. Even $A(4,2)$ is about $2{\cdot}10^{19728}$. This gives it a theoretical interest: although the function is computable, it can be shown that it grows faster than any function in the class of so-called 'primitive recursive' functions. Roughly speaking, they are functions that can be defined by recursions of fairly simple syntactic kind. They are computable, and for a short while after their articulation as a class of functions, they were thought to exhaust all the computable functions, until the Ackermann function provided a counterexample.

But what interests here us about the Ackermann function is the way in which the last clause of the definition makes the value of $A(m,n)$ depend on the value of the same function for the first argument diminished by 1, but paired with a value of the second argument *that might be larger than n*. As the function picks up speed, to calculate the value of $A(m,n)$ for a given m may require prior calculation of $A(m-1,n')$ for an extremely large $n' > n$.

Indeed, given the way in which the recursion condition reaches 'upwards' on the second variable, it is not immediately obvious that the three clauses taken together really succeed in defining a unique function. It can, however, be shown that they do, by introducing a suitable well-founded ordering on $\mathbf{N}^2$, and using the principle of well-founded recursion. We will explain the concept of a well-founded ordering later in this chapter.

Alice Box: Basis and Recursion Step

Alice: There is something about the definition of the Ackermann function that bothers me. There are three clauses in it, not two. So which is the basis and which is the recursion step?

Hatter: The first clause gives the basis. Although it covers infinitely many cases, and uses a function on the right hand side, the function A that is being defined does not appear on the right.

Alice: And the recursion step?

Hatter: It is given by the other two clauses together. The second clause covers the case $m > 0$ while $n = 0$; the third deals with the case when both m, $n > 0$. The distinguishing feature that marks them both as parts of the recursion step is the appearance of A on the right.

The Ackermann function illustrates dramatically the difference that can arise between calculating bottom-up or top-down, in the sense described in Sect. 4.4. The best way to appreciate the difference is to do the following exercise. You will see that this time it is the bottom-up approach that comes off worse. The reason is that, as already remarked, to calculate $A(m,n)$ for a given m may require prior calculation of $A(m-1,n')$ for a certain $n' > n$. We don't know in advance how large this n' will be, and it is inefficient to plough through all the values of n' until we reach the one that is actually needed.

Exercise 4.5.3 (2) Calculate the value of $A(2,3)$ bottom-up. Then calculate it top-down. Compare your calculations.

4.6 Structural Recursion and Induction

We now come to the form of recursion and induction that is perhaps the most frequently used by computer scientists – *structural*. It can be justified or replaced by recursion and induction on the natural numbers but may also be used, very conveniently, without ever mentioning them. It tends to be ignored in courses for mathematics students, but it is essential that those heading for computer science (or logic) be familiar with it.

We introduce structural recursion/induction in three stages, remembering as we go the rough dictum that behind every inductive proof lurks a recursive definition. First, we look at the business of defining sets by structural recursion. That will not be difficult, because back in Chaps. 2 and 3 on sets and functions we were already doing it without, however, paying attention to the recursive aspect. Then we turn to the task of proving things about these sets by structural induction, which will also be quite straightforward. Finally, we come to the rather delicate part: the task of taking a recursively defined set and using structural recursion to define a function with it as domain. That is where care must be taken, for such definitions are legitimate only when a special constraint is satisfied.

4.6.1 Defining Sets by Structural Recursion

In Chaps. 2 and 3, we introduced the notions of the image and the closure of a set under a relation or function. We now make intensive use of them. We begin by recalling their definitions, generalizing a little from binary (i.e. two-place) relations to relations of any finite number $n \geq 2$ of places.

Let A be any set, and let R be any relation (of at least two places) over the local universe within which we are working. Since m-argument functions are $(m+1)$-place relations, this covers functions of one or more arguments as well.

The *image* of A under the $(m+1)$-place relation R is defined by putting $x \in R(A)$ iff there are $a_1, \ldots, a_m \in A$ with $(a_1, \ldots a_m, x) \in R$. In the case that R is an m-argument function f, this is equivalent to saying: $x \in f(A)$ iff there are $a_1, \ldots, a_m \in A$ with $x = f(a_1, \ldots a_m)$.

The concept of image is not yet recursive; that comes with the closure $R[A]$, which for brevity we write as A^+. We saw that it can be defined in either of two ways.

The *way of union* (bottom-up) is by making a recursion on the natural numbers, then taking the union of all the sets thus produced:

$$A_0 = A$$
$$A_{n+1} = A_n \cup R(A_n), \text{ for each natural number } n$$
$$A^+ = \cup\{A_n: n \in \mathbf{N}\}.$$

The *way of intersection* ('top-down') dispenses with numbers altogether. For simplicity, we assume that we are working within a local universe (see Sect. 1.4.3), which, however, we will usually not bother to mention explicitly. Recall that a set X is *closed* under a relation R iff $R(X) \subseteq X$. The *closure* A^+ of A under the relation R is defined to be the intersection of all sets $X \supseteq A$ that are closed under R. That is, $A^+ = \cap\{X \subseteq U: A \subseteq X \supseteq R(X)\}$ where U is the local universe, assumed to include both A and R. So defined, A^+ is in fact *the least* superset of A that is closed under the relation R. On the one hand, since it is the intersection of all the $X \supseteq A$ closed under R, it is clearly included in each such X; on the other hand, it is easy to check that it is itself also closed under the relation R.

Exercise 4.6.1 (1)

(a) Verify the point just made, that the intersection of all sets $X \supseteq A$ that are closed under R is itself closed under R.

(b) For this exercise, to avoid confusion, use $A^{\cup}$ as temporary notation for A^{+} defined bottom-up, and $A^{\cap}$ as temporary notation for A^{+} defined top-down. Show that $A^{\cup} = A^{\cap}$ by establishing the two inclusions separately.

Hints for (b): To show that $A^{\cup} \subseteq A^{\cap}$, use induction on the natural numbers. For the converse, begin by checking that $A^{\cup}$ is closed under R.

The definition can evidently be extended to cover an arbitrary collection of relations, rather than just one. The *closure* A^{+} of A under a *collection* $\{R_i\}_{i \in I}$ of relations is defined to be the *intersection of all sets* $X \supseteq A$ that are closed under all the relations in the collection.

A set is said to be *defined by structural recursion* whenever it is introduced as the closure of some set (referred to as the *basis*, or initial set) under some collection of relations (often called *constructors* or *generators*).

In this context, it is intuitively helpful to think of each $(m + 1)$-place relation R as a *rule*: if $(a_1, \ldots a_m, x) \in R$, then, when building A^{+}, we are authorized to pass from items $a_1, \ldots a_m$ already in A^{+}, to put x in A^{+} too. Thus, A^{+} is also described as the closure of A under a collection of rules. This way of speaking often makes formulations more vivid. We illustrate the concept with five examples, two drawn from computer science, two from logic and one from abstract algebra:

- The *notion of a string*, already mentioned informally in an Alice box of Chap. 3. It is of central importance for formulating, for example, the theory of finite state machines. Let A be any alphabet, consisting of elementary signs. Let λ be an abstract object, distinct from all the letters in A, which is understood to serve as the empty string. The set of all strings over A, conventionally written as A^*, is the *closure* of $A \cup \{\lambda\}$ under the rule of concatenation, that is, the operation of taking two strings s,t and forming their concatenation by writing s immediately followed by t.
- We can define *specific sets of strings* by structural recursion. For instance, a string over an alphabet A is said to be a *palindrome* iff it reads the same from each end. Can we give this informal notion, known to grammarians since ancient times, a recursive definition? Very easily! The empty string λ reads the same way from each end, and is the shortest even palindrome. Each individual letter in A reads the same way from left and from right, and so these are the shortest odd palindromes. All other palindromes may be obtained by successive symmetric flanking of given palindromes. So we may take the set P of palindromes to be the *closure* of the set $\{\lambda\} \cup A$ under the rule permitting passage from a string s to a string xsx for any $x \in A$. In other words, it is the least set S including $\{\lambda\} \cup A$ such that $xsx \in S$ for any $s \in S$ and $x \in A$.

- Logicians also work with symbols, and constantly define sets by structural recursion. For example, the set of *formulae* of classical propositional logic (or any other logical system) is defined as the *closure* of an initial set A under some operations. In this case, A is a set of proposition letters. It is closed under the rules for forming compound formulae by means of the logical connectives allowed, for instance, $\neg,\wedge,\vee$ (with parentheses around each application of a connective, to ensure unambiguous reading). So defined, the set of propositional formulae is a proper subset of the set B^* of all strings in the alphabet $B = A\cup\{\neg,\wedge,\vee\}\cup\{(,)\}$. In case you have trouble reading it, $\{(,)\}$ is the set consisting of the left and right parentheses.
- The set of *theorems* (aka *theses*) of a formal logical system is also defined by structural recursion. It is the *closure* A^+ of some initial set A of formulae (known as the *axioms* of the system) under certain functions or relations between formulae (known as *derivation rules* of the system). A derivation rule often used in this context is modus ponens (aka detachment, also known as $\rightarrow^-$), permitting passage from formulae α and $\alpha \rightarrow \beta$ to the formula β.
- Algebraists also use this kind of definition, even though they are not, in general dealing with strings of symbols. For example, if A is a subset of an algebra, then the *subalgebra generated by A* has as its carrier (i.e. underlying set) the *closure* A^+ of A under the operations of the algebra, and as its operations the restriction to that carrier of the given operations over the whole algebra.

In all these cases, it is perfectly possible to use natural numbers as indices for successive sets $A_0, A_1, A_2, \ldots$ in the generation of the closure by the way of union, and replace the structural definition by one that makes a recursion on the indices. Indeed, this is a fairly common style of presentation, and, in some contexts, has its advantages. But in other cases, the use of indices and appeal to induction on numbers is an unnecessary detour.

Alice Box: Defining the Set of Natural Numbers Recursively

Alice: My friend studying the philosophy of mathematics tells me that even the set of natural numbers may be defined by structural recursion. This, he says, is the justification for induction over the integers. Is that possible?

Hatter: We can define the natural numbers in that way, if we are willing to identify them with sets. There are many ways of doing it. For example, we can take **N** to be the least set X that contains $\varnothing$ and is closed under the operation taking each set S to $S\cup\{S\}$. In this way, arithmetic is reduced to set theory.

Alice: Does that *justify* induction over the natural numbers?

Hatter: It depends on which way you are doing things. On the one hand, when arithmetic is axiomatized in its own terms, without reducing it to anything else, induction is simply treated as an axiom and so

(continued)

(continued)

is not given a formal justification. But when arithmetic is reduced to set theory along the lines that I just mentioned, induction is longer treated as an axiom, since it can be proven.

Alice: But which is the best way of proceeding?

Hatter: That depends on your philosophy of mathematics. In my view, it is not a question of one way being right and the other being wrong, but one of convenience for the tasks in hand. But discussing that any further would take us too far off our track.

Exercise 4.6.1 (2) (with solution) Define by structural recursion the set of even palindromes over an alphabet A, that is, the palindromes with an even number of letters.

Solution The set of even palindromes over an alphabet A is the closure of $\{\lambda\}$ under the same rule, that is, passage from a string s to a string xsx for any $x \in A$.

4.6.2 Proof by Structural Induction

We have seen that the procedure of defining a set by structural recursion, that is, as the closure of a set under given relations or functions, is pervasive in computer science, logic and abstract algebra. Piggybacking on this mode of definition is a mode of demonstration that we will now examine – proof by structural induction.

Let A be a set of any items whatsoever, let A^+ be the closure of A under a collection $\{R_i\}_{i \in I}$ of relations. Consider any property that we would like to show holds of all elements of A^+. We say that a relation R *preserves the property* iff whenever $a_1, \ldots a_m$ have the property and $(a_1, \ldots a_m, x) \in R$, then x also has it. When R is a function f, this amounts to requiring that whenever $a_1, \ldots a_m$ has the property, then $f(a_1, \ldots a_m)$ also has the property.

The *principle of proof by structural induction* may now be stated as follows. Again, let A be a set, and A^+ the closure of A under a collection $\{R_i\}_{i \in I}$ of relations. To show that every element of A^+ has a certain property, it suffices to show two things:

Basis: Every element of A has the property.
Induction step: Each relation $R \in \{R_i\}_{i \in I}$ preserves the property.

Proof of the principle is almost immediate given the definition of the closure A^+. Let X be the set of all items that have the property in question. Suppose that both basis and induction step hold. Since the basis holds, $A \subseteq X$. Since the induction step holds, X is closed under the relations R_i. Hence, by the definition of A^+ as the *least* set with those two features, we have $A^+ \subseteq X$, that is, every element of A^+ has the property, as desired.

For an example of the application of this principle, suppose we want to show that every even palindrome has the property that every letter that occurs in it occurs an even number of times. Recall that the set of even palindromes was defined as the closure of $\{\lambda\}$ under passage from a string s to a string xsx where $x \in A$. So we need only show two things: that λ has the property in question, and that whenever s has it then so does xsx. The former holds vacuously, since there are no letters in λ (remember, λ is not itself a letter). The latter is trivial, since the passage from s to xsx adds two more occurrences of the letter x without disturbing the other letters. Thus, the proof is complete.

We could have proven the same fact by an induction over the natural numbers, first defining a suitable measure of the length of a palindrome (in this case, just the number of letters from A occurring it), and then inducing on the length. But the passage through numbers is a quite unnecessary detour – in this example, just a short one, but in others, where the measure of length may be rather complex, it can be quite irksome.

Exercise 4.6.2

(a) Use proof by structural induction to show that in any (unabbreviated) formula of propositional logic, the number of left brackets equals the number of right brackets.

(b) If you were to prove the same by induction over the natural numbers, what would be a suitable measure of length?

(c) Let A be any set of formulae of propositional logic, and let R be the three-place relation of *detachment* (alias *modus ponens*) defined by putting $(a,b,c) \in R$ iff b is the formula $a \rightarrow c$. Let A^{+} be the closure of A under R. Use structural induction to show that every formula in A^{+} is a subformula of some formula in A.

4.6.3 Defining Functions by Structural Recursion on Their Domains

There is an important difference between the two examples in the last exercise. In the first one, we begin with a set A of elementary letters, and the closing functions (forming negations, conjunctions, disjunctions) produce longer and longer formulae, and so always give us something fresh. But in the second example, the closing relation of detachment may sometimes give us a formula that is already in the initial set A or was already available at an earlier stage of the closing process: *it is not guaranteed always to give us something fresh*.

This difference is of no significance for structural induction as a method of proof, but it is vital if we want to use structural recursion to define a function whose domain is a set that was already defined by structural recursion – a situation that arises quite frequently.

Suppose that we want to give a recursive definition of a function f whose domain is the closure A^{+} of a set A under a function g. For example, we might want to define a function $f: A^{+} \rightarrow \mathbf{N}$, as some kind of measure of complexity of the elements of the domain, by putting $f(a) = 0$ for all $a \in A$, and $f(g(a)) = f(a) + 1$. Is this legitimate?

Unfortunately, not always. If the function g is not injective, then we will have two distinct a, a' with $g(a) = g(a')$, and it may very well be that $f(a) \neq f(a')$ and thus $f(a) + 1 \neq f(a')+1$ so that $f(g(a))$ has not been given a unique value. Moreover, the injectivity of g is not quite enough, by itself, to ensure that f is well defined. It may happen that even when g is injective, $g(a)$ is already an element $a' \in A$, so that $f(g(a))$ is defined as 0 by the first part of the definition but as $f(a) + 1 > 0$ by the second part, again lacking a unique value.

To illustrate this second eventuality, let g: $\mathbf{N}\rightarrow\mathbf{N}$ the function of adding one to a natural number except that $g(99)$ is 0. That is, $g(n) = n + 1$ for $n \neq 99$, while $g(99) = 0$. Clearly, this function is injective. Put $A = \{0\}$. Then the closure A^+ of A under g is the set $\{0,\ldots,99\}$. Now suppose that we try to define a function f: $A^+\rightarrow\mathbf{N}$ by structural induction putting $f(0) = 0$ and $f(g(n)) = f(n) + 1$ for all $n \in A^+$. The recursion step is fine for values of $n < 99$; indeed we get f to be the identity function for those values. But it breaks down at $n = 99$. The recursion step tells us that $f(g(99)) = f(99) + 1 = 100$, but since $g(99) = 0$, the basis has already forced us to say that $f(g(99)) = f(0) = 0$. Although the basis and the recursion step make sense separately, they conflict, and we have not succeeded in defining a function!

In summary, whereas structural recursion is always legitimate as a method of defining a set as the closure of an initial set under relations or functions, it can fail *as a method of defining a function with a recursively defined set as its domain*, unless precautions are taken.

What precautions? What condition needs to be satisfied to guarantee that such definitions are legitimate? Fortunately, analysis of examples like the one above suggests an answer. To keep notation simple, we focus on the case that the closure is generated by functions.

Let A be a set and A^+ its closure under a collection $\{g_i\}_{i\in I}$ of functions. An element x of A^+ is said to be *uniquely decomposable* iff either (1) $x \in A$, and is not in the range of any of the functions g_i, or (2) $x \notin A$, and $x = g_i(y_1,\ldots,y_k)$ for a unique function g_i in the collection and a unique tuple $y_1,\ldots,y_k \in A^+$. Roughly speaking, this guarantees that there is a unique way in which x can have got into A^+. When the elements of A^+ are symbolic expressions of some kind, this property is also called *unique readability*. Unique decomposability/readability suffices to guarantee that a structural recursion on A^+ succeeds, that is, that it defines a unique function with A^+ as domain. To be precise, we have the following

Principle of structural recursive definition: Let A be a set, $\{g_i\}_{i\in I}$ a collection of functions, and A^+ the closure of A under the functions. Suppose that every element of A^+ is uniquely decomposable. Let X be any set, and let f: $A\rightarrow X$ be a given function on A into X. Then, for every collection $\{h_i\}_{i\in I}$ of functions h_i on powers of X into X with arities corresponding to those of the functions g_i, there is a unique function f^+: $A^+\rightarrow X$ satisfying the following recursively formulated conditions:

Case	*Conditions*
Basis: $x \in A$	$f^+(x) = f(x)$
Recursion step: $x = g_i(y_1,\ldots,y_k)$ is the unique decomposition of x	$f^+(x) = h_i(f^+(y_1),\ldots,f^+(y_k))$

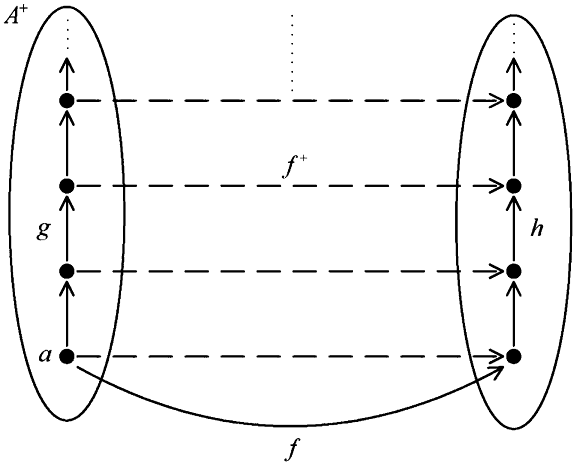

Fig. 4.1 Recursive structural definition

Another way of putting this principle, which will ring bells for readers who have done some abstract algebra, is as follows: We may legitimately extend a function f: $A{\rightarrow}X$ homomorphically to a function f^+: $A^+{\rightarrow}X$ if every element of A^+ is uniquely decomposable.

This is highly abstract, and may at first be rather difficult to follow. It may help to focus on the case that A is a singleton $\{a\}$ and A^+ is its closure under just one function g with just one argument, so that there is also just one function h under consideration, which likewise has just one argument. This basic case may also be represented by a diagram, as in Fig. 4.1. In the diagram, the unique decomposition condition becomes the requirement that in the left ellipse, the bottom element is not hit by any arrow, while the other elements in that ellipse are hit by just one arrow.

The good news is that when we look at specific recursions in computer science and logic, particularly for symbolic expressions, the unique decomposition/readability principle is often satisfied. Moreover, the applications can be very natural – so much so that they are sometimes carried out without mention, taking the existence of unique extensions for granted.

For a very simple example of this, suppose we are given an alphabet A, and let A^+ be the set of all (finite) lists that can be formed from this alphabet by the operation *cons*, that is, of prefixing a letter of the alphabet to an arbitrary list (see Sect. 3.6.2). We might define the *length* of a list recursively as follows, where $\langle\rangle$ is the empty list:

Basis: $length(\langle\rangle) = 0$.
Induction step: If $length(s) = n$ and $a \in A$, then $length(as) = n + 1$.

The principle tells us that this definition is legitimate, because it is clear that *each non-empty list has a unique decomposition into a head and a body*. On the other hand, if we had tried to make a similar definition using the *con* operation, of concatenating any two strings s and t and putting length $length\ (ts) = length(t){+}length(s)$, the unique readability condition would not hold: the string ts could have been broken down in other ways. To be sure, in this simple example, it would be possible to get around the failure of unique readability by showing that nevertheless whenever $ts = t's'$, then $length(t)+length(s) = length(t')+length(s')$. But that is a bit of a bother, and in other examples, such a patch may not be possible.

For another example of appeal to the unique readability condition, consider the customary definition of the *depth* of a formula of propositional logic. Suppose our formulae are built up using just negation, conjunction and disjunction. Usually we say:

Basis: $depth(p) = 0$ for any elementary letter p.
Induction step:
Case 1. If $depth(\alpha) = m$, then $depth(\neg\alpha) = m+1$
Case 2. If $depth(\alpha) = m$ and $depth(\beta) = n$, then $depth(\alpha\wedge\beta) = depth(\alpha\vee\beta) = max(m,n) + 1$,

and take it for granted that the function is well defined. The principle of structural recursive definition tells us that indeed it is so, because the formulae of propositional logic have unique decompositions under the operations of forming negations, conjunctions and disjunctions, provided that they were written with suitable bracketing. Indeed, we can say that the mathematical purpose of bracketing is to ensure unique decomposition.

Exercise 4.6.2 (with partial solution)

(a) Suppose we forget brackets from formulae of propositional logic, and make no bracket-saving conventions about the cohesive powers of connectives. Show that the depth of the expression $\neg p\wedge\neg q$ is not well defined.

(b) Let A be any alphabet, and A^* the set of all strings over A. Let $a \in A$ and $s \in A^*$. Intuitively, the substitution function $\sigma_{a,s}$ substitutes the string s for the letter a in all strings. Define this function by structural recursion.

(c) In propositional logic, two formulae α, β are said to be tautologically equivalent, and we write $\alpha \dashv\vdash \beta$, iff they receive the same truth-value as each other under every possible assignment of truth-values to elementary letters. Three well-known tautological equivalences are double negation $\alpha \dashv\vdash \neg\neg\alpha$ and the de Morgan principles $\neg(\alpha\wedge\beta) \dashv\vdash \neg\alpha\vee\neg\beta$ and $\neg(\alpha\vee\beta) \dashv\vdash \neg\alpha\wedge\neg\beta$. With these in mind, define by structural recursion a function that takes every propositional formula (built using the connectives $\neg,\wedge,\vee$) to a tautologically equivalent one in which negation is applied only to elementary letters. *Hint*: The recursion step will need to distinguish cases.

Solution to (a) When we omit brackets without making any conventions about the cohesive powers of negation and conjunction, the expression $\neg p\wedge\neg q$ has two decompositions, which we may write as $(\neg p\wedge\neg q)$ and $\neg(p\wedge\neg q)$. The depth of the former is 2, but of the latter is 3. However, in a later chapter, we will see a trick for restoring unique readability without needing brackets.

4.7 Recursion and Induction on Well-Founded Sets*

The concept of a well-founded set provides the most general context for recursion and induction. It permits us to apply these procedures to any domain whatsoever, provided we have available a well-founded relation over it. Every other form can in principle be derived from this one. We explain the basics.

4.7.1 Well-Founded Sets

We begin by defining the notion of a well-founded relation over a set. Let W be any set, and $<$ any irreflexive, transitive relation over W (attention: we are *not* requiring linearity, that is, that the relation is also complete, cf Sect. 2.6.2). We say that W is *well founded by* $<$ iff every non-empty subset $A \subseteq W$ has at least one minimal element. This definition is rather compact, and needs to be unfolded carefully:

- A *minimal* element of a set $A \subseteq W$ (under $<$) is any $a \in A$ such that there is no $b \in A$ with $b < a$.
- The definition requires that *every* non-empty subset $A \subseteq W$ has a minimal element. The word ‘every’ is vital. It is not enough to find *some* subset A of W with a minimal element, nor is it enough to show that W itself has a minimal element. Thus, the requirement is very demanding.

The definition can also be put in a quite different-looking, but nevertheless equivalent way. Let W be any set, and $<$ any irreflexive, transitive relation over W. Then W is well founded by $<$ iff there is no *infinite descending chain* $\ldots a_2 < a_1 < a_0$ of elements of W. To prove the equivalence, suppose on the one hand that there is such a chain. Then clearly the set $A = \{a_i: i \in \mathbf{N}\}$ is a subset of W with no minimal elements. For the converse, suppose that $A \subseteq W$ is non-empty and has no minimal elements. Choose $a_0 \in A$, which is possible since A is non-empty; and for each $i \in \mathbf{N}$, put a_{i+1} to be some $x \in A$ with $x < a_i$, which is always possible since a_i is never a minimal element of A. This gives us an infinitely descending chain $\ldots a_2 < a_1 < a_0$ of elements of $A \subseteq W$, and we are done.

This argument is short, but at the same time subtle. The second part of it, which looks so simple, implicitly made use of an axiom of set theory called the axiom of choice, which is not discussed in this chapter. For our purposes, the important point is to see the equivalence in intuitive terms.

To help appreciate the contours of the concept, we give some positive and negative examples. Clearly, the set $\mathbf{N}$ of all natural numbers is well founded under the usual ordering, since every non-empty set of natural numbers has a minimal element (in fact a unique least element). This contrasts with the set $\mathbf{Z}$ of all integers, which is not well founded under the customary ordering since it has a subset (for instance, that of the negative integers, or indeed the whole of $\mathbf{Z}$) that does not have a minimal element.

Exercise 4.7.1 (1) (with solutions)

(a) Is the set of all non-negative rational numbers well founded under its usual $<$?

(b) Is the empty set well founded under the empty relation?
(c) Show that every finite set A is well founded under any transitive irreflexive relation over it.
(d) Let A be any finite set. Show that its power set $\mathcal{P}(A)$ is well founded under the relation of proper set inclusion.
(e) Can you find a subset of **N** that is not well founded under the usual relation $<$?

Solutions

(a) No. Consider the subset consisting of all rationals greater than zero; it has no minimal element.
(b) Yes, vacuously, as it has no non-empty subsets.
(c) *Hint*: Take any non-empty subset $B \subseteq A$. Do an induction on the number of elements of B, making it clear where you use the irreflexivity and transitivity of the relation in question.
(d) Since A is finite, so is $\mathscr{P}(A)$, so we can apply (c).
(e) No: Every subset $A \subseteq \mathbf{N}$ is well founded under $<$, because all of its non-empty subsets are non-empty subsets of N and so have a minimal element.

Parts (c) and (d) of the exercise above give us examples of sets that are well founded under relations that are not linear, that is, we may have distinct elements a, b with neither $a < b$ nor $b < a$. For when a set A has more than one element, it will clearly have subsets that are incomparable under inclusion, for example, distinct singletons.

Exercise 4.7.1 (2)

(a) Show that the collection $\mathscr{P}(\mathbf{N})$ of all subsets (finite or infinite) of **N** is *not* well founded under inclusion. *Hint*: Use the definition in terms of infinite descending chains.
(b) Let A be any infinite set. Show that the collection $\mathscr{P}_f(A)$ of all its *finite* subsets is infinite, but is well founded under inclusion.

In the special case that a set is well founded by a linear relation, so that for all a,b in the set, either $a = b$ or $a < b$ or $b < a$, we say that it is well ordered. Thus, unpacking the definition, a set W is *well ordered* by a relation $<$ iff $<$ is a linear order of W satisfying the condition that every non-empty subset of W has a minimal element.

Thus, **N** and all of its subsets are well ordered under the usual $<$. However, a well-ordered set can in a natural sense be 'longer' than **N**. For example, if we take the natural numbers in their standard order, followed by the negative integers in the *converse* of their usual order, giving us the set $\{0, 1, 2, \ldots; -1, -2, -3, \ldots\}$, then this is well ordered in the order of listing. Clearly, the operation can be repeated as many times as we like. It opens the door to the theory of transfinite ordinal numbers, but we will stop short of going through it.

Exercise 4.7.1 (3)

(a) Reorder **N** itself in such a way as to obtain a relation that well orders it in a way that is, in the same natural sense, longer than that given by $<$. *Hint*: Don't do anything complicated.
(b) Describe a relation that well-orders $\mathbf{N}\times\mathbf{N}$, built out of the usual ordering $<$ of **N** itself. *Hint*: Try making an infinite table and zigzag from the top left corner.

(c) Show that if W is well ordered, then every non-empty subset $A \subseteq W$ has a *unique* minimal element, which is moreover *the least* element of the subset, in the sense that it is *less than* every other element of the subset.
(d) Show how, from any well-ordered set, we may form one that is not well ordered but is still well founded, by adding a single element.

4.7.2 Proof by Well-Founded Induction

Roughly speaking, a well-founded relation provides a ladder up which we can climb in a set. This intuition is expressed rigorously by the *principle of induction over well-founded sets*, as follows. Let W be any well-founded set, and consider any property. To show that every element of W has the property, it suffices to show:

Induction step: The property holds of an arbitrary element $a \in W$ whenever it holds of all $b \in W$ with $b < a$.

Note that the principle has no basis. In this, it is like cumulative induction on the positive integers. Indeed, it may be seen as a direct abstraction of the principle of cumulative induction, from the specific order over the natural numbers that we are familiar with, to any well-founded relation over any set whatsoever. Of course, we could write in a basis, which would be: every minimal element of W itself has the property in question. But it would be redundant, and so in this context is customarily omitted.

Exercise 4.7.2 (1) (with solution)
(a) Formulate the principle of induction over well-founded sets in contrapositive form.
(b) Formulate it as a statement about subsets of W rather than about properties of elements of W.
(c) Contrapose the formulation in (b).

Solution
(a) Let W be any well-founded set, and consider any property. If the property does not hold of all elements of W, then there is an $a \in W$ that lacks the property, although every $b \in W$ with $b < a$ has the property.
(b) Let W be any well-founded set, and let $A \subseteq W$. If $a \in A$ whenever $b \in A$ for every $b \in W$ with $b < a$, then $A = W$.
(c) Let W be any well-founded set, and let $A \subseteq W$. If $A \neq W$, then there is an $a \in W$ with $a \notin A$, although $b \in A$ for every $b \in W$ with $b < a$.

The proof of the principle of induction over well-founded sets is remarkably brief, considering its power. Let W be any well-founded set, and consider any property. Suppose that (1) the property holds of an arbitrary element $a \in W$ whenever it holds of all $b \in W$ with $b < a$, but (2) it does not hold of all elements of W. We get a contradiction.

Let A be the set consisting of those elements of W that do *not* have the property in question. By the second supposition, A is not empty, so, since W is well founded,

A has a minimal element a. Thus, on the one hand, a does not have the property. But on the other hand, since a is a minimal element of A, every $b \in W$ with $b < a$ is outside A, and so *does* have the property in question, contradicting supposition (1), and we are done.

Alice Box: Proving the Principle of Induction Over Well-Founded Sets

Alice: That's certainly brief for such a general principle – although I would not say it is easy. But there is something about the proof that bothers me. We showed that the principle holds for any set that is well founded under a relation.

Hatter: Yes, indeed.

Alice: In the definition of a well-founded set, we required that the relation be irreflexive and transitive, as well as satisfying the 'minimal element' condition that every non-empty subset has at least one minimal element. I see how we used the minimal element condition in the proof, but as far as I can see, we didn't use irreflexivity or transitivity. Does this mean that we can generalize the principle of well-founded induction by dropping those two requirements from the definition of a well-founded set?

Hatter: I guess so.

Alice and the Hatter guessed right, but we should perhaps say a bit more. Dropping the condition of irreflexivity from the definition of a well-founding relation does not really change anything. That is because irreflexivity is implied, anyway, by the minimal-element condition: If $a < a$, then the non-empty set $\{a\}$ does not have a minimal element under $<$. It can also be shown that dropping the condition of transitivity from the definition does not really strengthen the principle of well-founded induction: The 'strengthened' version can in fact be derived, in a rather devious manner, from the standard one that we have stated above. Moreover, applications of the principle almost always work with a transitive relation. So we rest content with that formulation.

Exercise 4.7.2 (2) Show that the minimal-element condition implies acyclicity.

When a proof by induction is combined with reductio ad absurdum, it is often set out in a special way, which we might call a *wlog presentation*. We may be considering a set W that is well founded under a relation $<$, and want to show that every element of that set has a certain property. We suppose for *reductio* that there is an $a \in W$ that lacks the property, and we let A be the set of all those elements. Since $a \in A$, we know that A is non-empty, so well ordering tells us that it has a minimal element a'. We then go on to get a contradiction by showing that there must nevertheless be a $b < a'$ lacking the property.

When this mode of argument is used, it is customary to abbreviate it. After supposing that there is an $a \in W$ that lacks the property, one says simply: ‘we may assume without loss of generality (briefly, *wlog*) that a is minimal’. Here, ‘minimal’ is understood to mean ‘minimal among the elements of W lacking the property’ (not to be confused with being a minimal element of W itself). This turn of phrase avoids introducing another variable a' as above, which can be a bit messy when the property under investigation is a complicated one. In the particular case that the well founding is linear and so a well ordering, then we say: ‘we may assume *wlog* that a is *the least* such element of W’.

This may all sound rather *recherché* and even perverse, but in fact it comes quite naturally, and it helps reduce notation. In the chapter on trees (specifically, in Sect. 7.6.2), we will see several examples of its use on the natural numbers under their usual ordering $<$.

Exercise 4.7.2 (3) Look up a standard textbook proof of the irrationality of $\sqrt{2}$ and put your finger on where it uses *wlog*.

Alice Box: Wlog Proofs

Alice : Is that the only situation in which we may assume something without loss of generality?

Hatter: No, but in practice, it is one of the most common.

Alice: When else can one do it?

Hatter : Whenever we are given that there is an a with a certain property P, and we can easily verify that whenever there is such an a, there is also an a' (not necessarily identical with a) that has both the property P and another property Q. Then, instead of restating and relettering everything for a', which can be quite a nuisance if the property P is a complex one, we simply say that *we may assume wlog* that a also has the property Q.

Once again, we can add a bit more to the Hatter’s response. There are still more occasions where an assumption may be made *wlog*. For example, suppose that we wish to show that all elements of a certain domain have a certain property. We may have at our disposal a *representation* or *normal form* theorem to the effect that every element of the domain is equivalent, in a suitable sense, to one of a special kind. Then we may begin our proof by taking an arbitrary element of the domain and assuming, without loss of generality, that the element is of the special kind. But care is needed. The move is legitimate only when such a normal form or representation theorem is available, and provided that the kind of equivalence that it talks about is sufficient to transfer the property that one is trying to establish.

4.7.3 Defining Functions by Well-Founded Recursion on Their Domains

Induction for well-founded sets is a principle of proof. Is there a corresponding principle for definition, guaranteeing the existence of functions that are defined by recursion on a well-founded domain? The answer is positive. However, stating the formal principle in its full generality is quite subtle, and we will not attempt it here. We supply a rather informal formulation that covers most of the cases that a computer scientist is likely to need.

Principle of recursive definition on well-founded sets: Let W be any set that is well founded under a relation $<$. Then we may safely define a function f: $W \to X$ by giving a rule that specifies, for every $a \in W$, the value of $f(a)$ in terms of the values of $f(b)$ for some collection of $b < a$, using any other functions and sets that are already well defined. 'Safely' here means that *there exists a unique function satisfying the definition*.

For confident and adventurous readers who have managed to follow so far, we illustrate the principle by using it to show that the Ackermann function is well defined. Those not so confident may skip directly to the next section. We recall the recursive definition of the Ackerman function:

$$\begin{aligned} &A(0,n) = n+1 \\ &A(m,0) = A(m-1,1) \quad \text{for } m > 0 \\ &A(m,n) = A(m-1, A(m, n-1)) \quad \text{for } m,n > 0. \end{aligned}$$

This function has two arguments, both from $\mathbf{N}$, so we turn it into a one-argument function on $\mathbf{N}^2 = \mathbf{N}\times\mathbf{N}$ by reading the round brackets in the definition as indicating ordered pairs. Given the familiar relation $<$ over $\mathbf{N}$, which we know to be well founded, indeed to be a well ordering of $\mathbf{N}$, we define the *lexicographic order* over $\mathbf{N}^2$ by the rule: $(m,n) < (m',n')$ iff either $m < m'$ or $m = m'$ and $n < n'$.

This is a very useful way of ordering the Cartesian product of any two well-founded sets, and deserves to be remembered in its own right. The reason for the name 'lexicographic' will be clear if you think of the case where instead of $\mathbf{N}$, we consider its initial segment $A = \{n\text{: } 0 \leq n \leq 26\}$ and identify these with the letters of the English alphabet. The lexicographic order of A^2 then corresponds to its dictionary order.

We first check that the lexicographic relation $<$ is irreflexive, transitive, and then that it well-founds $\mathbf{N}^2$ (as requested in the following exercise). Having done that, we are ready to apply the principle of recursive definition on well-founded sets. All we have to do is show that the two recursion clauses of the candidate definition specify, for every $(m,n) \in \mathbf{N}^2$, the value of $A(m,n)$ in terms of the values of $A(p,q)$ for a collection of pairs $(p,q) < (m,n)$, using any other functions and sets already known to exist.

This amounts to showing for the first recursion clause, that $(m-1,1) < (m,0)$ and, for the second clause of the recursion step, that $(m-1, A(m, n-1)) < (m,n)$, for all $m,n > 0$. But both of these are immediate from the definition of the lexicographic order $<$. We have $(m-1,1) < (m,0)$ because $m-1 < m$, regardless of the fact that

$1 > 0$. Likewise, we have $(m-1, A(m, n-1)) < (m,n)$ no matter how large $A(m, n-1)$ is, again because $m-1 < m$.

Exercise 4.7.3

(a) Check the claim made in the text, that the lexicographic order of $\mathbf{N}^2$ is irreflexive, transitive, and that it well-founds $\mathbf{N}^2$.

(b) Show that in addition it is linear and so a well ordering of $\mathbf{N}^2$.

4.7.4 Recursive Programs

What is the connection between all this and *recursive programs* for computers? Does the mathematical work on recursive definitions and inductive proofs do anything to help us understand how programs work and what they can do?

The link lies in the fact that a program can be seen as a finite battery of instructions to perform computations for a potentially infinite range of inputs. Simplifying a great deal, and restricting ourselves to the principle case of *deterministic* programs (those in which the instructions suffice to fully determine each step in terms of its predecessors), a program may be understood as a recipe for defining a function that takes an input a to a finite or infinite succession $a_0, a_1, \ldots$ of successively computed items. If the sequence is finite, the last item may be regarded as the output.

Equivalently, we are looking at a recipe for defining a two-place function taking each pair (a,n), where a is a possible input and n is a natural number, to an item a_n. The recipe is recursive in the sense that the value of a_n may depend upon the values of any of the earlier a_m, in whatever way that the program specifies, so long as each step to a_n from the earlier steps may actually be performed by the computer that is executing the program. Typically, the 'while' and 'until' clauses setting up loops in the program, form part of the recursion step in the definition of the associated function.

The question thus arises: When does the proposed recipe really give us a program, that is, when does it really define a function? If we are writing the program in a well-constructed programming language, then the tight constraints that have been imposed on the grammar of the language may suffice to guarantee that it is well defined. Hopefully, those who designed the language will have proven that by induction before making the language available to the user. If on the other hand we are working in a lax or informal language, the question of whether we have succeeded in specifying a unique function needs to be analyzed by the programmer, by expressing it in an explicitly recursive form and verifying by induction that it always gives a unique value.

When the recipe does give us a function, other questions remain. Is the program guaranteed to terminate, in other words, is the sequence of steps a_n finite? If it does terminate, does the output give us what we would like it to give? Answers to these two questions also require *proof*, and the proof will always involve inductive arguments of one kind or other, perhaps of many kinds. In some simple cases, it may be sufficient to induce on the number of loops in the program. For more complex programs, the inductive arguments can be less straightforward. In general, one needs to devise a suitable relation over an appropriate set of items, verify that it is well founded and then use well-founded induction over the relation.

End-of-Chapter Exercises

Exercise 4.1: Proof by simple induction

(a) Use simple induction to show that for every positive integer n, $5^n - 1$ is divisible by 4.

(b) Use simple induction to show that for every positive integer n, $n^3 - n$ is divisible by 3. *Hint*: In the induction step, you will need to make use of the arithmetic fact that $(k+1)^3 = k^3 + 3k^2 + 3k + 1$.

(c) Show by simple induction that for every natural number n, $\Sigma\{2^i\colon 0 \leq i \leq n\} = 2^{n+1} - 1$.

Exercise 4.2: Definition by simple recursion

(a) Let f: $\mathbf{N}\rightarrow\mathbf{N}$ be the function defined by putting $f(0) = 0$ and $f(n+1) = n$ for all $n \in N$.

 (i) Evaluate this function bottom-up for all arguments 0–5.

 (ii) Explain what f does by expressing it in explicit terms (i.e. without a recursion).

(b) Let g: $\mathbf{N}\mathrm{x}\mathbf{N}\rightarrow\mathbf{N}$ be defined by putting $g(m,0) = m$ for all $m \in N$ and $g(m,n+1) = f(g(m,n))$ where f is the function defined in the first part of this exercise.

 (i) Evaluate $f(3,4)$ top-down.

 (ii) Explain what f does by expressing it in explicit terms (i.e. without a recursion).

(c) Let f: $\mathbf{N}^+\rightarrow\mathbf{N}$ be the function that takes each positive integer n to the greatest natural number p with $2^p \leq n$. Define this function by a simple recursion. *Hint*: You will need to divide the recursion step into two cases.

Exercise 4.3: Proof by cumulative induction

(a) Use cumulative induction to show that any postage cost of four or more pence can be covered by two-pence and five-pence stamps.

(b) Use cumulative induction to show that for every natural number n, $F(n) \leq 2^n - 1$, where F is the Fibonacci function.

(c) Calculate $F(5)$ top-down, and then again bottom-up, where again F is the Fibonacci function.

(d) Express each of the numbers 14, 15 and 16 as a sum of 3s and/or 8s. Using this fact in your basis, show by cumulative induction that every positive integer $n \geq 14$ may be expressed as a sum of 3s and/or 8s.

(e) Show by induction that for every natural number n, $A(1,n) = n+2$, where A is the Ackermann function.

Exercise 4.4: Structural recursion and induction

(a) Define by structural recursion the set of all odd palindromes over a given alphabet.

(b) Show by structural induction that every even palindrome is either empty or contains two contiguous occurrences of the same letter.

(c) Define by structural recursion the set of all palindromes (over a given alphabet) that contain no two contiguous occurrences of the same letter. *Hint*: Use the fact given by (b).

(d) Show by structural induction that in classical propositional logic, no formula built using propositional letters and connectives from the set $\{\wedge,\vee,\rightarrow,\leftrightarrow\}$ is a contradiction. *Hint*: Guess an assignment of truth-values that looks like it will make all such formulae true, and show by structural induction that it really does so.

(e) Justify the principle of structural induction by using cumulative induction over the natural numbers. *Hint*: To keep the exposition simple, consider the closure A^+ of A under a single relation R, and use the bottom-up definition of A^+ as the union of sets $A_0, A_1, \ldots$.

Exercise 4.5: Definition of functions by structural recursion on their domains

(a) Consider any alphabet A and the set A^* of all strings made up from A. Intuitively, the *reversal* of a string s is the string obtained by reading all its letters from right to left instead of left to right. Provide a structural recursive definition for this operation on strings. *Hint*: the base should concern the empty string, called λ, and the induction step should use the operation *cons*.

(b) In presentations of propositional logic, it is customary to define an *assignment* of truth-values to be any function v: $A\rightarrow\{1,0\}$, where A is some stock of elementary letters $p, q, r, \ldots$. Any such assignment is then extended to a valuation v^+: $A^+\rightarrow\{1,0\}$ where A^+ is the set of all formulae (i.e. the closure of A under the operations of forming, say, negations, conjunctions and disjunctions) in a way that respects the usual truth tables. It is tacitly assumed that this extension is unique. Analyze what is being done in terms of structural recursion.

Exercise 4.6: Well-founded sets*

(a) Show that every subset of a well-founded set is well founded under the same relation (strictly speaking, under restriction of that relation to the subset, but let's not be too pedantic).

(b) Is the converse of a well-founding relation always a well-founding relation? Give proof or counterexample.

(c) Given well orderings $<_1$, $<_2$ of disjoint sets W_1, W_2, define a well ordering of their union $W_1 \cup W_2$. How would you modify the definition for the case that the two sets are not disjoint?

(d) Given well orderings $<_1$, $<_2$ of disjoint sets W_1, W_2, define a well ordering of their Cartesian product $W_1 \times W_2$.

(e) Let W be a set, R any relation over it, and let R^* be the transitive closure of R (Sect. 2.7.1). Let $A \subseteq W$ be non-empty. (i) Explain (in one sentence) why any infinite descending chain $\ldots a_2 R a_1 R a_0$ of elements of W under R is an infinite descending chain under R^*. (ii) Give a simple counterexample to the converse. (iii) Show that, nevertheless, if there is an infinite descending chain under R^*, then there is also an infinite descending chain under *R*. *Remark*: These facts underlie the 'devious' proof of equivalence mentioned in Sect. 4.7.2.

Selected Reading

Induction and recursion on the positive integers. There are plenty of texts, although most tend to neglect recursive definition in favour of inductive proof. Here are three among them:

Hein JL (2005) Discrete structures, logic and computability, 2nd edn. Jones and Bartlett, Boston, chapter 4.4

Schumacher C (2001) Chapter zero: fundamental notions of abstract mathematics. Pearson, Boston, chapter 3

Velleman DJ (2006) How to prove it: a structured approach, 2nd edn. Cambridge University Press, Cambridge, chapter 6

Well-founded induction and recursion. Introductory accounts tend to be written for students of mathematics rather than computer science and, again, tend to neglect recursive definition. Two texts accessible to computer science students are:

Lipschutz S (1998) Set theory and related topics, Schaum's outline series. McGraw Hill, New York, chapters 8–9

Halmos PR (2001) Naive set theory, New edn. Springer, New York, chapters 17–19

Structural induction and recursion. It is rather difficult to find introductory presentations. One of the purposes of the present chapter has been to fill the gap!

Counting Things: Combinatorics 5

Abstract

Up to now, our work has been almost entirely qualitative. The concepts of a set, relation, function, recursion and induction are non-numerical, although they have important numerical applications as, for example, sets of integers or recursive definitions on the natural numbers. In this chapter, we turn to quantitative matters, and specifically to problems of counting. We will tackle two kinds of question.

First: Are there rules for determining the number of elements of a large set from the number of elements of smaller ones? Here we will learn how to use two very simple rules: *addition* (for unions of disjoint sets) and *multiplication* (for Cartesian products of arbitrary sets).

Second: Are there rules for calculating the number of possible selections of k items out of a set with n elements? Here we will see that the question is less straightforward, as there are several different kinds of selection, and they give us very different outcomes. We will untangle *four basic modes of selection*, give arithmetical formulae for them and practice their application. In the final section, we will turn to the problems of counting *rearrangements* and *configured partitions* of a set.

5.1 Basic Principles for Addition and Multiplication

In earlier chapters, we already saw some principles for calculating the number of elements of one set, given the number in another. In particular, in Chap. 3, we noted the *equinumerosity principle*: two finite sets have the same number of elements iff there is a bijection from one to the other. In the exercises at the end of Chap. 1, we noted an important equality for difference, and another one for disjoint union. They provide our starting point in this chapter, and we begin by recalling them.

D. Makinson, *Sets, Logic and Maths for Computing*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-2500-6_5,

5.1.1 Principles Considered Separately

For the difference $A\backslash B$ of A with respect to B, that is, $\{a \in A: a \notin B\}$, we observed in Chap. 1:

Subtraction principle for difference: Let A, B be finite sets. Then $\#(A\backslash B) = \#(A) - \#(A \cap B)$.

For union we saw:

Addition principle for two disjoint sets: Let A, B be finite sets. If they are disjoint, then $\#(A \cup B) = \#(A) + \#(B)$.

The condition of disjointedness is essential here. For example, when $A = \{1,2\}$ and $B = \{2,3\}$, then $A \cup B = \{1,2,3\}$ with only three elements, not four.

Clearly, the addition principle can be generalized to n sets, provided they are *pairwise disjoint* in the sense of Chap. 1, that is, for any $i,j \leq n$, if $i \neq j$, then $A_i \cap A_j = \varnothing$.

Addition principle for many pairwise disjoint sets: Let $A_1, \ldots, A_n$ be pairwise disjoint finite sets. Then, $\#(\cup\{A_i\}_{i \leq n}) = \#(A_1) + \cdots + \#(A_n)$.

Exercise 5.1.1 (1) (with solution) Let A, B be finite sets. Formulate and verify a necessary and sufficient condition for $\#(A \cup B) = \#(A)$.

Solution $\#(A \cup B) = \#(A)$ iff $B \subseteq A$. Reason: If $B \subseteq A$, then $A \cup B = A$, so $\#(A \cup B) = \#(A)$. Conversely, if the inclusion does not hold, then there is a $b \in B\backslash A$, so $\#(A) < \#(A)+1 = \#(A \cup \{b\}) \leq \#(A \cup B)$.

Can we say anything for union when the sets A, B are not disjoint? Yes, by breaking them down into disjoint components. For example, we know that $A \cup B = (A \cap B) \cup (A\backslash B) \cup (B\backslash A)$, and these are disjoint, so $\#(A \cup B) = \#(A \cap B) + \#(A\backslash B) + \#(B\backslash A)$. But we can go further. Applying the subtraction principle to this, we have:

$$\begin{aligned}\#(A \cup B) &= \#(A \cap B) + \#(A) - \#(A \cap B) + \#(B) - \#(A \cap B)\\ &= \#(A \cap B) - \#(A \cap B) - \#(A \cap B) + \#(A) + \#(B)\\ &= \#(A) + \#(B) - \#(A \cap B).\end{aligned}$$

We have thus shown the following:

Addition principle for any two finite sets: Let A, B be any finite sets. Then, $\#(A \cup B) = \#(A) + \#(B) - \#(A \cap B)$.

Exercise 5.1.1 (2)

(a) Use a Venn diagram to illustrate the addition principle for two disjoint sets.
(b) Do the same for the non-disjoint version.
(c) Check the claim that $A \cup B = (A \cap B) \cup (A\backslash B) \cup (B\backslash A)$.

Alice Box: Addition Principle for Many Finite Sets

Alice: We generalized the addition principle for two disjoint sets to *n* disjoint sets. Can we make a similar generalization when the sets are not assumed to be disjoint?

Hatter: Indeed we can, but its formulation is rather more complex. To understand it properly, we need the notion of an arbitrary combination of *n* things *k* at a time, which we will get to later in the chapter. So let's take a rain-check on that one.

Exercise 5.1.1 (3) (with solution)

(a) The classroom contains 17 male students, 18 female students and the professor. How many altogether in the classroom?

(b) The logic class has 20 students who also take calculus, 9 who also take a philosophy unit, 11 who take neither, and 2 who take both calculus and a philosophy unit. How many students in the class?

Solution

(a) $17 + 18 + 1 = 36$, using the addition principle for $n = 3$ disjoint sets.

(b) $((20 + 9) - 2) + 11 = 38$, using the addition principle for two arbitrary finite sets to calculate the number taking either calculus or philosophy (in the outer parentheses), and then applying the addition principle for two disjoint sets to cover those who take neither.

Applications of the addition principles taken alone are usually quite trivial. Their power comes from their joint use with other rules, notably the multiplication principle, which tells us the following:

Multiplication rule for two sets. Let A, B be finite sets. Then, $\#(A\times B) = \#(A)\cdot\#(B)$.

In words: The number of elements in the Cartesian product of two sets is equal to the product of the numbers in each. Reason: If B has n elements, then for each $a \in A$, $\#\{(a,b)\colon b \in B\} = n$. If A has m elements, then there are m of these sets $\{(a,b)\colon b \in B\}$ for $a \in A$. Moreover, they are disjoint, and their union is $A\times B$. So, $\#(A\times B) = n + \ldots + n$ (m times) $= m\cdot n = \#(A)\cdot\#(B)$. The same reasoning gives us more generally:

Multiplication rule for many sets. $\#(A_1\times \ldots \times A_n) = \#(A) \cdot \ldots \cdot\#(B)$ for any finite sets $A_1, \ldots, A_n$.

Exercise 5.1.1 (4) (with solution) The menu at our restaurant allows choice: For first course, one can choose either soup or salad, the second course can be either beef, duck or vegetarian, followed by a choice of fruit or cheese, ending with black tea, green tea or coffee. How many selections are possible?

Solution $2{\cdot}3{\cdot}2{\cdot}3 = 36$. The selections are independent of each other, and so are in one-one correspondence with the Cartesian product of four sets with 2,3,2,3 elements, respectively.

Alice Box: The Maverick Diner

Alice: Not if I am there!
Hatter: What do you mean?
Alice: Well, I never drink anything with caffeine in it, so I would skip the last choice. And following an old European custom, I prefer to take my fruit before anything else, so I would not follow the standard order. So *my* selection would be none of your 36. It would be, say, (fruit, salad, duck, nothing).

Alice has put her finger on an important issue. When a real-life counting problem is described, its presentation is frequently underdetermined. In other words, certain aspects are left implicit, and often they can be filled in different ways. For example, in the restaurant problem, it was tacitly assumed that everyone chooses exactly one dish from each category, and that the order of the categories is fixed: any other reordering is either disallowed or regarded as equivalent to the listed order. If we drop these assumptions, we get quite different numbers of selections.

In general, the trickiest part of a real-life counting problem is not to be found in the calculations to be made when applying a standard mathematical formula. It lies in working out *which* (if any) of the mathematical formulae in one's toolkit is the appropriate one. And that depends on understanding what the problem is about and locating its possible nuances. We need to know what items are being selected, from what categories, in what manner. Especially, we need to know when two items, or two categories or two selections are to be regarded as identical – in other words, are to be counted as one, not as two. Only then can we safely give the problem an abstract representation that justifies the application of one of the formulae in the toolkit, rather than one of the others. We will see more examples of this as we go on.

Exercise 5.1.1 (5)

(a) Telephone numbers in the London area begin with 020 and continue with eight more digits. How many are there?
(b) How many distinct licence plates are there consisting of two letters other than O, I and S (to prevent possible visual confusion with similar-looking digits), followed by four digits?

5.1.2 Using the Two Principles Together

Often the solution of a problem requires the use of both addition and multiplication principles. Here is a simple example. How many four-digit numbers begin with 5 or with 73?

We begin by breaking our set (of all four-digit numbers beginning with 5 or with 73) into two disjoint subsets – those beginning with 5 and those beginning with 73. We determine their numbers separately, and then by the addition principle for disjoint sets, we add them. In the first case, with the digit 5, there are three digits still to be filled in, with 10 choices for each, so we get $10^3 = 1{,}000$ possibilities. In the second case, beginning with 73, we have two digits to fill in; hence, $10^2 = 100$ possibilities. So, the total is 1,100 four-digit numbers beginning with 5 or with 73.

This kind of procedure is quite common, and so we look at its general form. We are required to determine #(A) – the number of elements of a set A. We observe that $A = A_1 \cup \ldots \cup A_n$ where the sets A_i are pairwise disjoint. We then note that each of the sets A_i is (or may be put in one-one correspondence with) a Cartesian product $A_{\mathrm{i}1} \times \ldots \times A_{ik}$ (where k may depend on i). So we apply n times the multiplication rule to get the value of each #(A_i), and then apply once the addition rule for the disjoint sets to get the value of #(A) itself.

Exercise 5.1.2 (1) (with partial solution)

(a) You have six shirts, five ties and four pairs of jeans. You must wear a shirt and a pair of trousers, but maybe not a tie. How many outfits are possible?

(b) A tag consists of a sequence of four alphanumeric signs (letters or digits). How many tags with alternating letters and digits begin with either the digit 9 or the letter *m*? *Warning*: Pay attention to the requirement of alternation.

Solution to (a) The set of possible outfits is the union of two sets: $S \times J$ and $S \times J \times T$. Note that these two sets are disjoint (why?). There are $6 \cdot 4 = 24$ elements of $S \times J$, and $6 \cdot 4 \cdot 5 = 120$ elements of $S \times J \times T$. So there are 144 attires in which to sally forth. Another way of getting the same answer: Let T' be the six-element set consisting of the five ties plus 'no tie'. The set of possible outfits is $S \times J \times T'$, which has $6 \cdot 4 \cdot 6 = 144$ elements.

5.2 Four Ways of Selecting *k* Items Out of *n*

A club has 20 members, and volunteers are needed to set up a weekly roster to clean up the coffee room, one for each of the 6 days of the week that the club is open. How many possible ways of doing this?

