Undergraduate Topics in Computer Science

David Makinson

Sets, Logic and Maths for Computing

Third Edition

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 of which include fully worked solutions.

The UTiCS concept relies on high-quality, concise books in softback format, and generally a maximum of 275–300 pages. For undergraduate textbooks that are likely to be longer, more expository, Springer continues to offer the highly regarded Texts in Computer Science series, to which we refer potential authors.

More information about this series at http://www.springer.com/series/7592

David Makinson

Sets, Logic and Maths for Computing

Third Edition

Springer

David Makinson
Department of Philosophy,
Logic and Scientific Method
London School of Economics
London, UK

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-030-42217-2 ISBN 978-3-030-42218-9 (eBook)
https://doi.org/10.1007/978-3-030-42218-9

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

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 informatics or 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 required for entering the world of the information and computing 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, often 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 a compendium is, beginning students tend to feel intimidated, lost, and unclear about what is in it 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, from the world of sets,

- *Collecting* things together. In the jargon of mathematics, first steps with *sets*.
- *Comparing* things. This is the theory of *relations*.
- *Associating* one item with another. Introduces the key notion of a *function*.
- *Recycling outputs as inputs*. We explain the ideas of *recursion* and *induction*.

From other parts of finite mathematics,

- *Counting*. The mathematician calls it *combinatorics*.
- *Weighing the odds*. This is done with *probability*.
- *Squirrel math*. Here we make use of *trees*.

From logic,

- *Yea and Nay*. Just two truth-values underlie *propositional logic*.
- *Everything and nothing*. That is what *quantificational logic* is about.
- *Just supposing*. How *complex proofs* are built out of simple ones.
- *Sticking to the Point*. How to make logic sensitive to relevance (new to this edition).

How Should You Use It?

Without an understanding of 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. So, there is work to be done.

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 two essential things are to *understand* and to be able to *apply*.

But that is a 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 register 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 integral part of the learning process. Skip them at your peril. That is part of what is meant by the old proverb ‘there is no royal road in mathematics’. Although all exercises in this edition are accompanied by a solution, you will benefit much more if you first cover 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. Thus, they are also part of the development of the general theory.

By the same token, don’t get into the habit of skipping verifications when you read the text. Postpone, yes, but omit, no. In mathematics, you have never fully appreciated a fact unless you have also grasped *why* it is true, i.e. 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 has wrought disaster in some secondary and university education systems.

In practice, the tools that are 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 the competing virtues and shortcomings in different choices of material and ways of presenting it.

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 often it is disastrous. There are situations where the reverse is 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 minimally 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 for the same domain; broaden to their qualitatively formulated structural versions; finally, give the most general articulation, 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 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 approach is to familiarize readers 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 counting and probability to permit rapid 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. Again, 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 should learn which tool to choose when, and why, as well 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. Don't we 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 meaning 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 early chapters, notions of logic are identified briefly as they arise in the discussion and verification 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 four 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 to help study it.

From time to time there are boxes of a different nature—'Alice boxes'. This little trouble-maker 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. Quite often, we comment on such variations. In particular, two points in quantificational (first-order) logic always give trouble if variants are not compared explicitly. One concerns distinct, although ultimately equivalent, ways of reading the quantifiers; the other arises from differing conventions for using the terms 'true' and 'false' in application to formulae containing free variables.

Choice of Topics

Overall, our choice of topics is fairly standard, as can be seen from the chapter titles. If strapped for class time, an instructor can omit some later sections of some chapters, or even entire chapters that are not close to the interests of the students or the dictates of the curriculum. For example, few computer science curricula will require the material of Chaps. 10 and 11 (levels of inference, logic with relevance) and few philosophy students will be fired up by the Chap. 5 (counting) or the last part of Chap. 7 (on unrooted trees). But it is urged that Chaps. 1–3, the first four sections of Chap. 7, as well as 8 and 9, be taught uncut as just about everything in them is useful to everybody.

We have not included a chapter on the theory of graphs. That was a difficult call to make, and the reasons for the decision were as follows. Although trees are a particular kind of graph, there is no difficulty in covering everything we want to say about rooted 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 two 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 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 induction on the natural numbers it goes on to explain number-free forms of both inductive proof and recursive definition including, in particular the often-neglected structural forms that are so important for computer science, logic and theoretical linguistics. We also explain the very general versions based on arbitrary well-founded relations. Throughout the presentation, the interplay between recursive definition and inductive proof is brought out, with the latter piggy-backing on the former. This chapter ends up being the longest in the book.

Instructors may be surprised that the chapters on logic do not attempt to drill readers in building derivations in formal notation, following what is known as a system of 'natural deduction'. The reason is that the author, after years teaching a variety of such systems, has become convinced that they are of marginal pedagogical benefit. Students take a long time to become accustomed to the idiosyncrasies of whichever layout they are exposed to, tend to lose sight of essential principles in the finicky procedural details, and forget most of it as soon as the exams are over. Instead, Chap. 10, on proof and consequence, seeks to make clear the basic ideas that underlie natural deduction, explaining the difference between

first-level, second-level and split-level derivations as well as how to squeeze a derivation tree into a derivation sequence and how to flatten a split-level derivation into a format that mimics a first-level one.

Finally, a decision had to be made whether to include specific algorithms in the book. Most first-year students of computing will be taking courses, in parallel, 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. Instructors wishing 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 to master 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 case of philosophy there was, in the second half of the twentieth century, an unfortunate tendency to teach only elementary logic, leaving aside any