This question has no answer! Or rather, it has several different answers, according to how we interpret it. The general form is: how many ways of selecting k items (here 6 volunteers) out of a set with n elements (here, the pool of 20 members). There are two basic questions that always need to be asked before the counting can begin: whether order matters, and whether repetition is allowed.

Table 5.1 Cells not to count

	a	*b*	*c*	*d*	*e*
a	R−				
b	O−	R−			
c	O−	O−	R−		
d	O−	O−	O−	R−	
e	O−	O−	O−	O−	R−

5.2.1 Order and Repetition

Expressed in more detail, the two preliminary questions that we need to answer are the following:

- Does the *order* in which the selection is made matter for counting purposes? For example, is the roster with me serving on Monday and you on Tuesday regarded as different from the one that is the same for all other days but with our time-slots swapped? Do we count them as two rosters or as one?
- Is *repetition* allowed? For example, are you allowed to volunteer for more than one day, say, both Monday and Friday?

These two options evidently give rise to four cases, which we will need to distinguish carefully in our analysis.

Order matters, repetition allowed	O+R+
Order matters, repetition not allowed	O+R−
Order doesn't matter, repetition allowed	O−R+
Order doesn't matter, repetition not allowed	O−R−

To illustrate the difference in a two-dimensional table, consider a similar example with just two volunteers, and to keep the diagram small, take the total number of club members to be five, whom we call a,b,c,d,e. Our table will then contain $5{\cdot}5 = 25$ cells for the 25 pairs (x,y) of members. If we select volunteers (b,d) say, then this is represented by the cell for row b and column d (Table 5.1).

The decisions we make concerning order and repetition will affect which cells are considered possible and which are taken as distinct from each other. Thus, in both cases, these decisions determine *which cells not to count*.

Mode O+R+: If order matters and repetition is allowed, then each cell represents a possible selection of two volunteers out of five, and all are considered different. There are thus 25 possible selections.

Mode O+R−: If order matters but repetition is not allowed, then the diagonal cells do not represent allowed selections, but the remaining cells are all distinct. There are thus $25 - 5 = 20$ possible selections, subtracting the five diagonal ones (marked R−) from the total.

Mode O−R+: If order does not matter but repetition is permitted, then all cells are allowed, but those to the upper right of the diagonal give the same outcome as their images in the bottom left. For example, (b,d) represents the same selection as (d,b). In this case, we have only $25-10=15$ possible selections, subtracting the 10 to the bottom left of the diagonal (marked O−) to avoid counting them twice.

Mode O−R−: If order does not matter and repetition is not allowed, we have even less. We must subtract from the previous count, the non-permitted cells of the diagonal, leaving only those to the upper right. Thus, we get only $15-5=10$ selections. Equivalently, we could subtract from the count preceding that (the one for O+R−), the ten duplicating cells to the bottom left of the diagonal, likewise giving $20-10=10$ selections. In other words, we subtract from the total 25 both those marked R− (not permitted) and those marked O− (duplicates).

Thus, according to the way in which we understand the selection, we get four different answers: 25, 20, 15 or 10 possible selections! The moral of the story is that we must consider the four modes of selection separately.

Exercise 5.2.1 Construct a similar table for the four different modes of choosing 2 elements out of 6, marking the respective areas in the same way, and giving the number of selections under each mode.

5.2.2 Connections with Functions

It is helpful to notice connections between the above and what we have learned about functions in Chap. 3. In the first two modes, where order is significant, we are in effect counting functions, as follows:

> For mode O+R+: When order matters and repetition is allowed, then each selection of k items from an n-element set may be represented by an ordered k-tuple $(a_1,\ldots,a_k)$ of elements of the set. As we already know, such an ordered k-tuple may be understood as a function on the set $\{1,\ldots,k\}$ into the set $\{a_1,\ldots,a_n\}$. Thus, counting the selections of k items from an n-element set $\{a_1,\ldots,a_n\}$, understood in this mode, is the same as counting *all the functions* from the k-element set $\{1,\ldots,k\}$ into $\{a_1,\ldots,a_n\}$.
>
> For mode O+R−: When order matters but repetition is not allowed, then the number of selections of k items from an n-element set $\{a_1,\ldots,a_n\}$ equals the number of *injective functions* from the k-element set $\{1,\ldots,k\}$ into $\{a_1,\ldots,a_n\}$. Injective, because when $i \neq j, f(i)$ is not allowed to be the same as $f(j)$.

In the fourth mode, we are no longer counting functions, but something very close which reduces to something very simple.

> For mode O−R−: When order does not matter and repetition is not allowed, then the number of selections of k items from an n-element set $\{a_1,\ldots,a_n\}$ is the same as the number of *ranges of injective functions* from a k-element set into $\{a_1,\ldots,a_n\}$. If you think about it, this is the number of k-element *subsets* of $\{a_1,\ldots,a_n\}$. Reason: Every such range is a k-element subset of $\{a_1,\ldots,a_n\}$, and

every k-element subset of $\{a_1, \ldots, a_n\}$ is the range of some such function; the two collections are thus identical and so have the same number of elements.

This leaves us with the third mode of selection, O−R+. We can also describe this in terms of functions, but they are no longer functions from $\{1, \ldots, k\}$ into the selection-pool $\{a_1, \ldots, a_n\}$. They are functions from $\{a_1, \ldots, a_n\}$ into $\{0, \ldots, k\}$ satisfying a certain condition.

Mode O−R+: When order doesn't matter but repetition is allowed, then the number of selections of k items from an n-element set $\{a_1, \ldots, a_n\}$ is the same as the number of functions from $\{a_1, \ldots, a_n\}$ into $\{1, \ldots, k\}$ satisfying the condition that $f(a_1)+\ldots+f(a_n) = k$. At first, this may appear rather mysterious, but the reason is quite simple. The function tells us how many times each element of the selection-pool $\{a_1, \ldots, a_n\}$ actually appears in the selection. It can be zero, one or more, but the sum of all appearances must come to k, as we are selecting k items.

Alice Box: Multisets

Alice: This is getting a bit tricky.

Hatter: I can tell you about another way of understanding O−R+ selections of k items from a pool of n, but it uses a language we have not seen before.

Alice: OK, go ahead.

Hatter: We can see them as picking out all the k-element *multisets* of a *set* of n elements.

Alice: What is a multiset?

Hatter: Roughly speaking, it is like a set, but allows repetitions. For example, when a and b are distinct items, the *set* $\{a,a,b,b\}$ has only two elements and is identical to the set $\{a,b\}$; but the multiset $[a,a,b,b]$ has *four* members, and is distinct from the multiset $[a,b]$. Multisets, also sometimes called *bags*, have recently gained some popularity in certain areas of computer science, and also in certain areas of logic dealing with the analysis of proofs.

Alice: Oh no! So now I have to learn a new version of Chap. 1, about the behaviour of multisets? I protest!

Hatter: OK, let's forget about them for this course. If ever you want to know more, you can begin with the *Wikipedia* article about them.

Exercise 5.2.2 (with solution) When $k > n$, that is, when the number of items to be selected is greater than the number of items in the selection-pool, which of the four modes of selection become impossible?

Solution The two modes that prohibit repetition (O+R− and O−R−) prevent any possible selection when $k > n$. For they require that there be at least k *distinct* items selected, which is not possible when there are less than k waiting to be chosen.

Table 5.2 Names for four modes of selecting k items from an n-element set

Generic term	Mode of selection	Particular modes	Notation
Selection	O+R−	Permutations	$P(n,k)$ and similar
	O−R−	Combinations	$C(n,k)$ and similar
	O+R+	Permutations with repetition allowed	Varies with author
	O−R+	Combinations with repetition allowed	Varies with author

In contrast, the two modes permitting repetition (O+R+ and O−R+) still allow selection when $k > n$.

So far, we have been calling the four modes by their order/repetition codes; it is time to give them names. Two of the four modes – those where repetition is *not* allowed – have widely accepted names going back centuries, long before any systematic account along the lines given here:

- When repetition is not allowed but order matters, that is, O+R−, the mode of selection is traditionally called *permutation*. The function counting the number of permutations of n elements k at a time is written $P(n,k)$, or in some texts as nPk or ${}^{n}P_{k}$ or similar.
- When repetition is not allowed and order does not matter, that is, O−R−, the mode of selection is traditionally called *combination*. The function counting the number of combinations of n elements k at a time is written $C(n,k)$, or for some authors, nCk or ${}^{n}C_{k}$ or similar. A common two-dimensional notation, very convenient in complex calculations, puts the n above the k within large round brackets.

These two modes are the ones that tend to crop up most frequently in traditional mathematical practice, for example, in the formulation of the binomial theorem going back to the sixteenth century. They are also the most common modes in computer science practice.

For the modes allowing repetition, terminology is distressingly variable from author to author. The simplest names for them piggyback on the standard ones for their counterparts without repetition, as follows:

- When repetition is allowed and order matters, that is, when we are in the mode O+R+, we will speak of *permutations with repetition allowed*.
- When repetition is allowed but order does not matter, that is, when we are in the mode O−R+, we will speak of *combinations with repetition allowed*.

In Table 5.2, we summarize the nomenclature. Note that it lists the four modes in a sequence different from that which was convenient for our conceptual explanation. It is more convenient for numerical analysis; it corresponds to the order in which we will shortly establish their respective counting formulae. It also corresponds, very roughly, to the relative importance of the four modes in applications. In the table, we use the term *selection* to cover, quite generally, all four modes. The subject of counting selections is often nicknamed the theory of *perms and coms*.

5.3 Counting Formulae: Permutations and Combinations

So far, we have been doing essentially conceptual work, sorting out different modes of selection. We now present counting formulae for the four modes of selection that were distinguished. We begin with the two in which repetition is not allowed: permutations (O+R−) and combinations (O−R−). Table 5.3 gives the number of permutations and combinations (without repetition) of k items from an n-element set.

Exercise 5.3

(a) Check that the figures obtained in Sect. 5.3 for choosing 2 out of 5 agree with the formulae in the table.

(b) Calculate the figures for choosing 4 out of 9 under the two modes, using the formulae in the table.

At this stage, it is tempting to plunge into a river of examples, to practice applying the counting formulae. Indeed, this is necessary if one is to master the material. But by itself it is not enough. We also need to see *why* each of the formulae does its job. On that basis, we can also understand *when* it, rather than its neighbour, should be used in a given problem.

5.3.1 The Formula for Permutations

For intuition, we begin with an example. How many six-digit numerals are there in which no digit appears more than once? The formula in our table says that it is $P(n,k) = n!/(n-k)! = 10!/(10-6)! = 10!/4! = 10{\cdot}9{\cdot}8{\cdot}7{\cdot}6{\cdot}5 = 151{,}200$. How do we get this?

We use the multiplication principle. There are n ways of choosing the first item. As repeated choice of the same element is *not* allowed, we thus have only $n-1$ ways of choosing the second, then only $n-2$ ways of choosing the third, and so on. If we do this k times, the number of possible selections is thus: $n{\cdot}(n-1){\cdot}(n-2){\cdot}\ldots{\cdot}(n-(k-1))$. Multiplying this by $(n-k)!/(n-k)!$ gives us $n!/(n-k)!$ in agreement with the counting formula.

Table 5.3 Formulae for perms and coms (without repetition)

Mode of selection	Notation	Standard name	Formula	Proviso
O+R−	$P(n,k)$	Permutations	$n!/(n-k)!$	$k \leq n$
O−R−	$C(n,k)$	Combinations	$n!/k!(n-k)!$	$k \leq n$

Alice Box: When k = n

Alice: I understand why we have the condition $k \leq n$ for permutations; in Exercise 5.2.2, we showed that when $k > n$, selection is impossible for the modes that disallow repetition. But what happens in the limiting case that $k = n$? It worries me, because in that case, $n - k$ is 0, and in Chap. 4, the factorial function was defined only for *positive* integers.

Hatter: Nice point! To cover that case, it is conventional to extend the definition of factorial by setting $0! = 1$. So when $k = n$, we have $P(n,k) = P(n,n) = n!/(n - n)! = n!/0! = n!$.

Although the case $k = n$ for permutations is a limiting one, it is very important and often arises in applications. Then, instead of the long-winded phrase 'permutation of n items n at a time', it is customary to speak briefly of *a permutation of n items*. These permutations correspond to the bijections between an n-element set and itself. Thus, the number $P(n,n)$ of permutations of an n-element set is the same as the number of bijections on that set into itself, and is equal to $n!$.

Exercise 5.3.1 (1) (with partial solution)

(a) Use the formula to calculate each of $P(6,0), \ldots, P(6,6)$.

(b) You have 15 ties, and you want to wear a different one each day of the working week (5 days). For how long can you go on without ever repeating a weekly sequence?

Solution to (b) Our pool is the set of ties, with $n = 15$ elements, and we are selecting $k = 5$ with order significant and repetition not allowed. We can thus apply the formula for permutations. There are thus $15!/(15 - 5)! = 15!/10! = 15 \cdot 14 \cdot 13 \cdot 12 \cdot 11 =$ 360,360 possible weekly selections. That means you can go on for nearly 7,000 years without repeating a weekly sequence. Moral: you have far too many ties!

Exercise 5.3.1 (2) (with partial solution)

(a) In words of ordinary English, how would you interpret the meaning of $P(n,k)$ when $k = 0$? Is the counting formula reasonable in this limiting case?

(b) Compare the values of $P(n,n)$ and $P(n,0)$. Any comments?

(c) Verify the claim in the text that $P(n,n)$ counts the number of bijections between a set of n elements and itself. *Hint*: Use the connection between of permutations in general and injections given in Sect. 5.2.2 and an exercise on bijections in Sect. 3.3.3.

Solutions to (a) and (b)

(a) $P(n,0)$ is the number of ways of selecting nothing from a set of n elements. The counting formula gives the value $P(n,0) = n!/n! = 1$. That is, there is one way of selecting nothing (namely, doing nothing). That's reasonable enough for a limiting case.

(b) As Alice remarked, $P(n,n) = n!/(n-n)! = n!/0! = n!$, while as we have seen in part (a), $P(n,0) = 1$. Comment: clearly, $P(n,k)$ takes its largest possible value when $k = n$, and its smallest when $k = 0$.

5.3.2 Counting Combinations

For concreteness, we again begin with an example. How many ways are there of choosing a six-person subcommittee out of a full ten-person committee? It is implicitly understood that the order of the members of the subcommittee is of no consequence, and that all six members of the subcommittee must be different people. That is, we are in mode O−R−; we are counting *combinations*. The counting formula in the table gives us $C(n,k) = n!/(k!(n-k)!)$ so that $C(10,6) = 10!/(6!4!)$ $= (10{\cdot}9{\cdot}8{\cdot}7)/(4{\cdot}3{\cdot}2) = 5{\cdot}3{\cdot}2{\cdot}7 = 210$. This is a tiny fraction of the 151,200 for permutations: to be precise, it is one 6!th, that is, one 720th. Order matters!

How can we prove the general formula for combinations? Notice that it says, in effect, that $C(n,k){\cdot}k! = P(n,k)$ for $k \leq n$, so it suffices to prove that. Now $C(n,k)$ counts the number of k-element subsets of an n-element set. We already know that each such subset can be given $k! = P(k,k)$ orderings. Hence, the total number $P(n,k)$ of *ordered* subsets is $C(n,k){\cdot}k!$, and we are done.

Exercise 5.3.2 (1) (with partial solution)

(a) Use the formula to calculate each of $C(6,0), \ldots C(6,6)$.

(b) Draw a chart with the seven values of k ($0 \leq k \leq 6$) on the abscissa (horizontal axis) and the values of each of $P(6,k)$ and $C(6,k)$ on the ordinate (vertical axis).

(c) Your investment advisor has given you a list of eight stocks attractive for investment. You decide to invest in three of them. How many different selections are possible?

(d) Same scenario, except that you decide to invest \$1,000 in one, double that in another, and double that again in a third. How many different selections are possible?

Solutions to (c) and (d) These two questions illustrate how important it is to understand, before applying a formula, what kind of selection we are supposed to be making.

(c) It is implicit in the formulation that you wish to invest in three *different* stocks – no repetition. It is also implicit that the order of the selection is to be disregarded – we are interested only in the subset selected. So we are counting combinations (mode O−R−) and can apply the formula for $C(n,k)$. Once that is clear, the rest is just calculation: $C(8,3) = 8!/(3!5!) = (8{\cdot}7{\cdot}6)/(3{\cdot}2) = 4{\cdot}7{\cdot}2 = 56$.

(d) In this question, it is again assumed that there is no repetition, but the order of the selection is regarded as important – we are interested in which stock is bought in what volume. So we are back with permutations (mode O+R−), and should apply the counting formula for $P(n,k)$, to get $P(8,3) = 8!/5! = 8{\cdot}7{\cdot}6 = 336$.

It is instructive to consider an example that looks quite like the last two of the exercise, but which reveals greater complexity. Your investment advisor has given you a list of eight stocks attractive for investment, and a linear ordering of these eight by their estimated risk. You decide to invest \$1,000 in a comparatively risky one, double that in a less risky one, and double that again in an even less risky one. How many different selections are possible? Now, the first choice must not be the safest stock, nor the second safest, but may be from any of the six riskier ones. So far, so good: there are six ways of choosing the first stock. But the range of options open for the second choice depends on how we made the first one: it must be on a lesser risk level than the first choice, but still not the least risky one. And the third choice depends similarly on the second. We thus have quite an intricate problem – it requires more than a simple application of any one of the counting formulae in our table. We will not attempt to solve it here, but merely emphasize the lesson: *real-life counting problems, apparently simple in their verbal presentation, can be quite nasty to solve.*

Exercise 5.3.2 (2) (with partial solution)

(a) In words of ordinary English, how would you interpret the meaning of $C(n,k)$ when $k = 0$?

(b) Explain why $C(n,k) \leq P(n,k)$ for all n,k with $1 \leq k \leq n$.

(c) State and prove a necessary and sufficient condition on n,k for $C(n,k) < P(n,k)$.

(d) Compare the values of $C(n,n)$, $C(n,1)$ and $C(n,0)$ with their counterparts $P(n,n)$, $P(n,1)$ and $P(n,0)$ and explain the similarities/differences in intuitive terms.

(e) Show from the counting formula that $C(n,k) = C(n,n-k)$.

(f) Suppose that n is even, that is, $n = 2m$. Show that for $j < k \leq m$ we have $C(n,j) < C(n,k)$.

(g) Same question but with n odd, that is, $n = 2m + 1$.

Solutions to (d), (e)

(d) For the combinations, we have $C(n,n) = n!/(n!(n-n)!) = (n-n)! = 0! = 1$, while $C(n,1) = n!/(1!(n-1)!) = n!/(n-1)! = n$, and $C(n,0) = n!/(0!(n-0)!) = n!/n! = 1$. For permutations, as we have already seen $P(n,n) = n!$, $P(n,1) = n$ and $P(n,0) = 1$. Thus, $P(n,n) = n! > 1 = C(n,n)$, as we would expect since there is just one subset of all the n items in our pool, but $n!$ ways of putting that subset in order. On the other hand, $C(n,0) = 1 = P(n,0)$ since there is just one way of selecting nothing, irrespective of order; and $C(n,1) = n = P(n,1)$ because when you are selecting just one thing, the question of order of selection does not arise.

(e) By the counting formula, $C(n,n-k) = n!/[(n-k)!\cdot(n-(n-k))!] = n!/[(n-k)!k!] = n!/[k!(n-k)!] = C(n,k)$ again by the counting formula.

This is a good moment to fulfil a promise that Hatter made to Alice in the first section of this chapter, when discussing the addition principle for two arbitrary sets: $\#(A \cup B) = \#(A) + \#(B) - \#(A \cap B)$. The unanswered question was: How do we generalize this from 2 to n arbitrary sets?

Recall how we arrived at the principle for two sets. We counted the elements of A and B separately and added them; but as the two sets may not be disjoint, that counted the elements in $A \cap B$ twice, so we subtract them once. For three sets A,B,C the reasoning is similar. We begin by adding the number of elements in each of the three. But some of these may be in two or more of the sets, so we will have *overcounted*, by counting those elements at least twice. So in the next stage, we subtract the number of elements that are in the intersection of any two of the sets. But then we may have *undercounted*, for each item x that is in all three sets was counted three times in the first stage and then subtracted three times in the second stage, and thus needs to be added in again, which is what we do in the last stage. The rule for three sets is thus:

$$\#(A \cup B \cup C) = \#(A) + \#(B) + \#(C) - \#(A \cap B) - \#(B \cap C) - \#(A \cap C) + \#(A \cap B \cap C).$$

For the general rule, let $A_1, \ldots, A_n$ be any sets with $n \geq 1$. Then, $\#(A_1 \cup \ldots \cup A_n)$ equals:

+ (the sum of the cardinalities of the sets taken individually)
− (the sum of the cardinalities of the sets intersected two at a time)
+ (the sum of the cardinalities of the sets intersected three at a time)
− (the sum of the cardinalities of the sets intersected four at a time)
…………………………
± (the sum of the cardinalities of the n sets intersected n at a time)

The last term is marked ± because the sign depends on whether n is even or odd. If n is odd, then the sign is +; if n is even, then it is −. Of course, the rule can be written in a much more compact mathematical notation, but for our purposes, we can leave that aside. Traditionally, the rule is called the *principle of inclusion and exclusion*, because one first includes too much, then excludes too much and so on. It could also be called the principle of *overcounting and undercounting*.

Application of the rule ties in with the theory of combinations. For example, in the second line, we need to work out the sum of the cardinalities of the sets $A_i \cap A_j$ for all distinct $i,j \leq n$. The value of each $\#(A_i \cap A_j)$ of course depends on the specific example. But we also need to keep tabs on how many such terms we need to consider. Since there are $C(n,2)$ two-element subsets $\{A_i, A_j\}$ of the n-element set $\{A_1, \ldots, A_n\}$, there are $C(n,2)$ such terms to sum in the second line. Likewise, the third line of the rule asks for the sum of $C(n,3)$ terms, and so on.

Exercise 5.3.2 (3)

(a) Draw a Euler diagram representing three intersecting sets, label the relevant areas and paraphrase the rule of inclusion and exclusion in terms of the areas.

(b) Write out the principle of inclusion and exclusion in full for the case $n = 4$. Then calculate $C(4,k)$ for each $k = 1, \ldots, 4$ to check that at each stage you have summed the right number of terms.

5.4 Selections Allowing Repetition

We now consider the two modes of selection that allow repetition of the selected elements. We add these two modes as new rows at the bottom of the table that we had for the non-repetitive selections (Table 5.4).

Table 5.4 Formulae for four modes of selecting k items from an n-element set

Mode of selection	Notation	Standard name	Formula	Proviso	Example: $n=10$, $k=6$
O+R−	$P(n,k)$	Permutations	$n!/(n-k)!$	$k \leq n$	151,200
O−R−	$C(n,k)$	Combinations	$n!/k!(n-k)!$	$k \leq n$	210
O+R+		Permutations with repetition	n^k	none	1,000,000
O−R+		Combinations with repetition	$(n+k-1)!/k!(n-1)!$	$n \geq 1$	5,005

5.4.1 Permutations with Repetition Allowed

We are looking at the number of ways of choosing of k elements from an n-element set, this time allowing repetitions and distinguishing orders. To fix ideas, keep in mind a simple example: How many six-digit telephone numbers can be concocted with the usual ten digits 0–9? The formula in the table tells us that there are $10^6 = 1{,}000{,}000$. This is far more than the figure of 151,200 for plain permutations where repetition is not allowed, and dwarfing the 210 for combinations. What is the reasoning?

As for plain permutations, we apply the multiplication principle. Clearly, there are n ways of choosing the first item. But, as repetition is allowed, there are again n ways of choosing the second, and so on, thus giving us $n\cdot \ldots \cdot n$ (k times), that is, n^k possible selections.

In more abstract language: Each selection can be represented by an ordered k-tuple of elements of the n-element pool, so that the set of all the selections corresponds to the Cartesian product of the latter by itself k times which, as we have seen by the principle of multiplication, has cardinality n^k. As we remarked earlier, this is the same as the number of *functions* from a k-element set such as $\{1,\ldots,k\}$ into an n-element set $\{a_1,\ldots,a_n\}$.

We recall that this operation makes perfectly good sense even when $k > n$. For example, if only two digits may be used, we can construct six-digit telephone numbers out of them, and there will be 2^6 of them.

Exercise 5.4.1 (with partial solution)

(a) What happens in the limiting case that $n = 1$? Does the figure provided by the formula square with intuition?

(b) What about the limiting case that $k = 1$? Does the figure given by the formula make intuitive sense?

(c) The drinks machine has three kinds of coffee, four kinds of soft drink, still water and sparkling water. Every working day I buy a coffee to wake up in the morning, a bottle of water to wash down lunch and a soft drink for energy in the afternoon. Assuming that I start work on a Monday and work 5 days a week, how many weeks before I am obliged to repeat my daily selection?

Solution to (a)

(a) When $n = 1$, then $n^k = 1$ for any k. Intuitively, this is what we want, for there is only one way to write down the same thing k times.

The limiting case when $n = 0$ is a bit odd. On the one hand, when $n = 0$ but $k \geq 1$, we have $n^k = 0^k = 0$. This is fairly natural: in this case, no selection can be made since there is nothing to select from. On the other hand, the ultra-limiting case where $n = 0$ and $k = 0$ comes out differently: here $n^k = 0^0 = 1$, not 0. One could accommodate this with intuition by reflecting that in this case, we are asked to select nothing from nothing, and by doing nothing, we achieve just that, so that there is one selection (the empty selection) that does the job. This contrasts with being asked to select something from nothing (the case $n = 0$, $k \geq 1$), for there is no way of doing that – not even by doing nothing! So intuition, carefully cultivated, may be said to accord with the formula after all. Moreover, the formulae for permutations and combinations without repetition also yield value 1 in this case (and are not defined when $n = 0$ but $k \geq 1$).

That said, it should also be remarked that in mathematics, definitions quite often give rather odd-looking outcomes in their limiting cases. Nevertheless, so long as the definition behaves as desired in the principal cases, for the limiting ones the mathematician is pretty much free to stipulate whatever permits the smoothest formulation and proof of general principles. Reflection on the limiting cases sometimes suggests interesting philosophical perspectives, and usually one can find a way of seeing things that brings educated intuition into line with the mathematical convention adopted in the limiting case.

5.4.2 Combinations with Repetition Allowed

As usual, we begin with an example. A ten-person committee needs volunteers to handle six tasks. We are not interested in the order of the tasks, and dedicated members are allowed to volunteer for more than one task. How many different ways may volunteers come forward?

Since we are not interested in the order in which the tasks are performed, but we do allow multiple volunteering, we need to apply the formula for combinations with possible repetition in Table 5.4. This tells us that it is $(n+k-1)!/k!(n-1)! = (10+6-1)!/6!(10-1)! = 15!/6!9! = (15{\cdot}14{\cdot}13{\cdot}12{\cdot}11{\cdot}10)$ $/(6{\cdot}5{\cdot}4{\cdot}3{\cdot}2) = 7{\cdot}13{\cdot}11{\cdot}5 = 5{,}005$. More than the 210 for plain combinations, but still much less than the 151,200 for plain permutations, not to speak of the 1,000,000 for permutations with repetition allowed.

This mode was the trickiest to interpret in terms of functions in Sect. 5.3, and its counting formula is likewise the trickiest to prove. The basic idea of the argument is to reconceptualize the problem, of counting the selections under the O−R+ mode (combinations with repetition allowed) to one under the O−R− mode (plain combinations, in which repetitions are not allowed), from a larger pool of elements.

Recall that, as mentioned earlier, in the O−R+ mode, we are in effect counting the number of functions f from an n-element set $\{a_1, \ldots, a_n\}$ into $\{0, \ldots, k\}$ such that $f(a_1)+\ldots+f(a_n) = k$. Consider any such function f. To reconceptualize it, we think of ourselves as provided with a sufficiently long sequence of slots, which we fill up to a certain point in the following way. We write 1 in each of the first $f(a_1)$ slots, then a label + in the next slot, then 2 in each of the next $f(a_2)$ slots, then a label + in the next one, and so on up to the $(n-1)$th label +, and finally, write n in each of the next $f(a_n)$ slots, and stop. This makes sense provided $n \geq 1$. Careful: We do *not* put a plus sign at the end: we use $n-1$ plus signs just as there are in the expression $f(a_1)+\cdots+f(a_n)$. The construction will thus need $k+(n-1)$ slots, since we are requiring that $f(a_1)+\cdots+f(a_n) = k$ and we insert $n-1$ plus signs.

Now the slot pattern we have created is in fact fully determined by the total number $k+(n-1)$ of slots and the particular slots where the plus signs go. This is because we can recuperate the whole pattern by writing 1s in the empty slots to the left of the first plus sign, then 2s in the empty slots to the left of the second plus sign, and so on up to $(n-1)$s in the empty slots to the left of the $(n-1)$th plus sign and, not to be forgotten, writing ns in the remaining empty slots to the right of the last plus sign until all $k+(n-1)$ slots are occupied. So in effect, we are looking at the number of ways of choosing $n-1$ slots for plus signs from a set of $k+(n-1)$ slots. The $n-1$ plus sign slots must be distinct from each other (we can't put two plus signs in the same slot), and we are holding the order of the items fixed.

But for counting purposes, holding the order fixed is the same as taking the order to be of no significance, that is, identifying different orderings of the same items. This is because there is a trivial one-one correspondence between the two. So we are in effect making a selection, where order remains without significance but repetition is *not* allowed, of $n-1$ items from a larger pool of $k+(n-1)$ elements. We have reduced the counting problem for the O−R+ mode to one for the O−R− mode, that is, plain combinations, with different values for the variables. Specifically, we have shown that the number of selections when we choose k items from a pool of n elements, using the O−R+ mode of selection, equals $C(k+n-1, n-1)$.

The rest is calculation: $C(k+n-1,\ n-1) = (k+n-1)!/(n-1)!\cdot((k+n-1)-(n-1))! = (k+n-1)!/(n-1)!\cdot\ k! = (n+k-1)!/k!(n-1)!$ – which is the desired counting formula for the O−R+ mode in Table 5.4, so our proof of that formula is complete.

Exercise 5.4.2 (with partial solution)

(a) Use the formula to calculate the number of combinations with repetition of 6 items taken k at a time, for each $k = 1,\ldots,8$.

(b) The restaurant has five flavours of ice cream, and you can ask for one ball, two balls or three balls (unfortunately, not the same price). How many different servings are possible?

Solution to (b)

(b) We are looking at a set with 5 elements (the flavours), and at selections of one, two or three items (the balls), allowing repetition. Does order count? Well, the formulation of the problem is rather vague. Two interpretations suggest themselves.

First interpretation. If the balls are laid out in a row on a suitably shaped dish, we might regard the serving strawberry-chocolate-vanilla as different from chocolate-vanilla-strawberry. In that case, we are in the mode O+R+, and reason as follows. Break the set of selections down into three disjoint subsets – those with one ball, two balls, three balls. Calculate each separately. There are $5^1 = 5$ selections with one ball, $5^2 = 25$ with two balls, $5^3 = 125$ with three balls. So, in total, there are $5 + 25 + 125 = 155$ possible selections.

Second interpretation. If we don't care about the order in which the balls are distributed on the plate, then we are in mode O−R+, and reason as follows. Again, break the class of selections into three disjoint subclasses, and apply the counting formula to each separately. With one ball, there are still 5 selections, since $(5+1-1)!/1!(5-1)! = 5!/4! = 5$. But with two balls, there are now only $(n+k-1)!/k!(n-1)! = (5+2-1)!/2!(5-1)! = 6!/2!4! = 3{\cdot}5 = 15$ selections, and with three balls, just $(5+3-1)!/3!(5-1)! = 7!/3!4! = 7{\cdot}5 = 35$. We thus get a total of $5 + 15 + 35 = 55$ possible selections – only about one-third of the number under the first interpretation.

5.5 Rearrangements and Partitions*

We have covered all the four modes of *selection* of k items from a pool of n elements, but there are also other things that one can do. For example, one can rearrange the letters in a word, and partition a set into cells of specified sizes. Each of these operations gives rise to a counting problem: how many ways of doing it?

5.5.1 Rearrangements

How many ways of arranging the letters in *banana*? This problem sounds very simple, but its analysis is quite subtle. At first sight, it looks like a case of permutations allowing repetition: order definitely matters since *banana* $\neq$ *nabana*, and there are repeats since only three letters are filling six places. But there is also a big difference: *the amount of repetition is predetermined*: there must be exactly three *a*s, two *n*s and one *b*. Operations of this kind are usually called *rearrangements*.

It is important to approach the problem with the right *gestalt*. In the case of *banana*, don't think of yourself as selecting from a three-element set consisting of the letters *a*,*n*,*b*. Instead, focus on a six-element set consisting of the *six occurrences of letters* in the word or, alternatively, *six slots* ready for you to insert those occurrences.

These lead to two ways of analyzing the problem. They give the same numerical result but provide an instructive contrast: one uses permutations while the other uses combinations. We describe them in turn. For each we begin with an analysis of the example, then state the general rule for words with k letters and n letter-occurrences. To distinguish letter-occurrences from letters-as-types, we rewrite the word with subscripts: $b_1a_1n_1a_2n_2a_3$.

5.5.1.1 First Method

Analysis of the example: We know that there are $P(6,6) = 6! = 720$ permutations of the set $A = \{b_1,a_1,n_1,a_2,n_2,a_3\}$. Clearly, for each such permutation, there are $3! = 6$ that differ from it at most in the distribution of subscripts to the letter a. So dropping those subscripts means that we need to divide 720 by 6. But there are also $2! = 2$ differing from it at most in the distribution of subscripts to the letter n, so we also need to divide by 2. The letter b occurs just once, so as far as it is concerned, there is no need to divide. Thus, the number of rearrangements of the letters (including, of course, the original arrangement *banana*) is $720/6{\cdot}2 = 60$.

General rule: Suppose we are given a word with n letter-occurrences, made up of k letters, with those letters occurring $m_1, \ldots, m_k$ times. Then there are $n!/m_1! \ldots\ m_k!$ rearrangements of the letters in the word.

5.5.1.2 Second Method

For the second approach, we think in terms of six *slots* $s_1, \ldots, s_6$ ready to receive letter-occurrences.

Analysis of the example: The rearrangements of the letters can be thought of as functions f that take elements of the three-element set $\{a,n,b\}$ to *subsets* of $\{s_1, \ldots, s_6\}$ in such a way that $f(a)$ has three elements, $f(n)$ has two, $f(b)$ is a singleton and the family $\{f(a), f(n), f(b)\}$ partitions $\{s_1, \ldots, s_6\}$. How can we count these functions?

We know that there are $C(6,3)$ ways of choosing a three-element subset for $f(a)$, leaving 3 slots unfilled. We have $C(3,2)$ ways of choosing a two-element subset of those three slots for $f(n)$, leaving 1 slot to be filled with the letter b. That gives us $C(6,3){\cdot}C(3,2){\cdot}C(1,1)$ selections. Calculating using the formula already known for combinations:

$$\begin{aligned} C\,(6,3)\cdot C\,(3,2)\cdot C\,(1,1) &= [6!/3!(6-3)!]\cdot[3!/2!(3-2)!]\cdot[1!/1!(1-1)!] \\ &= [6!/3!3!]\cdot[3!/2!] \\ &= 20\cdot 3 = 60. \end{aligned}$$

This is the same figure as reached by the first method.

General rule: We have a set $\{a_1, \ldots, a_k\}$ of k letters to be written into a set $\{s_1, \ldots, s_n\}$ $(k \leq n)$ of slots, with each letter a_i being written in m_i slots. So we are looking at functions f that take elements of the set $\{a_1, \ldots, a_k\}$ to *subsets* of $\{s_1, \ldots, s_n\}$ in such a way that each $f(a_i)$ has m_i elements and the family $\{f(a_i) : i \leq k\}$ partitions $\{s_1, \ldots, s_n\}$. How can we count these functions?

We know that there are $C(n,m_1)$ ways of choosing an m_1-element subset for $f(a_1)$, leaving $n-m_1$ slots unfilled. We have $C(n-m_1,\ m_2)$ ways of choosing an m_2-element subset of those slots for $f(a_1)$, leaving $n-m_1-m_2$ slots to be filled, and so on until all the n slots are used up. Clearly this can be done in the following number of ways:

$$C(n, m_1)\cdot C(n-m_1, m_2)\cdot \ldots \cdot C(n-m_1-m_2-\ldots-m_{k-1}, m_k).$$

This formula looks pretty ghastly. But if we write it out using the formula for combinations and then do successive cancellations, it simplifies beautifully coming down to $n!/m_1! \ldots m_k!$ – which is the same formula as obtained by the other method.

Exercise 5.5.1

(a) Using the counting formula for combinations, write out the formula $C(n, m_1)\cdot C(n-m_1, m_2)\cdot \ldots \cdot C(n-m_1-m_2-\ldots-m_{k-1}, m_k)$ in full for the case that $n = 9$, $k = 4$. Perform the cancellations to check that it agrees with $9!/m_1! \ldots m_4!$.

(b) Apply the counting formula $n!/m_1! \ldots m_k!$ to determine how many ways of rearranging the letters in the word *Turramurra*.

5.5.2 Counting Configured Partitions

The second method for counting rearrangements can be expressed in an abstract way, without mentioning words and their component letters. Suppose we want to count the number of functions f that take elements of a k-element set $\{a_1, \ldots, a_k\}$ to subsets of an n-element set $\{s_1, \ldots, s_n\}$ $(k \leq n)$ such that each set $f(a_i)$ has m_i elements and the family $\{f(a_i) : i \leq k)\}$ partitions $\{s_1, \ldots, s_n\}$. Then by the same reasoning as for the second method, we may use the same formula:

$$C(n, m_1)\cdot C(n-m_1, m_2)\cdot \ldots \cdot C(n-m_1-m_2-\ldots-m_{k-1}, m_k) = n!/m_1! \ldots m_k!$$

Let's see how this plays out in an example. There are 9 students in a class. How many ways of dividing them into tutorial groups of 2, 3 and 4 students for tutorials on Monday, Wednesday and Friday? We are looking at a set $A = \{a_1, \ldots, a_9\}$ with 9 elements, and we want to count ways of partitioning it into three cells of specified sizes. But before applying a counting formula, we need to be more specific about what we are trying to count. Do we regard the assignment to days of the week as relevant? For example, suppose we put students a_1, a_2 in one tutorial group, a_3, a_4, a_5 in another, a_6, a_7, a_8, a_9 in the third; does it make a difference for our counting whether a_1, a_2 are taught on Monday rather than Wednesday?

Suppose that we wish to regard these as different. We know from the abstract version of the second method for counting rearrangements, given above, what formula to use for that. It is $n!/m_1! \ldots m_k!$ giving us $9!/2!3!4! = 1{,}260$ possible ways of assigning the students.

However, we may wish to ignore which day of the week a given tutorial group is to be taught. Perhaps we are merely constituting the three groups, leaving their scheduling for later decision. Then we should divide the above counting formula by 3! giving us $1{,}260/6 = 210$ possible ways of forming three tutorial groups of the

Table 5.5 Formulae for counting rearrangements and partitions

Description	Brief name	Counting formula
Number of rearrangements of k letters in a word of n letter occurrences	Rearrangements	$n!/m_1! \ldots m_k!$
Number of partitions with a given numerical configuration (k cells and cell sizes $m_1, \ldots, m_k$)	Configured partitions	$n!/m_1! \ldots m_k!\, k!$

specified sizes. With this way of interpreting the problem, we are in effect *counting the number of partitions* of the pool set ($n = 9$) into a specified number of cells ($k = 3$) of specified sizes (2,3,4), and using the formula $n!/m_1! \ldots m_k!k!$.

The answer we get depends thus on the method we use, which in turn depends on the exact interpretation of the problem. Sometimes there are only hints in its formulation to indicate which interpretation is meant. On other occasions, the formulation may simply be ambiguous, in which case we say that the problem is *underdetermined*. Some texts take a perverse pleasure in keeping the hints to an almost imperceptible minimum in their problems. Not here.

We summarize the results of the section in Table 5.5, which thus supplements Table 5.4 for the four O±R± modes.

Alice Box: Different but 'Identical'

Alice: I have been reading some other textbooks on this material, and there is something that leaves me quite confused.

Hatter: Go ahead.

Alice: I find problems that begin like this: 'Let A be a set of six elements, of which three are identical . . . '. But that doesn't make any sense!

Hatter: Why not?

Alice: You can't have different items that are nevertheless identical! If $A = \{a,b,c,d,e,f\}$ and a,b,c are identical to each other, then A has at most four elements, not six. These books seem to be speaking a different language!

Hatter: Well, at least a different dialect. It is rather confusing, indeed.

We should say a bit more about this than the Hatter does. The subject that this book calls 'sets, logic and finite mathematics' is a fairly recent assemblage. It is made up of different topics developed at quite distinct times in history. Each part evolved using its specific conceptual structures and terminologies, and these do not fit well with each other.

Textbooks like ours try to put it all into one coherent story, based on the very general notion of a set. But several of the topics are much older than the theory of sets. For example, the theory of counting was already highly active in the seventeenth century, and indeed goes back many centuries before – consider

Fibonacci in 1202! In contrast, set theory as we know it today appeared only in the late nineteenth century. As one would expect, the conceptual frameworks are quite different.

Consequently, textbooks like this one are torn between the demands of coherence of the overall structure, on the one side, and faithfulness to the deeply rooted terminologies of specific topics on the other. Usually something is given up on both sides. In this book, some concessions are made to traditional topic-specific terminologies, but coherence of the broad picture tends to take priority.

So how can you or Alice make sense of problems that speak of selections from a set with several identical elements? Above all, try to interpret what could be *meant* by the particular problem, taking into account its content as well as its structure and paying attention to little hints. Sometimes when the problem speaks of identical or indistinguishable elements, it may mean that we have no way of telling the difference between different orders of the elements, so that interest lies in selections where order is immaterial. On other occasions, it may be a way of referring obliquely to a partition (in other words, an equivalence relation) between elements. Then we may have a problem of counting partitions. There is no formal algorithm for ascertaining what is meant: it is an exercise in hermeneutics. Try to find the most natural interpretation of the problem, remembering there is not always a unique reading, as more than one may be plausible. The following exercise illustrates the task.

Exercise 5.5.2 (with hints for solution)

(a) How many ways are there to place 12 indistinguishable balls into 10 slots?

(b) In a cellar, there are 12 bottles of wine in a row, of which 5 are the same, another 4 the same and another 3 the same. They are taken down and put into a box. Then they are put back on the shelf. How many different results are possible?

Hints for solution

(a) The balls are described as 'indistinguishable'. Presumably we are not thinking of the trivial partition with just one cell. We are envisaging the selection, with repetition permitted but order ignored, of 12 slots from the set of 10, that is, 12-combinations with repetition from a 10-element set. Go on from there.

(b) From the phrasing, it appears that the problem setter is imagining a rack with 12 places on it, and functions on the three kinds of wine taking each kind to a subset, with a specified size, of the places. Go on from there.

End-of-Chapter Exercises

Exercise 5.1: Addition and multiplication

(a) A computer represents elements of **Z** using binary digits 0 and 1. The first digit represents the sign (negative or positive), and the remainder represents the magnitude. How many distinct integers can we represent with *n* binary digits?

Warning: Be careful with zero.

(b) I want to take two books with me on a trip. I have two logic books, three mathematics books and two novels. I want the two books that I take to be of different types. How many possible sets of two books can I take with me?
(c) In the USA, radio station identifiers consist of either 3 or 4 letters of the alphabet, with the first letter a K or a W. Determine the number of possible identifiers.
(d) You are in Paris, and you want to travel Paris-Biarritz-London-Paris or Paris-London-Biarritz-Paris. There are 5 flights each way per day between Paris and Biarritz, 13 each way between Paris and London, but only 3 per day from London to Biarritz and just 2 in the reverse direction. How many flight sequences are possible?

Exercise 5.2: The principle of inclusion and exclusion Amal telephones 5 different people, Bertrand telephones 10, Clarice telephones 7. But 3 of the people called by Amal are also called by Bertrand, 2 of those called by Bertrand are rung by Clarice, and 2 of those contacted by Clarice are called by Amal. One person was called by all three. How many people were called?

Exercise 5.3: Four basic modes of selection
(a) In the first round of a quiz show, a contestant has to answer correctly seven questions, each with a yes/no answer. Another contestant has to answer correctly two questions, each of which is multiple-choice with seven possible answers. Assuming that both contestants are totally ignorant and guess blindly, who faces the more daunting task?
(b) Australia currently has six states (not counting its various territories, notably the Northern Territory). A manager proposes to test a product in four of those states. In how many ways may the test states be selected?
(c) Another manager, more cautious, proposes to test the product in one state at a time. In how many ways may a test-run be selected?
(d) Even more cautious, the director decides to test the product in one state at a time, with the rule that the test-run is terminated if a negative result is obtained at any stage. How many test-runs are possible?
(e) A game-board is a 4 by 4 grid of squares, and you have to move your piece from the bottom left square to the top right one. The only moves allowed are 'one square to the right' and 'one square up'. How many possible routes?

Warning: This is more subtle than you may at first think. The answer is not 2^8.

(f) You have five dice of different colours, and throw one after another, keeping track of which is which. What is the natural concept of an outcome here? How many such outcomes are there?

(g) You throw five dice together, and are not able to distinguish one die from another. What is the natural concept of an outcome here, and how many are there?

Exercise 5.4: Rearrangements and configured partitions*

(a) The candidate wins a free holiday in the place whose name admits the greatest number of arrangements of its letters: *Mississippi*, *Ouagadougou* and *Woolloomooloo*. Make the calculations and give the answer to win the trip.

(b) How many arrangements of the letters in *Woolloomooloo* make all occurrences of any given letter contiguous?

(c) How many distributions of 10 problems are possible among Albert, Betty, Carlos and Deborah if Albert is to do 2, Betty 3 and Deborah 5.

(d) How many ways of dividing 10 problems up into three groups of 2, 3 and 5 problems?

(e) Which is larger: the number of partitions of a set with 12 elements into three cells each of four elements, or the number of partitions of the same set into four cells each of three elements?

Selected Reading

Almost every text on discrete mathematics has a chapter on this material somewhere in the middle and, although terminology, notation and order of presentation differ from book to book the material covered is usually much the same. One clear beginner's exposition is:

Haggarty R (2002) Discrete mathematics for computing. Pearson, Harlow, chapter 6

More detailed presentations include:

Johnsonbaugh R (2009) Discrete mathematics, 7th edn. Pearson, Upper Saddle River, chapter 6

Lipschutz S (1997) Discrete mathematics, Schaum's outline series. McGraw Hill, New York, chapter 6

For more advanced texts dedicated to the subject, see, for example:

Andreescu T et al (2004) Counting strategies: a path to combinatorics for undergraduates. Springer/Birkhauser, New York/Boston

Herman J et al (1997) Counting and configurations: problems in combinatorics, arithmetic and geometry, CMS books in mathematics. Springer, New York

As mentioned by the Hatter, a good place to begin reading about multisets is the relevant article in *Wikipedia*.

Weighing the Odds: Probability 6

Abstract

In this chapter, we introduce the elements of probability theory. In the spirit of the book, we confine ourselves to the discrete case, that is, probabilities on finite domains, leaving aside the infinite case.

We begin by defining *probability functions* on a finite sample space and identifying some of their basic properties. So much is simple mathematics. This is followed by some words on different *philosophies* of probability, and warnings of *traps* that arise in applications. Then back to the mathematical work, introducing the concept of *conditional probability* and setting out its properties, its connections with *independence* and its use in *Bayes' theorem*. An interlude presents the curious configuration known as Simpson's paradox. In the final section, we explain the notions of a *payoff function* and *expected value*.

6.1 Finite Probability Spaces

In the chapter on rules for counting, we remarked that the area is rather older than the modern theory of sets, and tends to keep some of its traditional ways of speaking even when its ideas can be expressed in a set-theoretic manner. The same is true of probability theory, creating problems for both reader and writer of a textbook like this. If we simply follow the rather loose traditional language, the reader may fail to see how it rests on fundamental notions. On the other hand, if we translate everything into set-theoretic terms the reader is ill-prepared for going on to other expositions couched in the traditional terminology.

For this reason, we follow a compromise approach. We make use of traditional probabilistic terminology, but at each step show how it is serving as shorthand for a uniform one in terms of sets and functions. A table will be used to keep a running tally of the correspondences.

D. Makinson, *Sets, Logic and Maths for Computing*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-2500-6_6,

It turns out that applications of probability theory to practical problems often need to deploy the basic counting principles that were developed in the preceding chapter, and so the reader should be ready to flip back and forth to refresh the mind whenever needed.

6.1.1 Basic Definitions

The first concept that we need in discrete probability theory is that of a *sample space*. Mathematically, this is just an arbitrary finite (but non-empty) set S. The term merely indicates that we intend to use it in a probabilistic framework.

A *probability distribution* (or just *distribution* for short) is an arbitrary function p: $S \rightarrow [0,1]$ such that $\Sigma\{p(s)\colon s \in S\} = 1$, that is, the sum of the values $p(s)$ for $s \in S$ equals 1. Recall that [0,1], often called *the real interval*, is the set of all real numbers from 0 to 1 included, that is, $[0,1] = \{x \in \mathbf{R}\colon 0 \leq x \leq 1\}$. Actually, in the context of discrete probability, that is, where S is a finite set, we could without loss of generality reduce the target of the probability function to the set of all rational numbers from 0 to 1. But we may as well allow the whole of the real interval, which gives us no extra trouble and is needed for the infinite case.

Exercise 6.1.1 (1) (with solution) Let $S = \{a,b,c,d,e\}$. Which of the following functions p are probability distributions on S? Give a reason in each case.
(a) $p(a) = 0.1, p(b) = 0.2, p(c) = 0.3, p(d) = 0.4, p(e) = 0.5$.
(b) $p(a) = 0.1, p(b) = 0.2, p(c) = 0.3, p(d) = 0.4, p(e) = 0$.
(c) $p(a) = 0, p(b) = 0, p(c) = 0, p(d) = 0, p(e) = 1$.
(d) $p(a) = -1, p(b) = 0, p(c) = 0, p(d) = 1, p(e) = 1$.
(e) $p(a) = p(b) = p(c) = p(d) = p(e) = 0.2$.

Solution
(a) No, the values do not add to one.
(b) Yes, values are in the real interval and add to one.
(c) Yes, same reason.
(d) No, not all values are in the real interval.
(e) Yes, values are in the real interval and add to one.

The last of the functions in the exercise is clearly a very special one: it is a constant function on S, with all the elements of the sample space receiving the same value. This is known as an *equiprobable* (or *uniform*) *distribution*, and is particularly easy to work with. Given a sample space S with n elements, there is evidently just one equiprobable distribution p: $S \rightarrow [0,1]$, and it puts $p(s) = 1/n$ for all $s \in S$. But it should be understood that this is a special case, not only mathematically but also in practice. Many problems involve unequal distributions, and we cannot confine ourselves to equiprobable ones.

Given a distribution p: $S \rightarrow [0,1]$, we can extend it to a function p^+ on the power set $\mathcal{P}(S)$ of S by putting $p^+(A) = \Sigma\{p(s)\colon s \in A\}$ when A is any non-empty subset of S, and $p^{+}(A) = 0$ in the limiting case that $A = \varnothing$. This is the *probability function* determined by the distribution. Even for a very small set S, there will be infinitely many distributions on S, and each of them determines a unique probability function p^{+} on $\mathscr{P}(S)$ or, as one also says, *over* S. The pair made up of the sample space S and the probability function p^{+}: $\mathscr{P}(S) \rightarrow [0,1]$ is often called a *probability space*. Traditionally, the elements of $\mathscr{P}(S)$, that is, the subsets of S, are called *events*.

Exercise 6.1.1 (2)

(a) Show that every probability function is indeed into [0,1].

(b) Show that whenever $\#(S) \geq 2$ then there are infinitely many distributions on S. *Hint*: No need for an induction; show it in one fell swoop. If you followed the Alice Box in Sect. 3.4.2 of Chap. 3, you can even show that there are uncountably many of them.

(c) Reformulate the above definition of p^{+} in an explicitly recursive manner, identifying its base and recursion step and why it is well defined (cf. Sect. 4.6.3).

(d) Consider the probability function p^{+}: $\mathscr{P}(S) \rightarrow [0,1]$ determined by the equiprobable distribution p: $S \rightarrow [0,1]$. Explain why $p^{+}(A)$ measures the proportion of elements of S that are in A, for every $A \subseteq S$.

(e) Explain why there are no equiprobable distributions over infinite sample spaces. *Hint*: Look up *Archimedes' principle* in arithmetic and apply it. There is a full explanation in Sect. 6.2.3.

Now that we are clear about the difference between p and p^{+}, we will often 'abuse notation' and keep things streamlined by dropping the superscript from p^{+}, calling it just p, in contexts where one can easily see which is meant.

6.1.2 Properties of Probability Functions

It is surprising how many useful properties of probability functions may be extracted from the definition. Some of the most important are covered in the following exercise. Try to do as much as possible before looking at the solution!

Exercise 6.1.2 (1) (with solution) Let p^{+}: $\mathscr{P}(S) \rightarrow [0,1]$, or more briefly p without the superscript, be a probability function over a finite sample space S. Show that p has the following properties (where A,B are arbitrary subsets of S):

(a) $p(\varnothing) = 0$.

(b) $p(S) = 1$.

(c) $p(A \cup B) = p(A) + p(B) - p(A \cap B)$.

(d) When A and B are disjoint then $p(A \cup B) = p(A) + p(B)$.

(e) $p(S \setminus A) = 1 - p(A)$.

(f) $p(A \cap B) = p(A) + p(B) - p(A \cup B)$.

(g) When $A \cup B = S$, then $p(A \cap B) = p(A) + p(B) - 1 = 1 - [p(S \setminus A) + p(S \setminus B)]$.

Solution

(a) Given explicitly in the definition of a probability function.

(b) $p(S) = \Sigma\{p(s)\colon s \in S\} = 1$ by the definition of a distribution function (for the second equality) and the definition of a probability function (for the first one).

(c) $p(A \cup B) = \Sigma\{p(s)\colon s \in A \cup B\} = \Sigma\{p(s)\colon s \in A\} + \Sigma\{p(s)\colon s \in B\} - \Sigma\{p(s)\colon s \in A \cap B\} = p(A) + p(B) - p(A \cap B)$.

(d) By (c) noting that when $A \cap B = \varnothing$, then $p(A \cap B) = p(\varnothing) = 0$.

(e) $S = A \cup (S \backslash A)$ and the sets A, $S \backslash A$ are disjoint, so by (d) we have $p(S) = p(A) + p(S \backslash A)$ and thus by arithmetic $p(S \backslash A) = p(S) - p(A) = 1 - p(A)$ by (b).

(f) This can be shown from first principles, but it is easier to get it by arithmetic manipulation of (c).

(g) The first equality is immediate from (f) noting that when $A \cup B = S$, then $p(A \cup B) = p(S) = 1$ by (b). For the second equality, $p(A) + p(B) - 1 = [1 - p(S \backslash A)] + [1 - p(S \backslash B)] - 1$ by (e), which simplifies to $1 - [p(S \backslash A) + p(S \backslash B)]$.

In the above exercise, we used the notation $S \backslash A$ for the complement of A with respect to the sample space S, in order to avoid any possible confusion with the arithmetic operation of subtraction. Although they are closely related, they are of course different. From now on, however, we will take the liberty of writing more briefly $–A$.

Here's a hint for simplifying your scratch-work when doing exercises like the preceding one, or the one to follow. It can be quite tiresome to write out the function sign p over and over again when calculating or proving in probability theory. For problems using only one probability function, we can use a more succinct notation. For each subset A of the sample space, we can write $p(A)$ as $\underline{A}$ with the underlining representing the probability function p. For example, we may write the equation $p(A \cup B) = p(A) + p(B) - p(A \cap B)$ as $\underline{A \cup B} = \underline{A} + \underline{B} - \underline{A \cap B}$. *But be careful*! When the problem involves more than one probability function (as for example in the iterated conditionalizations defined later in this chapter), this shorthand notation cannot be used, since it cannot keep track of the different functions. To be sure, if there are, say, two probability functions in play, one could in principle use two different kinds of underlining, but such devices quickly become cumbersome. For this reason, your instructor may not like you to employ underlining at all, so check on class policy before using it outside personal rough work. In what follows, we will stick to standard presentation.

Exercise 6.1.2 (1)

(a) Use results of the preceding exercise to show that if $A \subseteq B$ then $p(A) \le p(B)$.

(b) Use this to show that $p(A \cap B) \le p(A)$ and $p(A \cap B) \le p(B)$.

(c) Let $S = \{a,b,c,d\}$ and let p be the equiprobability distribution on S. Give examples of events, that is, sets $A,B \subseteq S$, such that, respectively, $p(A \cap B) < p(A) \cdot p(B)$, $p(A \cap B) = p(A) \cdot p(B)$ and $p(A \cap B) > p(A) \cdot p(B)$.

Table 6.1 Two languages for probability

	Traditional probabilistic	Modern set-theoretic
1	Sample space S	Arbitrary set S
2	Sample point	Element of S
3	Probability distribution	Function p: $S \rightarrow [0,1]$ whose values sum to 1
4	Event	Subset of S
5	Elementary event	Singleton subset of S
6	Null event	Empty set
7	Mutually exclusive events	Disjoint subsets of s
8	Experiment	Situation to be modelled probabilistically
9	$P(A\|B)$	$P(A,B)$
10	Random variable X	(Payoff) function f: $S \rightarrow \mathbf{R}$
11	Range space $\mathbf{R}_X$	Range $f(S)$ of the function f: $S \rightarrow \mathbf{R}$

Table 6.1 keeps track of translations between traditional probabilistic and modern set-theoretic terminology and notation. Rows 1 through 8 are about matters arising in this and the following two sections; row 9 concerns the notation for conditional probability to be discussed in Sect. 6.4; while rows 10 and 11 will come up in Sect. 6.8 on expectation.

6.2 Philosophy and Applications

Probability theory differs from any of the topics that we have studied so far in two respects, one philosophical and the other practical.

6.2.1 Philosophical Interpretations

Philosophically, the intuitive notion of probability is much more slippery than that of a set. It is even more perplexing than the notion of truth, which we used in logic boxes to define connectives such as negation, conjunction, disjunction and material implication. When we declare that an event is, say, highly improbable we are not committing ourselves to saying that it does not take place. Indeed, we may have a collection of events, each of which known to be highly improbable while we also know that at least one *must* take place. Consider for example a properly conducted lottery with many tickets. For each individual ticket, it is highly improbable that it will win; but it is nevertheless certain that one will do so.

So what is meant by saying that an event is probable, or improbable? There are two main perspectives on the question.

- One of them sees probability as a measure of uncertainty, understood as a state of mind. This is known as the *subjectivist* conception of probability. The general idea is that a probability function p is a measure of the degree of belief that a fully rational (but not omniscient) agent would have concerning various possible events, given some limited background knowledge that is not in general enough to permit unqualified judgement.

- The other sees probability as a manifestation of as something in the world itself, independent of the thinking of any agent. This is known as the *objectivist* conception of probability. It comes in several flavours. Often on this approach, the probability of an event is taken to be a measure of the long-run tendency for events similar to it to take place in contexts similar to the one under consideration. That particular version of the objectivist approach is called the *frequentist* conception.

Of course, these thumbnail sketches open more questions than they begin to answer. For example, under the subjectivist approach, what counts is not your degree of belief or mine, nor even that of some reputed expert or guru, but rather the degree that a fully *rational agent* would have. So how are we to understand rationality in this context? Surely, in the final analysis, rationality is not such a subjective matter. Properly unpacked, the approach may thus end up being not so subjective after all! On the other hand, the frequentist view leaves us with the difficult problem of explaining what is meant by ‘contexts similar to the one under consideration’. It also leaves us with questions such as why the long-run frequency of something should give us any confidence at all in the short term, and how long the long run is supposed to be. As the saying goes, in the long run we are all dead. It is difficult to justify confidence in real-life decisions, which usually need to be taken within a very limited time, in terms of the frequentist approach without making some equiprobability assumptions.

It is often felt by practitioners that some kinds of situation are more easily conceptualized in a subjectivist manner, while others are better seen in frequentist terms. The subjectivist approach seems to fit better for events that are rare or unique, or about which little data on past occurrences of similar events are available. For example, *prima facie* it may be rather difficult for a frequentist to deal with the question of the probability of the earth being rendered uninhabitable by a nuclear war in the coming century. On the other hand, there are other kinds of question on which lots of data are available, on which frequencies may be calculated, and for which the frequentist conception of probability seems appropriate. That is how insurance companies keep afloat, and part of the process by which new medicines are cleared for safety.

As this is not a text of the philosophy of science, we will not take such issues further. We simply note that they are difficult to formulate meaningfully, tricky to handle and have not obtained any consensual resolution despite the centuries with which probability theory has been studied mathematically and applied to everyday affairs. The amazing thing is that we are able to carry on with both the mathematics of probability theory and its application to the real world without resolving the philosophical issues about its interpretation.

6.2.2 The Art of Applying Probability Theory

The application of probability theory is an art as much as a science, and is more subtle than the mathematics itself. The first point to realize about it is that, unfortunately, you cannot get probabilities out of nothing. You have to make assumptions about the probabilities of some events in order to deduce probabilities of others. Such assumptions should always be made explicit, so that they can be recognized for what they are and then assessed. Whereas in textbook examples, the assumptions are often simply given as part of the problem statement, in real life they can be difficult to determine or justify.

In practice, this means that when confronted with a problem of probabilities, two steps should be made before attempting any calculation:

- *Identify the sample space of the problem*. What is the set S that we take as domain of the probability distribution in the question?
- *Specify the values of the distribution*. Can it reasonably be assumed to be an equiprobable distribution? If so, and we also know the size n of the sample space, then we already have value $p(s) = 1/n$ for each point s in the sample space, and we are ready to calculate the values $p^{+}(A)$ for subsets $A \subseteq S$. If it cannot reasonably be taken to be equiprobable, we must ask: Is there any other distribution that is reasonable to assume?

Many students are prone to premature calculation, and training is needed to overcome it. In this chapter, we will try always to be explicit about the sample space and the values of the distribution.

6.2.3 Digression: A Glimpse of the Infinite Case*

In this text, we study only *finite* probability spaces, so this subsection is only for the curious; others may skip it. What differences arise in the infinite case?

Those who have been doing the exercises for this chapter have already seen that when the sample space S is infinite, it cannot have an equiprobable distribution in the sense defined. The reason is that if the distribution is equiprobable, then either all points in S get the value zero, or else they all get the same value greater than zero. In the former case, the values add up to *less* than 1, for under the usual understanding of infinite sums, the sum of even infinitely many zeros is still zero. And in the latter case, the values sum to *more* than 1. Indeed, by the *principle of Archimedes*, for every real $r > 0$ (no matter how small) there is an integer n (whose size depends on the choice of r) with $n \cdot r > 1$. Thus, no matter how small we choose $r > 0$, assigning value r to all the elements of an infinite sample space will make even sufficiently large (finite) subspaces of it sum to more than 1.

The standard response to this predicament is to abandon the idea that in the infinite case, probability functions may always be generated by summing from distributions on the sample space S. Instead, they are treated directly, in a more abstract manner. They become functions p that take some (but not necessarily all) subsets A of the sample space S as arguments, to give values $p(A)$ between 0 and 1.

The domain of the functions is required to be a *field of subsets* of S, that is, a non-empty collection $\mathcal{C} \subseteq \mathcal{P}(S)$ that is closed under complement (with respect to S) and binary union (and so also binary intersection). The functions themselves are required to satisfy certain postulates, notably that $p(S) = 1$, and that when A,B are disjoint subsets of S then $p(A \cup B) = p(A) + p(B)$. But it is no longer required that for infinite A, $p(A)$ is in any sense the sum of the infinitely many values $p(\{a\})$ of the singleton subsets of A, nor indeed that such singleton subsets need be in the domain of the function p when A is.

As a result, it is not possible to generalize addition principles for finite unions to principles that cover arbitrary infinite unions of subsets of the sample space. However, a restricted form of the generalization, known as *σ-additivity*, may consistently be accepted, and often is. It requires that the field of subsets serving as domain of the probability function is closed under *countable unions* and, if the sample set is more than countable, that p satisfies a principle of addition for countable unions of pairwise disjoint sets. But this is going way beyond discrete mathematics.

Another approach to the definition of probability in the infinite case is to widen the set of the reals to contain also 'infinitesimals' as defined and exploited in what is called *non-standard analysis*. But that would take us even further afield.

6.3 Some Simple Problems

In this section, we run through a series of problems, to give practice in the task of applying the first principles of probability theory. The examples become progressively more sophisticated as they bring in additional devices.

Example 1 Throw a fair die. What is the probability that the number (of dots on the uppermost surface) is divisible by 3?

Solution First, we specify the sample space. In this case, it is the set $S = \{1,2,3,4,5,6\}$. Next, we specify the distribution. In this context, the term *fair* means that the questioner is assuming that we have an equiprobable distribution. The distribution thus puts $p(n) = 1/6$ for all positive $n \leq 6$. The event we have to consider is $A = \{3,6\}$ and so $p(A) = 1/6 + 1/6 = 1/3$.

Example 2 Select a playing card at random from a standard European pack. What is the probability that it is a hearts? That it is a court card (jack, queen, king)? That it is both? That it is hearts but not a court card? That it is either?

Solution First we specify the sample space. In this case, it is the set S of 52 cards in the standard European pack. Then we specify the distribution. In this context, the term *random* again means that we are asked to consider an equiprobable distribution. The distribution thus puts $p(c) = 1/52$ for each card c. We have five events to consider. The first two we call H, C, and the others are $H \cap C$, $H \setminus C$ and $H \cup C$. In a standard pack, there are 13 hearts and 12 court cards, so $p(H) = 13/52 = 1/4$

and $p(C) = 12/52 = 3/13$. There are only 3 cards in $H \cap C$, so $p(H \cap C) = 3/52$, so there are $13 - 3 = 10$ in $H \backslash C$ giving us $p(H \backslash C) = 10/52 = 5/26$. Finally, $\#(H \cup C) = \#(H) + \#(C) - \#(H \cap C) = 13 + 12 - 3 = 22$ so $p(H \cup C) = 22/52 = 11/26$.

Example 3 An unscrupulous gambler has a loaded die with probability distribution $p(1) = 0.1, p(2) = 0.2, p(3) = 0.1, p(4) = 0.2, p(5) = 0.1, p(6) = 0.3$. Which is more probable, that the die falls with an even number, or that it falls with a number greater than 3?

Solution As in the first problem, the sample space is $S = \{1,2,3,4,5,6\}$. But this time the distribution is not equiprobable. Writing E and G for the two events in question, we have $E = \{2,4,6\}$ while $G = \{4,5,6\}$, so $p(E) = p(2) + p(4) + p(6) = 0.2 + 0.2 + 0.3 = 0.7$, while $p(G) = p(4) + p(5) + p(6) = 0.2 + 0.1 + 0.3 = 0.6$. Thus, it is slightly more probable (difference 1/10) that this die falls even than that it falls greater than 3.

Example 4 We toss a fair coin three times. What is the probability that (a) at least two consecutive heads appear, (b) that exactly two heads (not necessarily consecutive) appear?

Solution The essential first step is to get clear about the sample space. It is *not* the pair {heads, tails} or {H,T} for short. It is the set $\{\mathrm{H,T}\}^3$ of all ordered triples with elements from {H,T}. Enumerating it in full, $S = \{$HHH, HHT, HTH, HTT, THH, THT, TTH, TTT$\}$. As the coin is supposed to be fair, these are considered equiprobable. Let A be the event of getting at least two consecutive heads, and B that of getting exactly two heads (in any order). Then $A = \{$HHH, HHT, THH$\}$, while $B = \{$HHT, HTH, THH$\}$. Thus, $p(A) = 3/8 = p(B)$.

Alice Box: Repeated Tosses

Alice: Not so fast! Something is funny here.

Hatter: What's wrong?

Alice: You seem to be making a hidden assumption. We are told that the coin is fair, so $p(H) = 0.5 = p(T)$. But how do you know that, say, $p(\mathrm{HHH}) = p(\mathrm{HTH}) = 1/8$? Perhaps the fact of landing heads on the first toss makes it less likely to land heads on the second toss. In that case the distribution over this sample space will not be equiprobable.

Hatter: Nice point! In fact, we *are* implicitly relying on such an assumption. We are assuming that $p(H)$ *always* has the value 0.5 no matter what happened on earlier tosses. Another way of putting it: the successive tosses have no influence on each other – H_{i+1} is independent of H_i, where H_i is the event 'heads on the ith toss'. We will be saying more about independence later in the chapter.

Example 5 Seven candidates are to be interviewed for a job. Of these, four are men and three are women. The interviews are conducted in random order. What is the probability that all the women are interviewed before any of the men?

Solution Let M be the four male candidates, and W be the three female ones. Then $M \cup W$ is the set of all candidates. We take the sample space to be the set of all permutations (i.e. selections in mode $\mathrm{O}+\mathrm{R}-$) of this seven-element set. As we know from the preceding chapter, there are $P(7,7) = 7!$ such permutations, and we assume that they are equiprobable as suggested by the word 'random' in the formulation of the problem. An interview in which all the women come before the men will be one in which we have some permutation of the women, followed by some permutation of the men, and there are clearly $P(3,3) \cdot P(4,4)$ of these. So the probability of such an interview pattern is $P(3,3) \cdot P(4,4)/P(7,7) = 3!4!/7! = 1/35$.

This is the first example in our discussion of probability that makes use of a rule from the chapter on counting. The sample space consisted of all permutations of a certain set and, since the space is assumed equiprobable, we are calculating the proportion of them with a certain property. If, on the other hand, a problem requires a sample space consisting of combinations, we need to apply the appropriate formula for counting them, as in the following example.

Example 6 Of the ten envelopes in my letterbox, two contain bills. If I pick three envelopes at random to open today, leaving the others for tomorrow, what is the probability that none of these three is a bill.

Solution We are interested in the proportion of 3-element subsets of S that contain no bills. So we take the sample space S to consist of the set of all 3-element subsets of a 10-element set. Thus S has $C(10,3) = 10!/3!7! = 120$ elements. How many of these subsets have no bills? There are 8 envelopes without bills, and the number of 3-element subsets of this 8-element set is $C(8,3) = 8!/3!5! = 56$. So the probability that none of our three selected envelopes contains a bill is $56/120 = 7/15$ – just under 0.5, unfortunately.

Alice Box: Combinations or Permutations?

Alice: Can't we do it with permutations too?

Hatter: Let's see. What's your sample space?

Alice: The set of all 3-element permutations (rather than subsets) of a 10-element set. This has $P(10,3) = 10!/7! = 10 \cdot 9 \cdot 8 = 720$ elements. And we are interested in the 3-element permutations (again, rather than subsets) of the 8-element no-bill set, of which there are $P(8,3) = 8!/5! = 8 \cdot 7 \cdot 6 = 336$ elements. As the sample space is equiprobable, the desired probability is given by the ratio $336/720 = 7/15$, the same as by the other method.

So there are problems that can be solved by either method. However, the solution using combinations has a smaller sample space. In effect, with permutations we are processing redundant information (the different orderings), giving us much larger numbers in the numerator and denominator, which we then cancel down. So it involves rather more calculation. Of course, much of the calculation could be avoided by cancelling as soon as possible, at the expression (8!/5!)/(10!/7!) before rewriting the numerator and denominator in decimal notation. But as a general rule, it is better to keep the sample space down from the beginning.

Exercise 6.3

(a) A pair of fair dice is thrown. What is the probability that the sum of the dots is 8? What is the probability that it is at least 5?

(b) Your debit card has a 4-digit pin number. You lose the card, and it is found by a dishonest person. The cash dispenser allows 3 attempts to enter the pin number. What is the probability that the person accesses your account, assuming that he/she enters candidate pin numbers at random, without repetition (but of course allowing repetition of individual digits).

(c) There are 25 people in a room. What is the probability that at least two of the people are born on the same day of the year (but not necessarily the same year)? For simplicity, assume that each year has 365 days, and that birthdays are randomly distributed through the year (both assumptions being, of course, rough approximations). *Hints*: (1) Be careful with your choice of sample space. (2) First find the probability that no two distinct people have the same birthday. (3) You will need a pocket calculator.

6.4 Conditional Probability

In this section, we introduce the general concept of conditional probability, its definition as a ratio, examples of its application and dangers of its misinterpretation.

6.4.1 The Ratio Definition

You throw a pair of fair dice. You already know how to calculate the probability of the event A that at least one of the dice is a 3. Your sample space is the set of all ordered pairs (n,m) where $n,m \in \{1,\ldots,6\}$ and you assume that the distribution on this space is equiprobable. There are 11 pairs (n,m) with either $n = 3$ or $m = 3$ (or both) so, by the assumption of equiprobability, $p(A) = 11/36$.

But what if we are informed, before looking at the result of the throw, that the sum of the two dice is 5? What is the probability that at least one die is a 3, *given that* the sum of the two dice is 5? This is a question of *conditional probability*.

In effect, we are changing the sample space. We are restricting it from the set S of all 36 ordered pairs (n,m) where $n,m \in \{1,\ldots,6\}$, to the subset $B \subseteq S$ consisting of those ordered pairs whose sum is 5. But it is very inconvenient to have to change the sample space each time we calculate a probability. It is easier to keep the sample space fixed and obtain the same result in another, equivalent, way. So we define the probability of A (e.g. that at least one die is a 3) *given* B (e.g. that the sum of the two faces is 5) as the ratio of $p(A \cap B)$ to $p(B)$. Writing the conditional probability of A given B as $p(A|B)$, we put:

$$p\,(A|B) = p(A \cap B)/\,p(B)$$

In the example given, there are four ordered pairs whose sum is 5, namely, the pairs (1,4), (2,3), (3,2), (4,1), so $p(B) = 4/36$. Just two of these contain a 3, namely, the pairs (2,3), (3,2), so $p(A \cap B) = 2/36$. Thus, $p(A|B) = (2/36)/(4/36) = 1/2$ – larger than the unconditional (alias *prior*) probability $p(A) = 11/36$.

Note that the conditional probability of A given B is not in general the same as the conditional probability of B given A. While $p(A|B) = p(A \cap B)/p(B)$, we have $p(B|A) = p(B \cap A)/p(A)$. The numerators are the same, since $A \cap B = B \cap A$, but the denominators are different: they are $p(B)$ and $p(A)$, respectively.

Exercise 6.4.1 (1) Calculate the value of $p(B|A)$ in the above die-throwing example to determine its relation to $p(A|B)$.

The non-convertibility (or non-commutativity) of conditional probability should not be surprising. In logic we know that the proposition $\alpha \rightarrow \beta$ is not the same as $\beta \rightarrow \alpha$, and in set theory, the inclusion $X \subseteq Y$ is not equivalent to $Y \subseteq X$.

Alice Box: Division by Zero

Alice: One moment! There is something that worries me in the definition of conditional probability. What happens to $p(A|B)$ when $p(B) = 0$, for example, when B is empty set? In this case, we can't put $p(A|B) = p(A \cap B)/\ p(B) = p(A \cap B)/0$, since division by zero is not possible.

Hatter: You are right! Division by zero is undefined in arithmetic, and so the definition that we gave for conditional probability does not make sense in this limiting case.

Alice: So what should we do?

Hatter: Just leave it undefined.

Actually, that is the most common of three possible reactions. It treats conditional probability, like division itself, as a *partial function* of two arguments (cf. Sect. 3.1 of Chap. 3). Its domain is not $\mathcal{P}(S)^2$ where S is the sample space, but rather $\mathcal{P}(S) \times \{X \subseteq S\colon p(X) \neq 0\}$. The second argument ranges over all the subsets of the sample space S whose probability under the function p is greater than zero. That excludes the empty set $\varnothing$, and it may also exclude some other subsets since it can happen that $p(X) = 0$ for a non-empty $X \subset S$. This option gives us the *ratio definition* of conditional probability: $p(A|B) = p(A \cap B)/p(B)$ except when $p(B) = 0$, in which case $p(A|B)$ is undefined. It is the definition that we follow in this chapter.

An alternative approach is to treat conditional probability as a full function on $\mathcal{P}(S)^2$, but to give $p(A|B)$ a constant value for all B with $p(B) = 0$, a 'dummy value' if you like to call it that. It is convenient to choose the constant value to be 1. This option gives us the *ratio/unit definition*: $p(A|B) = p(A \cap B)/p(B)$ except when $p(B) = 0$, in which case $p(A|B) = 1$. Really, these two definitions are not very different from each other; it is more a difference of style than of content.

There is also a third approach, which is significantly different. Like the first two, it puts $p(A|B) = p(A \cap B)/p(B)$ when $p(B) \neq 0$. Like the second, it treats $p(A|B)$ as well defined even $p(B) = 0$, thus taking the domain to be $\mathcal{P}(S)^2$. But unlike both, it allows the value of $p(A|B)$ to vary when $p(B) = 0$, provided it satisfies the *product rule* $p(A \cap B|C) = p(A|C) \cdot p(B|A \cap C)$. We will not go any further into this notion of *revisionary conditional probability*, except to say three things. (1) It has several main variants, corresponding to natural ways in which the details of the definition may be filled out for the 'critical zone' where $p(B) = 0$ but $B \neq \varnothing$; (2) it is quite popular among philosophers of probability; but (3) it is rarely used by statisticians, mathematicians and physicists, although it is employed by some economists in recent work on game theory.

Exercise 6.4.1 (2)

(a) Suppose that we know from records that, in a certain population, the probability $p(C)$ of high cholesterol is 0.3 and the probability $p(C \cap H)$ of high cholesterol and heart attack together is 0.1. Find the probability $p(H|C)$ of heart attack *given* high cholesterol.

(b) Suppose that we know from records from another population that the probability $p(H)$ of heart attack is 0.2 and the probability $p(H \cap C)$ of heart attack and high cholesterol together is 0.1. Find the probability $p(C|H)$ of high cholesterol *given* heart attack.

(c) Check the limiting case equalities: $p(\varnothing|B) = p(-B|B) = 0$, $p(S|B) = p(B|B) = 1$, and $p(A|S) = p(A)$.

(d) Show that (i) $p(A|B) = p((A \cap B)|B)$ and (ii) $p(A|B) \leq p(A'|B)$ whenever $A \subseteq A'$, both under the assumption that $p(B) \neq 0$.

(e) Show that the ratio definition of conditional probability satisfies the product rule $p(A \cap B|C) = p(A|C) \cdot p(B|A \cap C)$ whenever $p(A \cap C) \neq 0$.

6.4.2 Applying Conditional Probability

In the empirical sciences, conditional probability is often employed in the investigation of causal relationships. For example, as in the first parts of the last exercise, we may be interested in the extent to which high cholesterol leads to heart attack and use conditional probabilities like those in the exercise above when analysing the data. But great care is needed.

In the first place, conditional probability may make sense even when questions of time and causality do not arise. For example, in Exercise 6.4.1 (1) we calculated the conditional probabilities $p(A|B)$ and $p(B|A)$, where A is the event that at least one die is a 3 and B is the event that the sum of the two faces is 5. Neither of these events can be thought of as causally related to the other, nor does either happen before or after the other. Their conditional probabilities are merely a matter of numbers.

Second, when time and causality are relevant, we are just as entitled to look at the probability of an earlier event or suspected cause (e.g. high cholesterol) given a later one or suspected effect (e.g. heart attack), as conversely. Conditional probability makes sense in both directions.

Third, in the presence of time and causality, discovery of a value of $p(A|B)$ (even a value of 1) does not by itself establish any kind of causal link, direct or mediated, although it may lead us to suspect one. The mathematics says nothing about the existence of causal or temporal relationships between A and B in either direction.

Reasoning from high conditional probability to causation is fraught with philosophical difficulties and hedged with practical complications. Even inference in the reverse direction, from a causal relationship to a conditional probability, is not as straightforward as may appear. We will not attempt to unravel these fascinating and important issues, which are properly tackled in courses of statistics, experimental method and the philosophy of science. We remain with the confines of the basic mathematics of probability itself.

Exercise 6.4.2 (1) (with partial solution)

(a) In a writers' association, 60% of members write novels, 30% write poetry, but only 10% write both novels and poetry. What is (i) the probability that an arbitrarily chosen member writing poetry also writes novels, (ii) the converse conditional probability?

(b) In the same writers' association, 15% write songs, all of whom also write poetry, and 80% of the songsters write love poetry. But none of the novelists write love poetry. What is the probability that (i) an arbitrary poet writes songs, (ii) an arbitrary songster writes poetry, (iii) an arbitrary novelist writes love poetry, (iv) an arbitrary novelist writing love poetry also writes songs, (v) an arbitrary songster writes novels?

In this problem, understand all percentage figures given as *exact* (rather than at least, at most or approximate). Use the acronyms N,P,S,L for those members of the association who write novels, poetry, songs and about love.

Solution to (a), (b)(i), (b)(v)

(a) (i) $p(N|P) = p(N \cap P)/p(P) = 0.1/0.3 = 1/3$. (ii) $p(P|N) = p(P \cap N)/p(N) = 0.1/0.6 = 1/6$.

(b) (i) $p(S|P) = p(S \cap P)/p(P)$. But by hypothesis, $S \subseteq P$, so $S \cap P = S$, so $p(S|P) = p(S)/p(P) = 0.15/0.3 = 0.5$. (v) By the ratio definition, $p(N|S) = p(N \cap S)/p(S)$. But the data given in the problem do not suffice to calculate the value of $p(N \cap S)$, so we cannot solve the problem.

The last part of the exercise illustrates an important lesson. In real life, the information available is often insufficient to determine probabilities. One can sometimes get one by making rather arbitrary assumptions; for example, in the exercise we might assume that being a songster is independent of being a novelist (in a sense that we will define later) and use a principle for calculating the joint probability of independent events (also given later). But making such assumptions in an ad hoc or gratuitous manner can be dangerous. If introduced, they must be rendered explicit, so that they may be challenged and justified.

However, even when the data are insufficient to determine a precise probability for a property in which we are interested, they may permit us to work out a useful upper or lower bound. For example, in the exercise above, although we cannot calculate the exact value of $p(N \cap S)$, we can obtain an upper bound on it. Since exactly 15% of the association are songsters and exactly 80% of songster members write love poetry, we know that exactly 12% of the association are songsters writing love poetry. Since none of the love poets are novelists, *at least* 12% of the association (but not necessarily exactly 12%) are songster non-novelists. But recalling that exactly 15% of the association are songsters in the first place, we may conclude that *at most* 3% of the songsters are novelists, that is, $p(N \cap S) \leq 0.03$. So $p(N|S) = p(N \cap S)/p(S) \leq 0.03/0.15 \leq 0.2$. Rather tortuous reasoning, but it got us something.

Exercise 6.4.2 (2) Show that the data in the previous exercise are compatible with $N \cap S = \varnothing$. What does that tell us about a *lower* bound on $p(N|S)$?

When working with conditional probabilities it is easy to forget the fact that they are only partial functions, proceeding as if they are full ones. In other words, it is easy to neglect the fact that $p(A|B)$ is not defined in the limiting case that $p(B) = 0$. This can sometimes lead to serious errors. On the other hand, it is quite distracting to have to check out these limiting cases while trying to solve the problem; it can make you lose the thread of your thought.

How can these competing demands of rigour and insight be met? The best procedure is make two runs. When solving a problem involving conditional probability, make a first draft of your calculation or proof without worrying about the limiting cases, that is, as if conditional probability was always defined. When finished, go back and *check each step* to see that it is correct in those cases. This means verifying that each condition used in a conditional probability really has positive value, or finding an alternative argument for a step if the value of a condition is zero. These checks are usually quite straightforward, but sometimes there can be nasty surprises!

Exercise 6.4.2 (3) (with partial solution) Let p: $S \rightarrow [0,1]$ be a probability function. For each of the following statements, state the condition needed for all its conditional probabilities to be well defined, and show that it holds whenever well defined:

(a) $p(A_1 \cup A_2|B) = p(A_1|B) + p(A_2|B)$ whenever the sets $A_1 \cap B$, $A_2 \cap B$ are disjoint.
(b) $p(-A|B) = 1 - p(A|B)$.

(c) If $p(A) \geq p(B)$ then $p(A|B) \geq p(B|A)$.
(d) $p(A|B)/p(B|A) = p(A)/p(B)$.

Solution to (a)

(a) Condition needed: $p(B) \neq 0$. Proof: Suppose $A_1 \cap B$, $A_2 \cap B$ are disjoint. Then:

$p(A_1 \cup A_2|B) = p((A_1 \cup A_2) \cap B))/p(B)$ by definition of conditional probability

$= p((A_1 \cap B) \cup (A_2 \cap B))/p(B)$ by Boolean distribution in numerator

$= [p(A_1 \cap B) + p(A_2 \cap B)]/p(B)$ using disjointedness assumption

$= p(A_1 \cap B)/p(B) + p(A_2 \cap B)/p(B)$ by arithmetic

$= p\,(A_1|B) + p\,(A_2|B)$ by definition of conditional probability

Alice Box: What Is A | B?

Alice: I understand the definitions and can do the exercises. But there is still a question bothering me. *What is* $A|B$? The definition of conditional probability tells me how to calculate $p(A|B)$, but it does not say what $A|B$ itself is. Is it a subset of the sample space S like A,B, or some new kind of 'conditional object'?

Hatter: Neither one nor the other! Don't be misled by notation: $A|B$ is just a traditional shorthand way for writing the *ordered pair* (A,B). Whereas A,B are subsets of the sample space, $A|B$ is not: it is an ordered pair of such subsets. The vertical stroke is there merely to remind you that there is a division in the definition of $p(A,B)$. Conditional probability is thus a *partial two-place function*, not a one-place function with a funny argument. A notation uniform with the rest of mathematics would simply be $p(A,B)$.

Alice: Does an expression like $p(A|(B|C))$ or $p((A|B)|C)$ mean anything?

Hatter: Nothing whatsoever! Try to unpack the second one, say, according to the definition of conditional probability. We should have $p((A|B)|C) = p((A|B) \cap C)/p(C)$. The denominator is OK, but the numerator is meaningless, for $p(X \cap C)$ is defined only when X is a subset of the sample set S, and $A|B$ doesn't fit the bill.

Alice: So the operation of conditionalization can't be iterated?

Hatter: Not so fast! We *can* iterate conditionalization, but not like that! Back to the text to explain how.

Exercise 6.4.2 (4) Show that the other expression mentioned by Alice, $p(A|(B|C))$, is also meaningless.

As we have emphasized, a probability function p: $\mathscr{P}(S) \to [0,1]$ over a sample space S has only one argument, while its associated conditional probability function has two: it is a partial two-place function with domain $\mathscr{P}(S) \times \{X \subseteq S\colon p(X) \neq 0\}$. But like all two-place functions, full or partial, it can be transformed into a family of one-place functions by the operation of projection (Chap. 3, Sect. 3.5.3) from values of the left or right argument, and in this case, the interesting projections hold fixed the right argument. For every probability function p: $\mathscr{P}(S) \to [0,1]$ and $B \subseteq S$ with $p(B) \neq 0$, we have the one-place function p_B: $\mathscr{P}(S) \to [0,1]$ defined by putting $p_B(A) = p(A|B) = p(A \cap B)/p(B)$ for all $A \subseteq S$.

Why should these projections be of any interest? The reason is that each such function p_B: $\mathscr{P}(S) \to [0,1]$ *is also a probability function* over the same sample space S, as is easily verified and is set as an exercise below. It is called the *conditionalization* of p on B. This reveals a perfectly good sense in which the operation of conditionalization may be iterated. Although, as we have seen, Alice's expressions $p(A|(B|C))$ and $p((A|B)|C)$ are meaningless, the functions $(p_B)_C$ and $(p_C)_B$ are well defined when $p(B \cap C) \neq 0$. Moreover, and rather surprisingly, it is not difficult to show that the order of conditionalization is immaterial, and that every iterated conditionalization collapses into a suitable one-shot conditionalization. Specifically: $(p_B)_\mathrm{C} = p_{B \cap C} = p_{C \cap B} = (p_C)_B$ when all these functions are well defined, that is, when $p(B \cap C) \neq 0$. Verifying this last point is one of the end-of-chapter exercises.

Exercise 6.4.2 (5) (with partial solution) Let p: $\mathscr{P}(S) \to [0,1]$ be a probability function. Show the following:

(a) $p = p_S$.

(b) If $p(B) \neq 0$, then $p_B(A_1 \cup A_2) = p_B(A_1) + p_B(A_2)$ whenever the sets $A_1 \cap B$, $A_2 \cap B$ are disjoint.

(c) If $p(B) \neq 0$, then $p_B(-A) = 1 - p_B(A)$.

(d) For any probability function p: $\mathscr{P}(S) \to [0,1]$ and $B \subseteq S$ with $p(B) \neq 0$, the function p_B: $\mathscr{P}(S) \to [0,1]$ is well defined and is also a probability function over S.

Solution for (a) and Hints for the Others (a) By definition, $p_S(A) = p(A \cap S)/p(S) = p(A)/1 = p(A)$. Note that $p(S) = 1 \neq 0$. For (b), (c) use Exercise 6.4.2 (3). For (d), unpack the definition and make use of any properties already established.

6.5 Interlude: Simpson's Paradox*

We pause for a moment to note a surprising fact known as *Simpson's paradox*, named after the person who rediscovered it in 1951, although it had been noticed much earlier in 1903 by Yule, and even earlier in 1899 by Pearson. It shows – just in case you hadn't already noticed – that even elementary discrete probability theory is full of traps for the unwary!

Suppose we have a sample space S, a partition of S into two cells F, $-F$ and another partition of S into two cells C, $-C$. To fix ideas, take S to be the set of all applicants for vacant jobs in a university in a given period. It is assumed that all applicants are female or male (but not both), and that every application is addressed either to computer science or to mathematics (but not both). We can thus write F, $-F$ for the female and male applicants, respectively, and C, $-C$ for the applicants to the respective departments of computer science and mathematics. The four sets $F \cap C$, $F \cap -C$, $-F \cap C$, $-F \cap -C$ form a partition of S (Chap. 2, Sect. 2.5.3). Now let H be the set of applicants actually hired. Suppose that:

$$p(H|F \cap C) > p(H|-F \cap C);$$

that is, the probability that an arbitrarily chosen female candidate to the computer science department is hired, is greater than the probability that an arbitrarily chosen male candidate to the computer science department is hired. Suppose that the same strict inequality holds regarding the mathematics department, that is, that

$$p(H|F \cap -C) > p(H|-F \cap -C).$$

Intuitively, it seems inevitable that the same inequality will hold when we take unions on each side, so that

$$p\,(H|F) > p(H|-F);$$

in other words, that the probability that a female candidate is hired, is greater than that for a male candidate. But in fact, it need not do so! We give a counterexample (based on a lawsuit that was actually brought against a US university). Let S have 26 elements, neatly partitioned into 13 female and 13 male. Eight women apply to computer science, of whom two are hired, while five men apply to the same department, of whom one is hired. At the same time, five women apply to mathematics, with four hired, while eight men apply with six hired. We calculate the probabilities.

$$p(H|F \cap C) = p(H \cap F \cap C)/p(F \cap C) = 2/8 = 0.25 \text{ while}$$
$$p(H|-F \cap C) = p(H \cap -F \cap C)/p(-F \cap C) = 1/5 = 0.2,$$

so that the first inequality holds. Likewise

$$p(H|F \cap -C) = p(H \cap F \cap -C)/p(F \cap -C) = 4/5 = 0.8 \text{ while}$$

$$p(H|-F \cap -C) = p(H \cap -F \cap -C)/p(-F \cap -C) = 6/8 = 0.75,$$

and thus the second inequality also holds. But we also have:

$$p\,(H|F) = p(H \cap F)/p(F) = (2 + 4)\,/13 = 6/13 < 0.5$$
$$p(H|-F) = p(H \cap -F)/p(-F) = (1 + 6)\,/13 = 7/13 > 0.5,$$

and so the third inequality fails – giving rise to the lawsuit!

Of course, the 'paradox' is not a paradox in the strict sense of the term. It is not a contradiction arising from apparently unquestionable principles; it does not shake the foundations of probability theory. But it shows that there can be surprises even in elementary discrete probability. Intuition is not always a reliable guide in matters of probability, and even experts (as well as laymen and lawyers) can go wrong.

Simpson's paradox may also be regarded as a manifestation in a probabilistic setting of a simple and perhaps less surprising arithmetical fact: there are positive integers with $a_1/x_1 > a_2/x_2$ and $b_1/y_1 > b_2/y_2$ but $(a_1 + b_1)/(x_1 + y_1) < (a_2 + b_2)/(x_2 + y_2)$.

6.6 Independence

There are two equivalent ways of defining the independence of two events, using probability. One uses conditional probability, the other is in terms of multiplication of unconditional probabilities. We begin with the definition via conditional probability.

Let p: $\mathcal{P}(S) \to [0,1]$ be any probability function on a (finite) sample space S. Let A,B be events (i.e. subsets of S). We say that *A is independent of B* (modulo p) iff $p(A) = p(A|B)$ (in the principal case) or (limiting case) $p(B) = 0$. In the language of projections, iff $p(A) = p_B(A)$ when p_B is well defined. In rough terms, iff conditionalizing on B makes no difference to the probability of A.

Note that this is just mathematics; it is not a matter of the presence or absence of a causal connection. Remember too that independence is modulo the probability function chosen: A may be independent of B with respect to one probability function p on $\mathcal{P}(S)$, but not so with respect to another one p'.

The other definition, in terms of multiplication, is as follows: A is independent of B (modulo p) iff $p(A \cap B) = p(A) \cdot p(B)$. The two definitions are equivalent. Some people find the definition in terms of conditional probabilities more intuitive (except for its limiting case, which is a little annoying). But the account in terms of multiplication is usually handier to work with, and we adopt it as our official definition. The following exercise looks at some basic properties of independence.

Exercise 6.6 (1) (with partial solution)

(a) Show the equivalence of the two definitions of independence.
(b) Show that the relation of independence is symmetric.

(c) Show that A is independent of B in each of the four limiting cases that $p(B) = 0$, $p(B) = 1$, $p(A) = 0$, $p(A) = 1$.
(d) Show that for disjoint A,B, if A is independent of B then $p(A) = 0$ or $p(B) = 0$.

Solution to (a)

(a) Suppose first that $p(A \cap B) = p(A) \cdot p(B)$ and $p(B) \neq 0$. Then we may divide both sides by $p(B)$, giving us $p(A) = p(A \cap B)/p(B) = p(A|B)$ as desired. Conversely, suppose either $p(B) = 0$ or $p(A) = p(A|B)$. In the former case, $p(A \cap B) = 0 = p(A) \cdot p(B)$ as needed. In the latter case, $p(A) = p(A|B) = p(A \cap B)/p(B)$, so multiplying both sides by $p(B)$ we have $p(A \cap B) = p(A) \cdot p(B)$ and we are done.

An advantage of the multiplicative definition is that it generalizes elegantly to the independence of any finite collection of events. Let $\{A_1, \ldots, A_n\}$ be any such collection with $n \geq 1$. We say that the A_i are *independent* iff for every m with $1 \leq m \leq n$ we have: $p(A_1 \cap \ldots \cap A_m) = p(A_1) \cdot \ldots \cdot p(A_m)$. Equivalently, using a definition that may at first glance seem circular, but is in fact recursive (see Chap. 4): iff every proper non-empty subcollection of $\{A_1, \ldots, A_n\}$ is independent and $p(A_1 \cap \ldots \cap A_n) = p(A_1) \cdot \ldots \cdot p(A_n)$. It should be emphasized that the independence of n events does not follow from their pairwise independence; one of the end-of-chapter exercises asks you to find a simple counterexample.

When it holds, independence is a powerful tool for calculating probabilities. We illustrate with an example. Five per cent of computers sold by a store are defective. Today the store sold three computers. Assuming that these three form a random sample, with also independence as regards defects, what are: (i) the probability that all of them are defective, (i) the probability that none of them have defects, (iii) the probability that at least one is defective?

We are told that the probability $p(D_i)$ of defect for each computer c_i ($i \leq 3$) taken individually is 0.05, and that these probabilities are independent. For (i), the assumption of the independence of $\{D_1, D_2, D_3\}$ tells us that the probability $p(D_1 \cap D_2 \cap D_3)$ that all three are defective is $(0.05)^3 = 0.000125$. For (ii), the probability $p(-D_i)$ that a given computer is not defective is 0.95, so by independence again, the probability $p(-D_1 \cap -D_2 \cap -D_3)$ that none are defective is $(0.95)^3 = 0.857375$, which is significantly less than 0.95. For (iii), $p(D_1 \cup D_2 \cup D_3) = 1 - p(-D_1 \cap -D_2 \cap -D_3) = 1 - 0.857375 = 0.142625$.

Alice Box: A Flaw in the Argument?

Alice: Just a minute! I think that there is a flaw in that argument. In the statement of the problem, it was given that the sets D_i are independent. But in answering part (ii), we needed to assume that their *complements* $-D_i$ are independent.

Hatter: Gap in the argument, yes, and thanks for pointing it out. But not a flaw. In fact, if the D_i are independent, so are the $-D_i$. For the case $n = 2$, that will be part of the next exercise.

Exercise 6.6 (2) (with solution) Show that the following four conditions are all equivalent (modulo any given probability function): A and B are independent, A and $-B$ are independent, $-A$ and B are independent, $-A$ and $-B$ are independent.

Solution It suffices to show the equivalence of the first to the second. For, given that, we can reapply it to get the equivalence of the first to the third by using symmetry on the first. Then we get the equivalence of the third to the fourth by reapplying it again. So, suppose $p(A \cap B) = p(A) \cdot p(B)$; we want to show that $p(A \cap -B) = p(A) \cdot p(-B)$. Now $p(A) = p(A \cap B) + p(A \cap -B) = p(A) \cdot p(B) + p(A \cap -B)$. Hence, $p(A \cap -B) = p(A) - p(A) \cdot p(B) = p(A) - [p(A) \cdot (1 - p(-B))] = p(A) - [p(A) - p(A) \cdot p(-B)] = p(A) \cdot p(-B)$, as desired.

The usefulness of independence for making calculations should not, however, blind us to the fact that in real life it holds only exceptionally and should not be assumed without good reason. Examples abound. In one from recent UK legal history, an expert witness cited the accepted (low) figure for the probability of an infant cot death in a family, and obtained the probability of two successive such deaths simply by squaring the figure, which of course was very low indeed. This is legitimate only if we can safely assume that the probabilities of the two deaths are independent, which is highly questionable.

Exercise 6.6 (3)

(a) Two archers A_1, A_2 shoot at a target. The probability that A_1 gets a bull's-eye is 0.2, the probability that A_2 gets a bull's-eye is 0.3. Assuming independence, what are: (i) the probability that they both get a bull's-eye? (ii) That neither gets a bull's-eye? (iii) Assuming also independence for successive tries, that neither gets a bull's-eye in ten successive attempts each?

(b) Can an event ever be independent of itself?

(c) Is the relation of independence transitive?

6.7 Bayes' Theorem

We have emphasized that conditional probability does not commute: in general, $p(A|B) \neq p(B|A)$. But there are situations in which we can calculate the value of one from the other provided we are given some supplementary information. In this section, we see how this is done.

Assume that $p(A)$, $p(B)$ are non-zero. By definition, $p(A|B) = p(A \cap B)/p(B)$, so $p(A \cap B) = p(A|B) \cdot p(B)$. But likewise $p(B \cap A) = p(B|A) \cdot p(A)$. Since $A \cap B = B \cap A$ this gives us $p(A|B) \cdot p(B) = p(B|A) \cdot p(A)$ and so finally:

$$p(A|B) = p(B|A) \cdot p(A)/p(B).$$

Thus, we can calculate the conditional probability in one direction if we know it in the other direction and also know the two unconditional (also known as *prior*, or *base rate*) probabilities. Indeed, we can go further. Since

$p(B) = p(B|A){\cdot}p(A) + p(B|-A){\cdot}p(-A)$ when $p(A)$ and $p(-A)$ are non-zero, as shown in the next exercise, we can substitute in the denominator to get the following equation, which is a special case of what is known as *Bayes' theorem*:

> Assuming that $p(A)$, $p(-A)$, $p(B)$ are non-zero so $p(A|B)$, $p(B|A)$ and $p(B|-A)$ are well defined, $p(A|B) = p(B|A){\cdot}p(A)/[p(B|A){\cdot}p(A) + p(B|-A){\cdot}p(-A)]$.

So we can calculate the probability of $p(A|B)$ provided we know both of the conditional probabilities $p(B|A)$ and $p(B|-A)$ in the reverse direction, and the unconditional probability $p(A)$, from which we can of course get $p(-A)$.

Exercise 6.7 (1)

(a) Verify the claim made above, that $p(B) = p(B|A){\cdot}p(A) + p(B|-A){\cdot}p(-A)$ when $p(A)$ and $p(-A)$ are non-zero. *Hint*: Use the fact that $B = (B{\cap}A){\cup}(B{\cap}-A)$.

(b) When $p(B) = 0$ then $p(A|B)$ is not defined, but when $p(B) \neq 0$ then $p(A|B)$ is defined even if $p(A) = 0$ or $p(-A) = 0$. In these limiting two cases, we can still calculate the value of $p(A|B)$, not by Bayes' theorem but quite directly. What is the value in those cases?

Bayes' theorem is named after the eighteenth-century clergyman-mathematician who first noted it, and is surprisingly useful (also, as we will see, abusable). For it can often happen that we have no information permitting us to determine $p(A|B)$ directly, but do have statistical information, that is, records of frequencies, allowing us to give figures for $p(B|A)$ and for $p(B|-A)$, together with some kind of estimate of $p(A)$. With that information, Bayes' theorem tells us how to calculate $p(A|B)$.

However, applications of Bayes' theorem can be controversial. The figures coming out of the calculation are no better than those going into it: as the saying goes, garbage in, garbage out. In practice, it often happens that we have good statistics for estimating $p(B|A)$, perhaps less reliable ones for $p(B|-A)$, and only vague theoretical reasons for guessing approximately at $p(A)$. A good critical sense is needed to avoid being led into fantasy by the opportunity for quick calculation.

Indeed, probability theorists and statisticians tend to be divided into what are called *Bayesians* and their opponents. The difference is one of attitude or philosophy rather than mathematics. Bayesians tend to be happy with applying Bayes' theorem to get a value for $p(A|B)$ even in contexts where available statistics give us no serious indication of the value of the 'prior' $p(A)$. They are willing to rely on non-statistical reasons for its estimation or are confident that possible errors deriving from this source can be 'washed out' in the course of successive conditionalizations as new information comes in. Their opponents tend to be much more cautious, unwilling to assume a probability that does not have a serious statistical backing. As one would expect, subjectivists tend to be Bayesian, while frequentists do not.

It is beyond the scope of this book to enter into the debate, although the reader can presumably guess where the author's sympathies lie. And it should be remembered that whatever position one takes in the debate on applications, Bayes' theorem as a mathematical result, stated above, remains a provable fact. The disagreement is about the conditions under which it may reasonably be applied.

Exercise 6.7 (2) (with solution) Fifteen percent of patients in a ward suffer from the HIV virus. Further, of those that have the virus, about 90% react positively on a certain test, whereas only 3% of those lacking the virus react positively. (i) Find the probability that an arbitrarily chosen patient has the virus given that the test result is positive. (ii) What would that probability be if 60% of the patients in the ward suffered from the HIV virus, the other data remaining unchanged?

Solution First, we translate the data and goal into mathematical language. The sample set S is the set of all patients in the ward. We write V for the set of those with the HIV virus and P for the set of those who react positive to the test. The phrase 'arbitrarily chosen' indicates that we may use the percentages as probabilities. We are given the probabilities $p(V) = 0.15$, $p(P|V) = 0.9$, $p(P|-V) = 0.03$. For (i), we are asked to find the value of $p(V|P)$. Bayes' theorem tells us that $p(V|P) = p(P|V) \cdot p(V)/[p(P|V) \cdot p(V) + p(P|-V) \cdot p(-V)]$. The rest is calculation: $p(V|P) = 0.9 \cdot 0.15/(0.9 \cdot 0.15 + 0.03 \cdot 0.85) = 0.841$ to the first three decimal places. For (ii), we are given $p(V) = 0.6$ with the other data unchanged, so this time $p(V|P) = 0.9 \cdot 0.6/(0.9 \cdot 0.6 + 0.03 \cdot 0.4) = 0.978$ to the first three decimal places.

Notice how, in this exercise, the solution depends on the value of the unconditional probability (alias base rate, or prior) $p(V)$. Other things being equal, a higher prior gives a higher posterior probability: as $p(V)$ moved from 0.15 to 0.6, $p(V|P)$ moved from 0.841 to 0.948. This phenomenon holds quite generally.

Bayes' theorem can be stated in a more general manner. Clearly, the pair A,$-A$ gives us a two-cell partition of the sample space S (with complementation being taken as relative to that space), and it is natural to consider more generally any partition into n cells $A_1, \ldots, A_n$. Essentially the same argument as in the two-cell case then gives us the following *general version of Bayes' theorem*. It should be committed to memory. Consider any event B and partition $\{A_1, \ldots, A_n\}$ of the sample space such that B and each A_i has non-zero probability. Then for each $i \leq n$ we have:

$$p\left(A_i|B\right) = p\left(B|A_i\right) \cdot p\left(A_i\right) / \Sigma_{j \leq n} \left[p\left(B|A_j\right) \cdot p\left(A_j\right)\right].$$

Exercise 6.7 (3) A telephone helpline has three operators, Alfred, Betty, Clarice. They receive 33%, 37%, 30% of the calls, respectively. They manage to solve the problems of 60%, 70%, 90% of their respective calls. What is the probability that a successfully handled problem was dealt with by Alfred, Betty, Claire, respectively? *Hint*: First translate into mathematical language and then use Bayes' theorem with $n = 3$.

6.8 Expectation*

Let S be any finite set, and let f: $S \rightarrow \mathbf{R}$ be a function associating a magnitude, positive or negative, with each of the elements of S. For example, S may consist of runners in a horse race, and f may give the amount of money that will be won or lost, under some system of bets, when a horse wins. Or S could consist of the outcomes of throwing a die twice, with f specifying say the sum of the numbers appearing in the two throws. All we need to assume about f is that takes elements of S into the reals. It need not be surjective; in applications, the range may often be quite small. When the points in $f(S)$ are thought of as representing the 'worth' of elements of S, in money terms or otherwise, then we call $f\colon S \to \mathbf{R}$ a *payoff function*.

A terminological digression: Payoff functions are often called *random variables* and written $X\colon S \to \mathbf{R}$, particularly in texts covering the infinite as well as the finite case. The range of the function, which we are writing as $f(S)$ in set-theoretic notation, is then called its *range space* and written $\mathbf{R}_X$. Once again, this is a traditional topic-specific terminology, and the reader should be able to translate it into universal set-theoretic language. From a modern point of view, it can be quite misleading, since a 'random variable' is not a variable in the syntactic sense of a letter used to range over a domain, but rather a function. Nor is it necessarily random in the sense of being associated with an equiprobable distribution. End of digression.

Now suppose that we also have a probability distribution $p\colon S \to [0,1]$ on S (not necessarily equiprobable). Is there any way in which we can bring the payoff function and the probability to work together, to give the 'expected worth' of each element of S? The natural idea is to weigh one by the other. A low payoff for an $s \in S$ with a high probability may be as desirable as a large payoff for an s' with small probability. We may consider the probability-weighted payoff (or payoff-weighted probability) of an element $s \in S$ to be the product $f(s){\cdot}p(s)$ of its worth by its probability. This is also known as the *expected value* of s. For example, when S consists of the horses in a race, $p(s)$ is the probability that horse s will win (as determined from, say, its track record) and $f(s)$ is the amount that you stand to gain or lose from a bet if s wins, then $f(s){\cdot}p(s)$ is the gain or loss weighted by its probability.

This in turn leads naturally to the concept of *expected value* or, more briefly, *expectation*. It is a probability-weighted average for the payoff function itself. We introduce it through the same example of a three-horse race. Let $S = \{a,b,c\}$ be the runners, with probabilities of winning $p(a) = 0.1$, $p(b) = 0.3$, $p(c) = 0.6$. Suppose the payoff function puts $f(a) = 12$, $f(b) = -1$, $f(c) = -1$. Then the expectation of f given p is the sum $f(a){\cdot}p(a) + f(b){\cdot}p(b) + f(c){\cdot}p(c) = 12{\cdot}(0.1) - 1{\cdot}(0.3) - 1{\cdot}(0.6) = 0.3$.

In general terms, given a payoff function $f\colon S \to \mathbf{R}$ and a probability distribution $p\colon S \to [0,1]$ we define the *expectation of f given p*, written $\mu(f, p)$ or just μ, by the equation $\mu = \mu(f, p) = \Sigma_{s \in S}\{f(s){\cdot}p(s)\}$. Equivalently, in a form that can be useful when $f(S)$ is small and the probability function $p^{+}\colon \mathscr{P}(S) \to [0,1]$ determined by the distribution $p\colon S \to [0,1]$ is independently known, we have $\mu(f,p) = \Sigma_{x \in f(S)}\{x{\cdot}p^{+}(f^{-1}(x))\}$.

Note that to calculate the expectation we need to know both the payoff function and the probability distribution. Neither alone suffices. The pair (f,p) consisting of the two is sometimes referred to as a *gamble*. We can thus speak of the *expectation of a gamble*.

Exercise 6.8

(a) In the horse race example of the text, compare the expectation of *f* given *p* with the arithmetic mean (alias average) of values of *f*, calculated (without reference to probability) by the usual formula $mean(f) = \Sigma_{s \in S}\{f(s)\}/\#(S)$. Explain the basic reason for the difference.

(b) Show that in the special case that p: $S \to [0,1]$ is an equiprobable distribution expectation equals mean.

(c) Consider a pair of fair dice about to be thrown. We want to determine the expected value of the sum of the two outcomes. First, specify the sample space, the payoff function and the probability distribution. Use the preceding part of this exercise to calculate the expectation without reference to the probability function.

(d) This part is for the mathematically inclined. Show that the two definitions of the expectation of *f* given *p* are indeed equivalent.

When the expectation of a gamble comes out negative, then it is not a good bet to make – one is likely to lose money. When it is positive, the bet is favourable, provided of course the payoff function correctly records *all* your costs (perhaps including, as well as the actual outlay, taxes on earnings, time spent and opportunities foregone, stress) and benefits (perhaps including, as well as money won, the excitement and enjoyment given by the affair). In our horse race example, if those auxiliary costs and benefits are zero, then $\mu = 0.3$, and it is a good bet to make.

Alice Box: Really a Good Bet?

Alice: Not so fast! If I bet only once, then there are only two possibilities: either I win or I lose. The probability of winning is very low (0.1), while the probability of losing is correspondingly high (0.9). So, even though the benefit of winning is high at $12 and the cost of losing is low at $1 loss, I am much more likely to lose than to win. Not very attractive!

Hatter: If you bet only once, yes. But if you make the same bet many times, then it becomes highly probable that the outcome is close to the expected value. We will not show that here, but it can be proven. In that case, the bet becomes attractive.

Alice: Provided I have enough capital to be able to put up with some probable initial losses without undue hardship . . .

As usual, Alice has a good point. If there is a great disparity between the probabilities of winning and losing, then we may be likely to bankrupt ourselves while waiting for the lucky day. One might try to factor this into the payoff function, but this would be complex and perhaps rather artificial. Once again, we see that, while the mathematics of probability theory are quite straightforward – at least in the finite case, which we are dealing with here – its application can remain very tricky.

In the alternative characterization of expectation, we made use of the function $p^+{\circ}f^{-1}: f(S) \rightarrow [0,1]$, that is, the composition of the inverse of the payoff function and the probability function. We end by observing that this function turns out to be a probability distribution on $f(S)$. Thus, in turn, it determines another probability function $(p^+{\circ}f^{-1})^+: \mathscr{P}(f(S)) \rightarrow [0,1]$.

In this chapter, we have only scratched the surface of even finite probability theory. For example, as well as expectation μ, statisticians need measures of the *dispersion* of items around the expectation point, understood as the degree to which they bunch up around it or spread far out on each side. The first steps in that direction are the concepts of *variance* and *standard distribution*. We will not discuss them in this brief chapter; there are pointers to reading below.

End-of-Chapter Exercises

Exercise 6.1: Finite probability spaces

(a) Consider a loaded die, such that the numbers 1 through 5 are equally likely to appear, but 6 is twice as likely as any of the others. To what probability distribution does this correspond?

(b) Consider a loaded die in which the odd numbers are equally likely, the even numbers are also equally likely, but are three times more likely than the odd ones. What is the probability of getting a prime number?

(c) Consider a sample space S with n elements. How many probability distributions are there on this space in which $p(s) \in \{0,1\}$ for all $s \in S$? For such a probability distribution, formulate a criterion for $p(A) = 1$, where $A \subseteq S$.

(d) Show by induction on n that when $A_1, \ldots, A_n$ are pairwise disjoint, then $p(A_1 \cup \ldots \cup A_n) = p(A_1) + \cdots + p(A_n)$.

Exercise 6.2: Unconditional probabilities

(a) Consider the probability distribution in part (a) of the preceding exercise. Suppose that you roll the die three times. What is the probability that you get at least one 6?

(b) You have a loaded die with probability distribution like that in part (a) of the preceding exercise, and your friend has one with probability distribution like that in part (b). You both roll. What is the probability that you both get the same number?

(c) John has to take a multiple-choice examination. There are ten questions, each to be answered ‘yes’ or ‘no’. Marks are counted simply by the number of correct answers, and the pass mark is 5. Since he has not studied at all, John decides to answer the questions randomly. What is the probability that he passes?

(d) Mary is taking another multiple-choice examination, with a different marking system. There are again ten questions, each to be answered ‘yes’ or ‘no’, and the pass mark is again 5. But this time marks are counted by taking the number of correct answers and subtracting the number of incorrect ones. If a question is left unanswered, it is treated as incorrect. If Mary answers at random, what is the probability that she passes?

Exercise 6.3: Conditional probability

(a) In a sports club, 70% of the members play football, 30% swim and 20% do both. What are: (i) the probability that a randomly chosen member plays football, given that he/she swims? (ii) the converse probability?

(b) Two archers Alice and Betty shoot an arrow at a target. From their past records, we know that their probability of hitting the target are 1/4 and 1/5, respectively. We assume that their performances are independent of each other. If we learn that exactly one of them hit the target, what is the probability that it was Alice?

(c) Show, as claimed in the text, that for any probability function p: $\mathscr{P}(S) \rightarrow [0,1]$, if $p(B\cap C) \neq 0$ then $(p_B)_{\mathrm{C}} = p_{B\cap C} = p_{C\cap B} = (p_C)_B$.

(d) Using the above, apply induction to show that any finite series of conditionalizations can be reduced to a single one.

Exercise 6.4: Independence

(a) A fair coin is tossed three times. Consider the events A, B, C that the first toss gives tails, the second toss gives tails, and we get exactly two tails in a row. Specify the sample space, and identify the three events by enumerating their elements. Show that A and B are independent, and likewise A and C are independent, but B and C are not independent.

(b) Use results established in the text to show that A is independent of B iff $p(A|B) = p(A|-B)$.

(c) Show that if A is independent of each of two disjoint sets B,C, then it is independent of $B\cup C$.

(d) Construct an example of a sample space S and three events A, B, C that are pairwise independent but not jointly so.

Exercise 6.5: Bayes’ theorem

(a) Suppose that the probability of a person getting flu is 0.3, that the probability of a person having been vaccinated against flu is 0.4, and that the probability of a person getting flu given vaccination is 0.2. What is the probability of a person having been vaccinated given that he/she has flu?

(b) At any one time, approximately 3% of drivers have a blood alcohol level over the legal limit. About 98% of those over the limit react positively on a breath test, but 7% of those not over the limit also react positively. Find: (i) the probability that an arbitrarily chosen driver is over the limit given that the breath test is positive; (ii) the probability that a driver is not over the limit given that the breath test is negative; (iii) the corresponding results in a neighbouring country where, unfortunately, 20% of drivers are over the limit.

(c) Suppose that we redefine conditional probability in the alternative manner mentioned in Sect. 6.4.1, putting $p(A|B) = 1$ whenever $p(B) = 0$. What happens to Bayes' theorem in the limiting cases? *Suggestion*: First answer this question for the two-cell version of Bayes' theorem, then for the general version.

Exercise 6.6: Expectation*

(a) Consider a box of 10 gadgets, of which 4 are defective. A sample of three items (order immaterial, without replacement, that is, mode $\mathbf{O} - \mathbf{R}-$ in the notation of the chapter on counting) is drawn at random. What are: (i) the probability that it has exactly one defective item, (ii) the expected number of defective items?

(b) In a television contest, the guest is presented with four envelopes from which to choose at random. They contain $1, $10, $100, $1,000, respectively. Assuming that there are no costs to be taken into consideration, what is the expected gain?

(c) A loaded coin has $p(H) = 1/4$, $p(T) = 3/4$. It is tossed three times. Specify the corresponding sample space S. Let f be the payoff function on this sample space that gives the number of heads that appear. Specify $f(S)$. Calculate the expectation $\mu = \mu(f,p)$.

(d) For the more mathematically inclined: Let S be a finite sample space, f: $S \rightarrow \mathbf{R}$ a payoff function and p: $S \rightarrow [0,1]$ a probability distribution on S. Show, as claimed in the last paragraph of the chapter, that the function $p^{+} \circ f^{-1}$: $f(S) \rightarrow [0,1]$ is a probability distribution on $f(S)$.

Selected Reading

Introductory texts of discrete mathematics tend to say rather little on probability. However, the following two cover more or less the same ground as this chapter:

Lipschutz S, Lipson M (1997) Discrete mathematics, 2nd edn, Schaum's outline series. McGraw Hill, New York, chapter 7

Rosen K (2007) Discrete mathematics and its applications, 6th edn. McGraw Hill, New York, chapter 6

For those wishing to go further and include the infinite case, the authors of the first text have also written the following:

Lipschutz S, Lipson M (2000) Probability, 2nd edn, Schaum's outline series. McGraw Hill, New York

For all of the above, you may find it useful to use Table 6.1 to keep track of the more traditional terminology and notation that is often used.

Finally, if you had fun with the interlude on Simpson's paradox, you will enjoy the *Monty Hall problem*. See the *Wikipedia* entry on it, and if you can't stop thinking about it, continue with, say:

Rosenhouse J (2009) The Monty Hall problem. Oxford University Press, Oxford

Squirrel Math: Trees 7

Abstract

This chapter introduces a kind of structure that turns up everywhere in computer science: trees. We will be learning to speak their language – how to talk about their components, varieties and uses – more than proving things about them. The flavour of the chapter is thus rather different from that of the preceding one on probability: more use of spatial intuition, rather less in the way of demonstration.

We begin by looking at trees in their most intuitive form – *rooted* (*alias directed*) trees – first of all quite naked and then clothed with *labels* and finally *ordered*. Special attention will be given to the case of *binary trees* and their use in search procedures. Finally, we turn to *unrooted* (*or undirected*) trees and their application to *span* graphs. As always, we remain in the finite case.

7.1 My First Tree

Before giving any definitions, we look at an example. Consider the structure presented in Fig. 7.1, reading it from top to bottom.

This structure is a rooted tree. It consists of a set T of 15 elements $a, \ldots, o$, together with a relation over them. The relation is indicated by the arrows. The elements of the set are called *nodes* (aka *vertices*); the arrows representing pairs in the relation are called *links*. We note some features of the structure:

- The relation is acyclic, and so in particular asymmetric and irreflexive (as defined in the chapter on relations).
- There is a distinguished element a of the set, from which all others may be reached by following paths in the direction of the arrows. In the language of Chap. 2, all other elements are in the transitive closure of $\{a\}$ under the relation. This is called the *root* of the tree, and it is unique.
- Paths may fork, continue or stop. Thus, from the node a, we may pass to two nodes b and c. These are called the *children* of a. While a has just two children, d has three (i, j, k), and c has four (e, f, g, h). But b has only one (d), and m has none. Mixing biological with botanical metaphors, nodes without children are

D. Makinson, *Sets, Logic and Maths for Computing*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-2500-6_7,

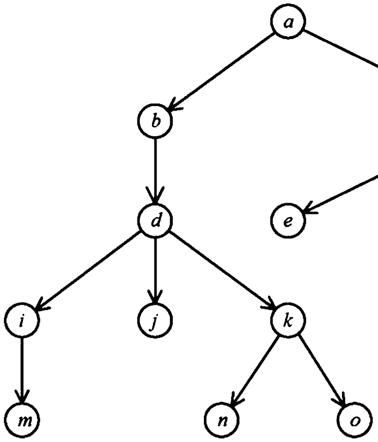

Fig. 7.1 Example of a rooted tree

called *leaves*. Any path from the root to a leaf is called a *branch* of the tree. Note that a branch is an entire path from root to a leaf; thus, neither (b,d,i) nor (a,c,h) nor (a,b,k,n) is a branch of the tree in the figure.

- Paths never meet once they have diverged: we never have two or more arrows going to a node. In other words, each node has at most one *parent*. An immediate consequence of this and irreflexivity is that the link relation is intransitive.

Given these features, it is clear that in our example we can leave the arrowheads off the links, with the direction of the relation understood implicitly from the layout.

Exercise 7.1 (with partial solution) This exercise concerns the tree in Fig. 7.1 and is intended to sharpen intuitions before we give a formal definition of what a tree is. Feel free to make use of any of the bulleted properties extracted from the figure.

(a) Identify and count all the leaves of the tree.
(b) Why are none of (b,d,i), (a,c,h), (a,b,k,n) branches of the tree?
(c) Identify all the branches of the tree and count them. Compare the result with that for leaves and comment.
(d) The *link-length* of a branch is understood to be the number of links making it up. How does this relate to the number of nodes in the branch? Find the link-length of each branch in the tree.
(e) Suppose you delete the node m and the link leading into it. Would the resulting structure be a tree? What if you delete the node but leave the link? And if you delete the link but not the node?
(f) Suppose you add a link from m to n. Would the resulting structure be a tree? And if you add one from c to l?
(g) Suppose you add a node p without any links. Would you have a tree? If you add p with a link from b to p? And if you add p but with a link from p to b? One from p to a?

(h) Given the notions of parent and child in a tree, define those of sibling, descendant and ancestor in the natural way. Identify the siblings, descendants and ancestors of the node *d* in the tree.

Solutions to (a), (b), (e), (g)

(a) There are eight leaves: *m*, *j*, *n*, *o*, *e*, *f*, *g*, *l*.

(b) The first does not begin from the root nor reach a leaf. The second does not reach a leaf. The third omits node *d* from the path that goes from the root to leaf *n*.

(e) If you delete node *m* and the arrow to it, the resulting structure is still a tree. But if you delete either alone, it will not be a tree.

(g) In the first case no, because the new node *p* is not reachable by a path from the root *a*; nor can it be considered a new root for the enlarged structure because no other node is reachable from it. But in the second case, we would have a tree. In the third case, we would not, as *p* would not be reachable from *a*, nor would *p* be a new root for the enlarged structure since *a* is not reachable from it. In the fourth case, we do get a tree.

7.2 Rooted Trees

There are two ways of defining the concept of a rooted tree. One is *explicit*, giving a necessary and sufficient condition for a structure to be a tree. The other is *recursive*, defining first the smallest rooted tree and then larger ones out of smaller ones. As usual, the recursive approach tends to be better for the computer, though not always for the human. Each way delivers its particular insights, and problems are sometimes more easily solved using one than another. In particular, the recursive approach provides support for inductive proofs.

Whichever definition we use, a limiting-case question poses itself from the very beginning. Do we wish to allow an empty rooted tree, or require that all rooted trees have at least one element? Intuition suggests the latter; after all, if a tree is empty, then it has no root! But it turns out convenient to allow the empty rooted tree as well, particularly when we come to the theory of binary trees. So that is what we will do.

7.2.1 Explicit Definition

We begin with the explicit approach. To get it going, we need the notion of a *directed graph* or briefly *digraph*. This is simply any pair (*G*,*R*) where *G* is a set and *R* is a two-place relation over *G*. A rooted tree is a special kind of directed graph, but we can zoom in on it immediately, with no special knowledge of general graph theory. The only notion we need that is not already available from Chap. 2 is that of a *path*. We used the notion informally in the preceding section, but we need a precise definition. If (*G*,*R*) is a directed graph, then a path is defined to be any finite sequence $a_0, \ldots, a_n$ ($n \geq 1$) of elements of *G* (not necessarily distinct from each other) such that each pair $(a_i, a_{i+1}) \in R$.

Note that in the definition of a path, we require that $n \geq 1$; thus, *we do not count empty or singleton sequences as paths*. Of course, they could also be counted if we wished, but that would tend to complicate formulations down the line, where we usually want to exclude the empty and singleton ones.

Note also that we do not require that all the a_i are distinct; thus, (b, b) is a path when $(b, b) \in R$, and (a, b, a, b) is a path when both of (a, b),$(b, a) \in R$. Nevertheless, it will follow from the definition of a tree that paths like these (called loops and cycles) never occur in trees.

A (finite) *rooted tree* is defined to be any finite directed graph (G, R) such that either (a) G is empty or (b) there is an element $a \in G$ (called the *root of the tree*) such that (i) for every $x \in G$ with $a \neq x$ there is a unique path from a to x but (ii) there is no path from a to a. When a graph (G, R) is a tree, we usually write its carrier set G as T, so that the tree is called (T, R). In this section, we will usually say, for brevity, just *tree* instead of *rooted tree*; but when we get to unrooted trees in the last section of the chapter, we will have to be more explicit.

From the definition, it is easy to establish some basic properties of trees. Let (T, R) be any tree. Then:

- R is acyclic.
- If $T \neq \varnothing$, then the tree has a unique root.
- If $T \neq \varnothing$, then its root has no parent.
- Every element of T other than the root has exactly one parent.
- No two diverging paths ever meet.

We verify these properties using proof by contradiction each time. The proofs make frequent use of the condition 'unique' in the definition of a rooted tree:

- For *acyclicity*, recall from Chap. 2 that a relation R is said to be *acyclic* iff there is no path from any element x to itself, that is, no path $x_0, \ldots ,x_n$ $(n \geq 1)$ with $x_0 = x_n$. Suppose that a tree fails acyclicity; we get a contradiction. By the supposition, there is a path from some element x to itself. So the tree is not empty. But by the definition of a tree, there is also a unique path from the root a to x. Form the composite path made up of the path from the root to x followed by the one from x to itself. This is clearly another path from the root to x, and it is distinct from the first path because it is longer. This contradicts uniqueness of the path from a to x.
- For *uniqueness of the root*, suppose for reductio ad absurdum that a and a' are distinct elements of T such that for every $x \in G$, if $a \neq x$ (resp. $a' \neq x$), there is a path from a to x (resp. from a' to x), but there is no path from a to a (resp. from a' to a'). From the first supposition, there is a path from a to a', and by the second, there is a path from a' to a. Putting these two paths together gives us one from a to a, giving us a contradiction.
- To show that the root has *no parent*, suppose $a \in T$ is the root of the tree, and that it has a parent x. By the definition, there is a path from a to x, so adding the link from x to a gives us a path from a to a, contradicting the definition.
- Let b be any element of T distinct from the root a. By the definition of a tree, there is a path from the root a to b, and clearly, its last link proceeds from a parent of b,

so b has at least one parent. Now suppose that b has two distinct parents x and x'. Then there are paths from the root a to each of x and x', and compounding them with the additional links from x and x' to b gives us two distinct paths from a to b, contradicting the definition of a tree.

- Finally, if two *diverging paths* ever meet, then the node where they meet would have two distinct parents, contradicting the preceding property.

The *link-height* (alias *level*) of a node in a tree is defined recursively: that of the root is 0 and that of each of the children of a node is one greater than that of the node. The *node-height* of a node (alias just its *height* in many texts) is defined by the same recursion, except that the node-height of the root is set at 1. Thus, for every node x, $\textit{node-height}(x) = \textit{link-height}(x) + 1$. As trees are usually drawn upside down, the term 'depth' is often used instead of 'height'. Depending on the problem under consideration, either one of these two measures may be more convenient to use than the other. The *height* (whether in terms of nodes or links) of a non-empty tree may be defined to be the greatest height (node/link, respectively) of any of its nodes. The height of the empty tree is zero.

These notions are closely related to that of the length of a path in a tree. Formally, the *link-length* of a path $x_0, \ldots, x_n$ $(n \geq 1)$ is n, its node-length is $n+1$. Thus, the height of a node (in terms of nodes or links) equals to the length (node/link, respectively) of the unique path from the root to that node – except for the case of the root node since we are not counting one-element sequences as paths.

Exercise 7.2.1 (with partial solution)

(a) What is the (link/node)-height of the tree in Fig. 7.1?

(b) Consider a set with $n \geq 2$ elements. What is the greatest possible (link/node)-height of any tree over that set? And the least possible?

(c) For $n = 4$, draw rooted trees with n nodes, one tree for each possible height.

(d) We have shown that the relation R of a rooted tree (T, R) is acyclic. Show that this immediately implies its irreflexivity and asymmetry.

(e) In Chap. 4 Sect. 4.5.1, we gave a table of calls that need to be made when calculating $F(8)$ where F is the Fibonacci function. Rewrite this table as a tree, carrying its construction through to the elimination of all occurrences of F.

Solutions to (a), (b) and (d)

(a) The link-height of the tree in Fig. 7.1 is 4, and its node-height is 5.

(b) Let $n \geq 2$. The greatest possible (link/node)-height is obtained when the tree consists of a single branch, that is, is a sequence of nodes. The node-length of such a branch is n, and its link-length is $n-1$. The least possible measure is obtained when the tree consists of the root with $n-1$ children; the link-length is then 1, and its node-length is 2.

(d) Irreflexivity and asymmetry both follow immediately from acyclicity by taking the cases $n = 1$ and $n = 2$.

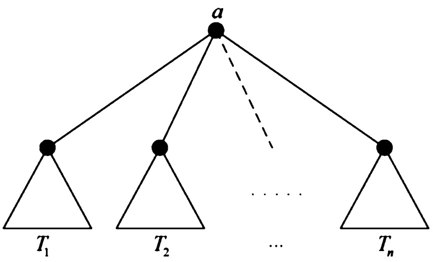

Fig. 7.2 Recursive definition of a tree, adding a new root

7.2.2 Recursive Definition

We pass now to *recursive definitions of a rooted tree*. There are two main ways of expressing them: the recursion step can bring in a new root or a new leaf. The most common presentation makes a *new root*, and we give it first.

- The basis of the definition stipulates that (∅, ∅), consisting of the empty set and empty relation over it, is a rooted tree (although it has no root).
- The recursion step tells us that whenever we have a finite non-empty collection of disjoint trees and a fresh item a, then we can form a new tree by taking a to be its root, linked to the roots of the given trees when the latter are non-empty. Formally, suppose (T_i, R_i) $(1 \leq i \leq n)$ are rooted trees. Let B be the set of their roots. Suppose that the T_i are pairwise disjoint and $a \notin \cup\{T_i\}_{i \leq n}$. Then, the structure (T,S) with $T = \cup\{T_i\}_{i \leq n} \cup \{a\}$ and $S = \cup\{R_i\}_{i \leq n} \cup \{(a,b)\colon b \in B\}_{i \leq n}$ is a tree with root a.

Note the conditions that the T_i must be disjoint and that a must be fresh. The recursion step may be illustrated by Fig. 7.2.

Here, the T_i are represented schematically by triangles. They are the *immediate* (*proper*) *subtrees* of the entire tree. They may be empty, singletons, or contain more than one node.

Exercise 7.2.2 (1) (with partial solution)

(a) Use the above recursive definition of a tree to construct five trees of respective node-heights 0, 1, 2, 3, 4, each one obtained from its predecessor by an application of the recursion step.

(b) How many rooted subtrees does a non-empty rooted tree with $n \geq 1$ nodes have?

Solution to (b)

(b) Clearly, there is a one-one correspondence between the nodes themselves and the non-empty subtrees with them as roots. This includes the given tree, which is a subtree of itself. Thus, there are n non-empty rooted subtrees. Counting also the empty subtree, this gives us $n + 1$ rooted subtrees.

We could also define rooted trees recursively by making *fresh leaves*. The basis is most conveniently formulated with two parts rather than one: The empty tree is a rooted tree, and so too is any singleton $T = \{a\}$, $R = \varnothing$ with a as root. The recursion step then says: Whenever we have a non-empty rooted tree and a fresh item x, then we can form a new tree by choosing a node b in it and linking b to x. In this case, the root of the new tree remains that of the old one.

This recursive definition corresponds more closely to most people's thoughts when they draw a tree on the page. One usually starts by placing a root, which retains that role throughout the construction, and then one adds new nodes by lengthening or dividing a branch. But the other recursive definition, by 'new root', lends itself more easily to inductive proofs and computations and so is more popular in computer science. We take it as our official definition.

Exercise 7.2.2 (2)

(a) Take the last tree that you constructed in the preceding exercise, and reconstruct it step by step following a 'new leaf' recursion.

(b) Use the 'new leaf' recursive definition to show that any tree with $n \geq 1$ nodes has $n - 1$ links.

(c) Do the same using the 'new root' recursive definition.

7.3 Working with Trees

Now that we have a definition for the notion of a rooted tree, we look at some situations in which trees can come up, and ways of annotating, or one might say, clothing or decorating them.

7.3.1 Trees Grow Everywhere

What are some typical examples of trees in everyday life? The first that springs to mind may be your family tree – but be careful! On the one hand, people have two parents, not one; and diverging branches of descendants can meet (legally as well as biologically) when cousins or more distant relations have children together. But if we make a family tree consisting of the male line only (or the female line only) of descendants of a given patriarch (resp. matriarch), then we do get a tree in the mathematical sense.

The most familiar example of a tree in today's world is given by the structure of folders and files that are available (or can be created) on your personal computer. Usually, these are constrained so as to form a tree. For those working as employees in an office, another familiar example of a tree is the staff organogram, in which the nodes are staff members (or rather their posts) and the links indicate the immediate subordinates of each node in the hierarchy. In a well-constructed pattern of responsibilities, this will often be a tree.

Trees also arise naturally whenever we investigate *grammatical structure*. In natural languages such as English, this reveals itself when we parse a sentence. In formal languages of computer science, mathematics and logic, parsing trees arise when we consider the structure of a formal expression. The syntactic analysis of a declaration in Prolog, for example, may take the form of a tree.

In logic, trees also arise in other ways. A proof or *derivation* can be represented as a tree, with the conclusion as root and the assumptions as leaves. And one popular way for checking whether a formula of propositional logic is a tautology proceeds by constructing what is called its *semantic decomposition tree*.

We will give some examples of parsing trees shortly. Proof trees and semantic decomposition trees will be described in the following chapters on logic. But first, we should introduce some refinements into our theory, introducing labelled and ordered trees.

Alice Box: Which Way Up?

Alice: Before going on, in Fig. 7.1 why did you draw the tree upside down, with the root at the top?

Hatter: Just a convention.

Alice: Is that all?

Hatter: Well, that's the short answer. Let's get on to labels.

A bit too short! In fact, we sometimes draw trees upside down, and sometimes, the right way up. The orientation depends, in part, on how we are constructing the tree. If we are thinking of it as built recursively from the root to the leaves by a 'new leaf' recursion, then it is natural to write the root at the top of the page and work downwards. As that is what humans most commonly do in small examples, diagrams are usually drawn with the root at the top. On the other hand, if we are building the tree by a 'new root' recursion, then it makes sense to put the leaves (which remain leaves throughout the construction) at the top of the page and work down to the root. Both of these modes of presentation show themselves in logic. In the following two chapters, formal derivations will be represented by trees with the leaves (assumptions) at the top and the root (conclusion) below, while semantic decomposition trees for testing validity will be upside down.

7.3.2 Labelled and Ordered Trees

In practice, we rarely work with naked trees. They are almost always clothed or decorated in some way – in the technical jargon, *labelled*. Indeed, in many applications, the identity of the nodes themselves is of little or no importance; they are thought of as just 'points'; what is important are the labels attached to them.

A *labelled tree* is a tree (T, R) accompanied by a function $\lambda: T \to L$ (λ and L are for 'label'). L can be any set, and the function λ can also be partial, that is, defined on a subset of T. Given a node x, $\lambda(x)$ indicates some object or property that is placed at that point in the tree. Thus, heuristically, the nodes are thought of as hooks on which to hang labels. We know that the hooks are there and that they are all distinct from each other, but usually what really interest us are the labels. In a diagram, we write the labels next to the nodes and may even omit to give names to the nodes, leaving them as dots on the page.

It is important to remember that the labelling function *need not be injective*. For example, in a tree of folders and files in your computer, we may put labels indicating the date of creation or of last modification. Different folders may thus have the same labels. In a graphic representation, dots at different positions in the tree may have the same label associated with them.

A given tree may come with several parallel systems for labelling its nodes. These could, of course, be amalgamated into one complex labelling function whose labels are ordered n-tuples of the component labels, but this is not always of any advantage. It is also sometimes useful to label the *links* rather than (or in addition to) the nodes. For example, we might want to represent some kind of distance, or cost of passage, between a parent and its children. Sometimes, the decision whether to place the labels on nodes or on links is just a matter of convention and convenience; in other cases, it may make a difference. Remember that trees are *tools for use*, as much as objects for study, and we have considerable freedom in adapting them to the needs of the task in hand.

An *ordered tree* is a tree in which the children of each node are put into a linear order and labelled with numbers $1, \ldots, n$ to record the ordering. In this case, the labelling function is a partial function on the tree (the root is not in its domain) into the positive integers. We give two examples of labelled, ordered trees.

Example 1 Consider the arithmetic expression $(5+(2{\cdot}x))-((y-3)+(2+x))$. Evidently, it is formed from two subexpressions, $5+(2{\cdot}x)$ and $(y-3)+(2+x)$ by inserting a sign for the operation of subtraction (and a pair of brackets). As this operation is not commutative, the order of application is important. We may represent the syntactic structure of the expression by a labelled tree, whose root is the whole expression, which has just two children, labelled by the two immediate subexpressions. Continuing this process until decomposition is no longer possible, we get the following labelled and ordered tree.

Alice Box: Drawing Trees

Alice: Why do we have to regard the expressions written alongside the nodes as *labels*? Why not just take them to be *the names* of the nodes? Does it make a difference?

Hatter: It does, because the labelling function need not be injective. In our example, we have two different nodes labelled by the numeral 2. We also have two different nodes labelled by the variable x. They have to be distinguished because they have different roles in the expression and so different places in the tree.

(continued)

(continued)

Alice: I see. But that leads me to another question. Where, in the figure, *are* the names of the nodes?

Hatter: We have not given them. In the diagram they are simply represented by dots. To avoid clutter, we name them only if we really need to.

Alice: One more question. If this is an ordered tree, then the children of each node must be given ordering labels. Where are they?

Hatter: We let the sheet of paper take care of that. In other words, when we draw a diagram for an ordered tree, we follow the convention of writing the children in the required order from left to right on the paper. This is another way of reducing clutter.

Alice: So this tree is different from its mirror image, in which left and right are reversed on the page?

Hatter: Yes, since it is an ordered tree. If it were meant as an unordered tree, they would be the same.

Actually, the tree in Fig. 7.3 can be written more economically. The interior nodes (i.e. the non-leaves) need not be labelled with entire subexpressions; we can just write their principal operations. This gives us the diagram in Fig. 7.4. It does not lose any information for we know that the operation acts on the two children in the order given. It can be regarded as an alternative presentation of the same tree.

Syntactic decomposition trees such as these are important for *evaluating* an expression, whether by hand or by computer. Suppose the variables x and y are

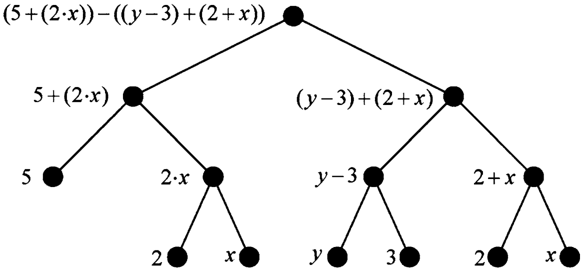

Fig. 7.3 Tree for syntactic structure of an arithmetic expression

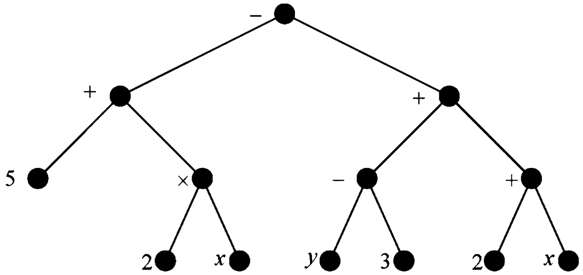

Fig. 7.4 Syntactic structure with economical labels

instantiated to 4 and 7, respectively. The most basic way of evaluating the expression as a whole is by 'climbing' the tree from leaves to root, always evaluating the children of a node before evaluating that node.

Example 2 Syntactic decomposition trees arise not only for arithmetic expressions but for any kind of 'algebraic' expression in mathematics or logic. Take, for example, the formula of propositional logic $(p \rightarrow \neg q) \rightarrow (\neg r \leftrightarrow (s \wedge \neg p))$. Clearly, the structure of this expression can be represented by a syntactic decomposition tree.

Exercise 7.3.2

(a) Evaluate the expression labelling the root of Fig. 7.3, for the values 4,7 for x,y, respectively, by writing the values in the diagram alongside the labels.

(b) (i) Draw the syntactic decomposition tree for the formula of Example 2, once with full labelling and again with abbreviated labelling. (ii) What are its node and link heights? (iii) Evaluate the formula for the values $p = 1$, $q = 1$, $r = 0$ and $s = 0$, writing the values as additional labels next to the nodes on the tree. *Reminder*: You will need to use the truth tables given in logic boxes in Chap. 1.

7.4 Interlude: Parenthesis-Free Notation

In our arithmetical and logical examples of syntactic decomposition trees, we used brackets. This is necessary to avoid ambiguity. Thus, the expression $5 + 2{\cdot}x$ is ambiguous unless brackets are put in (or a convention adopted, as indeed is usually done for multiplication, that it 'binds more strongly than' or 'has priority over' addition). Likewise, the logical expression $\neg r \leftrightarrow s \wedge \neg p$ is ambiguous unless parentheses are inserted or analogous conventions employed.

In practice, we frequently adopt priority conventions and also drop brackets whenever different syntactic structures have the same semantic content, as, for example, with $x + y + z$. Such conventions help prevent visual overload and are part of the endless battle against the tyranny of notation.

For a while in the early twentieth century, a system of dots was used by some logicians (notably Whitehead and Russell in their celebrated *Principia Mathematica*) to replace brackets. Thus, $(p \rightarrow \neg q) \rightarrow (\neg r \leftrightarrow (s \wedge \neg p))$ was written as $p \rightarrow \neg q. \rightarrow : \neg r \leftrightarrow .s \wedge \neg p$ where the dot after the q indicates a right bracket (the left one being omitted because it is at the beginning of the formula), the dot before the s marks a left bracket (its right partner omitted because it is at the end) and the two-dot colon marks another left bracket. The number of dots indicates the 'level' of the bracket. However, the system did not catch on and died out.

Is there any systematic way of doing without brackets altogether, without special priority conventions or alternative devices like dots? There is, and it was first noted by the philosopher/logician Łukasiewicz early in the twentieth century. Instead of writing the operation *between* its arguments, just write it *before* them! Then, brackets become redundant. For example, the arithmetical expression $(5 + (2{\cdot}x)) - ((y - 3) + (2 + x))$ is written as $- + 5{\cdot}2x + - y3 + 2x$.

To decipher this last sequence, construct its (unique!) decomposition tree from the leaves to the root. The leaves will be labelled by the constants and variables. $+2x$ will label the parent of nodes labelled by 2 and x; $-y3$ will label the parent of nodes labelled by 3 and y, etc.

This is known as *Polish notation*, after the nationality of its inventor. There is also *reverse Polish* notation, where the operation is written after its arguments, and which likewise makes bracketing redundant. The ordinary way of writing expressions is usually called *infix* notation, as the operation is written in between the arguments. The terms *prefix* and *postfix* are often used for Polish and reverse Polish notation, respectively.

Exercise 7.4 (with partial solution)

(a) Draw all possible decomposition trees for each of the ambiguous expressions $5+2{\cdot}x$ and $\neg r{\leftrightarrow}s{\wedge}\neg p$, writing the associated bracketed expression next to each tree.

(b) Draw the whole syntactic decomposition tree for the Polish expression $-+5{\cdot}2x+-y3+2x$. Compare it with the tree for $(5+(2{\cdot}x))-((y-3)+(2+x))$.

(c) Put the propositional formula $(p\rightarrow\neg q)\rightarrow(\neg r{\leftrightarrow}(s{\wedge}\neg p))$ in both Polish (prefix) and reverse Polish (postfix) notation.

(d) Insert a few redundant brackets into the outputs just obtained to make them friendlier to non-Polish readers.

Solutions to (c) and (d)

(c) In prefix notation, it is $\rightarrow\rightarrow p\neg q{\leftrightarrow}\neg r{\wedge}s\neg p$. In postfix notation, it is $pq\neg\rightarrow r\neg sp\neg{\wedge}{\leftrightarrow}\rightarrow$.

(d) $\rightarrow(\rightarrow p\neg q)({\leftrightarrow}(\neg r({\wedge}s\neg p))$ and $(pq\neg\rightarrow)(r\neg(sp\neg{\wedge}){\leftrightarrow})\rightarrow$.

7.5 Binary Trees

Computer science and logic both love binary concepts. In the case of logic, they appear at its very foundations: classical logic is two-valued with truth tables for the logical connectives defined using functions into the two-element set $\{0,1\}$. In computer science, binary concepts ultimately derive from a basic practical fact: a switch can be on or off. This means that an essential role is played by bits, and binary notions are propagated through the whole superstructure. Of all kinds of tree, the most intensively employed in computing is the binary one. So we look at it in more detail.

A *binary tree* is a rooted tree (possibly empty) in which each node has at most two children, equipped with an ordering that labels each child in the tree with a tag *left* or *right*. A peculiarity about the ordering required for binary trees is that even when a node has only one child, that child is also labelled ‘left’ or ‘right’. When drawing the tree on paper, we need not write these two labels explicitly; we can

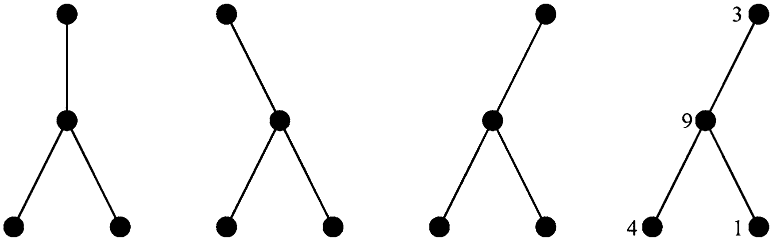

Fig. 7.5 Which are binary trees?

understand them as given by the position of nodes on the page. For this reason, in a binary tree we will never draw an only child vertically below its parent.

To illustrate, of the four trees in Fig. 7.5, the first is not a binary tree since the child of the root has not been given (explicitly or by orientation on the page) a left or right label. The remaining three are binary trees. The second and third are distinct binary trees since the second gives a right label to the child of the root, while the third gives it a left label. The third and the fourth are the same when considered simply as binary trees, but they differ in that the fourth has additional (numerical) labels.

What are binary trees used for? They, and more specifically a kind of binary tree known as a *binary search tree*, are convenient structures on which to record and manipulate data. Surprisingly, they do this more efficiently than linear sequences. There are very efficient algorithms for working with them, in particular, for *searching*, *deleting* and *inserting* items recorded on them. There are also good algorithms for *converting* lists into binary search trees and back again. Moreover, any finite tree can be reconfigured as a binary search tree in a natural way.

What is a binary *search* tree? Suppose we have a binary tree and a labelling function λ: $T \to L$ into a set L that is linearly ordered by a relation $<$. The set L may consist of numbers, but not necessarily so; the ordering may, for example, be a lexicographic ordering of expressions of a formal language. For simplicity, we will consider the case that the labelling is a full function (i.e. has the entire set T as domain) and is injective. In this situation, we may speak of a linear ordering $<$ of the nodes themselves, determined by the ordering of the labels. Generalization to non-injective labelling functions is not difficult, but a little more complex.

A binary tree so equipped is called *a binary search tree* (modulo the labelling function λ and linear relation $<$) iff each node x is greater than every node in the left subtree of x and is less than every node in the right subtree of x.

Exercise 7.5 (1) (with solution) Which of the four binary trees in Fig. 7.6 are binary search trees, where the order is the usual order between integers?

Solution The four trees differ only in their labelling systems. Tree (a) is not a binary search tree for three distinct reasons: $2 < 4$, $10 > 6$, $10 > 9$. Any one of these would

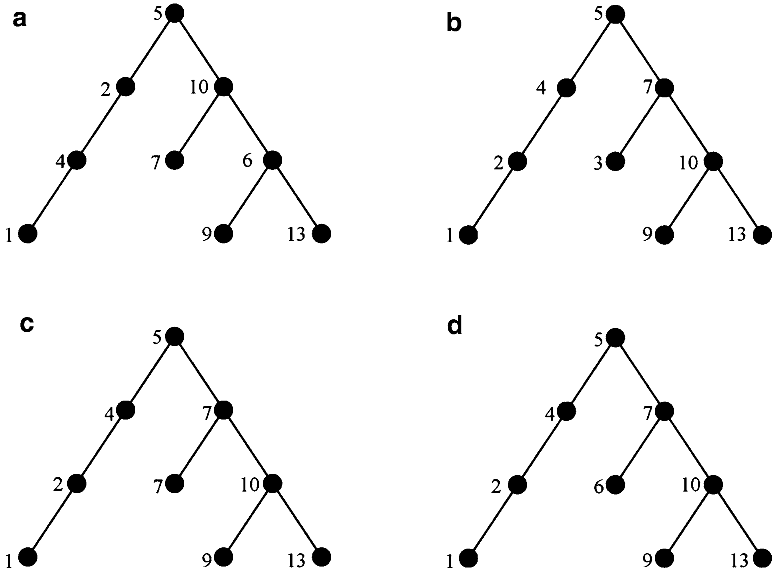

Fig. 7.6 Which are binary search trees?

be enough to disqualify it. Tree (b) isn't either because $5 > 3$. Nor is (c) a binary search tree in our sense of the term since the labelling is not injective. Only (d) is a binary search tree.

The exercise brings out the fact that in the definition of a binary search tree, we don't merely require that each node is greater than its left child and less than its right child. We require much more: Each node must be greater than *all nodes* in its left subtree and less than *all nodes* in its right subtree.

Suppose we are given a binary search tree T. How can we search it to find out whether it contains an item x? If the tree is empty, then evidently it does not contain x. If the tree is not empty, the following recursive procedure will do the job:
Compare x with the root a of T.

```
If $x = a$, then print yes.
If $x < a$ then
    If $a$ has no left child, then print no.
    If $a$ does have a left child, apply the recursion to that child.
If $x > a$ then
    If $a$ has no right child, then print no.
    If $a$ does have a right child, apply the recursion to that child.
```

Exercise 7.5 (2) (with partial solution)

(a) Execute the algorithm manually to search for 6 in the last tree of Fig. 7.6.
(b) Do the same to search for 8.
(c) For readers familiar with pseudo-code, express the above algorithm using it.

Solutions to (a) and (b)

(a) We compare 6 with the root 5. Since $6 > 5$, we compare 6 with the right child 7 of 5. Since $6 < 7$, we compare 6 with the left child 6 of 7. Since $6 = 6$, we print *yes*.

(b) We compare 8 with the root 5. Since $8 > 5$, we compare 8 with the right child 7 of 5. Since $8 > 7$, we compare 8 with the right child 10 of 7. Since $8 < 10$, we compare 8 with the left child 9 of 10. Since $8 < 9$ and 9 has no left child, we print *no*.

The interesting thing about this algorithm is its efficiency. The height of the tree will never be more than the length of the corresponding list of items, and usually very much less. At one end of the spectrum, when no node has two children, the node-height of the tree and the length of the list will be the same. At the other end of the spectrum, if every node other than the leaves has two children and all branches are the same length, then the non-empty tree will have $\Sigma\{2^i\colon 0 \leq i < h\} = 2^h - 1$ nodes where h is the node-height of the tree. A search using the algorithm down a branch of length h will thus usually be very much quicker, and never slower, than one plodding through a list of all $2^h - 1$ nodes.

The algorithm described above can be refined. There are also algorithms to insert nodes into a binary search tree and to delete nodes from it, in each case ensuring that the resulting structure is still a binary search tree. The one for insertion is quite simple, as it always inserts the new node as a leaf. That for deletion is rather more complex, as it deletes from anywhere in the tree, leaves or interior nodes. We sketch the insertion algorithm, omitting the deletion one.

To *insert* an item x, we begin by searching for it using the search algorithm. If we find x, we do nothing since it is already there. If we don't find it, we go to the node y where we learned that x is absent from the tree (it should have been in a left resp. right subtree of y, but y has no left resp. right child). We add x as a child of y with an appropriate left or right label.

Exercise 7.5 (3) (with solution) Consider again the binary search tree in Fig. 7.6. What is the tree that results from inserting the node 8 by means of the algorithm sketched above?

Solution Searching for 8 in the tree, we reach node 9 and note that 8 should be in its left subtree, but that 9 is a leaf. So we add 8 as left child of 9.

We can also iterate the insertion algorithm to construct a binary search tree from a list. Take, for example, the list l of nine words: *this, is, how, to, construct, a, binary, search, tree* and that our ordering relation $<$ for the tree construction is the lexicographic one (dictionary order). We begin with the empty tree and search in it for the first item in the list, the word 'this'. Evidently, it is not there, so we put it as root. We then take the next item in the list, the word 'is' and search for it in the tree so far constructed. We don't find it, but note that 'is' $<$ 'this', and so we add it as a left child of 'this'. We continue in this way until we have completed the tree.

Exercise 7.5 (4)

(a) Complete the construction of the binary search tree from the word-list $l =$ (this, is, how, to, construct, a, binary, search, tree).

(b) For readers familiar with pseudo-code, express the above algorithm for constructing a binary search tree using it.

Evidently, when constructed by this algorithm, the shape of the tree depends on the order in which the items are listed. For example, if in the above example the same words were presented in the list $l' =$ (a, binary, construct, how, is, search, this, to, tree), the binary search tree would be a chain going down diagonally to the right, and just as high as the list is long!

Clearly, we would like to keep the height of the tree as low as possible. The optimum is to have a binary tree with two children for every node other than the leaves, and all branches of the same length. This is possible only when the list to be encoded has node-length $2^h - 1$ for some $h \geq 0$. Nevertheless, for any value of $h \geq 0$, it is possible to construct a complete binary search tree in which no branch is more than one node longer than any other and where the longer branches are all to the left of the shorter ones. Moreover, there are good algorithms for carrying out this construction, although we will not describe them here.

7.6 Unrooted Trees

We now go back to the rooted tree at the beginning of this chapter in Fig. 7.1 and play around with it. Imagine that instead of the figure made of dots and lines on paper, we have physical model, made of beads connected with pieces of string or wire. We can then pick up the model, push the root a down, and bring up b, say, so that it becomes the root. Or we can rearrange the beads on a table so that the model loses its tree-like shape and looks more like a road map in which none of the nodes seems to have a special place. When we do this, we are treating the model as an unrooted tree.

7.6.1 Definition

Formally, an *unrooted tree* (alias *undirected tree*) may be defined as any structure (T,S) that can be formed out of a rooted tree (T,R) by taking S to be the symmetric closure of R.

Recall the definition of the symmetric closure S of R: it is the least symmetric relation (i.e. intersection of all symmetric relations) that includes R. Equivalently, and more simply, $S = R \cup R^{-1}$. Diagrammatically, it is the relation formed by deleting the arrowheads from the diagram of R (if there were any) and omitting any convention for reading a direction into the links.

Exercise 7.6.1 (1)

(a) Take the rooted tree in Fig. 7.1 and redraw it as an unrooted tree with the nodes scattered haphazardly on the page.
(b) Reroot the unrooted tree that you obtained by drawing it with node m as root.
(c) Check the equivalence of the two definitions of symmetric closure.

In the context of unrooted trees, terminology changes (yes, yet again). Nodes in unrooted trees are usually called *vertices*. More important, as well as speaking of links, which are *ordered* pairs (x,y), that is, elements of the relation, we also need to speak of *edges* (in some texts, *arcs*), identifying them with the *unordered* pairs $\{x,y\}$ such that both (x,y) and $(y,x) \in S$.

Exercise 7.6.1 (2) We observed in Exercise 7.2.2 (2) that a rooted tree (T,R) with $n \geq 1$ nodes has exactly $n-1$ links. Use this to show that an unrooted tree (T,S) with $n \geq 1$ vertices has exactly $n-1$ edges.

Alice Box: Unrooted Trees

Alice: That's a nice, simple definition – provided we are coming to the subject via rooted trees. But what if I wanted to study unrooted trees before the rooted ones? Could I define them in a direct way?

Hatter: No problem. In fact that is the way most textbooks do it. We will see how shortly, after noting some properties emerging from our present definition.

It appears that the mathematical concept of an unrooted tree was first articulated in the middle of the nineteenth century and popularized by the mathematician Arthur Cayley discussing problems of chemistry in 1857. Graphs were already being used to represent the structure of molecules, with vertices (nodes) representing atoms and (undirected) edges representing bonds. Cayley noticed that the saturated hydrocarbons – that is, the isomers of compounds of the form C_nH_{2n+2} – have a special structure: they are all what we now call unrooted trees. For rooted trees, intuitive uses go back much further. For example, in the third century AD, a neoplatonist philosopher commentating on the logic of Aristotle introduced what became known as the 'tree of Porphyry'.

7.6.2 Properties

Let (T,S) be an unrooted tree, with S the symmetric closure of the link relation R of a rooted tree (T,R). Then, it is easy to show that S is *connected* (over T), that is, for any two distinct elements x, y of T, there is an S-path from x to y, that is, a finite sequence $a_0, \ldots, a_n$ $(n \geq 1)$ of elements of T such that $x = a_0$, $y = a_n$, and each pair

$(a_i,a_{i+1}) \in S$. The proof is very straightforward. Since (T,R) is a rooted tree, it has a root a. We distinguish three cases. In the case that $a = x$, by the definition of a rooted tree we have a R-path from x to y, and this is an S-path. In the case that $a = y$, we have an R-path from y to x and thus, running it in reverse (which is legitimate since S is symmetric), we have an S-path from x to y. Finally, if $a \neq x$ and $a \neq y$, we know that there are R-paths from a to each of x, y considered separately. Reversing the path from a to x, we have an S-path from x to a; and continuing it from a to y (i.e. take the composition of the two paths), we end up with one from x to y.

Exercise 7.6.2 (1) Draw sample tree diagrams to illustrate the three cases of this proof and the constructions made.

Less obvious is the fact that S is a *minimal* connected relation over T. In other words, no proper subrelation of S is connected over T; that is, for any pair $(x, y) \in S$, the relation $S' = S \backslash \{(x,y)\}$ is *not* connected over T. To show this, note that either $(x, y) \in R$ or $(y,x) \in R$. Consider the former case; the latter is similar. We claim that there is no S'-path from a to y where a is the root of the rooted tree (T,R). Suppose for reductio ad absurdum that there is such a S'-path, that is, a finite sequence $a_0, \ldots ,a_n$ $(n \geq 1)$ of elements of T such that $a_0 = a$, $a_n = y$, and each pair $(a_i,a_{i+1}) \in S'$. We may assume without loss of generality that this is a shortest such path so that in particular never $a_i = a_{i+2}$. Since $(a_{n-1},a_n) \in S'$, we have either $(a_{n-1},a_n) \in R$ or conversely $(a_n,a_{n-1}) \in R$. But the former is impossible. Reason: Since $a_n = y$ and no node can have two R-parents, we must have $a_{n-1} = x$ so that (x,y) is the last link in the S'-path, contradicting the fact that $(x,y) \notin S'$. Thus, the latter alternative $(a_n,a_{n-1}) \in R$ must hold. But then by induction from n to 1, we must have each $(a_{i+1},a_i) \in R$, for otherwise we would have an a_i with two distinct R-parents. This gives us an R-path from y to the root a, which is impossible by the definition of a rooted tree.

This is a tricky little proof, partly because it uses one proof by contradiction within another, and partly because it also uses induction and wlog. Students are advised to read it carefully, at least twice.

Exercise 7.6.2 (2)

(a) Draw a diagram to illustrate the above argument.

(b) 'We may assume without loss of generality that this is a shortest such path'. What principle lies behind that remark? *Hint*: Reread the discussion of *wlog* proofs at the end of Sect. 4.7.2 of Chap. 4.

(c) 'For otherwise, we would have an a_i with two distinct R-parents'. Explain in more detail why.

The second key property of unrooted trees is that they have no cycles that are in a certain sense 'simple'. A *cycle* is a path whose first and last items are the same. So, unpacking the definition of a path, a cycle of an unrooted tree (T,S) is a sequence $a_0, \ldots ,a_n$ $(n \geq 1)$ of elements of T with each $(a_i,a_{i+1}) \in S$ and $a_n = a_0$. A *simple cycle* is one with no repeated edges, that is, for no $i < j < n$, do we have $\{a_i,a_{i+1}\} = \{a_j,a_{j+1}\}$. Expressed in terms of the relation R of the underlying rooted tree, the sequence $a_0, \ldots ,a_n = a_0$ never repeats or reverses an R-link. It may, however, repeat vertices.

For example, the cycles a,b,c,b,a and a,b,c,e,c,b,a are not simple, since each contains both of (b,c) and (c,b). The cycle a,b,c,e,b,c,a is not simple either, as it repeats the link (b,c). On the other hand, the cycle a,b,c,d,e,c,a is simple, despite the repetition of vertex c.

Exercise 7.6.2 (3)

(a) Take the first-mentioned cycle a,b,c,b,a and add c in second place, that is, after the initial a. Is it simple?

(b) Take the last-mentioned cycle a,b,c,d,e,c,a and drop the node b. Is it simple? What if we drop e?

Clearly, any unrooted tree with more than one vertex is full of cycles: whenever $(x,y) \in S$, then by symmetry $(y,x) \in S$ giving us the cycle x,y,x. The interesting point is that it never has any *simple* cycles. It can be a bit tricky for students to get this point clear in their minds, for its formulation uses two negations: there are *no* cycles that contain *no* repeated edges. To get a better handle on it, we can express it positively: *every cycle in* (T,S) *has at least one repeated edge*.

Alice Box: No Simple Cycles

Alice: No simple cycles: in other words, acyclic?

Hatter: Not so fast! Acyclicity, as we have defined it in Sect. 7.2 means that there are no cycles at all, that is, no paths $a_0, \ldots ,a_n$ $(n \geq 1)$ with each $(a_i,a_{i+1}) \in S$ and $a_n = a_0$. Here we are requiring less; we require only that there are no *simple* cycles. And in the present context, that is all we can ask for. As remarked in the text, when S is symmetric it will always contain non-simple cycles, by back-tracking.

Alice: But I think I have seen some other books using the term 'acyclic' in this way.

Hatter: Indeed, some do but then, of course, they need another term for not having any cycles at all! It would be nice to have special name for 'no simple cycles', say 'nocyclic', but let's not make a messy terminological landscape even more confusing.

Alice: I hope I don't get mixed up . . .

Hatter: If you do, go to the positive formulation above: 'no simple cycles' means 'every cycle has a repeated edge'.

To prove the claim, suppose for reductio ad absurdum that (T,S) is an unrooted tree containing a simple S-cycle $a_0, \ldots ,a_n = a_0$. We may assume without loss of generality that this is a shortest one so that in particular $a_i \neq a_j$ for distinct i,j except for the end points $a_0 = a_n$. We distinguish three cases and find a contradiction in each.

Case 1. Suppose $(a_0,a_1) \in R$. Then for all i with $0 \leq i < n$, we have $(a_i,a_{i+1}) \in R$, for otherwise some a_{i+1} would have two distinct R-parents a_i and a_{i+2}, which is impossible. Thus, the S-cycle $a_0, \ldots ,a_n = a_0$ is in fact an R-cycle, which we know from earlier in this chapter is impossible for the link relation R of a rooted tree.
Case 2. Suppose $(a_n,a_{n-1}) \in R$. A similar argument shows that this case is also impossible.
Case 3. Neither of the first two cases holds. Then, $(a_1 a_0) \in R$ and $(a_{n-1},a_n) \in R$. Since $a_0 = a_n$ and no node of a rooted tree can have two distinct parents, this implies that $a_1 = a_{n-1}$. But that gives us a shorter S-cycle $a_1, \ldots ,a_{n-1} = a_1$ which must also be simple, again giving us a contradiction.

Exercise 7.6.2 (4)
(a) Draw circular diagrams to illustrate the three cases of this proof.
(b) In Case 1, check fully the claim ‘which is impossible’.

Indeed, we can go further: when (T,S) is an unrooted tree then S is *maximally* without simple cycles. What does this mean? Let (T,S) be an unrooted tree, derived from a rooted tree (T,R). Let x,y be distinct elements of T with $(x,y) \notin S$. Then, the structure (T, S') where $S' = S \cup \{(x,y), (y,x)\}$ contains a simple cycle.

We will not give a full proof of this, just sketching the underlying construction. In the principal case that x, y are both distinct from the root a of (T,R), there are unique R-paths from a to each of them. Take the last node b that is common to these two R-paths and form an S-path from x up to b (using R^{-1}) and then down to y (using R). With (y,x) also in S', we thus get an S'-cycle from x to x, and it is not difficult to check that this cycle must be simple.

Exercise 7.6.2 (5)
(a) Draw a tree diagram to illustrate the above proof sketch.
(b) Fill in the details of the above proof by (i) covering the cases that either x or y is the root and (ii) in the case that neither is the root, showing that the constructed cycle is simple, as claimed in the text.

Summarizing what we have done so far in this section, we see that unrooted trees (T,S) have a symmetric relation S, and when T is non-empty, satisfy the following six conditions:

- S is the symmetric closure of the link relation of some rooted tree (T,R) (by definition).
- S minimally connects T.
- S connects T and has $n-1$ edges, where $n = \#(T)$.
- S connects T and has no simple cycles.
- S is maximally without simple cycles.
- S has no simple cycles and has $n-1$ edges, where $n = \#(T)$.

In fact, it turns out (although we do not prove it here) that these six conditions are mutually equivalent for any non-empty set T and symmetric relation $S \subseteq T^2$, so that any one of them could serve for a characterization of unrooted trees. The first bulleted condition defined unrooted trees out of rooted ones; each of the remaining ones answers a question that Alice raised: how can we define unrooted trees directly, without reference to rooted ones?

The equivalence of the six conditions should not be thought of as just a bit of technical wizardry; it has a deep conceptual content. In particular, the first three bullet points tell us that if we connect a set T with a symmetric relation in a most economical way possible then, whether we think of economy in terms of sets of edges (ordered by inclusion) or number of edges, we get an unrooted tree; and conversely.

Alice Box: Explicit Versus Recursive Definitions of Unrooted Trees

Alice: Thanks, this does answer my query. But one question leads to another. Could we also define unrooted trees *recursively*, as we did for rooted ones?

Hatter: Of course. Indeed, we need only take the recursive definition for rooted trees and tweak it a little. You might like to do it as an exercise.

Exercise 7.6.2 (6)

(a) Give a simple example of a structure (T,S) where T is a set with $\#(T) = n$ and S is a symmetric relation over T with $n - 1$ edges but is not an unrooted tree. Give another example in which S is moreover irreflexive.

(b) Without proof, using only your intuition, try making a suitable recursive definition of unrooted trees for Alice.

Solution to (a)

The simplest example has just two vertices, with an edge from one of the vertices to itself. If we also require irreflexivity, the simplest example has four vertices, with three connected in a simple cycle and the fourth not connected to anything.

7.6.3 Spanning Trees

The second of the above bulleted conditions gives rise to an important practical problem. A symmetric relation S that minimally connects a set T (thus satisfying the condition) is known as a *spanning tree* for T. Such minimality is valuable, as it helps reduce computation; how can we obtain it algorithmically?

Suppose we are given a set A connected by a symmetric relation S. There may be many S-paths between vertices and many simple cycles. In the limit, every element of A may be related to every one including itself, giving $n(n + 1)/2$ edges where n is the number of elements of A; in the case of an irreflexive relation, every element of A may be related to every other one, flooding us with $n(n - 1)/2$ edges. That is a lot of information, most of which may be redundant for our purposes. So the question arises: Is there an algorithm which, given a set A connected by a symmetric relation S over it, finds a minimal symmetric subrelation $S' \subseteq S$ that still connects A? In other words, is there an algorithm to find a spanning tree for A?

Clearly there is a 'top-down' procedure that does the job. We take the given relation S connecting A, and take off edges one by one in such a way as to leave A connected. To begin with, we can get rid of all the edges connecting a vertex with itself, and then we start to carefully remove edges between distinct vertices without damaging connectivity. When we get to a point where it is no longer possible to delete any edge without de-connecting A (which will happen when we have got down to $n-1$ edges, where n is the number of vertices), we stop. That leaves us with a *minimal* symmetric relation $S' \subseteq S$ connecting A – which is what we are looking for.

But there is also a 'bottom-up' procedure that accomplishes the task. We begin with the empty set of edges and add in edges from S one by one in such a way as never to create a simple cycle. When we get to a point where it is no longer possible to add an edge from S without creating a simple cycle (which will happen when we have got up to $n-1$ edges, where n is the number of vertices), we stop. That leaves us with a *maximal* relation $S' \subseteq S$ without simple cycles. By two of the equivalent conditions bulleted above, S' will also be a *minimal* symmetric relation connecting A – which is what we are looking for.

In general, the 'bottom-up' algorithm is much more efficient than the 'top-down' one for finding a spanning tree, since it is less costly computationally to check whether a given relation creates a simple cycle than to check whether a given set is connected by a relation.

Exercise 7.6.3 For readers familiar with pseudo-code, express each of the algorithms above using it.

In many practical situations, one needs to go further and consider symmetric relations whose edges have numerical weights attached to them. These weights may represent the distance between vertices, or the time or cost involved in passing from one to the other. In this context, we often want to do something more subtle than minimize the set of edges in a relation connecting the domain; we may wish to minimize total cost, that is, minimize *the sum of their weights*. In the usual terminology, we want to find a *minimal spanning tree* for A.

The 'bottom-up' algorithm that we have described for finding a spanning tree can be refined to solve this problem. However, doing so would take us beyond the limits of the present book. The references at the end of the chapter follow the subject further.

End-of-Chapter Exercises

Exercise 7.1: Properties of rooted trees

(a) Show that the relation R of a rooted tree (T,R) is always intransitive, in the sense defined in the chapter on relations. *Hint*: Use the explicit definition of a rooted tree and argue by using proof by contradiction.

(b) Consider the ‘new root’ definition of a rooted tree. Show that the tree constructed by an application of the recursion step has height (whether node-height or link-height) one larger than the maximal height (in the same sense) of its immediate subtrees.

(c) Do the same with the ‘new leaf’ definition of a rooted tree.

(d) In the text, we defined the notion of a subtree of a tree, but we did not verify that when so defined, it is always itself a tree. Check it.

Exercise 7.2: Definitions of rooted trees* Show the equivalence of the three definitions of a rooted tree – explicit, recursive by ‘new root’ and recursive by ‘new leaf’. *Remark*: This is a challenging exercise, and its answer will have to consist of at least three parts, thus taking some time. Perhaps the simplest strategy is to establish a cycle of implications.

Exercise 7.3: Labelled trees

(a) Construct the syntactic decomposition tree of the arithmetic expression $(8-(7+x))+(y^3+(x-5))$.

(b) Construct the syntactic decomposition tree of the arithmetic expression $-(-(8-(7+x)))$.

(c) What would be special about the shape of syntactic decomposition tree of an arithmetic expression formed from two-place arithmetic operations alone?

(d) Draw the mirror image of the tree in Fig. 7.3, with its labels. Write down the arithmetic expression to which it corresponds in both standard and Polish notation.

Exercise 7.4: Binary search trees

(a) Using the construction algorithm given in the text, construct a binary search tree for the list of letters (g, c, b, m, i, h, o, a) where the ordering relation $<$ is the usual alphabetical one.

(b) In the tree you have constructed, trace the steps in a search for the letter i using the search algorithm in the text.

(c) To the tree you have constructed, add the letter f using the insertion algorithm in the text.

Exercise 7.5 Unrooted trees*

Consider the *complete* (irreflexive) graph (G,R) with five vertices, that is, the graph in which there is an edge between each vertex and every other vertex.

(a) Construct a spanning tree for this graph by the top-down method, showing by successive diagrams each step.

(b) Construct a different spanning tree by the bottom-up method, again showing your steps by successive diagrams.

Selected Reading

Almost all introductory texts of discrete mathematics have a chapter on trees, usually preceded by one on the more general theory of graphs. More often than not, they treat unrooted trees before rooted ones. Of the two books below, both go into considerably more detail than we have done. The first starts from unrooted trees, while the approach of the second is closer to that of this chapter.

Johnsonbaugh R (2009) Discrete mathematics, 7th edn. Pearson, Upper Saddle River, chapter 9

Kolman B et al (2006) Discrete mathematical structures, 6th edn. Pearson, Upper Saddle River, chapter 7

Yea and Nay: Propositional Logic

8

Abstract

We have been using logic on every page of this book – in every proof, verification and informal justification. In the first four chapters, we inserted some 'logic boxes'; they gave just enough to be able to follow what was being done. Now we gather the material of these boxes together and develop their principles. Logic thus emerges as both a tool for reasoning and an object for study.

We begin by explaining different ways of approaching the subject and situating the kind of logic that we will be concerned with, then zooming into a detailed account of *classical propositional* logic. The basic topics there will be the *truth-functional connectives*, the family of concepts around *tautological implication*, the availability of *normal forms* and *unique minimalities* for formulae and the use of *semantic decomposition trees* as a shortcut method for testing logical status.

8.1 What Is Logic?

What we will be studying in this chapter is only one part of logic, but a very basic part. In a general sense, logic may be seen as the study of *reasoning* or, in more fashionable terms, *belief management*. It concerns ways in which agents (human or other) may develop and shape their beliefs, by inference, organization and change:

- *Inference* is the process by which a proposition is accepted on the basis of others, in other words, is considered as justified by them.
- *Belief organization* is the business of getting whatever we accept into an easily stocked, communicable and exploitable pattern. It is important in computer science for database management but also in mathematics, where it traditionally takes the form of *axiomatization*.
- *Belief change* takes place when we decide to abandon something that we had hitherto accepted, a process known to logicians as *contraction*. It can also take the form of *revision*, where we accept something that we previously ignored or even rejected, at the same time making sufficient contractions to maintain consistency of the whole. A closely related form of belief change, of particular

D. Makinson, *Sets, Logic and Maths for Computing*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-2500-6_8,

interest to computer science, is *update*, where we modify our records to keep up with changes that are taking place in the world. Evidently, this is very close to revision, but there are also subtle differences.

Of all these processes of belief management, the most basic is *inference*. It reappears as an ingredient within all the others, just as sets reappear in relations, functions, counting, probability and trees. For this reason, introductory logic books usually restrict themselves to inference, leaving other concerns for more advanced work. Even within that sphere, it is customary to look at only one kind of inference, admittedly the most fundamental one, underlying others: *deductive* inference. Reluctantly, we must do the same.

That this is a real limitation becomes apparent if one reflects on the kind of reasoning carried out in daily life, outside the study and without the aid of pen and paper. Often, it is not so much inference as articulating, marshalling and comparing information and points of view. Even when it takes the form of inference, it is seldom fully deductive. The conclusions one reaches are offered as plausible, reasonable, probable or convincing given the assumptions made, but they are rarely if ever absolutely certain relative to them. There is the possibility, perhaps remote, that *even if* the assumptions are true, the conclusion *might* still be false.

But within mathematics and most of computer science, the game is quite different. There we use only fully deductive inference, rendering the conclusions certain given the assumptions made. This is not to say that there is any certainty in the assumptions themselves, nor for that matter in the conclusions. It lies in the *link* between them: it is *impossible for the premises to be true without the conclusion also being so*.

That is the only kind of reasoning we will be studying in this chapter. It provides a basis that is needed before trying to tackle uncertain inference, whether that is expressed in qualitative terms or probabilistically, and before analyzing other kinds of belief management. Its study goes back 2,000 years; in it modern form, it began to take shape in the middle of the nineteenth century.

8.2 Truth-Functional Connectives

We begin by looking at some of the ways in which statements, often also called *propositions*, may be combined. The simplest ways of doing this are by means of the truth-functional connectives ‘not’, ‘and’, ‘or’, ‘if’, ‘iff’, etc. We have already introduced each of these one in a corresponding logic box in the first few chapters, and you are strongly advised to flip back to those boxes to get up to speed, for we will not repeat everything said there. However, for easy reference, we recall the truth tables, using α, β, . . . , for arbitrary propositions. The one-place connective ‘not’ has the following Table 8.1:

Table 8.1 Truth table for negation

α	$\neg\alpha$
1	0
0	1

Table 8.2 Truth table for familiar two-place connectives

α	β	$\alpha\wedge\beta$	$\alpha\vee\beta$	$\alpha\to\beta$	$\alpha\leftrightarrow\beta$
1	1	1	1	1	1
1	0	0	1	0	0
0	1	0	1	1	0
0	0	0	0	1	1

Here, 1 is for truth, and 0 is for falsehood. The two-place connectives 'and', 'or', 'if' and 'iff' are grouped together in Table 8.2.

A basic assumption made in these tables is the *principle of bivalence*: every proposition is either true or false, but not both. In other words, the truth-values of compound propositions may be represented by functions with domain taken to be the set of all propositions of the language under consideration (about which more shortly), into the two-element set {1,0}.

Alice Box: The Principle of Bivalence

Alice: What happens if we relax the principle of bivalence?

Hatter: There are two main ways of going about it. One is to allow that there may be truth-values other than truth and falsehood. This gives rise to the study of what is called *many-valued logic*.

Alice: And the other?

Hatter: We can allow that the values may not be exclusive, so that a proposition may be both true and false. That can also be accommodated within the first approach by using new values to represent *subsets* of the old values. For example, we might use four values (the two old ones and two new ones) to represent the four subsets of the two-element set consisting of the classical values 1 and 0.

Alice: Will we be looking at any of these?

Hatter: No. They are all *non-classical*, and beyond our compass. In any case, classical logic remains the standard base.

It is time to look more closely at the concept of a *truth function* in two-valued logic. A one-place truth function is simply a function on domain {1,0} into {1,0}. A two-place one is a function on $\{1,0\}^2$ into {1,0} and so on. This immediately suggests all kinds of questions. How many one-place truth functions are there? How many two-place and generally *n*-place ones? Can every truth function be represented by a truth table? Are the specific truth functions given in the tables above sufficient to represent all of them, or do we need further logical connectives to do so? We can answer these questions quite easily.

Clearly, there are just four one-place truth functions. Each can be represented by a truth table. In the leftmost column, we write the two truth-values 1 and 0 that a

Table 8.3 The one-place truth functions

α	$f_1(\alpha)$	$f_2(\alpha)$	$f_3(\alpha)$	$f_4(\alpha)$
1	1	1	0	0
0	1	0	1	0

proposition α can bear. In the remaining columns, we write the possible values of the functions for those two values of their arguments (Table 8.3):

Exercise 8.2 (1) (with solution)

(a) Which of these four truth functions f_i corresponds to negation?

(b) Can you express the other three truth functions in terms of connectives with which you are already familiar (¬, ∧, ∨, →, ↔)?

Solution

(a) Obviously, f_3.

(b) f_2 is the identity function, that is, $f_2(\alpha) = \alpha$, so we don't need to represent it by more than α itself. f_1 is the constant function with value 1, and so can be represented as any of $\alpha\vee\neg\alpha$, $\alpha\rightarrow\alpha$, or $\alpha\leftrightarrow\alpha$. Finally, f_4 is the constant function with value 0 and can be represented by $\alpha\wedge\neg\alpha$ or by the negation of any of those for f_1.

Going on to the two-place truth functions, the Cartesian product $\{1,0\}^2$ evidently has $2^2 = 4$ elements, which may be listed in a table with four rows, as in the tables for the familiar two-place connectives. For each pair $(a,b) \in \{1,0\}^2$, there are evidently two possible values for the function, which gives us $4^2 = 16$ columns to fill in, that is, 16 truth functions.

We always write the rows of a truth table in standard order. In a case of a two-place function with arguments α,β, we begin with the row (1,1) where both α and β are true, and end with the row (0,0) where both are false. The principle for constructing each row from its predecessor for truth functions of any number of arguments: take the last 1 in the preceding row, change it to 0 and replace all 0 s to its right to 1. The following exercise may look tedious, but you will learn a lot by carrying it out in full.

Exercise 8.2 (2)

(a) Write out a table grouping together all of the 16 two-place truth functions, calling them $f_1,\ldots,f_{16}$. *Hints*: You will need 16 columns for the sixteen functions, plus 2 on the left for their two arguments. Again, these columns should be written in standard order: begin with the column (1,1,1,1) and proceed to the last column (0,0,0,0), in the manner prescribed in the text.

(b) Identify which of the 16 columns represent truth functions with which you are already familiar. Try to express the others using combinations of the familiar ones.

The familiar truth functions ∧, ∨, →, ↔ are all two-place, while ¬ is one-place. Is there a gap in their expressive power, or can every truth function of two places, indeed of arbitrary *n* places, be captured using them? Fortunately, there is no gap.

Table 8.4 Sample two-place truth table

α	β	f_3(α,β)
1	1	1
1	0	1
0	1	0
0	0	1

It is easy to show that every truth function, of any finite number of places, may be represented using at most the three connectives ¬, ∧, ∨.

To see how, consider the case of the two-place functions. Take an arbitrary two-place truth function f_i, say f_3, with its table as worked out in Exercise 8.2 (2) (Table 8.4).

Take the rows in which $f_i(\alpha,\beta) = 1$ – there are three of them in this instance. For each such row, form the conjunction $\pm\alpha\wedge\pm\beta$ where ± is empty or negation according as the argument has 1 or 0. This gives us the three conjunctions $\alpha\wedge\beta$ (for the first row), $\alpha\wedge\neg\beta$ (for the second row) and $\neg\alpha\wedge\neg\beta$ (for the fourth row). Form their disjunction: $(\alpha\wedge\beta)\vee(\alpha\wedge\neg\beta)\vee(\neg\alpha\wedge\neg\beta)$. Then, this will express the same truth function f_3.

Why? By the table for disjunction, it will come out true just when at least one of the three disjuncts is true. But the first disjunct is true in just the first row, the second is true in just the second row and the third is true in just the last row. So the constructed expression has exactly the same truth table as f_3, that is, it expresses that truth function.

Exercise 8.2 (3)

(a) Of the 16 two-place truth functions, there is exactly one that cannot be expressed in this way. Which is it? Show how we can still represent it using ¬, ∧, ∨.

(b) Draw a truth table for some three-place truth function and express it using ¬, ∧, ∨ in the same way.

(c) Would it have made a difference if we had written exclusive disjunction ⊕ instead of inclusive disjunction ∨ in this construction? Why?

Thus, every truth function can be expressed using only the connectives ¬, ∧, ∨. Are all three really needed, or can we do the job with even fewer connectives? We will return to this question in the next section, after clarifying some fundamental concepts.

8.3 Tautological Relations

A special feature of logic, as compared with most other parts of mathematics, is the very careful attention that it gives to the *language* in which it is formulated. Whereas the theory of trees, say, is about certain kinds of abstract structure – arbitrary sets equipped with a relation satisfying a certain condition – logic is about the interconnections between certain kinds of language and the structures that may be used to interpret them. A good deal of logic thus *looks at the language* as well as at the structures in which the language is interpreted. This can take quite some time to get used to, since ordinarily in mathematics we look *through* the language at the structures alone. It is like learning to look at the window pane as well as at the landscape beyond it.

8.3.1 The Language of Propositional Logic

Our language should be able to express all truth functions. Since we know that the trio $\neg$, $\wedge$, $\vee$ are together sufficient for the task, we may as well use them. They are the *primitive connectives* of the language. We take some set of expressions p_1, p_2, $p_3, \ldots$, understood intuitively as ranging over propositions, and call them *propositional variables* (their most common name) or *elementary letters* (less common, but our preferred one). This set may be finite or infinite; the usual convention is to take it either as finite but of unspecified cardinality or as countably infinite. As subscripts are a pain, we usually write elementary letters as $p,q,r, \ldots$ bringing in the subscripts only when we run short of letters or they are convenient for formulations.

The *formulae* (in some texts, 'well-formed formulae') of our language are expressions that can be obtained recursively from elementary letters by applying connectives. If the chapter on recursion and induction has not been forgotten entirely, it should be clear what this means: the set of formulae is the least set L that contains all the elementary letters and is closed under the connectives, that is, the intersection of all such sets. That is, whenever $\alpha,\beta \in L$, then so are $(\neg\alpha)$, $(\alpha\wedge\beta)$ and $(\alpha\vee\beta)$, and only expressions that can be formed from elementary letters in a finite number of such steps are counted as formulae.

In the chapter on trees, we have already seen why at least some brackets are needed in propositional formulae. For example, we need to be able to distinguish $(p\wedge q)\vee r$ from $p\wedge(q\vee r)$ and likewise $\neg(p\wedge q)$ from $(\neg p)\wedge q$. We also saw how the brackets may in principle be dispensed with if we adopt a prefix (Polish) or postfix (reverse Polish) notation, and we learned how to draw the syntactic decomposition tree of a formula. We agreed to make reading easier by omitting brackets when this can be done without ambiguity, for example, by omitting the outermost brackets, which we have just done and will continue to do. Finally, we reduce brackets a bit further by using standard grouping conventions, notably for negation, for example, always reading $\neg p\wedge q$ as $(\neg p)\wedge q$ rather than as $\neg(p\wedge q)$.

Exercise 8.3.1 (1)

(a) For revision, draw the syntactic decomposition trees of the two formulae $(p\wedge q)\vee r$ and $p\wedge(q\vee r)$ and also write each of them in both Polish and in reverse Polish notation. Likewise for the formulae $\neg(p\wedge q)$ and $(\neg p)\wedge q$.

(b) If you did not do Exercise 4.6.2 in the chapter on recursion and induction, do it now. If you did answer it, review your solutions.

With this out of the way, we can get down to more serious business, defining the fundamental concepts of classical propositional logic. An *assignment* is defined to be a function v: $E\rightarrow\{1,0\}$ on the set E of elementary letters of the language into the two-element set $\{1,0\}$. Roughly speaking, assignments correspond to left-hand parts of a truth table. By structural induction on formulae, we can easily show that for each assignment v: $E \to \{1,0\}$, there is a unique function v^+: $L \to \{1,0\}$, where L is the set of all formulae, that satisfies the following two conditions:

(1) It agrees with v on E, that is, $v^+(p) = v(p)$ for every elementary letter p.
(2) It satisfies the truth-table conditions; that is, $v^+(\neg\alpha) = 1$ iff $v^+(\alpha) = 0$, $v^+(\alpha \wedge \beta) = 1$ iff $v^+(\alpha) = v^+(\beta) = 1$, and $v^+(\alpha \vee \beta) = 0$ iff $v^+(\alpha) = v^+(\beta) = 0$.

Such a v^+ is called a *valuation* of formulae – the valuation determined by the assignment v. For the verification of its existence and uniqueness, we needed to keep the difference between v and v^+ clear. However, now that the verification is done, in accord with the precept of minimizing notational fuss, we will soon be 'abusing notation' by dropping the superscript from v^+ and writing it too as plain v. Only in contexts where confusion could possibly arise will we put the superscript back in.

Exercise 8.3.1 (2) Write out the proof by structural induction that shows the 'unique extension' result. Keep the notational difference between v and v^+ clear for the duration of this proof.

We are interested in a group of notions that may be called the *tautologicality concepts*. There are four basic ones. Two are *relations* between formulae: tautological implication and tautological equivalence. The other two are *properties* of formulae: those of being a tautology and of being a contradiction. They are intimately linked to each other.

8.3.2 Tautological Implication

We begin with the relation of *tautological implication*, also known as *tautological consequence*. It is a relation between sets of formulae on the left and individual formulae on the right. Let A be a set of formulae, and β an individual formula. We say that A *tautologically implies* β and write $A \vdash \beta$ iff there is no valuation v such that $v(\alpha) = 1$ for all $\alpha \in A$ but $v(\beta) = 0$. In other words, for every valuation v, if $v(\alpha) = 1$ for all $\alpha \in A$, then $v(\beta) = 1$. When A is a singleton $\{\alpha\}$, this comes to requiring that $v(\beta) = 1$ whenever $v(\alpha) = 1$. To reduce notation, in this singleton case, we write $\alpha \vdash \beta$ instead of $\{\alpha\} \vdash \beta$.

The definition can be expressed in terms of truth tables. If we draw up a truth table that covers all the elementary letters that occur in formulae in $A \cup \{\beta\}$, then it is required that every row which has a 1 under each of the $\alpha \in A$ also has a 1 under β.

Warning: The symbol $\vdash$ is *not* one of the connectives of propositional logic like $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$. Whereas $p \to q$ is a formula of the language L of propositional logic, $p \vdash q$ is not. Rather, $\vdash$ is a symbol that we use when talking *about* formulae of propositional logic. In handy jargon, we say that it belongs to our *metalanguage* rather than to the *object language*. This distinction takes a little getting used to, but it is very important. Neglect can lead to inextricable confusion.

Table 8.5 Some important tautological implications

Name	LHS	RHS
Simplification, $\wedge^-$	$\alpha\wedge\beta$	α
	$\alpha\wedge\beta$	β
Conjunction, $\wedge^+$	α, β	$\alpha\wedge\beta$
Disjunction, $\vee^+$	α	$\alpha\vee\beta$
	β	$\alpha\vee\beta$
Modus ponens, MP, $\rightarrow^-$	$\alpha, \alpha\rightarrow\beta$	β
Modus tollens, MT	$\neg\beta, \alpha\rightarrow\beta$	$\neg\alpha$
Disjunctive syllogism, DS	$\alpha\vee\beta, \neg\alpha$	β
Transitivity	$\alpha\rightarrow\beta, \beta\rightarrow\gamma$	$\alpha\rightarrow\gamma$
Material implication	β	$\alpha\rightarrow\beta$
	$\neg\alpha$	$\alpha\rightarrow\beta$
Limiting cases	γ	$\beta\vee\neg\beta$
	$\alpha\wedge\neg\alpha$	γ

Table 8.6 Verification for modus tollens

α	β	$\alpha\rightarrow\beta$	$\neg\beta$	$\neg\alpha$
1	1	1	0	0
1	0	0	1	0
0	1	1	0	1
0	0	1	1	1

Table 8.5 gives some of the more important tautological implications. The left column gives their more common names, typically a traditional name followed in some cases by its acronym, in some instances also a modern symbolic name. The Greek letters α,β,γ stand for arbitrary formulae of propositional logic. In each row, the formula or set of formulas marked LHS (left-hand side) tautologically implies the formula marked RHS (right-hand side). In most cases, we have a singleton on the left, but in four rows, we have a two-element set whose elements are separated by a comma. Thus, for example, the premise set for modus ponens is the pair $\{\alpha, \alpha\rightarrow\beta\}$, and the conclusion is β.

We can check that modus tollens, for example, is a tautological implication by building a truth table (Table 8.6). The four possible truth-value combinations for α,β are given in the left part of the table. The resulting values of $\alpha\rightarrow\beta$, $\neg\beta$, $\neg\alpha$ are calculated step by step, from the inside to the outside, more precisely from the leaves of the syntactic decomposition trees for the three formulae to their roots (very short trees in this case). We ask: Is there a row in which the two premises $\alpha\rightarrow\beta$, $\neg\beta$ get 1 while the conclusion $\neg\alpha$ gets 0? No, so the premises tautologically imply the conclusion.

Exercise 8.3.2

(a) Draw truth tables to check out each of disjunctive syllogism, transitivity and two material implication consequences.

(b) Check the two limiting cases and also explain in general terms why they hold, even when the formula γ shares no elementary letters with α or β.

(c) Show that the relation of tautological implication is reflexive and transitive, but not symmetric.

(d) Show that the relation of tautological implication is neither antisymmetric nor linear.

Once understood, the entries in Table 8.5 should be committed firmly to memory. That is rather a bore, but it is necessary in order to make progress. It is like learning some basic equalities and inequalities in arithmetic; *you need to have them at your fingertips so as to apply them without hesitation.*

8.3.3 Tautological Equivalence

In a tautological implication such as $\alpha \vdash \alpha \vee \beta$, while the left implies the right, the converse does not hold in general. When p, q are distinct elementary letters, $p \vee q \nvdash p$, although we do have $\alpha \vee \beta \vdash \alpha$ for some special choices of α, β.

Exercise 8.3.3 (1) Show that $p \vee q \nvdash p$ and find formulae α, β such that $\alpha \vee \beta \vdash \alpha$.

When each of two formulae tautologically implies the other, we say that the two are tautologically equivalent. That is, α is *tautologically equivalent* to β, and we write $\alpha \dashv\vdash \beta$, iff both $\alpha \vdash \beta$ and $\beta \vdash \alpha$. Equivalently: $\alpha \dashv\vdash \beta$ iff $v(\alpha) = v(\beta)$ for every valuation v. In terms of truth tables: the two formulae are tautologically equivalent iff, when we draw up a truth table that covers all the elementary letters that occur in them, the column for α comes out exactly the same as the column for β. Once again, the symbol $\dashv\vdash$ does not belong to the object language of propositional logic but is part of our metalanguage.

We give two tables listing the most important tautological equivalences that can be expressed in up to three elementary letters. The first Table 8.7 contains equivalences using at most the connectives $\neg$, $\wedge$, $\vee$ while the second table also deals with $\rightarrow$, $\leftrightarrow$. They are all very important and should also be committed to memory after being understood.

Exercise 8.3.3 (2)

(a) Draw truth tables to verify one of the de Morgan equivalences, one of the distributions, one of the absorptions, one of the expansions and one of the limiting cases.

(b) What striking syntactic feature do the limiting case equivalences have, alone among all the equivalences in the table?

(c) What syntactic feature is shared by the limiting case equivalences, absorption and expansion, but none of the others?

(d) Show from the definition that tautological equivalence is indeed an equivalence relation.

We saw analogues of many of these equivalences in Chap. 1 on sets. For example, the de Morgan principles there took the form of identities between sets, two of them being $-(A \cap B) = -A \cup -B$ and $-(A \cup B) = -A \cap -B$, where $-$ is complementation

Table 8.7 Some important tautological equivalences for ¬, ∧, ∨

Name	LHS	RHS
Double negation	α	$\neg\neg\alpha$
Commutation for ∧	$\alpha\wedge\beta$	$\beta\wedge\alpha$
Association for ∧	$\alpha\wedge(\beta\wedge\gamma)$	$(\alpha\wedge\beta)\wedge\gamma$
Commutation for ∨	$\alpha\vee\beta$	$\beta\vee\alpha$
Association for ∨	$\alpha\vee(\beta\vee\gamma)$	$(\alpha\vee\beta)\vee\gamma$
Distribution of ∧ over ∨	$\alpha\wedge(\beta\vee\gamma)$	$(\alpha\wedge\beta)\vee(\alpha\wedge\gamma)$
Distribution of ∨ over ∧	$\alpha\vee(\beta\wedge\gamma)$	$(\alpha\vee\beta)\wedge(\alpha\vee\gamma)$
Absorption	α α	$\alpha\wedge(\alpha\vee\beta)$ $\alpha\vee(\alpha\wedge\beta)$
Expansion	α α	$(\alpha\wedge\beta)\vee(\alpha\wedge\neg\beta)$ $(\alpha\vee\beta)\wedge(\alpha\vee\neg\beta)$
de Morgan	$\neg(\alpha\wedge\beta)$ $\neg(\alpha\vee\beta)$ $\alpha\wedge\beta$ $\alpha\vee\beta$	$\neg\alpha\vee\neg\beta$ $\neg\alpha\wedge\neg\beta$ $\neg(\neg\alpha\vee\neg\beta)$ $\neg(\neg\alpha\wedge\neg\beta)$
Limiting cases	$\alpha\wedge\neg\alpha$ $\alpha\vee\neg\alpha$	$\beta\wedge\neg\beta$ $\beta\vee\neg\beta$

with respect to some local universe. This is not surprising, since intersection, union and complementation of sets are defined using 'and', 'or' and 'not', respectively. There is thus a *systematic correspondence* between tautological equivalences and Boolean identities between sets. Similarly, there is a correspondence between tautological implications, such as those in Table 8.5, and rules of inclusion between sets.

The de Morgan equivalences answer a question that we posed at the end of Sect. 8.2 about the connectives needed to be able to express all truth functions. They may all be represented using just the two connectives ¬, ∧, since from them we can get ∨ by the last of the four de Morgan equivalences, and we already know that with that trio, we may obtain all the others. Likewise, the pair ¬, ∨ suffices to express all possible truth functions via the third de Morgan equivalence.

Table 8.8 lists important equivalences that involve → or ↔, in some cases, plus connectives from ¬, ∧, ∨.

Exercise 8.3.3 (3)

(a) Use a truth table to verify association for ↔.

(b) Use information from the list of equivalences to show that the pair ¬,→ is enough to express all truth functions.

(c) Use information from the list of equivalences to show by structural induction that any formula built using at most the connectives ¬, ↔ is equivalent to one in which all occurrences of ¬ act on elementary letters.

(d) Show using the same resources and similar strategy that any formula α built using at most the connectives ¬, ↔ is equivalent to one of the form $\pm(\alpha)$, where ± may be negation or no connective at all and where α contains no occurrences of ¬.

Table 8.8 Some important tautological equivalences using →, ↔

Name	LHS	RHS
Contraposition	$\alpha\rightarrow\beta$	$\neg\beta\rightarrow\neg\alpha$
	$\alpha\rightarrow\neg\beta$	$\beta\rightarrow\neg\alpha$
	$\neg\alpha\rightarrow\beta$	$\neg\beta\rightarrow\alpha$
Import/export	$\alpha\rightarrow(\beta\rightarrow\gamma)$	$(\alpha\wedge\beta)\rightarrow\gamma$
	$\alpha\rightarrow(\beta\rightarrow\gamma)$	$\beta\rightarrow(\alpha\rightarrow\gamma)$
Consequential mirabilis (miraculous consequence)	$\alpha\rightarrow\neg\alpha$	$\neg\alpha$
	$\neg\alpha\rightarrow\alpha$	α
Commutation for ↔	$\alpha\leftrightarrow\beta$	$\beta\leftrightarrow\alpha$
Association for ↔	$\alpha\leftrightarrow(\beta\leftrightarrow\gamma)$	$(\alpha\leftrightarrow\beta)\leftrightarrow\gamma$
¬ through ↔	$\neg(\alpha\leftrightarrow\beta)$	$\alpha\leftrightarrow\neg\beta$
Translations between connectives	$\alpha\leftrightarrow\beta$	$(\alpha\rightarrow\beta)\wedge(\beta\rightarrow\alpha)$
	$\alpha\leftrightarrow\beta$	$(\alpha\wedge\beta)\vee(\neg\alpha\wedge\neg\beta)$
	$\alpha\rightarrow\beta$	$\neg(\alpha\wedge\neg\beta)$
	$\alpha\rightarrow\beta$	$\neg\alpha\vee\beta$
	$\alpha\vee\beta$	$\neg\alpha\rightarrow\beta$
Translations of negations of connectives	$\neg(\alpha\rightarrow\beta)$	$\alpha\wedge\neg\beta$
	$\neg(\alpha\wedge\beta)$	$\alpha\rightarrow\neg\beta$
	$\neg(\alpha\leftrightarrow\beta)$	$(\alpha\wedge\neg\beta)\vee(\beta\wedge\neg\alpha)$

A set of truth-functional connectives is said to be *functionally complete* if all truth functions, of any finite number of places, may be expressed using only connectives from that set. From what we have done so far, we know that the sets $\{\neg,\wedge,\vee\}$, $\{\wedge,\neg\}$, $\{\neg,\vee\}$ and $\{\neg,\rightarrow\}$ are all functionally complete.

Exercise 8.3.3 (4) Is there any two-place truth-connective that, taken alone, is functionally complete? *Hints*: Go through the table for all the 16 two-place truth functions that was constructed in an exercise for Sect. 8.2, use your intuition to pick likely candidates and check that they do the job. For the checking part, first try to express ¬, and then try to express either ∧ or ∨. If you succeed, you are done.

We already know that tautological equivalence is indeed an equivalence relation. We end this subsection by noting that it has two further properties. It is also a *congruence relation* with respect to every truth-functional connective. That is, whenever $\alpha \dashv\vdash \alpha'$, then $\neg\alpha \dashv\vdash \neg\alpha'$; whenever $\alpha \dashv\vdash \alpha'$ and $\beta \dashv\vdash \beta'$, then $\alpha\wedge\beta\dashv\vdash\alpha'\wedge\beta'$ and $\alpha\vee\beta \dashv\vdash\alpha'\vee\beta'$; and similarly for all other truth-functional connectives. Moreover, it has the *replacement* property. That is, whenever $\alpha \dashv\vdash \alpha'$, then if we take a formula γ and replace one or more occurrences of α in γ by α' to get a formula γ', then $\gamma\dashv\vdash\gamma'$.

Exercise 8.3.3 (5)

(a) Use structural induction to show that tautological equivalence is a congruence relation.

(b) Do the same to show that it has the replacement property. *Remark*: The proof is not difficult to see intuitively but can be tricky to formulate neatly.

The concept of tautological equivalence may evidently be 'lifted' to a relation between sets of formulae. If A, B are sets of formulae, we say that they are *tautologically equivalent* and write $A \dashv\vdash B$ iff $A \vdash \beta$ for all $\beta \in B$ and also $B \vdash \alpha$ for all $\alpha \in A$. Equivalently: For every valuation v, $v(\alpha) = 1$ for all $\alpha \in A$ iff $v(\beta) = 1$ for all $\beta \in B$.

Exercise 8.3.3 (6) Check that so defined tautological equivalence between sets of formulae is an equivalence relation.

8.3.4 Tautologies and Contradictions

Now that we have the relations of tautological implication and equivalence under our belt, the properties of being a tautology, contradiction or contingent are child's play. Let α be any formula.

- We say that α is a *tautology* iff $v(\alpha) = 1$ for every valuation v. In other words: iff α comes out with value 1 in every row of its truth table.
- We say that α is a *contradiction* iff $v(\alpha) = 0$ for every valuation v. In other words: iff α comes out with value 0 in every row of its truth table.
- We say that α is *contingent* iff it is neither a tautology nor a contradiction. In other words: iff $v(\alpha) = 1$ for some valuation v and also $v(\alpha) = 0$ for some valuation v.

Clearly, every formula is either a tautology, or a contradiction, or contingent, and only one of these. In other words, these three sets partition the set of all formulae into three cells.

Exercise 8.3.4 (1) (with solution) Classify the following formulae as tautologies, contradictions or contingent: (i) $p\vee\neg p$, (ii) $\neg(p\vee\neg p)$, (iii) $p\vee\neg q$, (iv) $\neg(p\vee\neg q)$, (v) $(p\wedge(\neg p\vee q))\rightarrow q$, (vi) $\neg(p\vee q)\leftrightarrow(\neg p\wedge\neg q)$, (vii) $p\wedge\neg p$, (viii) $p\rightarrow\neg p$, (ix) $p\leftrightarrow\neg p$, (x) $(r\wedge s)\vee\neg(r\wedge s)$, (xi) $(r\rightarrow s)\leftrightarrow\neg(r\rightarrow s)$.

Solution Tautologies: (i), (v), (vi), (x). Contradictions: (ii), (vii), (ix), (xi). Contingent: (iii), (iv), (viii).

If you went through this little exercise conscientiously, you will already have sensed several general lessons that can be extracted from it.

- A formula is a tautology iff its negation is a contradiction. Example: (i) and (ii).
- A formula is contingent iff its negation is contingent. Example: (iii) and (iv).
- A formula $\alpha\rightarrow\beta$ is a tautology iff $\alpha \vdash \beta$. Generally, $\{\alpha_1,\ldots,\alpha_n\} \vdash \beta$ iff the formula $(\alpha_1\wedge\ldots\wedge\alpha_n)\rightarrow\beta$ is a tautology. Example: formula (v) of the exercise and the tautological implication of disjunctive syllogism.
- A formula $\alpha\leftrightarrow\beta$ is a tautology iff $\alpha \dashv\vdash \beta$. Example: formula (vi) of the exercise and one of the de Morgan equivalences.

Exercise 8.3.4 (2) Verify each of the bulleted points from the definitions.

Examples (x) and (xi) of the penultimate exercise are also instructive. The former tells us that $(r\wedge s)\vee\neg(r\wedge s)$ is a tautology. Without making a truth table, we can see that it must be so since it is merely a substitution instance of the simple tautology $p\vee\neg p$. Likewise, $(r\rightarrow s)\leftrightarrow\neg(r\rightarrow s)$ is a contradiction, being a substitution instance of the contradiction $p\leftrightarrow\neg p$. Quite generally, we have the following principle, which we will prove in a moment:

- Every substitution instance of a tautology or a contradiction is a tautology or contradiction, respectively.

On the other hand, not every substitution instance of a contingent formula is contingent. For example, we saw that $p\vee\neg q$ is contingent, but its substitution instance $p\vee\neg p$ (formed by substituting p for q) is a tautology, while another of its substitution instances $(p\wedge\neg p)\vee\neg(q\vee\neg q)$, formed by substituting $p\wedge\neg p$ for p and $q\vee\neg q$ for q, is a contradiction.

This notion of a substitution in propositional logic can be given a very precise mathematical content. A *substitution* is a function $\sigma: L\rightarrow L$, where L is the set of all formulae, satisfying the following *homomorphism conditions*:

$$\begin{aligned}\sigma(\neg\alpha) &= \neg\sigma(\alpha)\\ \sigma(\alpha\wedge\beta) &= \sigma(\alpha)\wedge\sigma(\beta)\\ \sigma(\alpha\vee\beta) &= \sigma(\alpha)\vee\sigma(\beta)\end{aligned}$$

Note that in this definition, = is not just tautological equivalence; it is full identity between formulae, that is, the left and right sides stand for the very same formula. It is easy to show, by structural induction on the definition of a formula of propositional logic, that a substitution function is uniquely determined by its values for elementary letters.

Exercise 8.3.4 (3) (with partial solution)

(a) Suppose $\sigma(p)=q\wedge\neg r$, $\sigma(q)=\neg q$, $\sigma(r)=p\rightarrow s$. Identify the formulae (i) $\sigma(\neg p)$, (ii) $\sigma(p\vee\neg q)$, (iii) $\sigma(r\vee(q\rightarrow r))$.

(b) Treating $\alpha\rightarrow\beta$ and $\alpha\leftrightarrow\beta$ as abbreviations for $\neg(\alpha\wedge\neg\beta)$ and $(\alpha\wedge\beta)\vee(\neg\alpha\wedge\neg\beta)$, respectively, show that substitution functions satisfy analogous homomorphism conditions for them too.

Solution to (a)

(a) (i) $\sigma(\neg p)=\neg\sigma(p)=\neg(q\wedge\neg r)$, (ii) $\sigma(p\vee\neg q)=\sigma(p)\vee\sigma(\neg q)=\sigma(p)\vee\neg\sigma(q)=(q\wedge\neg r)\vee\neg\neg q$. For (iii), omitting the intermediate calculations, $\sigma(r\vee(q\rightarrow r))=(p\rightarrow s)\vee(\neg q\rightarrow(p\rightarrow s))$.

Note that substitutions are carried out simultaneously, not serially. For example, in part (a) (ii) of the exercise, $\sigma(p\vee\neg q)$ is obtained by simultaneously substituting $\sigma(p)$ for p and $\sigma(q)$ for q. If we were to replace serially, first substituting $\sigma(p)$ for p to get $(q\wedge\neg r)\vee\neg q$ and then substituting $\sigma(q)$ for q, we would get the result $(\neg q\wedge\neg r)\vee\neg\neg q$, which differs in the sign on the first occurrence of q.

We are now in a position to prove the claim that every substitution instance of a tautology is a tautology or, as we also say, the set of all tautologies is *closed under* *substitution*. Let α be any formula, and σ any substitution function. Suppose that $\sigma(\alpha)$ is not a tautology. Then there is a valuation v such that $v(\sigma(\alpha)) = 0$. Let v' be the valuation defined on the letters in α by putting $v'(p) = v(\sigma(p))$. Then it is easy to verify by structural induction that for every formula β, $v'(\beta) = v(\sigma(\beta))$. In particular, $v'(\alpha) = v(\sigma(\alpha)) = 0$, so that α is not a tautology.

Likewise, the set of all contradictions and the relations of tautological equivalence and tautological implication are closed under substitution. That is, for any substitution function σ, we have the following:

- Whenever α is a contradiction, then $\sigma(\alpha)$ is too.
- Whenever $\alpha \dashv\vdash \beta$, then $\sigma(\alpha) \dashv\vdash \sigma(\beta)$.
- Whenever $\alpha \vdash \beta$, then $\sigma(\alpha) \vdash \sigma(\beta)$.
- Whenever $A \vdash \beta$, then $\sigma(A) \vdash \sigma(\beta)$.

Here, A is any set of formulae, and $\sigma(A)$ is defined, as you would expect from the chapter on functions, as $\{\sigma(\alpha){:}\ \alpha \in A\}$.

Exercise 8.3.4 (4) Complete the inductive details in the verification that the set of all tautologies is closed under substitution.

8.4 Normal Forms

The formulae of propositional logic may be of any length, depth or level of complexity. It is important to be able to express them in the most transparent possible way. In this section, we will look briefly at four kinds of normal form. The first two focus on the positioning of connectives and show that they may be applied in a neat order. The next pair concerns the interplay of elementary letters; one gets rid of all redundant letters, while the other compartmentalizes their interaction as much as possible.

8.4.1 Disjunctive Normal Form

Normal forms are common in logic, mathematics and computer science. A *normal form* for an expression is another expression, equivalent (in a suitable sense) to the first, but with a nice simple structure. When the normal form is used a lot, the term *canonical form* is often employed. A *normal form theorem* is one telling us that every expression (from some broad category) has a normal form (of some specified kind). In propositional logic, the best-known such form is *disjunctive normal form*, abbreviated *dnf*. To explain it, we need the concepts of a literal and of a basic conjunction.

A *literal* is simply an elementary letter or its negation. A *basic conjunction* is any conjunction of (one or more) literals in which no letter occurs more than once. Thus, we do not allow repetitions of a letter, as in $p\wedge q\wedge p$, nor repetitions of the negation of a letter, as in $\neg p\wedge q\wedge\neg p$,nor a letter occurring both negated and unnegated, as in $\neg p\wedge q\wedge p$. We write a basic conjunction as $\pm p_1\wedge \ldots \wedge\pm p_n$, where the ± indicates the presence or absence of a negation sign. In the limiting case that $n = 1$, a basic conjunction is of the form $\pm p_1$, without any conjunction sign.

A formula is said to be in *disjunctive normal form* (*dnf*) iff it is a disjunction of (one or more) basic conjunctions. Again, in the limiting case that $n = 1$, a dnf will not contain a disjunction sign. A formula is said to be in *full dnf* iff it is in dnf, and moreover every letter in it occurs in each of the basic conjunctions.

Exercise 8.4.1 (1) (with solution)

(a) Which of the following are in disjunctive normal form? When your answer is negative, explain briefly why. (i) $((p\wedge q)\vee\neg r)\wedge\neg s$, (ii) $(p\vee q)\vee(q\rightarrow r)$, (iii) $(p\wedge q)\vee(\neg p\wedge\neg q)$, (iv) $(p\wedge q)\vee(\neg p\wedge\neg q\wedge p)$, (v) $(p\wedge q)\vee(\neg p\wedge\neg q\wedge\neg p)$, (vi) $p\wedge q\wedge r$, (vii) p, (viii) $\neg p$, (ix) $p\vee q$, (x) $p\vee\neg p$, (xi) $p\wedge\neg p$.

(b) Which of the above are in full dnf?

Solution

(a) No: there is a disjunction inside a conjunction. (ii) No: we have not eliminated $\rightarrow$. (iii) Yes. (iv) No: $\neg p\wedge\neg q\wedge p$ contains two occurrences of p and so is not a basic conjunction. (v) No: $\neg p\wedge\neg q\wedge\neg p$ contains two occurrences of $\neg p$. (vi) Yes. (vii) Yes. (viii) Yes. (ix) Yes. (x) Yes. (xi) No: $p\wedge\neg p$ has two occurrences of p.

(b) (iii), (vi), (vii), (viii), (x).

How can we find a disjunctive normal form for an arbitrarily given formula α? There are two basic algorithms. One is semantic, proceeding via the truth table for α. The other is syntactic, using successive transformations of α, justified by tautological equivalences from our table of basic equivalences.

The semantic construction is the simpler of the two. In fact, we already made use of it in Sect. 8.3 when we showed that the trio $\{\neg,\wedge,\vee\}$ of connectives is functionally complete. We begin by drawing up the truth table for α and checking whether it is a contradiction. Then:

- In the principal case that α is not a contradiction, the dnf of α is the disjunction of the basic conjunctions that correspond to rows of the table in which α receives value 1.
- In the limiting case that α is a contradiction, that is, when there are no such rows, the formula does not have a dnf.

It is clear from the construction that *every non-contradictory formula has a disjunctive normal form*. It is also clear that the dnf obtained is *unique* up to the ordering and bracketing of literals and basic conjuncts. Moreover, it is clearly *full*.

Exercise 8.4.1 (2) Find the full disjunctive normal form (if it exists) for each of the following formulae, using the above truth-table algorithm: (i) $p\leftrightarrow q$, (ii) $p\rightarrow(q\vee r)$, (iii) $\neg[(p\wedge q)\rightarrow\neg(r\vee s)]$, (iv) $p\rightarrow(q\rightarrow p)$, (v) $(p\vee\neg q)\rightarrow(r\wedge\neg s)$, (vi) $(p\vee\neg p)\rightarrow(r\wedge\neg r)$.

For the *syntactic method*, we start with the formula α and proceed by a series of transformations that massage it into the desired shape. The basic idea is fairly simple, although the details are rather fussy. The steps of the algorithm should be executed in the order given:

- Translate the connectives → and ↔ (and any others in the formula, such as exclusive disjunction) into ¬, ∧, ∨ using translation equivalences such as those in Table 8.8.
- Use the de Morgan rules $\neg(\alpha\wedge\beta) \dashv\vdash \neg\alpha\vee\neg\beta$ and $\neg(\alpha\vee\beta) \dashv\vdash \neg\alpha\wedge\neg\beta$ iteratively to move negation signs inwards until they act directly on elementary letters, eliminating double negations as you go by the rule $\neg\neg\alpha \dashv\vdash \alpha$.
- Use the distribution rule $\alpha\wedge(\beta\vee\gamma) \dashv\vdash (\alpha\wedge\beta)\vee(\alpha\wedge\gamma)$ to move all conjunctions inside disjunctions.
- Use idempotence, with help from commutation and association, to eliminate repetitions of letters or of their negations.
- Delete all basic conjunctions containing a letter and its negation.

This will give us as output a formula in disjunctive normal form, except when α is a contradiction, in which case it turns out that the output is empty. However, the dnf obtained in this way is rarely full: there may be basic conjunctions with less than the full baggage of letters occurring in them.

Exercise 8.4.1 (3)

(a) Use the syntactic algorithm to transform the formula $(p\vee\neg q)\rightarrow(r\wedge\neg s)$ into disjunctive normal form. Show your transformations step by step and compare the result with that obtained by the semantic method in the preceding exercise.

(b) How might you transform the dnf obtained above into a full one? *Hint*: Make use of the tautological equivalence of expansion $\alpha \dashv\vdash (\alpha\wedge\beta)\vee(\alpha\wedge\neg\beta)$.

8.4.2 Conjunctive Normal Form*

Conjunctive normal form is like disjunctive normal form but 'upside-down': the roles of disjunction and conjunction are reversed. Technically, they are called *duals* of each other. A *basic disjunction* is defined to be any disjunction of (one or more) literals in which no letter occurs more than once. A formula is said to be in *conjunctive normal form* (*cnf*) iff it is a conjunction of (one or more) basic disjunctions, and this is said to be a *full* conjunctive normal form of α iff every letter of α occurs in each of the basic disjunctions.

Cnfs may also be constructed semantically, syntactically or by piggybacking on dnfs. For the semantic construction, we look at the rows of the truth table that give α the value 0. For each such row, we construct a basic disjunction: this will be the disjunction of those letters with the value 0 and the negations of the letters with value 1. We then conjoin these basic disjunctions. This will give an output in conjunctive normal form, except when the initial formula α is a tautology. For the syntactic construction, we proceed just as we do for dnfs, except that we use the other distribution rule $\alpha\vee(\beta\wedge\gamma) \dashv\vdash (\alpha\vee\beta)\wedge(\alpha\vee\gamma)$ to move disjunctions inside conjunctions.

At first sight, this construction may seem a little mysterious. Why do we construct the cnf from the table the way described? Is there an underlying idea? There is, indeed. Instead of allowing the valuations that make the formula α true, we are excluding the valuations that make α false. To do that, we take each row that gives α the value 0 and declare it not to hold. That is the same as negating the basic conjunction of that row, and by de Morganizing that negation, we get the basic disjunction described in the construction. In effect, the basic disjunction says 'not in my row, thank you'.

Alice Box: Conjunctive Normal Form

Alice: OK, but why should we bother with cnfs when we already have dnfs? They are much less intuitive!

Hatter: Indeed, they are. But they have been found useful in a discipline known as *logic programming*. Such a program may, in the simplest case, be seen as a set or conjunction of *positive clauses* $(p_1 \wedge \ldots \wedge p_n) \rightarrow q$ with $n \geq 0$ (understood as just the letter q in the limiting case that $n = 0$, and sometimes known as *Horn clauses*). Such a formulae may be expressed as a basic disjunction with just one unnegated letter, $(\neg p_1 \vee \ldots \neg p_n) \vee q$ $(n \geq 0)$. A conjunction of such formulae is evidently in cnf.

We can also construct a cnf for a formula α by piggybacking on a dnf for its negation ¬α. Given α, we construct a dnf of its negation ¬α by whichever method you like (via truth tables or by successive transformations). This will be a disjunction $\beta_1 \vee \ldots \vee \beta_n$ of basic conjunctions β_i. Negate it, getting $\neg(\beta_1 \vee \ldots \vee \beta_n)$, which is tautologically equivalent to ¬¬α and thus by double negation to our original α. We can then use de Morgan to push all negations inside, getting $\neg\beta_1 \wedge \ldots \wedge \neg\beta_n$. Each $\neg\beta_i$ is of the form $\neg(\pm p_1 \wedge \ldots \wedge \pm p_n)$, so we can apply de Morgan again, with double negation as needed, to express it as a disjunction of literals, as desired.

Exercise 8.4.2

(a) Take again the formula $(p \vee \neg q) \rightarrow (r \wedge \neg s)$, and find a cnf for it by all three methods: (i) semantic, (ii) successive syntactic transformations, (iii) the indirect method.

(b) Evidently, in most cases, a formula that is in dnf will not be in cnf. But in some limiting cases, it will be in both forms. When can that happen?

(c) Check that $\neg p_1 \vee \ldots \neg p_n \vee q \dashv\vdash (p_1 \wedge \ldots \wedge p_n) \rightarrow q$.

8.4.3 Eliminating Redundant Letters

A formula of propositional logic may contain redundant letters. We have already seen an example in the table of important equivalences: absorption tells us that $p\wedge(p\vee q) \dashv\vdash p \dashv\vdash p\vee(p\wedge q)$; the letter q is thus redundant in each of the two outlying formulae. The expansion equivalences in the same table give another example. So does the equivalence $(p\rightarrow q)\wedge(\neg p\rightarrow q) \dashv\vdash q$, not in the table.

Suppose we expand our language a little to admit a zero-ary connective $\bot$ (called *bottom* or *falsum*) so that $\bot$ is a formula with no elementary letters. We stipulate that it receives the value 0 under every valuation. We also stipulate that $\sigma(\bot) = \bot$ for any substitution function σ. Then, contradictions like $p\wedge\neg p$ and tautologies like $p\vee\neg p$ will also have redundant letters, since $p\wedge\neg p \dashv\vdash \bot$ and $p\vee\neg p \dashv\vdash \neg\bot$. For our discussion of redundancy, it will be convenient to assume that our language contains $\bot$ as well as the ordinary connectives, as it will streamline formulations.

It is clear that for every formula α containing elementary letters $p_1,..,p_n$ ($n \geq 0$), there is a *minimal* set of letters in terms of which α may equivalently be expressed. That is, there is some minimal set E_α of letters such that $\alpha \dashv\vdash \alpha'$ for some formula α' all of whose letters are drawn from E_α. This is because α contains only finitely many letters to begin with, and so as we eliminate redundant letters, we must eventually come to a set (perhaps empty) from which no more can be eliminated.

But is this minimal set unique? In other words, is there a *least* such set – one that is included in every such set. In general, as noted in an exercise at the end of the chapter on relations, minimality does not in general imply leastness. However, in the present context, intuition suggests that surely there should be a least letter-set. And in this case, intuition is right.

The proof is not really difficult. Let α be a formula containing elementary letters $p_1, \ldots ,p_n$ and let $\{p_1, \ldots ,p_m\}$ ($m \leq n$) be a minimal letter-set for α, so that there is a formula $\alpha' \dashv\vdash \alpha$ containing only the letters $p_1, \ldots ,p_m$. We want to show that $\{p_1, \ldots ,p_m\}$ is a least letter-set for α. Suppose that it is not. Then there is a formula α'' equivalent to α and a letter $p_i \in \{p_1, \ldots ,p_m\}$ such that p_i does not occur in α''. We get a contradiction. Substitute the falsum $\bot$ for p_i leaving all other letters unchanged. Calling this substitution σ, we have $\sigma(\alpha') \dashv\vdash \sigma(\alpha'')$ since $\alpha' \dashv\vdash \alpha''$. Since p_i does not occur in α'', we have $\sigma(\alpha'') = \alpha'' \dashv\vdash \alpha$, so $\sigma(\alpha') \dashv\vdash \alpha$. But $\sigma(\alpha')$ has one less letter than α', namely, p_i (remember, $\bot$ is not an elementary letter), contradicting the assumption that the letters in α' form a minimal letter-set for α.

Thus, every formula has a least letter-set E_α, and it is trivial that this must be unique. Any formula α' equivalent to α that is built from those letters is known as a *least letter-set version* of α. The same considerations apply for arbitrary sets of formulae, even when they are infinite: Each set A of formulae has a unique least letter-set and thus also a least letter-set version A'.

It is possible to construct algorithms to find a least letter-set version of any formula, although they are too complex to be carried out easily by hand. So, for small examples like those of the next exercise, we use our experience with truth-functional formulae to inspire guesses, which we then settle by checking.

Exercise 8.7.3 (with solution)

(a) Find a least letter-set version for each of the following formulae: (i) $p \wedge \neg p$, (ii) $(p \wedge q \wedge r) \vee (p \wedge \neg q \wedge r)$, (iii) $(p \leftrightarrow q) \vee (p \leftrightarrow r) \vee (q \leftrightarrow r)$.

(b) Do the same for the following sets of formulae: (i) $\{p \vee q \vee r,\ p \vee \neg q \vee r \vee s,\ p \vee \neg q \vee r \vee \neg s\}$, (ii) $\{p \rightarrow q,\ p \rightarrow \neg q\}$.

(c) True or false? 'The least letter-set of a finite set of formulae is the same as the least letter-set of the conjunction of all of its elements'. Explain.

Solution

(a) (i) $\bot$, (ii) $p \wedge r$, (iii) $\neg\bot$. (b) (i) $p \vee r$, (ii) $\neg p$. (c) True: A finite set of formulae is tautologically equivalent to the conjunction of all of its elements, so the formulae that are equivalent to the set are the same as those equivalent to the conjunction.

8.4.4 Most Modular Version

The next kind of simplification is rather more subtle. It leads us to a representation that does not change the letter-set but makes their role as 'compartmentalized' or 'modular' as possible. The definition makes essential use of the notion of a partition, and you are advised to review the basic theory of partitions in Chap. 2 (Sect. 2.5.3) before going further.

Consider the formula set $A = \{\neg p,\ r \rightarrow ((\neg p \wedge s) \vee q),\ q \rightarrow p\}$. This set has three formulae as its elements. Between them, the formulae contain four elementary letters p, q, r, s. None of these letters is redundant – the least letter-set is still $\{p, q, r, s\}$. But the way in which the letters occur in formulae in A is unnecessarily 'mixed up': they can be separated out rather better from each other. In other words, we can make the presentation of A more 'modular', without reducing the set of letters involved.

Observe that A is tautologically equivalent to the set $A' = \{\neg p,\ r \rightarrow s,\ \neg q\}$. We have not eliminated any letters, but we have disentangled their role in the set. In effect, we have partitioned the letter-set $\{p, q, r, s\}$ of A into three cells $\{p\}$, $\{r,s\}$, $\{q\}$, with each formula in A' drawing all its letters from a single cell of the partition. Thus, the formula $\neg p$ takes its sole letter from the cell $\{p\}$, $r \rightarrow s$ draws its letters from the cell $\{r,s\}$ and $\neg q$ takes its letter from the cell $\{q\}$. We say that the partition $\{\{p\}, \{r,s\}, \{q\}\}$ of the letter-set $\{p, q, r, s\}$ is a *splitting* of A.

In general terms, here is the definition. Let A be any set of formulae, with E the set of its elementary letters. A *splitting* of A is a partition of E such that A is tautologically equivalent to some set A' of formulae, such that each formula in A' takes all of its letters from a single cell.

Now, as we saw in Exercise 2.5.3 of the chapter on relations, partitions of a set can be compared according to their fineness. Consider any two partitions of the same set. One partition is said to be at least as *fine* as another iff every cell of the former is a subset of some cell of the latter. This relation between partitions is a partial ordering in the sense also defined in the chapter on relations: reflexive, transitive, antisymmetric. As we also saw in the end-of-chapter exercises, one partition is at least as fine as another iff the equivalence relation corresponding to the former is a subrelation of that corresponding to the latter.

Since a splitting of a set of propositional formulae is a special kind of partition of the set of its elementary letters, it makes sense to compare splittings according to their fineness. In our example $A = \{\neg p, r \rightarrow ((\neg p \wedge s) \vee q), q \rightarrow p\}$, the three-cell splitting $\{\{p\}, \{r,s\}, \{q\}\}$ mentioned above is finer than the two-cell splitting $\{\{p,q\}, \{r,s\}\}$ that corresponds to the formula set $A' = \{\neg p \wedge \neg q, r \rightarrow s\}$ equivalent to A; and this is in turn finer than the trivial one-cell splitting $\{\{p, q, r, s\}\}$ that corresponds to A itself.

It turns out that each set A of formulae has a *unique finest splitting*. In the case of our example, it is the three-cell partition $\{\{p\}, \{r,s\}, \{q\}\}$ of E. The four-cell partition $\{\{p\}, \{r\}, \{s\}, \{q\}\}$ of E is finer, but it is not a splitting of A, since no set A'' of formulae, each of which draws its letters from a single cell of this partition, is equivalent to A. We will not prove this, but intuitively it is to be expected since each formula in A'' contains only a single letter.

Any formula set A' equivalent to A, using the same letters as A, but with the letters in each formula of A' taken from a single cell of the finest splitting of A, is called a *finest splitting version* of A.

Strictly speaking, the definitions above cover only the principal case that there is some elementary letter in some formula of A. For in the limiting case that there are no elementary letters, that is, when $E = \varnothing$, the notion of a partition of E is not defined. However, in this case, A must be tautologically equivalent either to $\bot$ or to $\neg\bot$, and it is convenient to take that as the finest splitting version of A in this case.

Thus, a finest splitting version of a set of formulae disentangles as much as possible the roles that are played by the different elementary letters. It makes the presentation of the set as modular as possible: we have reached the finest way of separating the letters such that no formula contains letters from two distinct cells. The finest splitting versions of A may thus also be called *the most modular presentations* of A. They could also be called the *minimal mixing* versions of A.

Whereas the idea of the least letter-set of a set of formulae is quite old, perhaps dating back as far as the nineteenth century, that of the finest splitting is surprisingly new. It was first formulated and verified for the finite case by Rohit Parikh in 1999; a proof of uniqueness for the infinite case was given (by the author) only in 2007!

Just as for least letter-set versions, any general algorithm for finding most modular versions is highly exponential and laborious to execute by hand in even the simplest examples. So for such small examples, we again use our experience with truth-functional formulae to inspire our guessing and follow it up by checking, as in the following exercise:

Exercise 8.4.4 (1) (with partial solution) Find the finest splitting and a most modular version for each of the following sets of formulae: $A = \{p \wedge q, r\}$, $B = \{p \rightarrow q, q \rightarrow r, r \rightarrow \neg p\}$, $C = \{(\neg p \wedge q \wedge r) \vee (q \wedge \neg s \wedge \neg p)\}$, $D = \{p \vee q, q \vee r, r \vee \neg p, \neg q \vee r\}$. *Remember*: You are not eliminating redundant letters (indeed, in these instances, no letters are redundant); you are splitting the existing letter-set.

Solution to (a) and (b) The finest splitting of A partitions its letter-set $E = \{p, q, r\}$ into singleton cells: $\{\{p\}, \{q\}, \{r\}\}$, with a most modular version of A being $A' = \{p, q, r\}$. The finest splitting of B partitions its letter-set $E = \{p, q, r\}$ into two cells: $\{\{p\}, \{q, r\}\}$, with a most modular version of B being $B' = \{\neg p, q \rightarrow r\}$.

Finally, we note that there is nothing to stop us combining these two kinds of letter management with each other and with either dnf or cnf. Given a set A of formulae, we can first eliminate all redundant letters, getting it into least letter-set form A'; then work on A' to put it into a most modular version A''; and then go on, if we wish, to express the separate formulae $\alpha \in A''$ in dnf or in cnf.

Exercise 8.4.4 (2) Find the least letter-set of the following set of formulae and put it in most modular form over that least letter-set: $\{(p \rightarrow q) \wedge (r \rightarrow s), \neg q \wedge (p \vee u), (u \vee r) \rightarrow (s \vee r)\}$.

8.5 Semantic Decomposition Trees

By now, you are probably sick of drawing up truth tables, even small ones of four or eight rows, to test for the various kinds of tautological relations and properties (tautological consequence and equivalence, tautologies and contradictions). We describe a shortcut method that can be used instead.

The method is *semantic*, in the sense that it is formulated in terms of truth-values; *algorithmic*, in the sense that its steps can be carried out by a computer; and *two-sided*, in the sense that it gives a way of determining, in a finite time, whether or not a formula is, for example, a tautological consequence of another. It thus supplies us with a *decision procedure*. It is called the method of *semantic decomposition trees*, also known as *semantic tableaux*.

We begin with an example. We already know that the de Morgan formula $\alpha = \neg(p \wedge q) \rightarrow (\neg p \vee \neg q)$ is a tautology, but let's check it without making a table. Suppose that $v(\alpha) = 0$ for some valuation v. We show by successive decompositions of the formula that this supposition leads to a violation of bivalence. From the supposition, we have by the table for the arrow that $v(\neg(p \wedge q)) = 1$ while $v(\neg p \vee \neg q) = 0$. From the latter by the table for disjunction, $v(\neg p) = 0$ and $v(\neg q) = 0$, so by the table for negation, $v(p) = 1$ and $v(q) = 1$. On the other hand, since $v(\neg(p \wedge q)) = 1$, we have $v(p \wedge q) = 0$. so by the table for conjunction, either $v(p) = 0$ or $v(q) = 0$. In the first case, we get a contradiction with $v(p) = 1$, and in the second case, a contradiction with $v(q) = 1$. Thus, the initial supposition that $v(\alpha) = 0$ is impossible, so $\neg(p \wedge q) \rightarrow (\neg p \vee \neg q)$ is a tautology.

This reasoning can be set out in the form of a labelled tree, as in Fig. 8.1. It is constructed from the root down and should be read in the same way.

To see what is going on, note the following features of the way in which this tree is constructed:

- The *root* is labelled with the formula that we are testing, together with a truth-value. In our example, we are testing whether α is a tautology, that is, whether it

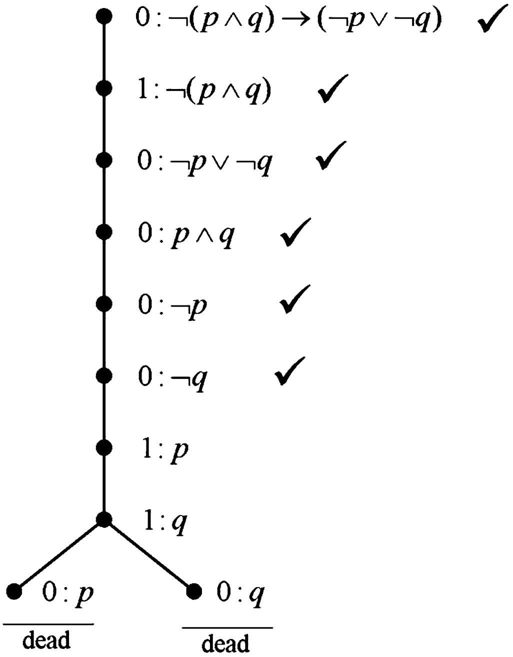

Fig. 8.1 Semantic decomposition tree for $\neg(p\wedge q)\rightarrow(\neg p\vee\neg q)$

is impossible for it to receive value 0, so the label is 0: α. If we had been testing whether α is a contradiction, the label would have been 1: α.

- At each step, we *decompose* the current formula, passing from information about its truth-value to resulting information about its *immediate subformulae*. Never in the opposite direction. The information is supplied by the truth table for the particular connective that is being decomposed at that point.
- When we get *disjunctive* information about the immediate subformulae of φ, we *divide* our branch into two subbranches, with one of the alternatives on one branch and the other alternative on the other.
- When we get *definite* information about the immediate subformulae of φ, we put it on *every branch* below the node for φ. One way of not forgetting this is by writing it down immediately before any further dividing takes place.

Having constructed the decomposition tree, we need to be able to read our answer from it. We make sure that we have decomposed each node in the tree whenever possible, that is, when it is not an elementary letter. In our example, the ticks next to nodes keep a record that we have actually carried out the decomposition. Then we read the completed tree as follows:

- If *every* branch contains an *explicit contradiction* (two nodes labelled by the same formula with opposite signs, 1: φ and 0: φ), then the label of the root is impossible. This is what happens in our example: there are two branches, one containing both $v(p) = 1$ and $v(p) = 0$ and the other containing both $v(q) = 1$ and

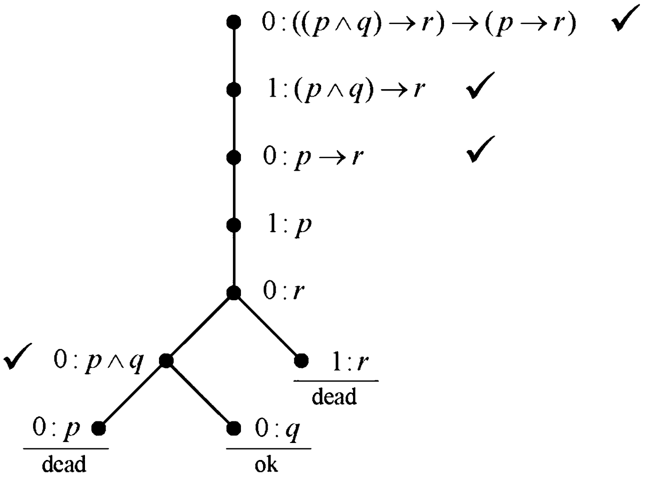

Fig. 8.2 Semantic decomposition tree for $((p\wedge q)\rightarrow r)\rightarrow(p\rightarrow r)$

$v(q) = 0$. We label these branches *dead* and conclude that $v(\alpha) = 0$ is impossible, that is, that α is a tautology.

- On the other hand, if *at least one* branch is without explicit contradiction, then the label of the root is possible, provided we have really decomposed all decomposable formulae in that branch. We label these branches *ok*, and read off any one of them a valuation that gives the root formula the value indicated in the label.

We give a second example that illustrates the latter situation. Consider the formula $\alpha = ((p\wedge q)\rightarrow r)\rightarrow(p\rightarrow r)$ and test to determine whether or not it is a tautology. We get the labelled tree of Fig. 8.2.

This decomposition tree contains three branches. The leftmost and rightmost ones each contain an explicit contradiction (1: p and 0: p in the left one, 0: r and 1: r in the right one), and so we label them *dead*. The middle branch does not contain any explicit contradiction. We check carefully that we have completed all decompositions in this branch – that we have ticked every formula in it other than elementary letters. We collect from the branch the assignment $v(p) = 1$, $v(r) = v(q) = 0$. This valuation generated by this assignment gives every formula in the branch its labelled value, and in particular gives the root formula α the labelled value 0, so that it is not a tautology.

Evidently, this method is algorithmic and can be programmed. It is usually quicker to calculate (whether by human or by machine) and more fun than a full table, although it must be admitted that in worst-case examples, it turns out to be just as exponential as a truth table. The construction always terminates: as we go down the branches, we deal with shorter and shorter formulae until we reach elementary letters and can decompose no further. This is intuitively clear: a rigorous proof would be by structural induction on formulae.

Table 8.9 Definite decomposition rules

$1: \neg\alpha$	$0: \neg\alpha$	$1: \alpha\wedge\beta$	$0: \alpha\vee\beta$	$0: \alpha\rightarrow\beta$
$0: \alpha$	$1: \alpha$	$1: \alpha$	$0: \alpha$	$1: \alpha$
		$1: \beta$	$0: \beta$	$0: \beta$

Table 8.10 Branching decomposition rules

$1: \alpha\vee\beta$		$0: \alpha\wedge\beta$		$1: \alpha\rightarrow\beta$		$1: \alpha\leftrightarrow\beta$		$0: \alpha\leftrightarrow\beta$	
$1: \alpha$	$1: \beta$	$0: \alpha$	$0: \beta$	$0: \alpha$	$1: \beta$	$1: \alpha$	$0: \alpha$	$1: \alpha$	$0: \alpha$
						$1: \beta$	$0: \beta$	$0: \beta$	$1: \beta$

There are many ways in which the method can be streamlined to maximize its efficiency. Some gains may be obtained by controlling the order in which decompositions are carried out. In our second example, after decomposing the root to introduce the second and third nodes, we had a choice of which of those to decompose first. We opted to decompose the third node before the second, but we could perfectly well have done the reverse. Our choice was motivated by the fact that the third node gives definite information, and we were following a policy of *postponing branching for as long as possible*. This is a control heuristic that works well, at least for humans in small finite examples, because it can reduce the number of branches and repetitions of labelled nodes on them. Whichever order we follow, we get the same final verdict.

Exercise 8.5 (1) Reconstruct the semantic decomposition tree for $((p\wedge q)\rightarrow r)\rightarrow(p\rightarrow r)$, decomposing the second node before the third one.

Also important as a way of reducing unnecessary work is the fact that we can sometimes stop constructing the tree before it is finished. This can be done in two distinct ways:

- If a branch contains an explicit contradiction before having been fully decomposed, we can already declare that branch is dead without completing its decomposition and pass on to other branches. This sometimes saves a little work.
- If we have a branch, all whose nodes have been decomposed, and it is free of explicit contradictions, then we can declare that this branch is *ok* so that the label of the root (and of everything on the branch) is possible, without bothering to complete any other branches. This sometimes saves a lot of work.

In addition to these economies, one can develop *heuristics* to guide the choice, when we branch, of which side to construct first and how far before looking at the other, in other words balancing *depth-first* and *breadth-first* strategies. But such heuristics take us beyond the limits of this introduction, and we leave them aside. Tables 8.9 and 8.10 recapitulate the definite and branching decompositions that one may make with the various truth-functional connectives. Their justification is immediate from the truth tables.

We thus have two rules for decomposing each connective, one for sign 1 and the other for sign 0. Note that the rule for ¬ is definite no matter what the sign (1 or 0); the rules for ∧, ∨, → are definite or branching depending on the sign, and the one for

$\leftrightarrow$ branches irrespective of sign. Most common student errors: (1) misremembering the rule for $\rightarrow$, usually from not having remembered its truth table well enough and (2) forgetting that for material equivalence, we always need to make two entries on each side of the fork.

Exercise 8.5 (2)

(a) What would the decomposition rules for exclusive disjunction look like? Would they be definite or disjunctive? Can you see any connection with the rules for any of the other connectives?

(b) Determine the status of the following formulae by the method of decomposition trees. First, test to see whether it is a tautology. If it is not a tautology, test to see whether it is a contradiction. Your answer will specify one of the three possibilities: tautology, contradiction, contingent.

(i) $(p\rightarrow(q\rightarrow r))\rightarrow((p\rightarrow q)\rightarrow(p\rightarrow r))$
(ii) $(p\rightarrow q)\vee(q\rightarrow p)$
(iii) $(p\leftrightarrow q)\vee(q\leftrightarrow p)$
(iv) $(p\leftrightarrow q)\vee(p\leftrightarrow r)\vee(q\leftrightarrow r)$
(v) $(p\rightarrow q)\wedge(q\rightarrow r)\wedge(q\rightarrow\neg p)$
(vi) $((p\vee q)\wedge(p\vee r))\rightarrow(p\wedge(q\vee r))$.

Clearly, the method may also be applied to determine whether two formulae are tautologically equivalent, whether one formula tautologically implies another, and more generally, whether a finite set of formulae tautologically implies a formula.

- To test whether $\alpha \dashv\vdash \beta$, test whether the formula $\alpha\leftrightarrow\beta$ is a tautology.
- To test whether $\alpha \vdash \beta$, test whether the formula $\alpha\rightarrow\beta$ is a tautology.
- To test whether $\{\alpha_1,\ldots,\alpha_n\} \vdash \beta$, test whether the formula $(\alpha_1\wedge\ldots\wedge\alpha_n)\rightarrow\beta$ is a tautology.

Exercise 8.5 (3)

(a) Use the method of decomposition trees to determine whether (i) $(p\vee q)\rightarrow r \dashv\vdash (p\rightarrow r)\wedge(q\rightarrow r)$, (ii) $(p\wedge q)\rightarrow r \dashv\vdash (p\rightarrow r)\vee(q\rightarrow r)$, (iii) $p\wedge\neg p \dashv\vdash q\leftrightarrow\neg q$.

(b) Use the method of decomposition trees to determine whether $\{\neg q\vee p, \neg r\rightarrow\neg p, s, s\rightarrow\neg r, t\rightarrow p\} \vdash \neg t\wedge\neg q$.

End-of-Chapter Exercises

Exercise 8.1 Truth-functional connectives

(a) Construct the full disjunctive normal form for each of the two-place truth functions f_2, f_{11} and f_{15} constructed for Exercise 8.2 (2). Can any of them be expressed more simply?

(b) Show that the connective-set $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$ is not functionally complete. *Hint*: Think about the top row of the truth table to suggest a truth function that cannot be expressed by formulae built using only these connectives. Then use structural induction on those formulae to check out that this is really so.

(c) Show that the pair $\{\neg, \leftrightarrow\}$ is not functionally complete. This exercise is quite challenging, so we give plenty of *hints*. Aim at showing that $\{\neg, \leftrightarrow\}$ do not suffice to express $\wedge$. As a preliminary, first, show that if they do suffice to express $\wedge$, then they can do so by a formula with just two elementary letters. The main part will then be to show by structural induction that any formula with just two elementary letters and with at most $\neg$, $\leftrightarrow$ as connectives, is true in an even number of the four rows of its truth table. This can be done directly by structural induction on formulae. Alternatively, it can be done via another induction, showing that every formula with at most two letters p, q and at most those connectives is equivalent to one of eight specific formulae.

Exercise 8.2 Tautological relations

(a) In informal mathematics, we often say, when asserting a chain of equivalences, 'α iff β iff γ', just as in arithmetic, we say $a = b = c$. But what does it mean: $\alpha\leftrightarrow(\beta\leftrightarrow\gamma)$, $(\alpha\leftrightarrow\beta)\leftrightarrow\gamma$, or $(\alpha\leftrightarrow\beta)\wedge(\beta\leftrightarrow\gamma)$? Are they tautologically equivalent?

(b) Explain why the following two properties are equivalent: (i) α is a contradiction, (ii) $\alpha \vdash \beta$ for every formula β.

(c) Explain why the following three properties are equivalent: (i) α is a tautology, (ii) $\beta \vdash \alpha$ for every formula β, (iii) $\varnothing \vdash \alpha$.

(d) A set A of formulae is said to be *inconsistent* iff there is no valuation v such that $v(\alpha) = 1$ for all $\alpha \in A$. How does this concept, applied to finite sets of formulae, relate to that of a contradiction?

(e) Show, as claimed in the text, that the set of all contradictions and the relations of tautological equivalence and tautological implication are closed under substitution. *Hint*: You may do this in either of two ways – either directly from their definitions or from their connections with the property of being a tautology and the fact (shown in the text) that the set of all tautologies is closed under substitution.

(f) Show by structural induction that every formula constructed using the zero-ary connective $\perp$ and the usual $\neg,\wedge,\vee$ but without any elementary letters is either a tautology or a contradiction.

(g) In Exercise 1.4.3 (1) of Chap. 1, we saw *inter alia* that $(A\setminus B)\setminus C = (A\setminus C)\setminus B$ and $A\setminus(B\setminus C) \subseteq (A\setminus B)\cup C$ for arbitrary sets A, B, C. What would the counterpart of these be in propositional logic, expressed using $\wedge$, $\neg$, $\vee$?

Exercise 8.3 Normal forms

(a) Find a dnf for each of the following formulae, (i) $r\rightarrow\neg(q\vee p)$, (ii) $p\wedge\neg(r\vee\neg(q\rightarrow\neg p))$, using the semantic method for one and the method of successive syntactic transformations for the other.

(b)* Find a cnf for the formula $p\wedge\neg(r\vee\neg(q\rightarrow\neg p))$ by each of the three methods: (i) semantic, (ii) successive syntactic transformations, (iii) indirect method (via the dnf of its negation).

(c) For each of the following sets of formulae, find a least letter-set version: $A = \{p\leftrightarrow q, q\leftrightarrow r, r\leftrightarrow\neg p\}$, $B = \{(p\wedge q\wedge r)\vee(s\wedge r)\}$, $C = \{(p\wedge q\wedge r)\vee(s\wedge q), \neg p\}$.

(d) True or false? For each, give a proof or counterexample. (i) The least letter-set of a formula is empty iff the formula is either a tautology or a contradiction. (ii) If a letter is redundant in each of α,β, then it is redundant in $\alpha\wedge\beta$. (iii) The least letter-set of a disjunction is the union of the least letter-sets of its disjuncts.

(e)* Find most modular versions of your answers to (c).

Exercise 8.4 Semantic decomposition trees*

(a) Use a semantic decomposition tree to show that $(p\wedge q)\to r \nvdash (p\wedge(q\vee s))\to r$.

(b) Use semantic decomposition trees to determine whether each the following hold: (i) $(p\vee q)\to r \dashv\vdash (p\to r)\wedge(q\to r)$, (ii) $p\to(q\wedge r) \dashv\vdash (p\to q)\wedge(p\to r)$, (iii) $(p\wedge q)\to r \dashv\vdash (p\to r)\vee(q\to r)$, (iv) $p\to(q\vee r) \dashv\vdash (p\to q)\vee(p\to r)$.

(c) Use semantic decomposition trees to determine whether $p\to(q\to r)\vdash(p\to q)\to r$ and conversely.

Selected Reading

Introductions to discrete mathematics tend to put their chapters on logic right at the beginning. In the order of nature, this makes good sense but it also makes it difficult to use tools like set, relation and function in the presentation. One computer science text that introduces logic after having presented those notions is:

Hein J (2002) Discrete structures, logic and computability, 2nd edn. Jones and Bartlett, Boston, chapter 6

Five well-known books dedicated to elementary logic are listed below. The first is written specifically for students of computer science without much mathematics, the second for the same students but with more mathematical baggage, while the last three are aimed respectively at students of philosophy, linguistics and the general reader.

Huth M, Ryan M (2000) Logic in computer science. Cambridge University Press, Cambridge, chapter 1

Ben-Ami M (2001) Mathematical logic for computer science, 2nd edn. Springer, London, chapters 1–4

Howson C (1997) Logic with trees. Routledge, London/New York, chapters 1–4

Gamut LTF (1991) Logic, language, and meaning, vol I, Introduction to logic. University of Chicago Press, Chicago, chapters 1,2

Hodges W (1977) Logic. Penguin, Harmondsworth, sections 1–25

Something About Everything: Quantificational Logic 9

Abstract

Although fundamental, the logic of truth-functional connectives has very limited expressive power. In this chapter we go further, explaining the basic ideas of *quantificational* (alias *first-order* or again *predicate*) logic, which is sufficiently expressive to cover most of the deductive reasoning that is carried out in standard mathematics and computer science.

We begin by presenting its language, built around the *universal and existential quantifiers*, and the ways they can be used to express complex relationships. With no more than an intuitive understanding of the quantifiers, some of the basic *logical equivalences* involving them can already be appreciated. For a deeper understanding, we then present the *semantics* of the language, which is still bivalent but goes beyond truth tables. This semantics may be given in two versions, *substitutional* and *x-variant*, which however are equivalent under a suitable conditions. After explaining the distinction between *free* and *bound* occurrences of variables, and the notion of a *clean substitution*, the chapter ends with a review of some of the most important *logical implications* with quantifiers and with the identity relation.

9.1 The Language of Quantifiers

We have been using quantifiers informally throughout the book, and have made a few remarks on them in a logic box in Chap. 2. Recall that there are two quantifiers ∀ and ∃, meaning 'for all' and 'for some'. They are always used with an attached variable.

D. Makinson, *Sets, Logic and Maths for Computing*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-2500-6_9,

Table 9.1 Examples of quantified statements

	English	Symbols
1	All composer are poets	$\forall x(Cx \to Px)$
2	Some composers are poets	$\exists x(Cx \wedge Px)$
3	No poets are composers	$\forall x(Px \to \neg Cx)$
4	Everybody loves someone	$\forall x \exists y(Lxy)$
5	There is someone who is loved by everyone	$\exists y \forall x(Lxy)$
6	There is a prime number less than 5	$\exists x(Px \wedge (x < 5))$
7	Behind every successful man stands an ambitious woman	$\forall x[(Mx \wedge Sx) \to \exists y(Wy \wedge Ay \wedge Byx)]$
8	No man is older than his father	$\neg \exists x(Oxf(x))$
9	The successors of distinct integers are distinct	$\forall x \forall y[\neg(x \equiv y) \to \neg(s(x) \equiv s(y)]$
10	The English should be familiar from Chap. 4!	$[P0 \wedge \forall x\{Px \to Px+1\}] \to \forall x(Px)$

9.1.1 Some Examples

Before getting systematic, we give some examples of statements of ordinary English and their representation using quantifiers, commenting on their salient features (Table 9.1). Each of these examples raises a host of questions, whose answers will emerge as we continue through the chapter.

1. This symbolization uses the universal quantifier ∀, variable x, two predicate letters, truth-functional →. Could we use a different variable, say, y? Why are we using → here instead of, say, ∧? Can we express this using ∃ instead of ∀? What is its relation to $\forall x(Px \to Cx)$?
2. Why are we using ∧ here instead of →? Can we express this using ∀ instead of ∃? Is it logically implied by statement (1)? What is its relation to $\exists x(Cx \wedge \neg Px)$?
3. Can we express this using ∃, ∧, ¬? What is its relation to $\forall x(Cx \to \neg Px)$?
4. Why haven't we used a predicate P for 'is a person'? Does the meaning change if we write $\forall x \exists y(Lyx)$?
5. Is this equivalent to 4? Does either logically imply the other?
6. Could we somehow replace the individual constant by a predicate?
7. Can we express this equivalently with both quantifiers up the front?
8. Is it possible to express the statement more 'positively'? Could we express it with a relation symbol F and two variables?
9. Can we simplify this by converting the part inside square brackets? Does the meaning change if we reverse the initial quantifiers? Is the identity relation any different, from the point of view of logic, than other relations? Why aren't we using the usual sign = for identity?
10. Why haven't we used a predicate N for 'is a natural number'? Shouldn't the second universal quantifier take a different variable? Can we move the second quantifier right out the front? What about the first quantifier?

Exercise 9.1.1 On the basis of your informal experience with quantifiers, have a go at answering the questions. If you feel confident about most of your answers,

Table 9.2 Ingredients of the language of quantificational logic

Broad category	Specific items	Signs used	Purpose
Basic terms	Constants	$a, b, c, \ldots$	Name-specific objects: for example, 5, Charlie Chaplin, London
	Variables	$x, y, z, \ldots$	Range over specific objects, combine with quantifiers to express generality
Function letters	2-Place	$f, g, h, \ldots$	Form compound terms out of simpler terms, starting from the basic ones
	n-Place		
Predicates	1-Place	$P, Q, R, \ldots$	For example, is prime, is funny, is polluted
	2-Place		For example, is smaller than, resembles
	n-Place		For example, lies between (3-place)
	Special relation sign	$\equiv$	Identity
Quantifiers	Universal	$\forall$	For all
	Existential	$\exists$	There is
Connectives	'Not' etc.	$\neg, \wedge, \vee, \rightarrow$	Usual truth tables
Auxiliary	Parentheses and commas		To ensure unique decomposition and make formulae easier to read

congratulations – you seem to have a head start! But keep a record of your answers to check later! If on the other hand the questions leave you puzzled or uncertain, don't worry; by the end of the chapter, all will be clear.

9.1.2 Systematic Presentation of the Language

The basic components of the language of quantificational logic are set out in Table 9.2. The familiar truth-functional connectives are of course there, and the new quantifiers. But to allow the quantifiers to do their work, we also need some further ingredients.

In each of the categories of basic terms, function letters and predicates, we assume that we have an infinite supply. These can be referred to by using numerical subscripts, for example, $f_1, f_2, \ldots,$ for the function letters. We could also have numerical superscripts to indicate the arity of the function and predicate letters, so that the two-place predicate letters, say, are listed as $R_1{}^2, R_2{}^2, \ldots$ But as this is rather cumbersome, we ordinarily use a few chosen letters as indicated in the table, with context or comment indicating the arity.

How are these ingredients put together to build formulae? Recursively, as you would expect, but in two stages. First, we define recursively the notion of a *term*, and then afterwards we use that to define, again recursively, the notion of a *(well-formed) formula*. We begin with terms.

Basis: Constants and variables are terms (we call them *basic terms*, in some texts 'atomic terms').
Recursion step: If f is an n-place function symbol and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

Exercise 9.1.2 (1) (with solution) Which of the following are terms, where f is two-place and g is one-place? In the negative cases, give a brief reason. (i) a, (ii) ax, (iii) $f(x,b)$, (iv) $f(c,g(y))$, (v) $g(g(x,y))$, (vi) $f(f(b,b),f(a,y))$, (vii) $g(g(a))$, (viii) $g(g(a)$, (ix) $gg(a)$.

Solution (i) Yes, it is a basic term. (ii) No, it juxtaposes two basic terms but is not itself basic, nor is it formed using a function symbol. (iii) Yes. (iv) Yes. (v) No, g is one-place. (vi) Yes. (vii) Yes, (viii) Not quite, a right-hand parenthesis forgotten. (ix) Strictly speaking no, but this often used to abbreviate $g(g(a))$, and we will do the same in what follows.

Given the terms, we can now define formulae recursively.

Basis: If R is an n-place predicate and $t_1, \ldots, t_n$ are terms, then $R(t_1, \ldots, t_n)$ is a *formula* (sometimes called an *atomic formula*), and so is $(t_1 \equiv t_2)$ where $\equiv$ is the symbol for the identity relation.
Recursion step: If α,β are formulae and x is a variable, then the following are formulae: $(\neg\alpha)$, $(\alpha\wedge\beta)$, $(\alpha\vee\beta)$, $(\alpha \rightarrow \beta)$, $\forall x(\alpha)$, $\exists x(\alpha)$.

As in propositional logic, we drop parentheses whenever context or convention suffices to ensure unique decomposition. For ease of reading, when there are multiple parentheses, we also use different styles, for example, square and curly brackets. The special relation sign $\equiv$ for identity is customarily infixed. All other predicates and also the function symbols are prefixed; for this reason, the parentheses and commas in terms and atomic formulae without identity can in principle be omitted without ambiguity, and they will be when it improves readability.

Alice Box: Use and Mention

Alice: I notice that when you talk about a specific sign, say, the conjunction sign or the universal quantifier, you do not put it in inverted commas. For example, you say 'The symbol ∀ is the universal quantifier', rather than 'The symbol '∀' is the universal quantifier'. Is this legitimate? Isn't it like saying 'The word London has six letters' when you should be saying 'The word 'London' has six letters'?

(continued)

(continued)

Hatter: Yes and no. Strictly speaking, one should distinguish between use and mention: the *use* of a symbol like the quantifier in our object language and its *mention* when talking about it in our metalanguage. That could be done by using quotation marks, italics, underlining, a different font, etc. But when carried out systematically, it is very tiresome. As is customary in computer science and mathematics writing, we omit them except when really needed to avoid confusion.

Alice: So presentations of logic are in general rather less rigorous than the logics they present!

Hatter: I wouldn't say less rigorous. Less formal would be more accurate.

The Hatter is right. Another example of informality in our presentation can be seen in the above definition of formulae. In the recursion step, we have used one particular variable, namely, x. But what we really mean is that we can use any of the variables x, y, $z, \ldots$ in this way. So when α is a formula, then $\forall y(\alpha), \forall z(\alpha), \ldots$ are also formulae.

The reason for such informality is that it keeps down the notation. In our metalanguage, we already have *meta-variables* α, $\beta, \ldots$ ranging over formulae; we don't want to add special signs (say Gothic letters) to serve as meta-variables that range over, say, the set of variables of the object language, unless we really have to. In the early twentieth century, in obeisance to a misplaced sense of rigour, a few logic texts did attempt to work with multiple arrays of meta-variables ranging over different categories of symbols of the object language, and to use inverted commas whenever mentioning rather than using a symbol. But the result was cumbersome, annoying and almost unreadable. Nobody does it any more, although everybody knows that it *can* be done, and that for utter strictness should be done.

Exercise 9.1.2 (2) (with solution) (a) Which of the following are formulae, where R is a two-place predicate, S is a three-place predicate, and P is a one-place predicate, and f, g are as in the previous exercise? In the negative cases, give a brief reason. (i) $R(a,a)$, (ii) $R(x,b)\wedge\neg P(g(y))$, (iii) $R(x,P(b))$, (iv) $P(P(x))$, (v) $\forall(Rxy)$, (vi) $\exists P(Px)$, (vii) $\forall x\forall x(Px)$, (viii) $\forall x(Px\rightarrow Rxy)$, (ix) $\exists y\exists y[R(f(x,y),g(x))]$, (x) $\forall x(\exists y(Rxy))$, (xi) $\forall x\exists y(Rxy)$, (xii) $\exists x(Ray)$, (xiii) $\exists x\forall x(Ray)$, (xiv) $\forall x\exists y\forall z(Sxyz)$.

Solution (i) and (ii) Yes. (iii) No, $P(b)$ is a *formula*, not a *term*, and we need a term in this position if the whole expression is to be a formula. Note this carefully, as it is a common student error. (iv) No, same reason. (v) No, the quantifier does not have a variable attached to it. (vi) No, the quantifier has a predicate attached to it, rather than a variable. (vii) Yes, despite the fact that both quantifiers have the same variable attached to them. (viii), (ix) and (x) Yes. (xi) Strictly speaking, there should be another pair of parentheses, as in (x). But when we have a string of quantifiers following each other like this, we can drop such parentheses without ambiguity, and will do so. (xii) and (xiii) Yes. (xiv) Yes.

Examples (vi) and (xiv) in the exercise deserve further comment. Although (vi) is not a formula of quantificational logic, also known as first-order logic, it is admitted in what is called *second-order logic*, which goes beyond our remit. In (xiv), we have *three alternating quantifiers* in the order $\forall\exists\forall$. You may have come across this level of complexity when defining the notion of a limit in real number theory.

Exercise 9.1.2 (3) (with some solutions, hints, comments) Express the following statements in the language of quantificational logic, using naturally suggestive letters for the predicates. For example, in (a), use L for the predicate 'is a lion', T for the predicate 'is a tiger' and D for the predicate 'is dangerous'.

(a) Lions and tigers are dangerous. (b) If a triangle is right-angled, then it is not equilateral. (c) Anyone who likes Albert likes Betty. (d) Albert doesn't like everybody Betty likes. (e) Albert doesn't like anybody Betty likes. (f) Everyone who loves someone is loved by that person. (g) Everyone who loves someone is loved by someone. (h) There are exactly two prime numbers less than 5. (i) Every integer has exactly one successor. (j) My mother's father is older than my father's mother.

Some solutions, hints and comments

(a) Can be $\forall x(Lx \rightarrow Dx) \wedge \forall x(Tx \rightarrow Dx)$ or $\forall x((Lx \vee Tx) \rightarrow Dx)$. It cannot be expressed as $\forall x((Lx \wedge Tx) \rightarrow Dx)$ – try to understand why! Note that the universal quantifier is not explicit in the English, but it evidently part of the meaning.

(b) $\forall x((Tx \wedge Rx) \rightarrow \neg Ex)$. Here we are using T for 'is a triangle'. You can also simplify life by declaring the set of all triangles as your *domain* (also called *universe*) of discourse and write simply $\forall x(Rx \rightarrow \neg Ex)$. This kind of simplification, by declaring a domain of discourse, is often useful when symbolizing. But it is legitimate only if the domain is chosen so that the statement says nothing about anything outside the domain.

(c) $\forall x(Lxa \rightarrow Lxb)$. We declare our domain of discourse to be the set of all people, write Lxy stands for 'x loves y', and we use individual constants for Albert and Betty.

(d) and (e) Try to understand the difference of meaning between the two. The English words 'every' and 'any' tend do much the same work when they occur positively, but when occurring negatively, they can do quite different jobs. Declare your domain of discourse.

(f) and (g) Try to understand the difference of meaning, and declare your domain of discourse. In (f), you will need two quantifiers, both universal (despite the idiomatic English), while in (g), you will need three quantifiers.

(h) The problem here is how to say that there are *exactly two* items with a certain property. Try paraphrasing it as: There is an x and there is a y such that (1) they both have the property and (2) everything with the property is identical to at least one of them. This goes into the language of quantificational logic smoothly. Take your domain of discourse to be the set of positive integers.

(i) If you can express 'exactly two' it should be easy to express 'exactly one'.

(j) Declare your domain of discourse. Use one-place function letters for 'father of' and 'mother of'. Use a constant to stand for yourself. Be careful with order.

9.1.3 Freedom and Bondage

To understand the internal structure of a formula of quantificational logic, three notions are essential – the *scope* of a quantifier and *free* versus *bound* occurrences of a variable. We explain them through some examples.

Consider the formula $\forall z[Rxz \rightarrow \exists y(Rzy)]$. It has two quantifiers. The *scope* of each quantifier is the material in the parentheses immediately following it. Thus, the scope of the first quantifier is the material between the square brackets, and the scope of the second one likewise is given by the round brackets. Note how the scope of one quantifier may lie inside the scope of another, as in this example, or be entirely separate from it; but they never overlap.

In a quantified formula $\forall x(\alpha)$ or $\exists x(\alpha)$, the quantifier is said to *bind* the occurrence of the variable x that is attached to it and all occurrences of the same variable x occurring in α, unless some other quantifier occurring inside α already binds them. An occurrence of a variable x in a formula α is said to be *bound in* α iff there is some quantifier in α that binds it. Occurrences of a variable that are not bound in a formula are called *free* in the formula. Finally, a formula with no free occurrences of any variables is said to be *closed*, otherwise *open*. Closed formulae are often called *sentences*.

The translation into symbols of any complete sentence of English should have all its variables bound, that is, it should be closed, since a free variable is not specifying anything in particular, nor yet expressing a universal or existential quantification. Go back to Table 9.1 to check that all its symbolic representations are closed formulae.

Exercise 9.1.3 (with partial solution)

(a) Identify the free and bound occurrences of variables in the formula $\forall z[Rxz \rightarrow \exists y(Rzy)]$. Use marks from above to mark the bound occurrences, marks from below for the free ones.

(b) In the formula $\forall x\exists y(Rxy) \vee \exists z(Py \rightarrow \forall x(Rzx \wedge Ryx))$, replace some of the round brackets by square brackets to help make salient the scopes of the quantifiers in the formula. Add further square brackets for those that are implicit under the convention for writing strings of contiguous quantifiers. Identify free and bound occurrences of variables in the formula.

(c) Use the last example to illustrate how a single variable can have one occurrence bound, another free, in the same formula.

(d) Give a simple example to illustrate how an occurrence of a variable may be bound in a formula but free in one of its subformulae.

(e) Consider the formula $\forall x\exists y\forall x\exists y(Rxy)$. Which quantifiers bind which occurrences of x? Which bind which occurrences of y?

(f) Which of the formulae mentioned in this Exercise are closed?

(g) Express the definition of an occurrence of a variable being free in a formula more rigorously, as a recursion on the structure of the formula.

Solutions to (b)–(f)

(b) $\forall x\exists y(Rxy)\vee\exists z[Py\to\forall x(Rzx\wedge Ryx)]$ and $\forall x[\exists y(Rxy)]\vee\exists z[Py\to\forall x(Rzx\wedge Ryx)]$ with added parentheses.

(c) The occurrences of y to the left of $\vee$ are bound, those on the right are free.

(d) In the last example, x is bound in the whole formula, and also in the subformula $\forall x\exists y(Rxy)$, but is free in the subformula $\exists y(Rxy)$.

(e) In the formula $\forall x\exists y\forall x\exists y(Rxy)$, the x in Rxy is bound by the innermost $\forall$. The outer $\forall$ binds only its attached occurrence of x. Likewise, the y in Rxy is bound by the innermost $\exists$. The outer $\exists$ binds only its attached occurrence of y.

(f) Closed formula: $\forall x\exists y\forall x\exists y(Rxy)$. Open: $\forall x\exists y(Rxy)\vee\exists z(Py\to\forall x(Rzx\wedge Ryx))$, $\forall z[Rxz\to\exists y(Rzy)]$.

9.2 Some Basic Logical Equivalences

At this point, we could set out the semantics for quantificational formulae, with rigorous definitions of logical relationships such as consequence and equivalence. But we will postpone that until the next section and, in the meantime, cultivate intuitions. There are a number of basic logical equivalences that can already be appreciated when you read $\forall x(\alpha)$ and $\exists x(\alpha)$ intuitively as saying, respectively, 'α holds for every x in our domain of discourse' and 'α holds for at least one x in our domain of discourse'.

9.2.1 Quantifier Interchange

First among these equivalences are the *quantifier interchange* principles, which show that anything expressed by one of our two quantifiers may equivalently be expressed by the other, with the judicious help of negation. In Table 9.3, the formulae on the left are logically equivalent to those on the right.

Here, α stands for any formula of quantificational logic. We will reuse the sign $\dashv\vdash$, familiar from propositional logic, for the intuitively understood, but not yet formally defined, concept of logical equivalence for quantificational formulae.

Alice Box: **$\exists x(\neg\alpha)$ or $\exists x\neg(\alpha)$?**

Alice: A question about the right column in the table. Should we write $\exists x(\neg\alpha)$, $\forall x(\neg\alpha)$ or $\exists x\neg(\alpha),\forall x\neg(\alpha)$?

Hatter: *Hatter*: It doesn't matter. With all brackets present, we would have $\exists x(\neg(\alpha)),\forall x(\neg(\alpha))$, and we can leave off one of the two pairs without ambiguity.

Table 9.3 Quantifier interchange equivalences

$\neg\forall x(\alpha)$	$\exists x(\neg\alpha)$
$\neg\exists x(\alpha)$	$\forall x(\neg\alpha)$
$\forall x(\alpha)$	$\neg\exists x(\neg\alpha)$
$\exists x(\alpha)$	$\neg\forall x(\neg\alpha)$

Actually, any one of the four equivalences can be obtained from any other by means of double negation and the *principle of replacement of logically equivalent formulae*. For example, suppose we have the first one and want to get the last one. Since $\alpha \dashv\vdash \neg\neg\alpha$, we have $\exists x(\alpha) \dashv\vdash \exists x(\neg\neg\alpha) \dashv\vdash \neg\forall x(\neg\alpha)$ by replacement and then the first equivalence, and we are done.

Exercise 9.2.1 (1)

(a) Obtain the second and third quantifier interchange principles from the first one by a similar procedure.

(b) Use quantifier interchange and suitable truth-functional equivalences to show that (i) $\neg\forall x(\alpha\to\beta) \dashv\vdash \exists x(\alpha\wedge\neg\beta)$, (ii) $\neg\exists x(\alpha\wedge\beta) \dashv\vdash \forall x(\alpha\to\neg\beta)$, (iii) $\forall x(\alpha\to\beta) \dashv\vdash \neg\exists x(\alpha\wedge\neg\beta)$ and (iv) $\exists x(\alpha\wedge\beta) \dashv\vdash \neg\forall x(\alpha\to\neg\beta)$. Make free use of the principle of replacement in your work.

The equivalences in the exercise are important, because the formulae correspond to familiar kinds of statement in English. For example, in (b)(i), the left side says 'not all αs are βs', while the right one says 'at least one α is not a β'.

Exercise 9.2.1 (2) To what English statements do the equivalences (b)(ii) through (b)(iv) correspond?

9.2.2 Distribution

The next group of equivalences may be described as *distribution principles*. They show the way in which universal quantification distributes over conjunction, while the existential distributes over disjunction (Table 9.4).

Why does the universal quantifier get on so well with conjunction? Essentially because it is like a long conjunction itself. Suppose our domain of discourse has just n elements, named by n individual constants $a_1, \ldots, a_n$. Then saying $\forall x(Px)$ amounts to saying, of each element in the domain, that it has the property P, that is, that $Pa_1 \wedge \ldots \wedge Pa_n$. This formula is called a *finite transform* of $\forall x(Px)$.

Of course if we change the size of the domain, then we change the length of the conjunction; and to make the transform work, we must have enough constants to name all the elements of the domain. But even with these provisos, it is clear that the principles for $\forall$ reflect those for $\wedge$, while those for $\exists$ resemble familiar ones for $\vee$. The idea of working with such *finite transforms* of quantified formulae can be put to systematic use to obtain negative results. We will return to this later.

Table 9.4 Distribution equivalences

$\forall x(\alpha\wedge\beta)$	$\forall x(\alpha)\wedge\forall x(\beta)$
$\exists x(\alpha\vee\beta)$	$\exists x(\alpha)\vee\exists x(\beta)$

Exercise 9.2.2

(a) Write out $\forall x(\alpha\wedge\beta)$ as a conjunction in a domain of three elements, with α taken to be the atomic formula Px and β the atomic formula Qx. Then write out $\forall x(\alpha)\wedge\forall x(\beta)$ in the same way and check that they are equivalent. What truth-functional principles do you appeal to in that check?

(b) What does $\exists x(Px)$ amount to in a domain of n elements?

(c) Write out $\exists x(\alpha\vee\beta)$ as a disjunction in a domain of three elements, with α chosen as Px and β as Qx. Then write out $\exists x(\alpha)\vee\exists x(\beta)$ in the same way and check that they are equivalent. What truth-functional principles do you appeal to?

(d) If we think of the universal and existential quantifier as expressing generalized conjunctions and disjunctions, respectively, to what familiar truth-functional equivalences do the entries in Table 9.3 correspond?

(e) Illustrate your answer to (d) by constructing finite transforms for the third row of Table 9.3 in a domain of three elements with α chosen as Px.

9.2.3 Vacuity and Relettering

The last two intuitive equivalences that we mention are the vacuity and relettering principles. The vacuity equivalences are expressed in Table 9.5, where all formulae in a given row are equivalent.

In other words, quantifying twice on the same variable does no more than doing it once. To say 'for every x it is true that for every x we have Px', as in the top row middle column, is just a long-winded way of saying 'Px for every x'. Note that in these vacuity equivalences, *the variables must be the same*: $\forall x(Rxy)$ is *not* logically equivalent to either of $\forall y\forall x(Rxy)$ and $\exists y\forall x(Rxy)$.

The equivalences of the table are special cases of a general *principle of vacuous quantification*: *When there are no free occurrences of x in* φ, *then each of* $\forall x(\varphi)$ *and* $\exists x(\varphi)$ *is logically equivalent to* φ. To see how the table fits under this principle, in the top row, take φ to be $\forall x(\alpha)$, and in the bottom row, take φ to be $\exists x(\alpha)$.

Exercise 9.2.3 (1) Use vacuity and distribution equivalences to show that $\forall x[Px\rightarrow\exists y(Qy)] \dashv\vdash \forall x\exists y[Px\rightarrow Qy]$ and $\exists x[Px\wedge\forall y(Qy)] \dashv\vdash \exists x\forall y[Px\wedge Qy]$. Do you notice anything interesting about the shapes of the two right-hand formulae? *Hint*: Begin the first equivalence by getting rid of the conditional in favour of disjunction and negation, using propositional logic and the replacement rule.

For the relettering equivalence, we begin with some examples. Intuitively, it is clear that $\forall x(Px) \dashv\vdash \forall y(Py)$, although $\forall x(Px\wedge Qy)$ does not seem to be equivalent to $\forall y(Py\wedge Qy)$. Again, $\forall x\exists y(Rxy) \dashv\vdash \forall x\exists z(Rxz) \dashv\vdash \forall y\exists z(Ryz) \dashv\vdash \forall y\exists x(Ryx)$ although $\forall x\exists y(Rxy)$ is certainly not equivalent to $\forall x\exists x(Rxx)$.

Table 9.5 Vacuity equivalences

$\forall x(\alpha)$	$\forall x \forall x(\alpha)$	$\exists x \forall x(\alpha)$
$\exists x(\alpha)$	$\exists x \exists x(\alpha)$	$\forall x \exists x(\alpha)$

The useful principle of *relettering of bound variables* may be put as follows. Let φ be a formula and x any variable. Let y be a *fresh* variable, in the sense that it does not occur at all (whether free or bound) in φ. Then $\varphi \dashv\vdash \varphi'$, where φ' is obtained by replacing all bound occurrences of x in φ by y. The proviso that y is fresh for φ is essential!

Exercise 9.2.3 (2) (with solution)

(a) Use iterated relettering to show that $\exists x \forall y(Rxy) \dashv\vdash \exists y \forall x(Ryx)$.
(b) Can relettering be used to show that $\exists x \forall y(Rxy) \dashv\vdash \exists y \forall x(Rxy)$?
(c) Is there an analogous principle of relettering of free variables?

Solution

(a) You need to proceed in several steps, bringing in and eliminating fresh variables z,w: $\exists x \forall y(Rxy) \dashv\vdash \exists z \forall y(Rzy) \dashv\vdash \exists z \forall w(Rzw) \dashv\vdash \exists y \forall w(Ryw) \dashv\vdash \exists y \forall x(Ryx)$.
(b) No. Roughly speaking, the reason is that no matter how many replacements you make, the variable attached to each quantifier will continue to occupy the same place in the part that is quantified. In fact, in this example, the left- and right-hand sides of (b) are not equivalent.
(c) No. For example, Px is not logically equivalent to Py. However, there is a weaker principle for free variables. Let φ be any formula and x any variable. Let y be a variable fresh to φ, and form φ' by replacing all *free* occurrences of x in φ by y. Then φ is logically true iff φ' is logically true. If the subtle difference between this weaker but valid principle and the stronger but invalid one is not intuitively clear to you, return to it after completing the chapter.

9.3 Two Semantics for Quantificational Logic

It is time to get rigorous. In this section, we present the semantics for quantificational logic. To keep a sense of direction, remember that we are working towards analogues for our enriched language of concepts familiar from propositional logic: a valuation, logical consequence, logical truth, etc.

9.3.1 The Common Part of the Two Semantics

We begin with the notion of an *interpretation with universe D*, where D is a non-empty set. This is defined to be any function v that assigns a value to each constant, variable, predicate letter and function letter of the language in the following way:

- For each constant a, $v(a)$ is an element of the domain, that is, $v(a) \in D$.
- Likewise, for each variable x, $v(x)$ is an element of the domain, that is, $v(x) \in D$.

- For each n-place function letter f, $v(f)$ is an n-argument function on the domain, that is, $v(f)$: $D^n \to D$.
- For each n-place predicate P other than the identity symbol, $v(P)$ is an n-place relation over the domain, that is, $v(P) \subseteq D^n$.
- For the identity symbol, we put $v(\equiv)$ to be the identity relation over the domain.

The last clause of this definition means that we are giving the identity symbol a special treatment. Whereas every other predicate symbol may be interpreted as any relation over the domain of the right arity (i.e. number of places), the identity symbol is always read as *genuine identity* over the domain. Semantics following this rule are often said to be *standard* with respect to identity. It is also possible to give a non-standard treatment, in which the identity sign is interpreted as any equivalence relation over the domain that is also congruential with respect to all function letters and predicates. However, we will not go further into the non-standard approach, confining ourselves as is customary to standard interpretations.

Given an interpretation with domain D, we can define the value $v^+(t)$ of a term recursively, following the definition of the terms themselves. Remember that terms are built from constants and variables by applying function letters. The recursive definition of v^+ follows suit, with two base clauses and a recursion step.

$$v^+(a) = v(a) \text{ for every constant } a \text{ of the language}$$
$$v^+(x) = v(x) \text{ for every variable } x \text{ of the language}$$
$$v^+(f(t_1, \ldots, t_n)) = (v^+(f))(v^+(t_1), \ldots, v^+(t_n)).$$

The recursion step needs some commentary. We are in effect defining $v^+(f(t_1, \ldots, t_n))$ *homomorphically*. The value of a term $f(t_1, \ldots, t_n)$ is defined by taking the interpretation of the function letter f, which will be a function on D^n into D, and applying that function to the interpretations of the terms $t_1, \ldots, t_n$, which will all be elements of D, to get a value that will also be an element of D. The notation makes it look complicated at first, but the underlying idea is natural and simple.

As v^+ is a uniquely determined extension of v, we will follow the same ‘abuse of notation’ as in propositional logic and write it simply as v. There are some contexts in which the notation could be simplified further. When considering only *one* interpretation function v, we could reduce clutter by writing $v(t)$ as $\underline{t}$, simply underlining the term t. But in contexts where we are considering more than one interpretation (which is most of the time), this simple notation is not open to us, unless of course we use more than one kind of underlining.

Exercise 9.3.1 Rewrite the recursion step of the definition of v^+ using the underlining notation.

Finally, we may go on to define the notion of the *truth* or *falsehood* of a formula α under an interpretation v with domain D. Once again, our definition is recursive, following the recursive construction of the formulae. We are defining a function v: $L \to \{0,1\}$ from the set L of all formulae into the set $\{0,1\}$. We will merely specify when $v(\alpha) = 1$, leaving it understood that otherwise $v(\alpha) = 0$.

- *Basis, first part*: For any atomic formula of the form $Pt_1, \ldots, t_n$: $v(Pt_1, \ldots, t_n) = 1$ iff $(v(t_1), \ldots, v(t_n)) \in v(P)$.
- *Basis, second part*: For any atomic formula of the form $t_1 \equiv t_2$: $v(t_1 \equiv t_2) = 1$ iff $(v(t_1), v(t_2)) \in v(\equiv)$. Since $v(\equiv)$ is required always to be the identity relation over D, this means that $v(t_1 \equiv t_2) = 1$ iff $v(t_1) = v(t_2)$.

The recursion step also has two parts, one for the truth-functional connectives and the other for the quantifiers. The first part is easy.

- *Recursion step for the truth-functional connectives*: $v(\neg\alpha) = 1$ iff $v(\alpha) = 0$ and so on for the other truth-functional connectives, just as in propositional logic.

The *recursion step for the quantifiers* is the heart of the semantics. But it is subtle and needs careful formulation. In the literature, there are two ways of going about it. They are equivalent under a suitable condition, and from an abstract point of view, one can see them as essentially two ways of doing the same thing. But passions can run high over which is better to use, and it is good policy to understand both of them. They are known as the *substitutional* and *x-variant* readings of the quantifiers. We explain each in turn, beginning with the substitutional reading as students usually find it easier to grasp.

9.3.2 Substitutional Reading

This way of interpreting the quantifiers is not often used by mathematicians, but it is often employed by computer scientists and philosophers. Given a formula α and a variable x, write $\alpha[t/x]$ to stand for the result of *substituting the term t for all free occurrences of x in* α. For example, if α is the formula $\forall z[Rxz \rightarrow \exists x(Rzx)]$, then $\alpha[a/x] = \forall z[Raz \rightarrow \exists x(Rzx)]$, obtained by replacing the unique free occurrence of x in α by the constant a (leaving bound occurrences of x untouched).

Exercise 9.3.2 (1)

(a) Let α be the formula $\forall z[Rxz \rightarrow \exists x(Rzx)]$. Write out $\alpha[b/x]$, $\alpha[b/y]$.
(b) Do the same for the formula $\forall x\exists y(Rxy) \vee \exists z[Py \rightarrow \forall x(Rzx \wedge Ryx)]$.

We are now ready to give the substitutional reading of the quantifiers under an interpretation v with domain D. We first make sure that the function v, restricted to the set of all constants of the language, is *onto D*, in other words, that every element of D is the value under v of some constant symbol. If it is not already onto D, we add enough constants to the language to be able to extend the interpretation function to ensure that it is onto D. Then, we evaluate the quantifiers as follows:

$v(\forall x(\alpha)) = 1$ iff $v(\alpha[a/x]) = 1$ for every constant symbol a of the language (thus expanded).

$v(\exists x(\alpha)) = 1$ iff $v(\alpha[a/x]) = 1$ for at least one constant symbol a of the language (thus expanded).

The reason for requiring that every element of the domain is the value, under v, of some constant symbol, is to guarantee that $\forall x$ really means 'for every element in the domain' and not merely 'for every element of the domain that happens to have a name in our language'.

Alice Box: Oddities

Alice: One moment, there is something funny here! The requirement that every element of the domain D is the image under v of some constant symbol has a strange consequence. The language itself is not fixed, since the supply of constants depends on the particular domain of discourse under consideration. That's odd!

Hatter: Odd, yes, but not incoherent. It works perfectly well. But it is one reason why some logicians prefer the other reading of the quantifiers, which we will give shortly.

Alice: Another puzzle. Reading the above rules, it seems that we are using quantifiers in our metalanguage when defining the semantics of quantifiers in the object language.

Hatter: Sure, just as we used truth-functional connectives in the metalanguage when defining their semantics in the object language. There's no other way.

Exercise 9.3.2 (2) (with partial solution)

(a) With a domain D consisting of two items 1 and 2, construct an interpretation that makes the formula $\forall x(Px \vee Qx)$ true but makes $\forall x(Px) \vee \forall x(Qx)$ false. Show your calculations of the respective truth-values.

(b) Do the same for the pair $\exists x(Px) \wedge \exists x(Qx)$ and $\exists x(Px \wedge Qx)$.

(c) Again for the pair $\forall x \exists y(Rxy)$ and $\exists y \forall x(Rxy)$.

Sample solution to (a) Put $v(P) = \{1\}$ and $v(Q) = \{2\}$, and let a, b be constants with $v(a) = 1$ and $v(b) = 2$. We claim that $v(\forall x(Px \vee Qx)) = 1$ while $v(\forall x(Px) \vee \forall x(Qx)) = 0$. For the former, it suffices to show that both $v(Pa \vee Qa) = 1$ and $v(Pb \vee Qb) = 1$. But since $v(Pa) = 1$, we have $v(Pa \vee Qa) = 1$, and likewise since $v(Qb) = 1$, we have $v(Pb \vee Qb) = 1$. For the latter, $v(Pb) = 0$ so that $v(\forall x(Px)) = 0$, and likewise $v(Qa) = 0$ so that $v(\forall x(Qx)) = 0$, and so finally, $v(\forall x(Px) \vee \forall x(Qx)) = 0$ as desired.

The substitutional account of the quantifiers throws light on the finite transforms of a quantified formula, which we discussed briefly in the preceding section. Let v be any interpretation with domain D. Then, under the substitutional reading, for any universally quantified formula, we have $v(\forall x(\alpha)) = 1$ iff $v(\alpha[a/x]) = 1$ for every constant symbol a of the language. If D is finite and $a_1, \ldots, a_n$ are constants naming all its elements, then this holds iff $v(\alpha[a_1/x] \wedge \ldots \wedge (\alpha[a_n/x]) = 1$. Similarly, for the existential quantifier, $v(\exists x(\alpha)) = 1$ iff $v(\alpha[a_1/x] \vee \ldots \vee (\alpha[a_n/x]) = 1$.

Thus, the truth-value of a formula under an interpretation v with a finite domain D of n elements can also be calculated by a *translation into a quantifier-free formula* (i.e. with only truth-functional connectives). We recursively translate $\forall x(\alpha)$ and $\exists x(\alpha)$ into $\alpha[a_1/x] \wedge \ldots \wedge \alpha[a_n/x]$ and $\alpha[a_1/x] \vee \ldots \vee \alpha[a_n/x]$, respectively, where $a_1, \ldots, a_n$ are constants chosen to name all elements of D.

Exercise 9.3.2 (3) (with partial solution) Do Exercise 9.3.2 (2) again, but this time translating into quantifier-free formulae and assigning truth-values.

Solution Take constants a, b. The translation of $\forall x(Px \lor Qx)$ is $(Pa \lor Qa) \land (Pb \lor Qb)$ while the translation of $\forall x(Px) \lor \forall x(Qx)$ is $(Pa \land Pb) \lor (Qa \land Qb)$. Let v be the propositional assignment that, for example, puts $v(Pa) = 1 = v(Qb)$ and $v(Pb) = 0 = v(Qa)$. Then, by truth tables, $v((Pa \lor Qa) \land (Pb \lor Qb)) = 1$ while $v((Pa \land Pb) \lor (Qa \land Qb)) = 0$ as desired.

9.3.3 The *x*-Variant Reading

We now explain the rival x-variant reading of the quantifiers. Roughly speaking, whereas the substitutional reading of a formula $\forall x(\alpha)$ under a valuation v looked at the values of certain formulae *different* from α under the *same* valuation v, the x-variant reading will consider the values of the *same* formula α under certain valuations, most of which are *different* from v.

Let v be any interpretation with domain D, and let x be any variable of the language. By an *x-variant* of v, we mean any interpretation v' with the same domain D that agrees with v in the values given to all constants, function letters and predicates (i.e. $v'(a) = v(a)$, $v'(f) = v(f)$, $v'(P) = v(P)$ for all letters of the respective kinds), and also agrees on the interpretation given to all variables *except possibly the particular variable* x (so that $v'(y) = v(y)$ for every variable y of the language other than the variable x). With this notion in hand, we evaluate the quantifiers for a valuation v with domain D, as follows:

$v(\forall x(\alpha)) = 1$ iff $v'(\alpha) = 1$ for every x-variant interpretation v of v.
$v(\exists x(\alpha)) = 1$ $v'(\alpha) = 1$ for at least one x-variant interpretation v of v.

It is instructive to re-solve Exercise 9.3.2 (2) using the x-variant reading of the quantifiers and compare the calculations.

Exercise 9.3.3 (with partial solution)

(a) Construct an interpretation v with domain $D = \{1,2\}$, such that under the x-variant reading of the quantifiers $v(\forall x(Px \lor Qx)) = 1$ while $v(\forall x(Px) \lor \forall x(Qx)) = 0$. Show your calculations.

(b) Do the same for the pair $\exists x(Px) \land \exists x(Qx)$ and $\exists x(Px \land Qx)$.

(c) Again for the pair $\forall x \exists y(Rxy)$ and $\exists y \forall x(Rxy)$.

Sample solution to (a) Put $v(P) = \{1\}$, $v(Q) = \{2\}$ and $v(x)$ any element of D (it will not matter how it is chosen in D since the formulae we are evaluating are closed). We claim that $v(\forall x(Px \lor Qx)) = 1$ while $v(\forall x(Px) \lor \forall x(Qx)) = 0$. For the former, it suffices to show that $v'(Px \lor Qx) = 1$ for every x-variant interpretation v'. But there are only two such x-variants, $v_{x=1}$ and $v_{x=2}$, defined by putting $v_{x=1}(x) = 1$ and $v_{x=2}(x) = 2$. Then, we have $v_{x=1}(Px) = 1$ while $v_{x=2}(Qx) = 1$, so $v_{x=1}(Px \lor Qx) = 1 = v_{x=2}(Px \lor Qx)$ and, thus, $v(\forall x(Px \lor Qx)) = 1$. For the latter, we have $v_{x=2}(Px) = 0 = v_{x=1}(Qx)$ so that $v(\forall x(Px)) = 0 = v(\forall x(Qx))$, and so finally, $v(\forall x(Px) \lor \forall x(Qx)) = 0$ as desired.

In this exercise, we have used a notation for x-variants ($v_{x=1}$, $v_{x=2}$, etc.) that permits us to keep track of what is going on. It also runs parallel to the notation for the substitutional reading ($v_{x=1}(\alpha)$, etc., in place of $v(\alpha[1/x])$, etc.), making it clear that at some deep level, the two procedures are 'doing the same thing'.

Evidently, detailed truth-verifications like those in the exercises are rather tedious, and having done them, you already begin to 'see' the value of fairly simple formulae under a given interpretation, without writing out all the details. It should also be clear from the exercises that the truth-value of a formula under an interpretation is independent of the value given to the variables that do not occur free in it. For example, in the formula $\forall x(Px \vee Qx)$, the variable x does not occur free – all its occurrences are bound – and so its truth-value under an interpretation v is independent of the value of $v(x)$, although, of course, the truth-value of $Px \vee Qx$ under an interpretation v does in general depend on the value of $v(x)$.

Alice Box: Values of Formulae That Are Not Closed

Alice: There is something about this that bothers me. I have been browsing in several other books on logic for students of mathematics rather than computer science. They use the x-variant account but in a rather different way. Under the definition above, every formula comes out as *true*, or as *false*, under a given interpretation with specified domain, that is, we always have either $v(\alpha) = 1$ or $v(\alpha) = 0$. But the texts that I have been looking at insist that when a formula α has free occurrences of variables in it (in other words, is not closed, is not a sentence), then in general it is *neither true nor false under an interpretation*! What is going on?

Hatter: That is another way of building the x-variant semantics. It is different in some of the terminology and details, but in the end, it does the same job. The two accounts agree on closed formulae.

As usual, the Hatter has given the short answer. Alice deserves rather more, although students who have not been exposed to the other way of proceeding may happily skip to the next section.

Under the alternative account, terms and formulae are evaluated under an interpretation in exactly the same way as we have done but with a terminological difference. Rather than speaking of truth and falsity and writing $v(\alpha) = 1$ and $v(\alpha) = 0$, the interpretation is said to 'satisfy' or 'not satisfy' the formula, writing $v \vDash \alpha$ or $v \nvDash \alpha$. A 'model' is understood as an interpretation stripped of the values assigned to variables, and a formula is called be true in a model itself iff it is satisfied by *every interpretation* extending that model, that is, every interpretation that agrees with the model but in addition gives values to variables. A formula is said to be false in the model iff it is satisfied by *none* of those interpretations.

A formula containing free occurrences of variables will in general be neither true nor false in a given model: some assignments of values to its free variables may satisfy it, others not. The simplest example is the basic formula Px. The model with domain $D = \{1,2\}$ and $v(P) = \{1\}$ does not make Px true, nor false, since the interpretation with $v(x) = 1$ satisfies Px while the interpretation with $v(x) = 2$ does not. But when a formula is closed (i.e. has no free occurrences of variables) then the gap disappears: for any given model, a closed formula is either true in the model or false in the model. Moreover, it turns out that an arbitrary formula α is true in a model iff its universal closure $\forall x_1 \ldots \forall x_n(\alpha)$ is true in that same model.

The reason why this text (like quite a few others) has not followed the alternative manner of presentation is that it creates unnecessary difficulties for students. Having become accustomed to bivalent 'truth' and 'falsehood' in propositional logic, when they come to quantificational logic, they are asked to express the same bivalence using the different terms 'satisfaction' and 'non-satisfaction' and to employ the terms 'truth' and 'falsehood' in a non-bivalent way. A sure recipe for classroom confusion!

9.4 Semantic Analysis

We now apply the semantic tools developed in Sect. 9.3 to analyze the notion of logical implication in quantificational languages and establish some fundamental rules – even more fundamental than the equivalences listed in Sect. 9.2.

9.4.1 Logical Implication

We have all the apparatus for extending our definitions of relations such as logical implication from propositional to quantificational logic. We simply take the same formulations as for truth-functional logic and plug in the more elaborate notion of an interpretation:

- A set A of formulae is said to *logically imply* a formula β iff there is no interpretation v such that $v(\alpha) = 1$ for all $\alpha \in A$ but $v(\beta) = 0$. This relation is written with the same 'gate' or 'turnstile' sign as in propositional logic, $A \vdash \beta$, and we also say that β is a *logical consequence* of A.
- α is *logically equivalent* to β, and we write $\alpha \dashv\vdash \beta$, iff both $\alpha \vdash \beta$ and $\beta \vdash \alpha$. Equivalently, $\alpha \dashv\vdash \beta$ iff $v(\alpha) = v(\beta)$ for every interpretation v.
- α is a *logically true* iff $v(\alpha) = 1$ for every interpretation v.
- A formula α is a *contradiction* iff $v(\alpha) = 0$ for every interpretation v. More generally, a set A of formulae is *inconsistent* iff for every interpretation v there is a formula $\alpha \in A$ with $v(\alpha) = 0$.
- α is *contingent* iff it is neither logically true nor a contradiction.

Evidently, the five bulleted concepts above are interrelated in the same way as their counterparts in propositional logic (see Sect. 8.3.4) with the same verifications. For example, α is logically true iff $\varnothing \vdash \alpha$.

In the exercises of Sect. 9.3, we checked some negative results. In effect, we saw that $\forall x(Px \vee Qx) \nvdash \forall x(Px) \vee \forall x(Qx)$, $\exists x(Px) \wedge \exists x(Qx) \nvdash \exists x(Px \wedge Qx)$ and $\forall x \exists y(Rxy)$

$$\begin{tikzcd}
\exists x\exists y(...) \arrow[r, leftrightarrow] & \exists y\exists x(...) \\
\forall x\exists y(...) \arrow[u] & \forall y\exists x(...) \arrow[u] \\
\exists x\forall y(...) \arrow[u, phantom] \arrow[ur] & \exists y\forall x(...) \arrow[ul] \\
\forall x\forall y(...) \arrow[u] \arrow[r, leftrightarrow] & \forall y\forall x(...) \arrow[u]
\end{tikzcd}$$

Fig. 9.1 Logical relations between alternating quantifiers

$\nvdash \exists y\forall x(Rxy)$. In each instance, we found an interpretation in a very small domain (two elements) that does the job. For more complex non-implications, one often needs larger domains. Indeed, there are formulae α, β such that $\alpha \nvdash \beta$ but such that $v(\beta) = 1$ for every interpretation v in any finite domain with $v(\alpha) = 1$. In other words, such non-implications can be witnessed only in infinite domains (which, however, can always be chosen to be countable). But that is beyond our remit, and we remain with more elementary matters.

The last of the three non-implications above suggests the general question of which alternating quantifiers imply which. This is answered in Fig. 9.1.

The double-headed arrows in the diagram indicate logical equivalence; single-headed ones are for logical implication. We are looking at formulae with two initial quantifiers with attached variables; the left column changes the quantifiers leaving the attached variables in the fixed order x,y ($2^2 = 4$ cases), while the right column does the same with the attached variables in reverse order y,x (another $2^2 = 4$ cases), thus 8 cases in all. In each case, the material quantified remains unchanged. The diagram is complete in the sense that when there is no path from φ to ψ in the figure, then $\varphi \nvdash \psi$.

Exercise 9.4.1 (1) Verify the non-implication $\exists x\forall y(\alpha) \nvdash \exists y\forall x(\alpha)$ by choosing α to be Rxy and considering a suitable interpretation. Use either the x-variant or substitutional account of the quantifier in your verification, as you prefer. *Advice*: (1) Make life easy by using the smallest domain that you can get away with. (2) If using the x-variant reading, use the same subscript notation as in the solution to Exercise 9.3.3.

So far, we have been using the semantics to get negative results. We now use it to get a positive one by verifying the implication $\forall x(\alpha) \vdash \exists x(\alpha)$ using the substitutional reading of the quantifiers. Let v be a valuation with domain D, and suppose $v(\forall x(\alpha)) = 1$. Then for every constant a of the language, we have $v(\alpha[a/x]) = 1$. Since the domain is assumed to be non-empty and every element of the domain is given a constant as name, there is at least one constant in the language, and so we have $v(\alpha[a/x]) = 1$ for at least one constant a, so $v(\exists x(\alpha)) = 1$ as desired.

Alice Box: That's a Fraud!

Alice: You make me laugh! You are trying to prove something intuitively obvious, and in proving it, you use the very same principle to prove it (OK, it is in the metalanguage rather than the object language), and the resulting argument is more complicated and less transparent than the original fact. That's a bit of a fraud!

Hatter: A disappointment, yes; and a pain in the neck, I agree. But not a fraud. To establish a principle about the object language, we must reason in our metalanguage, and if we make everything explicit, it can become rather complex.

For our second example, we show that $\exists x\forall y(\alpha) \vdash \forall y\exists x(\alpha)$ using this time the *x*-variant reading of the quantifiers. Let *v* be a valuation with domain *D*, and suppose $v(\exists x\forall y(\alpha)) = 1$ while $v(\forall y\exists x(\alpha)) = 0$; we get a contradiction. By the first supposition, there is an *x*-variant of *v*, which we write as $v_{x=a}$, with $v_{x=a}(\forall y(\alpha)) = 1$. By the second supposition, there is a *y*-variant $v_{y=b}$ of *v* with $v_{y=b}(\exists x(\alpha)) = 0$. Now take the *y*-variant $v_{x=a,y=b}$ of $v_{x=a}$. Then by the first remark, $v_{x=a,y=b}(\alpha) = 1$, while by the second remark, $v_{y=b,x=a}(\alpha) = 0$. But clearly, the two valuations $v_{x=a,y=b}$ and $v_{y=b,x=a}$ are the same, so long as *x* and *y* are distinct variables, as presupposed in the formulation of the problem. For the former is obtained by respecifying the value given to *x* and then respecifying the value given to the different variable *y*, while the latter respecifies in reverse order. So $v_{x=a,y=b}(\alpha) = 1$ while $v_{x=a,y=b}(\alpha) = 0$, giving us a contradiction.

Exercise 9.4.1 (2)

(a) Do the first example again using the *x*-variant method, and the second example using the substitutional method. *Hint*: Follow the same patterns, just changing the notation.

(b) Verify one of the quantifier interchange principles, one of the distribution principles, and one of the vacuous quantification principles of Sect. 9.2. In each case, give a proof using the substitutional method or the *x*-variant method one, as you prefer, but using each method at least once.

(c) Correct or incorrect? (i) $\forall x(Px \to Qx) \vdash \forall x(Px) \to \forall x(Qx)$, (ii) the converse. Same instructions as for (b).

Alice Box: This Is Tedious . . .

Alice: Do I have to do much of this? I've got the idea, but the calculations are tedious, and by now I can guess the results intuitively in simple examples.

Hatter: One example at the end of the chapter for revision.

Alice: Will these calculations be in the exam?

Hatter: You'd better ask your instructor.

9.4.2 Clean Substitutions

All the logical relations between quantifiers rest on four fundamental principles that we have not yet enunciated. To formulate them, we need a further concept – that of a *clean* substitution of a term t for a variable x.

Recall that a term t may contain variables as well as constants and function letters; indeed, a variable standing alone is a term. So when we substitute t for free occurrences of x in α, it may happen that t contains a variable y that is 'captured' by some quantifier of α. For example, when α is the formula $\exists y(Rxy)$ and we substitute y for the sole free occurrence of x in it, we get $\alpha[y/x] = \exists y(Ryy)$, where the y that is introduced is in the scope of the existential quantifier. Such substitutions are undesirable from the point of view of the principles that we are about to articulate. We say that a substitution $\alpha[t/x]$ is *clean* iff no free occurrence of x in α falls within the scope of a quantifier binding some variable occurring in t.

Exercise 9.4.2 (with solution) Let $\alpha = \exists y\{Py \vee \forall z[Rzx \wedge \forall x(Sax)]\}$. Which of the following substitutions are clean: (i) $\alpha[y/x]$, (ii) $\alpha[a/x]$, (iii) $\alpha[x/x]$, (iv) $\alpha[z/x]$, (v) $\alpha[z/y]$, (vi) $\alpha[y/w]$?

Solution (i) Not clean, because the sole free occurrence of x is in the scope of $\exists y$. (ii), (iii) Both clean. (iv) No, because the free occurrence of x is in the scope of $\forall z$. For (v), note that there are no free occurrences of y. Hence, vacuously, the substitution $\alpha[z/y]$ is clean. (vi) Again vacuously clean: there are no occurrences of w in α at all.

This exercise illustrates the following general facts, which are immediate from the definition:

- The substitution of a constant for a variable is always clean.
- The identity substitution is always clean.
- A substitution is clean whenever the variable being replaced has no free occurrences in the formula.
- In particular, a substitution is clean whenever the variable being replaced does not occur in the formula at all.

9.4.3 Fundamental Rules

We can now formulate the promised fundamental principles about $\forall$. The first is a logical consequence (and known as a *first-level* rule); the second is a rule allowing us to get one logical consequence from another (a *second-level* rule). They are known as $\forall^-$ (universal instantiation, UI) and $\forall^+$ (universal generalization, UG):

- $\forall^-$: $\forall x(\alpha) \vdash \alpha[t/x]$, provided the substitution $\alpha[t/x]$ is clean.
- $\forall^+$: Whenever $\alpha_1, \ldots, \alpha_n \vdash \alpha$ then $\alpha_1, \ldots, \alpha_n \vdash \forall x(\alpha)$, provided the variable x has no free occurrences in any of $\alpha_1, \ldots, \alpha_n$.

As one would expect, there are two dual principles for $\exists$. They are dubbed $\exists^+$ (existential generalization, EG) and $\exists^-$ (existential instantiation, EI):

- $\exists^+$: $\alpha[t/x] \vdash \exists x(\alpha)$, provided the substitution $\alpha[t/x]$ is clean.
- $\exists^-$: Whenever $\alpha_1, \ldots, \alpha_{n-1}, \alpha \vdash \alpha_n$, then $\alpha_1, \ldots, \alpha_{n-1}, \exists x(\alpha) \vdash \alpha_n$, provided the variable x has no free occurrences in any of $\alpha_1, \ldots, \alpha_n$.

In the second-level rule $\forall^+$, the entire consequence $\alpha_1, \ldots, \alpha_n \vdash \forall x(\alpha)$ serves as output of the rule. The input of the rule is the consequence $\alpha_1, \ldots, \alpha_n \vdash \alpha$. Similarly, the input and output of $\exists^-$ are both entire consequences.

Note carefully that the two first-level implications $\forall^-$ and $\exists^+$ make substitutions and require that these substitutions be clean. In contrast, the second-level ones $\forall^+$ and $\exists$ do not make substitutions and have the different proviso that the quantified variable has no free occurrences in certain formulae. Commit to memory to avoid confusion!

We draw attention to a detail about the first-level rule $\exists^+$ that sometimes throws students. While the implication goes from $\alpha[t/x]$ to $\exists x(\alpha)$, the substitution goes in the reverse direction, from the body α of $\exists x(\alpha)$ to $\alpha[t/x]$. This contrasts with the situation in the first-level rule $\forall^-$, where the direction of the substitution is the same as that of the inference.

The following exercise is very boring, but you need to do it in order to make sure that you have really understood what these two rules are saying.

Exercise 9.4.3 (1) (with partial solution)

(a) Which of the following are instances of the first-level rule $\forall^-$? (i) $\forall x(Px \to Qx) \vdash Px \to Qx$, (ii) $\forall x(Px \to Qx) \vdash Pb \to Qb$, (iii) $\forall x(Px \wedge Qx) \vdash Px \wedge Qy$, (iv) $\forall x \exists y(Rxy) \vdash \exists y(Rxy)$, (v) $\forall x \exists y(Rxy) \vdash \exists y(Ryy)$, (vi) $\forall x \exists x(Rxy) \vdash \exists x(Ray)$, (vii) $\forall x \exists y(Rxy) \vdash \exists y(Ryx)$, (viii) $\forall x \exists y(Rxy) \vdash \exists y(Rzy)$, (ix) $\forall x \exists y(Rxy) \vdash \exists y(Ray)$, (x) $\forall x \exists y(Rxy) \vdash Rxy$.

(b) Which of the following are instances of the first-level rule $\exists^+$? (i) $Pa \vdash \exists x(Px)$, (ii) $Rxy \vdash \exists x(Rxx)$, (iii) $Rxy \vdash \exists z(Rzy)$, (iv) $Rxy \vdash \exists z(Rzx)$, (v) $\exists y(Rxy) \vdash \exists z \exists y(Rzy)$.

(c) We know that $\forall x \exists y(Rxyz) \vdash \exists y(Rzyz)$. Which of the following may be obtained from it by an application of the second-level rule $\forall^+$? (i) $\forall x \exists y(Rxyz) \vdash \forall z \exists y(Rzyz)$, (ii) $\forall x \exists y(Rxyz) \vdash \forall w \exists y(Rzyz)$, (iii) $\forall x \exists y(Rxyz) \vdash \forall w \exists y(Rwyz)$, (iv) $\forall x \exists y(Rxyz) \vdash \exists x \exists y(Rzyz)$, (v) $\forall x \exists y(Rxyz) \vdash \forall y \exists y(Rzyz)$.

(d) We know that $Px, \forall y(Ryz) \vdash Px \wedge \exists z(Rxz)$. Which of the following may be obtained from it by an application of the second-level rule $\exists^-$? (i) $Px, \exists z \forall y(Ryz) \vdash Px \wedge \exists z(Rxz)$, (ii) $Px, \exists x \forall y(Ryz) \vdash Px \wedge \exists z(Rxz)$, (iii) $Px, \exists w \forall y(Ryw) \vdash Px \wedge \exists z(Rxz)$, (iv) $Px, \exists w \forall y(Ryz) \vdash Px \wedge \exists z(Rxz)$, (v) $\exists x(Px), \forall y(Ryz) \vdash Px \wedge \exists z(Rxz)$.

Solutions to (a) and (c)

(a): (i) and (ii) yes; (iii) no; (iv) yes; (v), (vi) and (vii) no; (viii) and (ix) yes; (x) no.

(c): (i) No; (ii) yes; (iii), (iv) no; (v) yes.

Alice Box: Why the Minus in $\exists^{-}$?

Alice: I'm a bit bothered by the name of one of these four rules. I can see why you call the first three as you do, but not the fourth. Why the minus sign in $\exists^{-}$? Surely we are *adding* the existential quantifier $\exists x$ to α.

Hatter: It is a procedural elimination. Suppose we are given $\alpha_1,\ldots,\alpha_{n-1},\exists x(\alpha)$ as premises and we want to infer a conclusion α_n. The rule tells us that to do that, it suffices to *strip off* the existential quantifier from $\exists x(\alpha)$ and use α as an additional premise, heading for the same goal α_n. While we are not *inferring* α from $\exists x(\alpha)$ (and it would be quite invalid to do so), nevertheless, we are *moving* procedurally from $\exists x(\alpha)$ to the α in the process of building our deduction.

In this respect, $\exists^{-}$ is similar to the rule $\vee^{-}$ of disjunctive proof in propositional logic. There we are given $\alpha_1,\ldots,\alpha_n$, $\beta_1\vee\beta_2$ as premises, and we want to infer a conclusion γ. The rule $\vee^{-}$ tells us that to do that, it suffices to strip the disjunction out of $\beta_1\vee\beta_2$ and first use β_1 as an additional premise, heading for the same goal γ, then use β_2 as the additional premise, heading again for γ.

9.4.4 Identity

So far we have ignored the identity relation sign. Recall from the preceding section that this symbol is always given a highly constrained interpretation. At the most lax, it is interpreted as a congruence relation over elements of the domain; in the standard semantics used here, it is always taken as the identity relation over the domain. Whichever of these readings one follows, standard or non-standard, some special properties emerge.

As would be expected, these include formulae expressing the reflexivity, symmetry and transitivity of the relation, thus its status as an equivalence relation. To be precise, the following formulae are logically true, in the sense defined in Sect. 9.4.1: $\forall x(x\equiv x)$, $\forall x\forall y[(x\equiv y)\rightarrow(y\equiv x)]$, $\forall x\forall y\forall z[\{(x\equiv y)\wedge(y\equiv z)\rightarrow(x\equiv z)\}]$.

A more subtle principle for identity is known as *replacement*. It reflects the fact that whenever x is identical with y, then whatever is true of x is true of y. Sometimes dubbed the principle of the 'indiscernibility of identicals', its formulation in the language of first-order logic is a little trickier than one might anticipate. To state the rule, let t, t' be terms and α a formula. Remember that terms may be constants, variables or more complex items made up from constants and/or variables by iterated application of function letters. We say that an occurrence of a term t in a formula α is *free* iff it does not fall within the scope of a quantifier of α that binds a variable in t. A *replacement* of t by t' in α, written $\alpha[t'\colon t]$, is defined to be the result of taking one or more free occurrences of the term t in α and replacing them by t'. Such a replacement $\alpha[t'\colon t]$ is said to be *clean* iff the occurrences of t' introduced are also free in $\alpha[t'\colon t]$. Using these concepts, we can articulate the following *principle of replacement*:

$\alpha, t \equiv t' \vdash \alpha[t'\colon t]$, provided the replacement is clean.

For example, taking α to be $\exists y(Rxy)$, t to be x and t' to be z, we have:

$$\exists y(Rxy), x \equiv z \vdash \exists y(Rzy)$$

since the introduced variable z is free in $\exists y(Rzy)$. On the other hand, if we take t' to be y and propose the inference $\exists y(Rxy), x \equiv y \vdash \exists y(Ryy)$, the introduced occurrence of y y is not free in $\exists y(Ryy)$, and so, the principle *does not authorize* the step, which indeed is not valid. Intuitively speaking, the y that is free in the formula $x \equiv y$ has 'nothing to do' with the y that is bound in $\exists y(Ryy)$; that formula is logically equivalent to $\exists z(Rzz)$ which does not contain y at all.

The proof of the principle of the replacement is by a rather tedious induction on the complexity of the formula α, which we omit.

Exercise 9.4.4 (1) (with solution)

(a) Put α to be $\exists y[R(f(x,a),y)]$ and t to be $f(x,a)$, which is free in α. Let t_1 be $g(x)$ and t_2 be $f(y,z)$. Write out $\alpha[t_1\colon t]$ and $\alpha[t_2\colon t]$, and determine in each case whether the principle authorizes the consequence $\alpha, t \equiv t_i \vdash \alpha[t_i\colon t]$.

(b) Show that the principle of replacement authorizes the consequence $x \equiv x, x \equiv y \vdash y \equiv x$.

Solution

(a) In this example, $\alpha[t_1\colon t]$ is $\exists y[R(g(x),y)]$, in which the introduced $g(x)$ is free, so the replacement is clean. The principle thus authorizes the desired implication. On the other hand, $\alpha[t_2\colon t]$ is $\exists y[R(f(y,z),y)]$, in which the y of the introduced $f(y,z)$ is bound. So, the principle does not authorize the proposed implication.

(b) Put α to be $x \equiv x$, t to be x and t' to be y. Let $\alpha[t'\colon t]$ replace the *first* occurrence of t in α by t'. Since there are no quantifiers, the replacement is clean. Thus, we have $\alpha, t \equiv t' \vdash \alpha[t'\colon t]$, that is, $x \equiv x, x \equiv y \vdash y \equiv x$.

Actually, symmetry and transitivity for $\equiv$ may be obtained from reflexivity by clever use of replacement. In the case of symmetry, the essential idea behind such a derivation is given in the last exercise. But we will leave the full derivations until we have looked at higher-level proof strategies in the next chapter.

End-of-Chapter Exercises

Exercise 9.1: The language of quantifiers

(a) Return to the ten questions that were raised after Table 9.1 and answer them in the light of what you have learned in this chapter.

(b) Express the following statements in the language of first-order logic. In each case, specify explicitly an appropriate domain of discourse and suitable constants, function letters and/or predicates:
   (i) Zero is less than or equal to every natural number.
   (ii) If one real number is less than another, there is a third between the two.
   (iii) Every computer program has a bug.
   (iv) Any student who can solve this problem can solve all problems.
   (v) Squaring on the natural numbers is injective but is not onto.
   (vi) Nobody loves everybody, but nobody loves nobody.

(c) Let α be the formula $\forall x\exists y\{(y\equiv z)\rightarrow\forall w\exists y[R(f(u,w),y)]\}$. Identify the free and bound occurrences of variables and terms in it. Identify also the vacuous quantifiers, if any. Which of the substitutions $\alpha[x/z]$, $\alpha[y/z]$, $\alpha[w/z]$, and their three converse substitutions, are clean?

Exercise 9.2: Interpretations

(a) Consider the relation between interpretations of being x-variants (for a fixed variable x). Is it an equivalence relation? And if the variable x is not held fixed? Justify.

(b) Find a formula using the identity predicate, which is true under every interpretation whose domain has less than three elements but is false under every interpretation whose domain has three or more elements.

(c) Sketch an explanation why there is no hope of finding a formula that does not contain the identity predicate, with the same properties.

(d) In the definition of an interpretation, we required that the domain of discourse is non-empty. Would the definition make sense, as it stands, if we allowed the empty domain? How might you reformulate it if you wanted to admit the empty domain? What would the result be for the relationship between ∀ and ∃?

Exercise 9.3: Semantic analysis

(a) Justify the principle of vacuous quantification semantically, first using the substitutional reading of the quantifiers, and then using the x-variant reading. Which, if any, is easier?

(b) Which of the following claims are correct? (i) $\forall x(Px\rightarrow Qx)\vdash\exists x(Px)\rightarrow\exists x(Qx)$, (ii) $\exists x(Px\rightarrow Qx)\vdash\exists x(Px)\rightarrow\exists x(Qx)$, (iii) $\forall x(Px\rightarrow Qx)\vdash\exists x(Px)\rightarrow\forall x(Qx)$, (iv) $\forall x(Px\rightarrow Qx)\vdash\forall x(Px)\rightarrow\exists x(Qx)$ and (v)–(viii) their converses. In the negative cases, give an interpretation in a small finite domain to serve as witness. In the positive cases, sketch a semantic verification using either the x-variant or the substitutional reading of the quantifiers.

(c) Find a simple formula α of quantificational logic such that $\forall x(\alpha)$ is a contradiction but α is not. Use this to get a simple formula β such that $\exists x(\beta)$ is logically true but β is not. *Hint*: It is possible to do it without the identity predicate.

(d) Recall from the text that a set A of formulae is said to be *consistent* iff there is some interpretation that makes all of its elements true. Verify the following: (i) The empty set is consistent. (ii) A singleton set is consistent iff its unique element is not a contradiction. (iii) An arbitrary set A of formulae is consistent iff $A \nvdash p\wedge\neg p$. (iv) Every subset of a consistent set of formulae is consistent. (v) A finite set of formulae is consistent iff the conjunction of all its elements is consistent.

Selected Reading

The same books as for propositional logic, with different chapters. In detail:

Hein J (2002) Discrete structures, logic and computability, 2nd edn. Jones and Bartlett, Boston, chapter 7

Huth M, Ryan M (2000) Logic in computer science. Cambridge University Press, Cambridge

Ben-Ami M (2001) Mathematical logic for computer science, 2nd edn. Springer, London, chapters 5–7

Gamut LTF (1991) Logic, language, and meaning. Volume I: Introduction to logic. University of Chicago Press, Chicago, chapters 3,4

Howson C (1997) Logic with trees. Routledge, London/New York, chapters 5–11

Hodges W (1977) Logic. Penguin, Harmondsworth, sections 34–41

A more advanced text, despite its title, is:

Hedman S (2004) A first course in logic. Oxford University Press, Oxford

Just Supposing: Proof and Consequence 10

Abstract

In the last two chapters, we learned quite a lot about propositional and quantificational logic and in particular their relations of logical implication. In this chapter, we look at how simple implications may be put together to make a deductively valid argument or *proof*. At first glance, this may seem trivial: just string them together! But although it starts like that, it goes well beyond, and is indeed quite subtle.

We begin by looking at the easy process of *chaining*, which creates *elementary derivations*, and show how its validity is linked with the *Tarski conditions* defining *consequence relations/operations*. We then review several *higher-level proof strategies* used in everyday mathematics and uncover the logic behind them. These include the strategies traditionally known as conditional proof, disjunctive proof and proof by cases, proof by contradiction and argument to and from an arbitrary instance. Their analysis leads us to distinguish *second-level* from *split-level* rules, articulate their recursive structures and explain the informal procedure of *flattening* a split-level proof into its familiar ‘suppositional’ form.

10.1 Elementary Derivations

We begin with an example and then see what is required in order to make it work. While reading this section, the reader is advised to have at hand the tables of basic tautological implications and equivalences from Chap. 8.

10.1.1 My First Derivation

Suppose that we are given four propositions as assumptions (usually called *premises* in logic), and that they are of the form $(p\vee\neg q) \to (s \to r)$, $\neg s \to q$ and $\neg q$. We want to infer r. How can we do it?

D. Makinson, *Sets, Logic and Maths for Computing*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-2500-6_10,

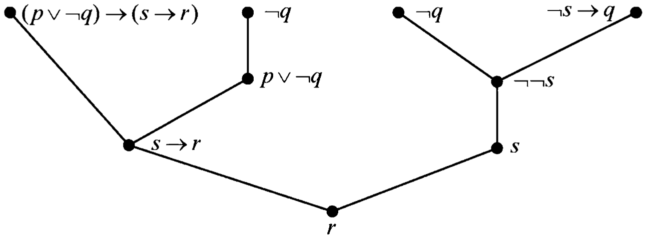

Fig. 10.1 A labelled derivation tree

This is, in effect, a matter of showing that $\{(p\vee\neg q)\rightarrow(s\rightarrow r),\ \neg s\rightarrow q,\ \neg q\}\vdash r$, so in principle we could use a truth table with $2^4=16$ rows or a semantic decomposition tree. But there is another method, which parallels much more closely how we would handle such a problem intuitively. We can apply some basic tautological implications or equivalences with which we are already familiar (e.g. from the tables in Sects. 8.2 and 8.3 of Chap. 8) and chain them together to get the desired conclusion out of the assumptions given.

Specifically, from the third premise, we have $p\vee\neg q$ by addition (aka $\vee^+$), and from this together with the first premise, we get $s\rightarrow r$ by modus ponens ($\rightarrow^-$). But from the third premise again, combined this time with the second one, we have $\neg\neg s$ by modus tollens, which evidently gives us s by double negation elimination. Combining this with $s\rightarrow r$ gives us r by modus ponens, as desired. One way of setting this out is as a *labelled derivation tree*, as in Fig. 10.1.

Such a tree is conventionally written with leaves at the top and is usually constructed from the leaves to the root. The leaves are labelled by the premises (alias assumptions) of the inference. The parent of a node (or pair of nodes) is labelled by the proposition immediately inferred from it/them. If desired, we may also label each node with the usual name of the inference rule used to get it or, in the case of the leaves, give it the label 'premise'. Note that the labelling function in the example is not quite injective: there are two distinct leaves labelled by the third premise.

Exercise 10.1.1 (1) (with partial solution)

(a) In the tree of Fig. 10.1, label the interior nodes (i.e. the non-leaves) with the rules used.

(b) What happens to the tree in the figure if we merge the two nodes labelled $\neg q$?

Solution to (b) It would no longer be a tree, only a graph. It turns out to be much more convenient to represent derivations as trees rather than as graphs.

Given the convention of linearity that is inherent in just about all traditional human writing systems, a derivation such as the above would not normally be written out as a tree but as a finite *sequence*. This is always possible and may be accompanied by annotations to permit unambiguous reconstruction of the tree from the sequence. The process of transforming a derivation from tree to sequence layout may be called *squeezing* the tree into linear form.

Table 10.1 A derivation as a sequence with annotations

Number	Formula	Obtained from	By rule
1	$(p\vee\neg q)\rightarrow(s\rightarrow r)$		Premise
2	$\neg s\rightarrow q$		Premise
3	$\neg q$		Premise
4	$p\vee\neg q$	3	Addition
5	$s\rightarrow r$	4, 1	Modus ponens
6	$\neg\neg s$	3, 2	Modus tollens
7	s	6	Double negation
8	r	7, 5	Modus ponens

Table 10.1 gives an annotated sequence corresponding to the tree in Fig. 10.1. The sequence itself is in the second column, with its elements numbered in the first column and annotations in the third and fourth columns.

The linear form is the one with which the layman first thinks of, with vague memories of proofs done at school. One begins with the premises and works one's way, proposition by proposition, to the conclusion. Building a derivation like this is often called *chaining*.

Tree displays reveal their structure more readily to the eye, but linearization has practical advantages. It takes up less space on the handwritten page, and it is easier to write down using standard word-processing software, at least with current systems. A further advantage to linearization is that it eliminates the need to repeat propositions that may have occurred at more than one node in the tree. The saving may appear derisory; indeed it is so in our example, which is short and where the repeated formula $\neg q$ is a premise, thus a leaf in the tree. But it is easy to imagine a long derivation in which a proposition is appealed to twice (or more) near the end. In tree form, this will be represented by a large tree with a node repeated near the root, and that forces repetition of the entire subtree above it from the node in question to the leaves from which it was obtained. In such cases, the amount of repetition can become very significant.

Exercise 10.1.1 (2) On the basis of the example considered in the above figure and table, give in words a simple algorithm for squeezing a derivation tree into an annotated derivation sequence that (1) 'preserves the order', in the sense that every node of the tree is preceded, in the sequence, by its children, and (2) does not repeat any propositions.

10.1.2 The Logic Behind Chaining

This is all very easy, but it gives rise to a rather subtle question. What guarantee do we have that when each of the individual steps in a derivation sequence/tree is valid (i.e. corresponds to a tautological implication) then the sequence as a whole is too? In other words, how can we be sure that the last item/root is implied by the premises taken together, no matter how distant? In the example, how can we be sure that the propositions (1) through (3) given as premises really do tautologically imply the end proposition (8)? The first answer that comes to mind is that we are just applying the transitivity of $\vdash$. But while transitivity is indeed involved, there is more to it than that.

In our example, we want to show that each formula α in the sequence is tautologically implied by the three premises (1) through (3) taken together. We consider them one by one. First, we want to check that formula (1) is implied by the set $\{1,2,3\}$. Clearly, this requires a principle even more basic than transitivity, namely, that a proposition is a consequence of any set of propositions of which it is an element. That is, we need the following principle of *strong reflexivity*:

$$A \vdash \alpha \text{ whenever } \alpha \in A.$$

Propositions (2) and (3) are checked in a similar manner. What about (4)? We saw that it is tautologically implied by (3), but we need to show that it is tautologically implied by the set $\{1,2,3\}$ of propositions. For this we need a principle called *monotony*. It tells us that whenever γ is a consequence of a set A, then it remains a consequence of any set obtained by adding any other propositions to A. In other words, increasing the stock of premises never leads us to drop a conclusion:

$$B \vdash \beta \text{ whenever both } A \vdash \beta \text{ and } A \subseteq B.$$

In our example, $A = \{3\}$, $\beta = (4)$ and $B = \{1,2,3\}$. What about (5)? In the annotations, we noted that it is tautologically implied by $\{1,4\}$, so monotony tells us that it implied by $\{1,2,3,4\}$. But again we need to show that it is tautologically implied by $\{1,2,3\}$, so we need to 'get rid of' (4). We are in effect using the following principle, called *cumulative transitivity* where, in our example, $A = \{1,2,3\}$, $B = \{4\}$ and $\gamma = (5)$.

$$A \vdash \gamma \text{ whenever both } A \vdash \beta \text{ for all propositions } \beta \in B \text{ and } A \cup B \vdash \gamma.$$

All three of these principles – strong reflexivity, monotony and cumulative transitivity – are needed to ensure that when each individual step in a sequence is justified by an inference relation $\Rightarrow$ (in this example, the relation $\vdash$ of tautological implication), then so too is the sequence as a whole. The surprising thing is that the three are not only necessary for the job; together they are sufficient! Inference relations satisfying them are thus very important and are called *consequence relations*. In the next section, we study them closely and prove that they suffice to guarantee the validity of chaining.

Exercise 10.1.2 (with partial solution)

(a) Complete the enumeration for the sequence in Table 10.1 by considering formulae (6), (7), (8) in turn and noting the application of the principles of strong reflexivity, monotony and/or transitivity.

(b) In Chap. 8, we used the method of decomposition trees to show that $\{\neg q\vee p, \neg r \to \neg p, s, s \to \neg r, t \to p\} \vdash \neg t\wedge\neg q$. Now do it by constructing an elementary derivation, setting out the result first as a labelled derivation tree, and then as an annotated derivation sequence.

Partial solution to (a) To check that {1,2,3} ⊢ (6), we first note that {2,3} ⊢ (6) (by modus tollens, as recorded in the annotation) and then apply monotony. To check that {1,2,3} ⊢ (7), we first note that (6) ⊢ (7) (by double negation elimination as recorded in the annotation), so {1,2,3,6} ⊢ (7) by monotony. But we already know that {1,2,3} ⊢ (6), so we may apply cumulative transitivity to get {1,2,3} ⊢ (7), as desired. The last check, that {1,2,3} ⊢ (8), is done in similar fashion.

10.2 Consequence Relations

We now abstract on the material of Sect. 10.1 to define the general concept of a consequence relation, establish its relationship to elementary derivations and show how it may conveniently be expressed as an operation.

10.2.1 The Tarski Conditions

Let ⇒ be any relation between sets of formulae on its left (the formulae coming from the language of propositional, quantificational or some other logic) and individual formulae (of the same language) on its right. It is called a *consequence relation* iff it satisfies the three conditions that we isolated in the previous section:

Strong reflexivity: $A \Rightarrow \alpha$ whenever $\alpha \in A$.

Monotony: Whenever $A \Rightarrow \beta$ and $A \subseteq B$, then $B \Rightarrow \beta$.

Cumulative transitivity: Whenever $A \Rightarrow \beta$ for all $\beta \in B$ and $A \cup B \Rightarrow \gamma$, then $A \Rightarrow \gamma$.

A few words on variant terminologies may be useful. Strong reflexivity is sometimes called ‘identity’, though that could possibly give rise to confusion with principles for the identity relation in first-order logic. It is sometimes also called (plain) reflexivity, although we are already using the term in its standard and weaker sense that $\beta \Rightarrow \beta$ (when the left side is a singleton set, we omit the braces to reduce clutter). ‘Monotony’ is sometimes written ‘monotonicity’. Cumulative transitivity is often called ‘cut’.

The three conditions are often called the *Tarski conditions*, in honour of Alfred Tarski who first realized their joint importance. Algebraically-minded logicians often call consequence relations *logical closure relations*, as they are just a particular case of the closure relations that arise in abstract algebra.

Exercise 10.2.1 (1) (with partial solution)

(a) Show that both tautological implication in propositional logic and logical implication in quantificational logic are indeed consequence relations. *Hint*: You can use a single argument to cover both. Consider any valuation v, without bothering to specify which of the two languages you are considering.

(b) Show that (i) strong reflexivity immediately implies plain reflexivity and (ii) plain reflexivity together with monotony implies strong reflexivity.

(c) Give a simple example of a relation (between sets of formulae on the left and formulae on the right) that satisfies both monotony and cumulative transitivity but does not satisfy reflexivity.

(d) Show that monotony may be expressed equivalently in either of the following two ways: (i) whenever $A \Rightarrow \gamma$, then $A \cup B \Rightarrow \gamma$; (ii) whenever $A \cap B \Rightarrow \gamma$, then $A \Rightarrow \gamma$.

Partial solutions to (a) and (c)

(a) We check out cumulative transitivity. Suppose $A \Rightarrow \beta$ for all $\beta \in B$ and $A \cup B \Rightarrow \gamma$; we need to show $A \Rightarrow \gamma$. Let v be any valuation, and suppose $v(\alpha) = 1$ for all $\alpha \in A$; we need to get $v(\gamma) = 1$. By the first supposition (and the definition of tautological/logical implication), we have $v(\beta) = 1$ for all $\beta \in B$. Thus, $v(\chi) = 1$ for all $\chi \in A \cup B$. Hence, by the second supposition, $v(\gamma) = 1$. The other two verifications are even easier.

(c) The empty relation is the simplest.

Although, as verified in the above exercise, classical logical implication satisfies all three of the Tarski conditions, it should be appreciated that neither monotony nor cumulative transitivity is appropriate (in unrestricted form) for relations of *uncertain inference*. In particular, monotony fails for both qualitative and probabilistic accounts of uncertain reasoning, and while cumulative transitivity is acceptable under some qualitative perspectives, it too fails probabilistically. But that goes beyond our present concerns.

Exercise 10.2.1 (2) (with partial solution)

(a) Sketch an intuitive example of non-deductive reasoning (say, an imaginary one concerning the guilt of a suspect in a criminal case) that illustrates how the principle of monotony may fail in such contexts.

(b) Is plain transitivity always appropriate for uncertain inference? Give an intuitive counterexample or explanation.

Solution to (b) Plain transitivity fails for uncertain inference. One way of illustrating this is in terms of an intuitive notion of risk. When we pass from α to β non-deductively, there is a small risk of losing truth, and passage from β to γ may likewise incur a small risk. Taking both steps compounds the risk, which may thereby exceed a reasonable threshold that each of the separate ones respects. In effect, the strength of the chain may be *even less than* the strength of its weakest link.

10.2.2 Consequence and Chaining

We are almost ready to articulate in a rigorous manner the fact that the chaining is always justified for consequence relations. First we define, more carefully than before, the notion of an elementary derivation.

Let $\Rightarrow_0$ be any relation between sets of formulae and individual formulae of a formal language. To guide intuition, think of $\Rightarrow_0$ as (the union of the instances of) some small finite set of rules like $\wedge^+$, $\vee^+$, $\rightarrow^-$, or others that you select from the logical implication tables of Chaps. 8 and 9, or yet others that you find attractive. Let A be any set of formulae, α an individual formula. We say that a finite sequence $\sigma = \alpha_1, \ldots, \alpha_n$ of formulae is an *elementary derivation* of α from A using the relation $\Rightarrow_0$ iff (1) $\alpha_n = \alpha$ and (2) for every $i \leq n$, either $\alpha_i \in A$ or $B \Rightarrow_0 \alpha_i$ for some $B \subseteq \{\alpha_1, \ldots, \alpha_{i-1}\}$.

This definition is in terms of sequences. It may equivalently be expressed in terms of trees. We can say that a finite tree τ whose nodes are labelled by formulae is an *elementary derivation* of α from A using the relation $\Rightarrow_0$ iff (1) its root is labelled by α, (2) every leaf node in the tree is labelled by a formula in A and (3) each non-leaf node x is labelled by a formula β such that the labels of the children of x form a set B of formulae with $B \Rightarrow_0 \beta$. Visually, the tree representation is more graphic, and conceptually it is also clearer; but verbally, it is more complex to formulate and use. We will speak in both ways, according to which is handier on the occasion.

The *soundness theorem for chaining* tells us: For any inference relation $\Rightarrow$ and subrelation $\Rightarrow_0$ of $\Rightarrow$, if (i) there is an elementary derivation of α from A using $\Rightarrow_0$, and (ii) $\Rightarrow$ satisfies the three Tarski conditions, then we have $A \Rightarrow \alpha$. The proof of this is easy once you see it, but it can be difficult for a student to set it up from scratch, so we give it in full detail. It is most simply expressed in terms of sequences (rather than trees) and proceeds by cumulative induction on the length of the derivation (i.e. the number of terms in the sequence). The reader is advised to refresh memories of cumulative induction from Sect. 4.5.2. In particular, it should be noted that in this form of induction, we do not need a base (although we can put one in if one wants), only an induction step.

Let $\Rightarrow_0$ be any relation between sets of formulae and individual formulae of a formal language. Suppose that $\Rightarrow_0 \subseteq \Rightarrow$ and that $\Rightarrow$ is a consequence relation (i.e. satisfies the three Tarski conditions). We want to show that for any elementary derivation $\sigma = \alpha_1, \ldots, \alpha_n$ of α from A using the relation $\Rightarrow_0$, we have $A \Rightarrow \alpha$. (Note that we are *not* trying to show that $A \Rightarrow_0 \alpha$; in general, that would fail.) Suppose that this holds for all $i < n$. Now, by the definition of an elementary derivation, either $\alpha_n \in A$ or $B \Rightarrow_0 \alpha_n$ for some $B \subseteq \{\alpha_1, \ldots, \alpha_{n-1}\}$. In the former case, we have $A \Rightarrow \alpha$ by strong reflexivity of $\Rightarrow$. In the latter case, we have $B \Rightarrow \alpha_n$ by the supposition that $\Rightarrow_0 \subseteq \Rightarrow$, and thus in turn (1) $A \cup B \Rightarrow \alpha_n$ by monotony of $\Rightarrow$. But each $\alpha_i \in B$ is obtained by an elementary derivation of length $i < n$. So by our induction hypothesis, (2) $A \Rightarrow \alpha_i$ for each $\alpha_i \in B$. Applying cumulative transitivity to (1) and (2) gives us $A \Rightarrow \alpha$, as desired.

Exercise 10.2.2 (1) Uneasy about not having a base in this induction? Add it in. That is, show that the soundness theorem for chaining holds (trivially) for derivations of length 1.

Alice Box: What Did We Use in This Proof?

Alice: This little proof does not seem to use the full power of monotony or cumulative transitivity.

Hatter: How is that?

Alice: When applying cumulative transitivity, we needed only the case that B is finite, although A may be infinite. And similarly, when applying monotony understood in the form 'whenever $A \Rightarrow \gamma$, then $A \cup B \Rightarrow \gamma$', only the case that B is finite was needed. I wonder whether there is any significance in that!

Hatter: I have no idea . . .

Eagle eye! Alice is going beyond the boundaries of the text, but we can say a few words for those who are interested. Indeed, we don't quite need the full force of the Tarski conditions to establish the soundness theorem for chaining. As Alice noticed, we can do it with restricted versions of monotony and cumulative transitivity, in which certain sets are finite. For many inference relations ⇒, including classical consequence in both propositional and quantificational logic, the full and restricted versions of those Tarski conditions are in fact equivalent, because those relations are *compact* in the sense that whenever $A \Rightarrow \alpha$, then there is some finite subset $A' \subseteq A$ with $A' \Rightarrow \alpha$. But for relations ⇒ that are not compact, the restricted versions of monotony and cumulative transitivity can sometimes be strictly weaker than their full versions.

Another lesson can be drawn from Alice's observation. It concerns a possible converse for the soundness theorem for chaining. That would be a theorem to the effect that if ⇒ fails one of the three Tarski conditions, then chaining with a subrelation $\Rightarrow_0$ may lead us outside ⇒. It could be called a *completeness theorem for chaining*. Such a result can very easily be proven, but only for the restricted versions of the Tarski conditions, or for relations ⇒ that are compact in the sense defined above. But this is beyond the syllabus, and if you have trouble understanding it, just skip the next exercise.

Exercise 10.2.2 (2) (optional, for confident students only)*

(a) Explain why the converse of compactness holds trivially for every consequence relation.

(b) Show that if a relation ⇒ is both compact and monotonic, then, whenever A is infinite and $A \Rightarrow \beta$, there are infinitely many finite subsets $A' \subseteq A$ with $A' \Rightarrow \beta$.

(c) Formulate and prove a completeness theorem for chaining, using the full versions of the Tarski conditions but restricting the statement of the theorem to relations ⇒ that are compact.

(d) Do the same but with the restricted versions of monotony and cumulative instead of requiring compactness.

10.2.3 Consequence as an Operation*

One can express the notion of logical consequence equivalently as an *operation* instead of a relation. This is often convenient, since its behaviour can usually be stated rather more succinctly in operational terms. On the other hand, the operational versions do take a bit of time to get used to.

Let A be any set of propositions, expressed in a formal language like propositional or first-order logic, and let ⇒ be any inference relation for that language. We define Cn quite simply as gathering together the propositions β such that $A \Rightarrow \beta$. That is, $Cn(A) = \{\beta: A \Rightarrow \beta\}$. The function Cn thus takes *sets* of propositions to *sets* of propositions; it is on $\mathscr{P}(L)$ into $\mathscr{P}(L)$, where L is the set of all propositions of the language under consideration and $\mathscr{P}$ is the power-set operation, familiar from the chapter on sets. The notation for Cn is chosen to recall the word 'consequence'.

When ⇒ is a consequence relation, that is, satisfies the three Tarski conditions, then the operation Cn has the following three properties:

Inclusion:	$A \subseteq Cn(A)$
Idempotence:	$Cn(A) = Cn(Cn(A))$
Monotony:	$Cn(A) \subseteq Cn(B)$ whenever $A \subseteq B$

These properties correspond to those for consequence as a relation and can be derived from them. Again, they are called *Tarski conditions*, and consequence operations are known to the algebraically or topologically inclined as *closure* operations. In topology, however, closure operations are sometimes required to satisfy the equality $Cn(A) \cup Cn(B) = Cn(A \cup B)$, which does not hold for classical consequence and the logical operations in general.

We distinguished notationally between an arbitrary consequence relation ⇒ and the particular relation of classical consequence ⊢. Similarly, while we write Cn for an arbitrary consequence relation, $Cn_{\vdash}$ stands for the specific operation of classical consequence. By the definition, we have $Cn_{\vdash}(A) = \{\beta: A \vdash \beta\}$.

Exercise 10.2.3 (with partial solution)

(a) Give a very simple example to show that (i) classical consequence does not satisfy the equality $Cn_{\vdash}(A) \cup Cn_{\vdash}(B) = Cn_{\vdash}(A \cup B)$. Show that nevertheless, (ii) the LHS ⊆ RHS half of it holds for every consequence relation, including classical consequence.

(b) Show that the three Tarski conditions for logical consequence as an operation follow from those for consequence relations, via the definition $Cn(A) = \{\beta: A \Rightarrow \beta\}$ given in the text.

(c) In your answer to (b), did you obtain idempotence from cumulative transitivity alone, or did you need the cooperation of one of the other conditions for consequence relations?

(d) What would be the natural way of defining consequence as a relation from consequence as an operation?

(e) Show that the three Tarski conditions for logical consequence as a relation follow from those for consequence operations, via the definition given in answer to (d).

(f) Express the equality $Cn\{\alpha\}\cap Cn\{\beta\} = Cn(\{\alpha\vee\beta\})$ in terms of consequence relations $\Rightarrow$. Show that it holds for classical consequence. *Remark*: You may answer in terms of $\vdash$ or of $Cn_{\vdash}$.

(g) Express in terms of consequence relations the principle that when A is finite, $Cn(A) = Cn(\wedge A)$ where $\wedge A$ is the conjunction of all the elements of A. Show that it holds for classical consequence. *Remark*: Same as before.

Solutions to (a) and (d)

(a) (i) Take two distinct elementary letters p, q and put $A = \{p\}$, $B = \{q\}$. Then, $p\wedge q \in$ RHS but $p\wedge q \notin$ LHS. (ii) $A \subseteq A\cup B$ and also $A \subseteq A\cup B$; so by monotony, $Cn(A) \subseteq Cn(A\cup B)$ and also $Cn(B) \subseteq Cn(A\cup B)$, so $Cn(A)\cup Cn(B) \subseteq Cn(A\cup B)$.

(d) Put $A \Rightarrow \beta$ iff $\beta \in Cn(A)$.

10.3 A Higher-Level Proof Strategy

In this section and the next, we review some of the proof strategies that are used in traditional mathematical reasoning. These include conditional proof, disjunctive proof and its variant proof by cases, proof by contradiction and proof to and from an arbitrary instance. They are typically marked by terms like ‘suppose’, ‘*let x* be an *arbitrary* such-and-such’ or ‘*choose* any so-and-so’. From a logical point of view, they make use of *higher-level rules* as well as the implications deployed in chaining, which, for contrast, we can refer to as *first-level rules*.

This section focuses on the strategy known as *conditional proof*. We already said a few words about it in a logic box (Chap. 1, Sect. 1.2.2), but now we put it under the microscope and analyze the logic behind it. We do so slowly, as conditional proof serves as an exemplar. After examining it carefully, we can describe the other higher-level strategies more succinctly in the following section.

10.3.1 Informal Conditional Proof

Consider a situation where we want to prove a conditional proposition $\beta \rightarrow \gamma$ given background propositions $\alpha_1, \ldots, \alpha_n$ that we are willing to take as known, say as axioms or theorems already established. The standard way of doing this is to *make a supposition* and *change the goal*. We suppose the antecedent β of the conditional and seek to establish, on this basis, the consequent γ of that conditional. If we manage to get γ out of $\{\alpha_1, \ldots, \alpha_n, \beta\}$, then our proof is complete. From that point on, we disregard the supposition; it is like a ladder that we kick away after we have climbed to our destination. This kind of reasoning is known as *conditional proof.*

In everyday mathematics, it is relatively rare to prove a bare conditional $\beta \rightarrow \gamma$; usually there is an initial universal quantifier as well. For example, we might want to prove (using facts we already know) that whenever a positive integer n is even, then so is n^2, which is of the form $\forall n(En \rightarrow En^2)$ taking the set of all positive integers as domain. To prove that we begin by stripping off the initial quantifier 'whenever', we first let n be an *arbitrary* positive integer, and only then suppose that n is even and go on to establish that n^2 is even. However, in our analysis, we want to separate out the moves made for different logical connectives. Proof methods for the quantifiers will be discussed later; for now we consider an example in which conditional proof is the only method going beyond familiar chaining.

Take three premises $(p \wedge q) \rightarrow r$, $(t \rightarrow \neg s)$, $(r \rightarrow (t \vee u))$ (these are the $\alpha_1, \ldots, \alpha_n$ of our schematic description) and try to prove $(p \wedge s) \rightarrow (q \rightarrow u)$ (which is $\beta \rightarrow \gamma$ in the scheme). What do we do? We *suppose* $p \wedge s$ (β in the scheme) and reset our goal as $q \rightarrow u$ (γ in the scheme). That happens still to be a conditional, so we may iterate the procedure by *supposing* its antecedent q, with the desired conclusion now reset as u. If we succeed in getting u out of the five assumptions $\alpha_1, \ldots, \alpha_5$ (three initial premises plus two suppositions), then our proof is complete. Filling in the rest is straightforward; we make use of some of the tautological implications in Chap. 8 to construct an elementary derivation that does the job.

Exercise 10.3.1 (1) (with sketch of a solution) Write out the above proof in full, including the steps described as straightforward.

Sketch of a solution Make your suppositions as above, and try applying simplification, conjunction, modus ponens, modus tollens and disjunctive syllogism, some of these perhaps more than once. Some variation in order of application is possible.

Sometimes, when we want to prove a conditional $\beta \rightarrow \gamma$, it is more convenient to prove its contrapositive $\neg\gamma \rightarrow \neg\beta$. In that case, we apply the same method to $\neg\gamma \rightarrow \neg\beta$: we suppose $\neg\gamma$ and try to get $\neg\beta$. If we succeed, we have proven $\neg\gamma \rightarrow \neg\beta$ from whatever the original premises were, and so we can finally apply the first-level implication of contraposition to end with the desired $\beta \rightarrow \gamma$. This procedure is often known as *contraposed* or *indirect conditional proof.* However, it is important to realize that its central feature is still plain conditional proof, and for the purposes of logical analysis, we continue to focus on that. A text more concerned with heuristics than logic would give the contraposed version more attention.

Exercise 10.3.1 (2) Carry out the same proof by first supposing $p \wedge s$ and then supposing (contrapositively) $\neg u$, heading for $\neg q$. The choice and/or order of tautological implications applied may differ slightly.

Alice Box: Why Bother with Suppositions?

Alice: Why bother making suppositions? Can't we do it without them?
Hatter: How?
Alice: Well, by a using truth table or a semantic decomposition tree.
Hatter: The last example has 6 elementary letters, so the table will have $2^6 = 64$ rows. The decomposition tree will also be quite big. You can do it, but it is neither elegant nor convenient.
Alice: Well, by an elementary derivation, without suppositions?
Hatter: Try it. It will be tricky to find, longer and less transparent.

10.3.2 Conditional Proof as a Formal Rule

Let's articulate the logical rule underlying the procedure, expressing it in terms of an arbitrary inference relation $\Rightarrow$. Quite simply, it is that $A \Rightarrow (\beta \rightarrow \gamma)$ whenever $A \cup \{\beta\} \Rightarrow \gamma$. Writing a slash for the central transition, it may be expressed as

$$[A \cup \{\beta\} \Rightarrow \gamma] / [A \Rightarrow (\beta \rightarrow \gamma)]$$

Omitting the square brackets as understood and abbreviating $A \cup \{\beta\}$ as A, β:

$$A, \beta \Rightarrow \gamma / A \Rightarrow (\beta \rightarrow \gamma) .$$

In the rule, $A \Rightarrow \beta \rightarrow \gamma$ is called the *principal* (or *main*) *inference*, while the one from which that is obtained, A, $\beta \Rightarrow \gamma$, is called the *subordinate inference* (or *subproof*). The proposition β is the *supposition* of the subordinate inference and becomes the antecedent of the conclusion of the principal inference. Like the informal procedure that it underlies, the rule as a whole is known as *conditional proof* with acronym CP. It is often called $\rightarrow^+$ because an arrow is introduced into the conclusion of the principal inference. Sometimes for display purposes, instead of using a slash, a line is drawn below the subordinate inference and the principal one is written below.

There are several important points that we should immediately note about conditional proof. They concern its shape, status and benefits and apply, *mutatis mutandis*, to the other higher-level rules that will be explained in the following section:

- In its present formulation, it is a *second-level rule*, in the sense that it takes us from the validity of an *entire inference* A, $\beta \Rightarrow \gamma$ to the validity of another entire inference $A \Rightarrow \beta \rightarrow \gamma$. In this respect, it contrasts with the tautological implications and equivalences in tables of Chap. 8, all of which take us from various *propositions* as premises to a proposition as conclusion.

- It is *correct* when the relation $\Rightarrow$ is classical consequence, that is, tautological or first-order logical implication $\vdash$. That is the only specific context that we will be considering in this chapter, but we continue to formulate conditional proof in a general way using the double arrow, and note in passing that the rule is also correct for many other consequence relations in more complex or variant languages.
- It makes life easier when carrying out a deduction, because it gives us more premises to play around with. When the principal inference $A \Rightarrow \beta \to \gamma$ has n premises, the subordinate inference $A, \beta \Rightarrow \gamma$ has $n + 1$. That gives us *one more premise to grab*. If conditional proof is iterated, as in our example, then each application makes another premise available.
- Finally, the conclusion γ of the subordinate argument has *one less occurrence of a connective* than the conclusion $\beta \to \gamma$ of the principal argument. In this way, it reduces the question of obtaining a conditional conclusion to that of getting a logically simpler one.

Not all of the higher-order rules that follow actually *increase* the number of assumptions in the subordinate inference: one of them (for the universal quantifier) leaves the premise-set unchanged, making no supposition. But in all of them, it is possible to formulate the rule in such a way that the premise-set of the subordinate inference is a *superset* (usually but not always proper) of the premise-set of the principal one. To mark this property, we say that the rules are *incremental*.

Finally, we observe that conditional proof may be written in another way, as a *split-level* rule taking propositions plus an entire inference to a proposition. What is the difference? Whereas the second-level formulation tells us that whenever the inference A, $\beta \Rightarrow \gamma$ is valid, then so is the inference $A \Rightarrow \beta \to \gamma$, the *split-level* presentation tells us that whenever all propositions in A are true and the inference A, $\beta \Rightarrow \gamma$ is valid, then the proposition $\beta \to \gamma$ is true. In a brief display, where we have added a semicolon and restored some brackets for readability:

$$A; (A, \beta \Rightarrow \gamma) \,/\, (\beta \to \gamma)\,.$$

The difference between the two formulations is mathematically trivial; we have only moved the A around. But it can be helpful, as we will see shortly. It is rather like working with the left or right projection of a two-argument function instead of the function itself. In both cases, we have a notational reformulation based on a trivial equivalence, which is surprisingly useful. All of the second-level rules that will be considered can be rewritten in a split-level version.

Exercise 10.3.2

(a) Show that, as claimed in the text, conditional proof is correct for classical consequence. *Hint*: You can consider the second-level or the split-level version. You may also use a single argument to cover both propositional and first-order consequence: consider any valuation v, without bothering to specify which of the two languages you are considering.

(b) The *converse* of the second-level rule of conditional proof reverses its principal and subordinate inferences; that is, its principal inference is the old subordinate one and vice versa. Write this converse explicitly, rewrite it in split-level form and show that it is classically correct.

(c) Explain in informal terms why converse conditional proof is of little interest.

10.3.3 Flattening Split-Level Proofs

It is difficult to combine second-level rules directly with first-level ones, as the premises and conclusion of the latter are all propositions, while the premises and conclusion of the former are all inferences. But split-level rules cooperate easily with first-level ones. Since the output and some of the inputs of the split-level rule are propositions, we can use them as premises or conclusions of first-level rules. A proof using both split-level and first-level rules will have a *mix* of nodes: some will be propositions and some will be summary records (premises and associated conclusions) of entire inferences.

When all the higher-level rules used are incremental, then we can rewrite a split-level proof as a sequence (or tree, but we will focus on sequences) of propositions only, by a process called *flattening*. At the nodes that were labelled by summary records of entire inferences, we insert a sequence of propositions beginning with the additional premises (if any) of the subproof, calling them *suppositions*, and continuing with the propositions of the subproof until its goal is reached, at which point the supposition is cancelled (or as is often said, *discharged*).

In the particular case of conditional proof, we add to the premises A of the main proof the further premise β of the subproof, calling it a supposition, and make free use of it as well as the original premises in A until we reach the goal γ of the subproof, where we annotate to record the discharge of the supposition, and then conclude $\beta \to \gamma$ by the rule of conditional proof on the basis of the original premises A alone. Whereas the split-level proof has the form:

Given the premises in A, and wanting to get $\beta \to \gamma$,
carry out separately a proof of γ from $A \cup \{\beta\}$, and then
conclude $\beta \to \gamma$ from A alone;

its flattened or suppositional version has the form:

Given the premises in A, and wanting to get $\beta \to \gamma$,
add in β as a further premise, called a supposition,
argue to γ,
discharge (i.e. unsuppose) β,
conclude $\beta \to \gamma$ on the basis of A.

What is the advantage? It *cuts down on repetition*: we do not have to repeat all the premises in A. That may seem a very meagre gain, but in practice, it makes things much more manageable, especially with multiple subproofs.

All of this applies to the other higher-level strategies that we will shortly be explaining. In each case, the higher-level rule is incremental, so that flattening becomes possible and eliminates the need to repeat premises.

Exercise 10.3.3 (with sketch of solution) In Exercise 10.3.1 (1), we constructed an informal argument using conditional proof. These were expressed in traditional form with suppositions, that is, as flattened versions of split-level proofs. Reconstruct the full split-level proof that lies behind it, and compare the amount of repetition.

Sketch of Solution In the principal proof (call it **A**), where we supposed $p \wedge s$, you need to write a summary record (premises and conclusion) of a subproof and also write out that subproof (call it **B**) separately. In the subproof, where we supposed q, you should write a summary record (premises and conclusion) of a sub-subproof and give separately the sub-subproof (call it **C**).

Thus, flattening, like squeezing (Sect. 10.1), reduces repetition in a proof. But the two processes should not be confused:

- *Squeezing* may be applied to any derivation in tree form, even an elementary one. By squeezing a tree into a sequence, we avoid the need to repeat propositions that are appealed to more than once, which also eliminates repetition of the subtrees above them.
- On the other hand, *flattening* transforms arguments combining first-level with split-level rules (whether in tree or sequence form), into structures that superficially resemble elementary derivations in that they contain only propositions, with the application of higher-level rules signalled by markers such as 'suppose'.

Flattening split-level proofs into suppositional form has the advantage of streamlining, but it has the conceptual side effect of moving the recursive mechanism out of the spotlight to somewhere at the back of the stage. A good suppositional presentation should retain enough annotative signals to let the reader know where the recursions are and to permit their recovery in full if needed.

10.4 Other Higher-Level Proof Strategies

Now that we have a clear picture of what is going on when one carries out a conditional proof, we can review more easily other higher-level proof strategies, namely, disjunctive proof (and the closely related proof by cases), proof by contradiction and proofs to and from arbitrary instances. We consider each in turn.

10.4.1 Disjunctive Proof and Proof by Cases

We said a little about disjunctive proof in a logic box of Chap. 1; here we dig deeper. Consider a situation where we have among our premises, or have already inferred, a disjunction $\beta_1 \vee \beta_2$. We wish to establish a conclusion γ. The disjunction doesn't give us much definite information; how can we make good use of it?

One way is to translate it into a conditional $\neg\beta_1 \rightarrow \beta_2$ and try to apply modus ponens or modus tollens. But that will work only when our information is sufficient to logically imply one of $\neg\beta_1$ and $\neg\beta_2$, which is not often the case. But we can tackle the problem in another way: divide and rule. We *first suppose* β_1 and try to get the same conclusion γ as before. If we succeed, we forget about that supposition, and *suppose instead* β_2, aiming again at γ. If we succeed again, the proof is complete. This is *disjunctive proof*, acronym DP, code name $\vee^-$.

Again, it is not usual for disjunctive proof to occur in isolation in a mathematical argument. More commonly, the information available will be of a form like $\forall x(\beta_1 \vee \beta_2)$ or $\forall x[\alpha \to (\beta_1 \vee \beta_2)]$, and we have to peel off the outer operators before working on the disjunction. For example, we might wish to show that for any sets A, B, $\mathcal{P}(A) \cup \mathcal{P}(B) \subseteq \mathcal{P}(A \cup B)$. We first let A, B be arbitrary sets, suppose for conditional proof that $X \in \mathcal{P}(A) \cup \mathcal{P}(B)$ and reset our goal as $X \in \mathcal{P}(A \cup B)$. From the supposition, we know that either $X \in \mathcal{P}(A)$ or $X \in \mathcal{P}(B)$, so we are now in a position to apply disjunctive proof. We first suppose that $X \in \mathcal{P}(A)$ and argue to our current goal $X \in \mathcal{P}(A \cup B)$; then we suppose that $X \in \mathcal{P}(B)$ and do the same. If we succeed in both, we are done.

Exercise 10.4.1 (1) (with partial solution)

(a) Fill in the missing parts of the above proof of $\mathcal{P}(A) \cup \mathcal{P}(B) \subseteq \mathcal{P}(A \cup B)$.

(b) Does it matter which disjunct we handle first?

(c) Do we need to require that the two disjuncts are exclusive, that is, that they cannot both be true?

(d) Construct informal verbal arguments using disjunctive proof to show that $(A \cup B) \times C \subseteq (A \times C) \cup (B \times C)$ and conversely.

Partial solutions to (b) and (c)

(b) No, because the two subordinate arguments are independent of each other. In practice, it is good style to cover the easier one first.

(c) No. The situation where both disjuncts are true is covered by each of the two subproofs.

For an example in which disjunctive proof is the only device used other than chaining, we derive s from the three premises $\neg p \vee q$, $(\neg p \vee r) \to s$, $\neg s \to \neg q$. Noting that one of the premises is a disjunction $\neg p \vee q$, we first suppose $\neg p$ and use familiar tautological implications to reach the unchanged goal s. We then suppose q and do the same. With both done, we are finished.

Exercise 10.4.1 (2) Fill in the gaps of the above proof.

What is the underlying logical rule? Expressing it in an incremental form, it says: Whenever A, $\beta_1 \vee \beta_2$, $\beta_1 \Rightarrow \gamma$ and A, $\beta_1 \vee \beta_2$, $\beta_2 \Rightarrow \gamma$, then A, $\beta_1 \vee \beta_2 \Rightarrow \gamma$. Of course, for classical consequence (or anything reasonably like it), we have $\beta_1 \Rightarrow \beta_1 \vee \beta_2$ and $\beta_2 \Rightarrow \beta_1 \vee \beta_2$, so for such consequence relations, the rule may be expressed more succinctly without $\beta_1 \vee \beta_2$ appearing as a premise in the two subordinate inferences. That gives us the following more common formulation, with the same bracketing conventions as for the rule for conditional proof.

$$A, \beta_1 \Rightarrow \gamma; A, \beta_2 \Rightarrow \gamma \,/\, A, \beta_1 \vee \beta_2 \Rightarrow \gamma.$$

Exercise 10.4.1 (3)

(a) Identify the principal inference and the two subordinate inferences in the above scheme for disjunctive proof.
(b) Write in brackets to make sure that you understand the structure of the rule.
(c) Verify that the rule is correct for classical consequence.

A split-level formulation moves the A across, telling us that when $\beta_1 \vee \beta_2$ and all propositions in A are true and the inferences $A,\beta_1 \Rightarrow \gamma$ and $A,\beta_2 \Rightarrow \gamma$ are both valid, then the proposition γ is true. In a brief display, with brackets for readability:

$$[(A, \beta_1 \vee \beta_2)\,;(A, \beta_1 \Rightarrow \gamma)\,;(A, \beta_2 \Rightarrow \gamma)]\,/\gamma\,.$$

A *flattened* suppositional version looks like the following:

Given $\beta_1 \vee \beta_2$ and premises in A, and wanting to get γ,
suppose β_1,
argue to γ,
discharge β_1,
suppose $\beta_{2,}$
argue to γ again,
discharge β_2,
and finally conclude γ.

In disjunctive proof, flattening reduces repetition even more than it does in conditional proof. A single application of the split-level version would write all the propositions in A three times; a flattened version writes them only once.

In mathematical practice, disjunctive proof most often appears in a variant (though classically equivalent) form known as *proof by cases*. We have various premises $\alpha_1, \ldots, \alpha_n$, and we wish to infer a conclusion γ. We may not be able to find a single form of argument that works in all possible cases. So we divide the possible cases into two. We think up a suitable proposition β and consider separately the cases for β and $\neg\beta$. Each of these may open the way to a neat proof of γ. The two proofs may be more or less similar or very different. If both succeed, we are done.

For example, we may want to show in the arithmetic of the natural numbers that for all n, $n^2 + n$ is even. The natural way to do it is by first considering the case that n is even, then the case that n is not even (so odd). Again, we might want to show that the remainder when a square natural number n^2 is divided by 4 is either 0 or 1; here too, the natural breakdown is into the cases that n is even and that it is not even. In the chapter on trees, proof by cases was used (inside a proof by contradiction) in Sect. 7.6.2 when showing that every cycle in an unrooted tree contains at least one repeated edge.

Exercise 10.4.1 (4) Complete the two informal arithmetic proofs, with the use of cases marked clearly.

The underlying second-level rule for proof by cases is

$$[A, \beta \Rightarrow \gamma; A, \neg\beta \Rightarrow \gamma]\,/\,A \Rightarrow \gamma\,.$$

Expressed as a split-level rule this becomes

$$[A; (A, \beta \Rightarrow \gamma) ; (A, \neg\beta \Rightarrow \gamma)] / \gamma .$$

Exercise 10.4.1 (5)

(a) Verify that the rule of proof by cases is correct for classical consequence. You may focus on the second-level or the split-level formulation.

(b) Write out an informal flattened version of proof by cases, modelling it on the flattened version of disjunctive proof.

In our two arithmetical examples of proof by cases, it was rather obvious how best to choose the proposition β. In other examples, it may not be so clear. In difficult problems, finding a profitable way to split the situation may be the greater part of solving the problem. Experience helps, just as it helps us cut a roast turkey at the joints.

Exercise 10.4.1 (6) Use proof by cases to give a simple proof that there are positive irrational numbers x, y such that x^y is rational.

Sometimes in a proof, one finds oneself iterating the splitting to get subcases, sub-subcases, and so on, ending up with dozens of them. Such arguments are rather inelegant and are disdained by mathematicians when a less fragmented one can be found. However, computers are not so averse to them. A famous example is the solution of the four-colour problem in graph theory, which required a breakdown into a myriad of cases treated separately with the assistance of a computer.

10.4.2 Proof by Contradiction

Our third method of argument has been famous since Greek antiquity. It is known as *proof by contradiction* or, using its Latin name, reductio ad absurdum, acronym RAA. It was sketched in an Alice box in Chap. 2; here we go further.

Imagine that we are given information $\alpha_1, \ldots, \alpha_n$ and we want to get a conclusion β. We may even have tried various methods and got stuck, so we try *proof by contradiction*. We suppose the opposite proposition ¬β and seek to establish, on this basis, a contradiction. If we manage to do so, our proof of β from $\alpha_1, \ldots, \alpha_n$ is complete.

What is meant by 'a contradiction' here? We mean an *explicit* contradiction, that is, any statement γ and its negation ¬γ, whether obtained separately or conjoined as $\gamma\wedge\neg\gamma$. As remarked by the Hatter in the Alice box of Chap. 2, it does not matter what contradiction it is, that is, how γ is chosen. When constructing the proof, we usually do not have a clear idea which contradiction we will end up with, but we don't care in the least.

Proof by contradiction is a universal rule: Neither the premises nor the conclusion has to be in any particular form, in contrast with conditional proof (for conditional conclusions) and disjunctive proof (for disjunctive premises). Sometimes it will make the argument a great deal easier, due to the availability of the supposition as something more to work with, but sometimes it will not make much difference. Some mathematicians prefer to use *reductio* only when they have to; others apply it at the slightest pretext.

When RAA is used to establish the existence of an item with a certain property, it will usually be *non-constructive*. That is, it shows the *existence* of something with the desired feature by getting a contradiction out of supposing that nothing has that feature, but without thereby specifying a *particular* object possessing it. Actually, the same can happen in a less radical way with proof by cases; indeed, the argument for exercise 10.4.1 (6) shows that there exists a pair (x,y) positive irrational numbers x, y such that x^y is rational, and moreover shows that at least one of two pairs must do the job, but does not specify a single such pair. Intuitionists do not accept such proofs.

The most famous example of proof by contradiction, dating back to Greek antiquity, is the standard proof that $\surd 2$ is irrational. It begins by supposing that $\surd 2$ is rational and, with the assistance of some basic arithmetic facts, derives a contradiction.

Exercise 10.4.2 (1) An exercise of Chap. 4 asked you to look up a standard textbook proof of the irrationality of $\surd 2$ and put your finger on where it uses *wlog*. Look it up again to appreciate how it uses *reductio*.

We used proof by contradiction many times in previous chapters. In particular, in Sect. 7.2.1 of the chapter on trees, we used it five times to establish as many bulleted properties from the explicit definition of a rooted tree. In Sect. 7.6.2, when proving that $S = R \cup R^{-1}$ is a *minimal* connected relation over a rooted tree (T,R), we used two *reductios*, one inside the other! Readers are advised to review those proofs again now. The first five are very easy; the last is rather tricky, partly because of the iterated RAA.

For a simple example using only propositional moves, consider the two premises $s \rightarrow t$, $(r \vee t) \rightarrow \neg p$ and try to get the conclusion $\neg(p \wedge s)$. As the conclusion is already negative, we can cut a small corner: instead of supposing $\neg\neg(p \wedge s)$ and eliminating the double negation, we suppose $p \wedge s$ itself. We then use familiar tautological implications to chain through to a contradiction, and we are done.

Exercise 10.4.2 (2)

(a) Complete the above proof by constructing an elementary derivation of a contradiction, using only familiar tautological implications from the tables in Chap. 8.

(b) What contradiction did you come to? Can you modify the argument to reach a different contradiction just as naturally? *Hint*: Think of using modus tollens rather than modus ponens or vice versa.

Writing $\perp$ to stand for any explicit contradiction, we can state the logical rule underlying the procedure: Whenever A, $\neg\beta \Rightarrow \perp$, then $A \Rightarrow \beta$. Writing a slash for the central transition, it may be expressed as

$$A, \neg\beta \Rightarrow \perp / A \Rightarrow \beta.$$

In *split-level* form, this becomes

$$[A; (A, \neg\beta \Rightarrow \bot)] / \beta.$$

Finally, a flattened suppositional rendering:

Given premises in A,
suppose $\neg\beta$,
argue to $\bot$,
discharge β,
conclude β on the basis of A.

Exercise 10.4.2 (3) Show that proof by contradiction is correct for classical consequence.

Alice Box: Indirect Inference

Alice: Is proof by contradiction the same as *indirect inference*? Or does that term cover, more broadly, all higher-level strategies?

Hatter: Neither, so be careful! 'Indirect inference' is more of a stylistic description than a logical one. It is used to mean any way of writing out an argument in which the order of development 'swims against' the stream of implication.

Alice: And so which are deemed direct, and which indirect?

Hatter: Chaining is of course considered direct. But so too is conditional proof (although logically it rests on a higher-level inference) while the variant that we called contraposed or indirect conditional proof is, as its name suggests, indirect. Both disjunctive and case-based proof are normally regarded as direct, while proof by contradiction is seen as the most indirect of all. Not only does it swim against the current of implication but it supposes the very opposite of what we want to prove.

Of course, there is a sense in which given any one of the three higher-level rules that we have been discussing, the others become redundant. For example, given conditional proof, we can make it do the work of disjunctive proof by supposing β_1 to get γ, conditionalizing to have $\beta_1 \rightarrow \gamma$ with the supposition discharged, then supposing β_2 to get to γ again, conditionalizing to obtain $\beta_2 \rightarrow \gamma$ with that supposition likewise discharged, and finally using the tautological implication $\beta_1 \rightarrow \gamma, \beta_2 \rightarrow \gamma, \beta_1 \vee \beta_2 \vdash \gamma$, which one could add to the list in the table of Chap. 8, to conclude γ, as desired. But from the standpoint of real-life reasoning, this is an artificial way to proceed, even though it is only a couple of steps longer. That is why in mathematical practice, all three strategies are considered in their own right.

Exercise 10.4.2 (4)

(a) Show that a similar formal trick can be done, given conditional proof, to carry out proof by contradiction. Identify the tautological implication that would be used.

(b) Similar formal tricks could be used to make disjunctive proof do the work of the other two, likewise to make *reductio* do it all. Describe those tricks, isolating the relevant tautological implication in each case.

We end with some remarks for the philosophically inclined; others may skip to the next subsection. There are a few mathematicians, and perhaps rather more philosophers of mathematics, who are wary of the use of truth-values in the discipline. For them, the notions of truth and falsehood have no meaning in mathematics beyond 'intuitively provable' and 'intuitively provable that we cannot have an intuitive proof'. For this reason, they feel that the whole of classical logic must be reconstructed. In the resulting *intuitionistic logic*, some classical principles go out the window.

The most notorious of these is the tautology of excluded middle $\alpha\vee\neg\alpha$, which is not intuitionistically acceptable. But some other principles are also affected; in particular, a number of classical equivalences involving negation are accepted in one direction only. For example, classical double negation $\neg\neg\alpha \dashv\vdash \alpha$ is reduced to $\alpha \vdash \neg\neg\alpha$; contraposition $\alpha\to\beta \dashv\vdash\neg\beta\to\neg\alpha$ drops to $\alpha\to\beta \vdash \neg\beta\to\neg\alpha$; de Morgan $\neg(\alpha\wedge\beta) \dashv\vdash \neg\alpha\vee\neg\beta$ is cut to $\neg\alpha\vee\neg\beta \vdash \neg(\alpha\wedge\beta)$; quantifier interchange $\neg\forall x(\alpha)\dashv\vdash\exists x\neg(\alpha)$ becomes only $\exists x\neg(\alpha) \vdash \neg\forall x(\alpha)$. There are also losses involving implication without negation: for example, the classical tautology $(\alpha\to\beta)\vee(\beta\to\alpha)$ is not intuitionistically acceptable, nor is the following classical tautology using implication alone: $((\alpha\to\beta)\to\alpha)\to\alpha$.

As one would expect, such cutbacks have repercussions for higher-level inference rules. In intuitionistic logic, conditional proof is accepted, but its contraposed version is not; disjunctive proof is retained, but proof by cases in any form that presumes excluded middle is out; proof by contradiction is not accepted in the form that we have given it but only in a form with interchanged negations: A, $\beta \Rightarrow \bot/A \Rightarrow \neg\beta$. However, all that is beyond our remit.

10.4.3 Rules with Quantifiers

We have already seen the second-level rules $\forall^+$ and $\exists^-$ for the two quantifiers in Chap. 9, contrasting them with the first-level logical implications $\forall^-$ and $\exists^+$. Their formulations were:

$\forall^+$: Whenever $\alpha_1,\ldots,\alpha_n \vdash \alpha$, then $\alpha_1,\ldots,\alpha_n \vdash \forall x(\alpha)$, provided the variable x has no free occurrences in any of $\alpha_1,\ldots,\alpha_n$

$\exists^-$: Whenever $\alpha_1,\ldots,\alpha_{n-1}$, $\alpha \vdash \alpha_n$, then $\alpha_1,\ldots,\alpha_{n-1}$, $\exists x(\alpha) \vdash \alpha_n$, provided the variable x has no free occurrences in any of $\alpha_1,\ldots,\alpha_n$

As we also noted, the usual names and acronyms in English are 'universal generalization' (UG) and 'existential instantiation' (EI), although their higher-order nature is perhaps brought out more clearly under the names *proof to* and *from* an *arbitrary instance*.

There is one respect in which $\forall^+$ differs from both its dual $\exists^-$ and all the other second-level rules for propositional connectives that we have considered. *It makes no supposition*! The premises of the subordinate inference are exactly the same as those of the principal inference. So, we cannot simply identify higher-level inference with the making and discharging of suppositions. Suppositions are a salient but not invariable feature of such inferences. Nevertheless, the rule $\forall^+$ is clearly incremental, since the principal and subordinate inferences have the same premises.

On the other hand, the rule $\exists^-$, it is not incremental as we have written it: the premise $\exists x(\alpha)$ of the principal inference is not a premise of the subordinate one. However, just as with disjunctive proof, this is not a difficulty. Since $\alpha \vdash \exists x(\alpha)$, we may equivalently reformulate the rule, a little redundantly, as follows:

$\exists^-$: Whenever $\alpha_1,\ldots,\alpha_{n-1}$, $\exists x(\alpha)$, $\alpha \vdash \alpha_n$, then $\alpha_1,\ldots,\alpha_{n-1}$, $\exists x(\alpha) \vdash \alpha_n$, provided the variable x has no free occurrences in any of $\alpha_1,\ldots,\alpha_n$

Clearly, in this formulation, $\exists^-$ is incremental, like all our other higher-level rules. However, for simplicity, we will usually give it in its first formulation.

Exercise 10.4.3 (1)

(a) Give a simple example of how the rule $\forall^+$ can fail if its proviso is not respected.

(b) Give two simple examples of how $\exists^-$ can go awry if its proviso is ignored, one with a free occurrence of x in α_n, the other with a free occurrence of x in one of $\alpha_1,\ldots,\alpha_{n-1}$.

How do $\forall^+$ and $\exists^-$ look in a more schematic presentation? We first rewrite them as conditions on an arbitrary inference relation $\Rightarrow$. At the same time, we also use explicit notation for sets of formulae, which forces us to rephrase the formulation of the proviso for $\exists^-$. We also abbreviate the wording of the provisos in a manner that should still be clear.

$\forall^+$: Whenever $A \Rightarrow \alpha$, then $A \Rightarrow \forall x(\alpha)$ (x not free in A)
$\exists^-$: Whenever $A,\alpha \Rightarrow \beta$, then A, $\exists x(\alpha) \Rightarrow \beta$ (x not free in β or in A)

In the schematic slash manner:

$\forall^+$: $A \Rightarrow \alpha$ / $A \Rightarrow \forall x(\alpha)$ (x not free in A)
$\exists^-$: A, $\alpha \Rightarrow \beta$ / A, $\exists x(\alpha) \Rightarrow \beta$ (x not free in β or in A)

Written as split-level rules:

$\forall^+$: A ; ($A \Rightarrow \alpha$) / $\forall x(\alpha)$ (x not free in A)
$\exists^-$: A, $\exists x(\alpha)$; (A, $\alpha \Rightarrow \beta$) / β (x not free in β or in A)

Exercise 10.4.3 (2) Take the incremental formulation of $\exists^-$, and rewrite it as a split-level rule too.

When flattened, using the traditional terminology of ‘arbitrary’ and ‘choose’, these split-level rules become:

Given premises in A, consider arbitrary x. Argue to $\alpha(x)$. Conclude $\forall x(\alpha)$.

Given $\exists x(\alpha)$ and premises A, arbitrarily choose x_0 with $\alpha(x_0)$**.** Argue to β. Conclude β.

Here we face a terminological problem. Strictly speaking, we should not call the flattened presentation of $\forall^+$ a suppositional argument because, as we have observed, it makes no supposition: the subordinate inference of the second-order rule $\forall^+$ has exactly the same premises as the principal one. But it would be pedantic to invent a new terminology to replace one so well entrenched, so we will follow custom by continuing to call such flattened arguments 'suppositional', in a broad sense of that term.

Alice Box: Arbitrary Triangles?

Alice: I have a philosophical worry about this talk of arbitrary items. In the flattened version of $\forall^+$ you speak of an arbitrary x in the domain – say an arbitrary positive integer or triangle. And $\exists^-$ is also glossed in terms of choosing x_0 in an arbitrary way. But there is no such thing as an arbitrary triangle or arbitrary number. Every integer is either even or odd, every triangle is either equilateral, isosceles or scalene. Arbitrary triangles do not exist!

Hatter: If I were as mad as hatters are made out to be, I might try to disagree. But of course you are absolutely right.

Alice: So how can we legitimately use the word?

Hatter: Well, it's just a manner of speaking . . .

Alice was right to be puzzled. Her question was raised by the eighteenth-century philosopher Berkeley in a forceful critique of the mathematics of his time. The Hatter's response is also correct, but we should flesh it out a bit. Let's look at $\forall^+$. Although there is no such thing as an arbitrary element of a domain, we can prove theorems using a variable x, ranging over the domain, which is treated arbitrarily *as far as the particular proof is concerned*. What does that mean? It means that the premises of the proof in question do not ascribe any particular attributes to x, other than those that it asserts for all elements of the domain. But this is still rather vague! We make it perfectly precise in a syntactic manner: the variable x should not occur *free* in any of the premises – which is just the formal proviso of the rule. Likewise for $\exists^-$: treating the variable x in an arbitrary manner means that it does not occur free in the conclusion or any of the other premises of the application. In this way, a rather perplexing semantic notion is rendered clear by a banal syntactic criterion.

For quick reference, we end this section by putting the main second-level schemas for propositional connectives and quantifiers into a table, omitting their split-level versions, which are easily constructed out of them (Table 10.2).

Table 10.2 Second-level inference rules

	Conditional proof ($\rightarrow^+$)	Disjunctive proof ($\vee^-$)	Proof by contradiction	Proof to an arbitrary instance ($\forall^+$)	Proof from an arbitrary instance ($\exists^-$)
Subordinate inference(s)	$A,\beta \Rightarrow \gamma$	$A,\beta_1 \Rightarrow \gamma$; $A,\beta_2 \Rightarrow \gamma$	$A,\neg\beta \Rightarrow \gamma\wedge\neg\gamma$	$A \Rightarrow \alpha$	$A,\alpha \Rightarrow \beta$
Principal inference	$A \Rightarrow \beta \rightarrow \gamma$	$A,\beta_1\vee\beta_2 \Rightarrow \gamma$	$A \Rightarrow \beta$	$A \Rightarrow \forall x(\alpha)$	$A,\exists x(\alpha) \Rightarrow \beta$
Provisos				x not free in A	x not free in β or in A

10.5 Proofs as Recursive Structures*

In Sect. 10.2.2, we gave a formal definition of the notion of an elementary derivation in terms of sequences and (equivalently) in terms of trees. In the former mode, a finite sequence $\sigma = \alpha_1, \ldots, \alpha_n$ of formulae is an *elementary derivation* of α from A using the relation $\Rightarrow_0$ iff (1) $\alpha_n = \alpha$ and (2) for every $i \leq n$, either $\alpha_i \in A$ or $B \Rightarrow_0 \alpha_i$ for some $B \subseteq \{\alpha_1, \ldots, \alpha_{i-1}\}$. How can we generalize to define, in an equally rigorous manner, the idea of a higher-level proof? We have already expressed the essential ideas in an informal way; our task is now to do it formally.

10.5.1 Second-Level Proofs

A *second-level proof* uses second-level rules alone, without any first-level rules (nor split-level ones). In contrast, a *split-level proof* employs first-level plus split-level rules (but no second-level ones). We consider second-level proofs in this subsection and split-level ones in the next.

How can one get away with only second-level rules, without any help at all from first-level ones? For example, how can we dispense with the first-level rule of modus ponens ($\rightarrow^-$): $\alpha, \alpha \rightarrow \beta \Rightarrow \beta$? The answer is that it can be reformulated as a second-level rule $[A \Rightarrow \alpha; A \Rightarrow (\alpha \rightarrow \beta)]$ / $[A \Rightarrow \beta]$. Similarly, the first-level rule $\alpha \Rightarrow \alpha\vee\beta$ of addition ($\vee^+$) may be put into the second-level form $[A \Rightarrow \alpha]$ / $[A \Rightarrow \alpha\vee\beta]$. Other first-level rules are transformed in the same simple way: a first-level rule $\varphi_1, \ldots, \varphi_n \vdash \psi$ becomes the second-level rule $[A \vdash \varphi_1, \ \ldots, A \vdash \varphi_n]$ / $[A \vdash \psi]$.

Exercise 10.5.1 (1) (with partial solution)

(a) Express modus tollens as a second-level rule.

(b) Likewise for the first-level rules $\alpha\wedge\beta \Rightarrow \alpha$ and $\alpha\wedge\beta \Rightarrow \beta$.

(c) What would be the second-order version of the first-level rule of reflexivity $\alpha \Rightarrow \alpha$?

Solutions to (a) and (c)

(a) $A \Rightarrow (\alpha \rightarrow \beta), A \Rightarrow \neg\beta \,/\, A \Rightarrow \neg\alpha$.

(b) $A \Rightarrow \alpha \,/\, A \Rightarrow \alpha$.

The notion of a proof using only second-level rules may be defined in exactly the same way as we defined an elementary derivation. The only difference is that the sequence (or tree) is no longer labelled by propositional or first-order formulae but by *sequents*, that is, by expressions $A \Rightarrow \alpha$ with finite sets A (or in some versions, multisets or sequences) of formulae on the left and individual formulae on the right. We use the second-level rules to pass from earlier sequents in the sequence to later ones (from children to parents in the tree), and we allow a special set of sequents as not needing further justification (leaves in the tree). In pure logic, these are usually of the form $\alpha \Rightarrow \alpha$; in mathematics, they may also reflect the substantive axioms being employed.

In general, such sequent calculi are unsuitable for constructing deductions by hand, due to the immense amount of repetition that they create. The transformation of first-level rules into second-level ones brings with it the obligation to write formulae over and over again. For example, when we rewrite first-level modus ponens $\alpha, \alpha \rightarrow \beta \Rightarrow \beta$ as a second-level rule $[A \Rightarrow \alpha; A \Rightarrow (\alpha \rightarrow \beta)] \,/\, [A \Rightarrow \beta]$, the set A of propositions figures thrice, so that a single application of the rule must write out all the formulae in A three times! Since a typical proof will apply such 'up-levelled' rules many times, this effect is multiplied. Nevertheless, sequent calculi can sometimes be used in automated reasoning. While the copying is enough to drive a human to distraction, it need not bother a computer too much so long as it does not become exponential.

However, the main use of sequent calculi is for certain kinds of theoretical work, notably to show that all proofs from suitable initial sequents have desirable properties. The idea is to choose a well-balanced collection of second-level rules to work with, show that proof using only those rules may be put into a suitable normal form, define a well ordering on the normal derivations to serve as a measure of their length or complexity and establish desired properties by inducing on the well ordering. Such investigations make up what is known as *proof theory*. It forms a world of its own, with a vast literature; we will not discuss it further.

Exercise 10.5.1 (2) (with solution) Write the second-level Tarski conditions of monotony and cumulative transitivity for inference relations in schematic slash manner.

Solution Monotony is most concisely expressed as $[A \Rightarrow \beta] \,/\, [A \cup X \Rightarrow \beta]$, although one could of course write: $[A \Rightarrow \beta] \,/\, [B \Rightarrow \beta]$ whenever $A \subseteq B$. Cumulative transitivity may be written: $[A \Rightarrow \beta$ for all $\beta \in B, A \cup B \Rightarrow \gamma] \,/\, [A \Rightarrow \gamma]$.

10.5.2 Split-Level Proofs

Next, we look at split-level proofs (aka subproof systems) which, as remarked, use first-level and split-level rules in cooperation with each other. Their definition may be expressed in terms of trees or of sequences; technically the version with sequences is more straightforward, and so we use it here.

Let $\Rightarrow_0$ be a relation between sets of formulae on the left and individual formulae on the right. To guide intuition, think of it as (the union of the instances of) some small finite set of *first-level* rules. Similarly, let $\Rightarrow_1$ be a relation between *both* formulae *and* sets of formulae on the left and formulae on the right. This represents the (union of the instances of) a small finite set of *split-level* rules. These two relations, $\Rightarrow_0$ and $\Rightarrow_1$, are what we use to build, recursively, split-level proofs.

For any set A of formulae and individual formula α, we say that a finite sequence σ is a *split-level proof* of α from A using $\Rightarrow_0$ and $\Rightarrow_1$ iff (1) its last term is α, (2) every point in the sequence is either (a) a formula in A, or (b) a split-level proof (of some formula from some set of formulae), or (c) a formula β such that either $B \Rightarrow_0 \beta$ or $B \Rightarrow_1 \beta$ for some set B of earlier points in the sequence.

Note the recursion in clause 2(b): it is this that makes the definition more subtle than that for a purely first-level proof (i.e. an elementary derivation) or even for a second-level one. Some points in the sequence may be split-level proofs, and thus themselves finite sequences, and some of their points may in turn be finite sequences and so on. With trees, the split-level proofs mentioned in clause (2b) will all be (labels of) *leaves* of the tree, but they will also be trees themselves and so on recursively. In situations like this, it would be nice to have an n-dimensional page on which to display the proof!

Exercise 10.5.2 (with solution) Write the Tarski conditions of monotony and cumulative transitivity for inference relations as split-level rules.

Solution Monotony is most concisely expressed as $[A \cup X;\ A \Rightarrow \beta]\ /\ \beta$, although one could of course write: $[B;\ A \Rightarrow \beta]\ /\ \beta$ whenever $A \subseteq B$. Cumulative transitivity may be written: $[A;\ A \Rightarrow \beta$ for all $\beta \in B,\ A \cup B \Rightarrow \gamma]\ /\ \gamma$. When B is a singleton, this reduces to $[A;\ A \Rightarrow \beta,\ A \cup \{\beta\} \Rightarrow \gamma]\ /\ \gamma$.

Alice Box: Third-Level Rules?

Alice: We have first- and second-level rules as well as the split-level versions of the latter. Are there such things as third-level rules and so on?

Hatter: Of course, we could define them formally. But neither traditional mathematical reasoning nor current logical analysis has found any serious use for them. Perhaps this will change in the future.

Alice: I hope not. I already have enough on my plate!

10.5.3 Four Views of Mount Fuji

The material of this section has been rather abstract. We now illustrate it by an example. We take a single inference problem and look at it from four different angles: (1) as a short informal proof, entirely in English, (2) a semiformal one that reveals more traces of higher-level inference, (3) a complete split-level proof, and (4) a purely second-level one. Not as many perspectives as in Hokusai's hundred views of Mount Fuji, but they do illustrate the theoretical discussion carried out so far.

10.5.3.1 An Informal Argument in English

The example itself is simple; you don't need to have taken a course in logic to recognize its validity. Our task is to put it under the microscope. *Given* that all wombats are marsupials, the problem is to *show* that if there is an animal in the garden that is not a marsupial, then not every animal in the garden is a wombat.

In plain English, with no logical symbols, nor even any variables, the following short argument suffices:

> Assume the premise; we want to get the conclusion. *Suppose* that there is an animal in the garden that is not a marsupial. Since it is not a marsupial, the premise tells us that it is not a wombat. So, not every animal in the garden is a wombat.

For most purposes, this is perfectly adequate. But it leaves the underlying logical mechanisms rather hidden. While the word 'suppose' hints at the presence of a higher-level step, namely, a conditional proof, there is a second one that is almost invisible. As a first move to bring a bit more logic to the surface, we reformulate the problem in the language of first-order logic and set out the argument in a more formal way but still in suppositional form.

10.5.3.2 Semiformal Suppositional Presentation

Expressed in the symbolism of first-order logic, the problem is *given* $\forall x[Wx \to Mx]$, *show* $\exists x[Ax \wedge Gx \wedge \neg Mx] \to \neg\forall x[(Ax \wedge Gx) \to Wx]$. In our semiformal but still suppositional proof, we make free use of any basic implications or equivalences from tables in Chaps. 8 and 9.

> We are given (1) $\forall x[Wx \to Mx]$. *Suppose* (2) $\exists x[Ax \wedge Gx \wedge \neg Mx]$; it will suffice to get $\neg\forall x[(Ax \wedge Gx) \to Wx]$. By (2), *we may choose* an x such that (3) $Ax \wedge Gx \wedge \neg Mx$, so that in particular $\neg Mx$. Applying $\forall -$ to (1), we have $Wx \to Mx$, so by modus tollens, we get $\neg Wx$. Putting this together with $Ax \wedge Gx$, which also comes from (3), we have (4) $Ax \wedge Gx \wedge \neg Wx$, so by $\exists^{+}$, we infer (5) $\exists x[Ax \wedge Gx \wedge \neg Wx]$, in other words by a familiar tautological equivalence, $\exists x \neg[(Ax \wedge Gx) \to Wx]$ and thus by quantifier interchange (QI) $\neg\forall x[(Ax \wedge Gx) \to Wx]$, as desired.

This already renders more of the logic visible, and in particular, the italicized words tell us that it is appealing to *two* higher-level strategies: conditional (CP) proof and $\exists^{-}$. This is more or less the way in which the argument would be presented in a logic text centred on what is called 'natural deduction'. However, it is still flat in the sense of Sect. 10.3.3, that is, just an annotated sequence of propositions. We now write it explicitly as a split-level proof, replacing the suppositions by subproofs.

10.5.3.3 As a Split-Level Proof

The split-level version consists of a main proof, a subproof and, within the latter, a sub-subproof that has no further subproofs and is thus an elementary derivation.

Main Proof **A**

A1. $\forall x[Wx \rightarrow Mx]$	premise
A2. Insert subproof *B*	
A3. $\exists x[Ax \wedge Gx \wedge \neg Mx] \rightarrow \neg\forall x[(Ax \wedge Gx) \rightarrow Wx]$	conclusion: from A1,A2 by CP

Subproof **B**

B1. $\forall x[Wx \rightarrow Mx]$	premise
B2. $\exists x[Ax \wedge Gx \wedge \neg Mx]$	premise
B3. Insert sub-subproof **C**	
B4. $\neg\forall x[(Ax \wedge Gx) \rightarrow Wx]$	new conclusion: from B1, B2, B3 by $\exists^-$

Sub-subproof **C**

C1. $\forall x[Wx \rightarrow Mx]$	premise
C2. $\exists x[Ax \wedge Gx \wedge \neg Mx]$	premise
C3. $Ax \wedge Gx \wedge \neg Mx$	premise
C4. $\neg Mx$	from C3 by $\wedge^-$
C5. $Wx \rightarrow Mx$	from C1 by $\forall^-$
C6. $\neg Wx$	from C4,C5 by MT
C7. $Ax \wedge Gx$	from C3 by $\wedge^-$
C8. $Ax \wedge Gx \wedge \neg Wx$	from C6,C7 by $\wedge^+$
C9. $\neg[(Ax \wedge Gx) \rightarrow Wx]$	from C8 by a taut. equivalence
C10. $\exists x \neg[(Ax \wedge Gx) \rightarrow Wx]$	from C9 by $\exists^+$
C11. $\neg\forall x[(Ax \wedge Gx) \rightarrow Wx]$	same conclusion: from C10 by QI

What were suppositions of a single argument now become additional premises of subordinate arguments. Evidently, this is more repetitious and thus longer than the suppositional version. It is also less familiar and thus perhaps more difficult to assimilate on first acquaintance. But it gives a fuller record of what is really going on.

10.5.3.4 As a Second-Level Proof

Finally, we present essentially the same little argument as a second-level proof, in which the only rules used are second-level ones. 'Essentially', because in order to keep things reasonably brief, we forget about the 'animal in the garden' part and consider the following simpler version: *Given* that all wombats are marsupials, *show* that if there is something that is not a marsupial, then not everything is a wombat. In other words, we prove the sequent $\forall x[Wx \rightarrow Mx] \vdash \exists x[\neg Mx] \rightarrow \neg\forall x[Wx]$.

Conditional proof and $\exists^-$ are reformulated in their second-level rather than split-level versions. Familiar first-level rules are also recast as second-level ones, in the manner we have explained, with one exception: inferences of the form $\alpha \vdash \alpha$ (identity inferences) are allowed without needing further justification (in a tree presentation they would occur as leaves).

D1. $Wx \to Mx \vdash Wx \to Mx$	identity inference
D2. $Wx \vdash Wx$	identity inference
D3. $Wx \to Mx, Wx \vdash Mx$	from D1, D2 by second-level MP
D4. $Wx \to Mx, \neg Mx, Wx \vdash Mx$	from D3 by monotony
D5. $\neg Mx \vdash \neg Mx$	identity inference
D6. $Wx \to Mx, \neg Mx, Wx \vdash \neg Mx$	from D5 by monotony
D7. $Wx \to Mx, \neg Mx \vdash \neg Wx$	from D4, D6 by reductio ad absurdum
D8. $\forall x[Wx \to Mx], \neg Mx \vdash \neg Wx$	from D7 by second-level $\forall^-$
D9. $\forall x[Wx \to Mx], \neg Mx \vdash \exists x[\neg Wx]$	from D8 by second-level $\exists^+$
D10. $\forall x[Wx \to Mx], \neg Mx \vdash \neg \forall x[Wx]$	from D9 by second-level qfr. interchange
D11. $\forall x[Wx \to Mx], \exists x[\neg Mx] \vdash \neg \forall x[Wx]$	from D10 by $\exists^-$
D12. $\forall x[Wx \to Mx] \vdash \exists x[\neg Mx] \to \neg \forall x[Wx]$	from D10 by conditional proof

The second-level versions of first-level rules are obtained by the transformation described earlier. In particular, in line D3, the second-level version of modus ponens is the rule $\alpha \vdash \varphi, \beta \vdash (\varphi \to \psi)$ / $\alpha, \beta \vdash \psi$. In D8, second-level $\forall^-$ is the rule $\alpha \vdash \forall x[\varphi]$ / $\alpha \vdash \varphi[t/x]$ whenever the substitution is clean. Line D10 uses the second-level version of quantifier interchange: $\alpha \vdash \exists x[\neg\varphi]$ / $\alpha \vdash \neg\forall x[\varphi]$. On the other hand, the special first-level rule $\alpha \vdash \alpha$ is left unchanged so that the derivation can start with something.

Exercise 10.5.3 (1)

(a) Specify the substitutions that make lines D3, D8 and D10 instances of the second-level rules mentioned.

(b) Elaborate on this second-level proof for the simplified version of the example to construct one for the example itself.

In the example, we have allowed any familiar first-level rule from tables in Chaps. 8 and 9 to be transformed into a second-level one. It should be noted, however, that most presentations of sequent calculi are much more parsimonious. They allow only a very few rules: one ‘introduction’ and one ‘elimination’ rule for almost all connectives (although an exception must be made for negation and for the falsum, which do not have any classically correct introduction rules unless we admit sets of formulae on the right of sequents). Despite their parsimony, these nevertheless suffice to generate all classically valid inferences, and because of the same parsimony, they permit normal forms for second-level derivations, which in turn make inductive arguments about them possible. That, however, would take us beyond the limits of this book.

Exercise 10.5.3 (2) Consider the following simple inference: If one person is a relative of another, who is a relative of a third, then the first is a relative of the third. If one person is a relative of another, then that other is a relative of the first. So if a person is a relative of two others, then they are relatives of each other.

(a) Establish its validity in English (with the help of variables if you wish but no symbols).
(b) Translate it into the symbolism of first-order logic, and establish validity by a semiformal suppositional argument.
(c) Rewrite the argument as a split-level proof containing subproofs.
(d) Rewrite it as purely second-level proof.

Exercise 10.5.3 (3) In Chap. 9, we remarked that in first-order logic, the symmetry and transitivity of identity may be derived from its reflexivity and the replacement rule. It is hardly possible to express that derivation in English without any symbols, since the replacement rule itself is difficult to formulate except in a symbolic manner. So:

(a) Construct a semiformal suppositional argument for each of symmetry and transitivity. *Hint*: The central step in that argument was made in exercise 9.4.4 (1) (b).
(b) Rewrite the argument as a split-level proof containing subproofs.

End-of-Chapter Exercises

Exercise 10.1: Consequence Relations

(a) Show that cumulative reflexivity taken together with monotony implies (plain) transitivity (the principle that whenever $\alpha \Rightarrow \beta$ and $\alpha \Rightarrow \beta$, then $\alpha \Rightarrow \gamma$).
(b) Let $\Rightarrow$ be the relation defined by putting $A \Rightarrow \beta$ iff either (i) A is a singleton $\{\alpha\}$ and $\alpha = \beta$ or (ii) A has more than one element. Show that $\Rightarrow$ satisfies strong reflexivity, monotony and transitivity, but not cumulative transitivity.

Exercise 10.2: Consequence Operations*

(a) In an exercise, we noted that the LHS $\subseteq$ RHS half of the 'distribution' equality $Cn_{\vdash}(A \cup B) = Cn_{\vdash}(A) \cup Cn_{\vdash}(B)$ fails for classical consequence $Cn_{\vdash}$. (i) Show that the equality formed by 'closing' the RHS, namely, $Cn(A \cup B) = Cn(Cn(A) \cup Cn(B))$, holds for all consequence operations, and (ii) express this equality in terms of consequence relations.
(b) (i) Show that for any consequence operation Cn, if $A \subseteq Cn(B)$ and $B \subseteq Cn(A)$, then $Cn(A) = Cn(B)$, and (ii) express the principle in terms of consequence relations.
(c) Show that for classical consequence, we have the equality $Cn_{\vdash}(A) \cap Cn_{\vdash}(B) = Cn_{\vdash}[Cn_{\vdash}(A) \cap Cn_{\vdash}(B)]$. *Hint*: One half is easy; the other is quite difficult. It needs the compactness of classical consequence and uses disjunction and distribution.

Exercise 10.3: Proof Construction

(a) Given premises $p \rightarrow ((r \vee s) \rightarrow t)$, $q \rightarrow (\neg(s \vee u) \vee t)$, s, get the conclusion $(p \vee q) \rightarrow t$, by a semiformal suppositional proof using CP and DP.

(b) Rewrite the semiformal suppositional proof as an explicit split-level proof.

(c) Construct an informal proof, from no premises at all, that there is no set whose elements are precisely the sets that are not elements of themselves. *Remark*: Those familiar with the paradoxes of naïve set theory will recognize the Russell paradox here.

(d) Express the last statement in the language of first-order logic, taking the class of all sets as your domain of discourse (thus, no predicate for '. . . is a set' is needed) and elementhood as the sole relation. Rewrite your informal proof as a semiformal suppositional argument, then as a split-level proof.

Exercise 10.4: Higher-Level Rules

(a) Show that when $\Rightarrow$ is monotonic, then conditional proof may equivalently be formulated in the following more general way: $A', \beta \Rightarrow \gamma \,/\, A \Rightarrow \beta \rightarrow \gamma$ whenever $A' \subseteq A$.

(b) Explain informally how the rule of proof by cases may be obtained from disjunctive proof. *Hint*: Make use of a simple tautology.

(c) Explain informally how, conversely, disjunctive proof may be obtained from proof by cases. *Hint*: Make use of disjunctive syllogism.

(d) Write out the converses of disjunctive proof, *reductio*, $\forall^+$, $\exists^-$ and check whether they are classically correct. Comment on the status of the provisos for $\forall^+$, $\exists^-$ in the converse rules. *Remark*: As disjunctive proof is a two-premise rule, you will need to define its 'converse' a little creatively.

Exercise 10.5: Introduction and Elimination Rules* A second-level inference rule $A_1 \Rightarrow \alpha_1, \ldots, A_n \Rightarrow \alpha_n \,/\, B \Rightarrow \beta$ for propositional logic is said to be *structural* iff it contains no connectives at all, and is an *introduction* (resp. *elimination*) rule iff it contains only one connective, and that connective occurs just once, with the unique occurrence in β (resp. in some formula in B).

(a) Classify the second-level rules discussed in this chapter (monotony, cumulative transitivity, conditional proof, disjunctive proof, proof by cases, reductio, $\forall^+$, $\exists^-$) as structural, introduction, elimination or 'none of the above'.

(b) Call a truth-functional connective $*(p_1, \ldots, p_n)$ *contrarian* iff $v(*(p_1, \ldots, p_n)) = 0$ when all $v(p_n) = 1$. (i) Identify five familiar contrarian truth-functional connectives. (ii) Show by a simple argument that no contrarian connective has a classically correct introduction rule. *Hint*: Read the definition of an introduction rule carefully.

Exercise 10.6: Wild Card Exercises – Uncertain Inference Relations*

(a) Google 'lottery paradox' and 'preface paradox', write up short descriptions of them and articulate what they tell us about the status of second-level rule for $\wedge^+$ in uncertain inference.

(b) In the light of Chap. 6 on probability, what would you expect the status of disjunctive proof to be in probabilistic inference? What about proof by cases?

Selected Reading

The references given for Chaps. 8 and 9 do not contain anything on the material of this chapter. The concept of a consequence operation/relation is explained in the following sources:

Wikipedia entry on consequence operators. http://en.wikipedia.org/wiki/Consequence_operators

Makinson D (2007) Bridges from classical to nonmonotonic logic. College Publications, London, chapter 1

Wójcicki R (1988) Theory of logical calculi: basic theory of consequence operations. Synthese library, vol 199. Reidel, Dordrecht, chapter 1

The Wikipedia article highlights algebraic and topological connections of the concept, with lots of useful links. The Makinson text also goes on to discuss uncertain inference relations, mainly qualitative but also probabilistic.

The following are two very clear textbook discussions of traditional informal proof strategies in mathematics. Section 2.6 of the Bloch text also contains useful advice on writing proofs in coherent and elegant English:

Bloch ED (2011) Proofs and fundamentals: a first course in abstract mathematics, 2nd edn. Springer, New York, chapter 2

Velleman DJ (2006) How to prove it: a structured approach, 2nd edn. Cambridge University Press, Cambridge, chapter 3

For a formal study of higher-level rules and proof theory, a good entry point would be:

von Plato J The development of proof theory. In: Stanford encyclopedia of philosophy. http://plato.stanford.edu/entries/proof-theory-development

For an aerial view of the jungle of textbook systems of 'natural deduction', with also a brief introduction to proof theory, see:

Pelletier J, Hazen A (2012) Natural deduction. In: Gabbay D, Woods J (eds) Handbook of the history of logic. Central concepts, vol 11. Elsevier North-Holland, Amsterdam

Index

D. Makinson, *Sets, Logic and Maths for Computing*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-2500-6,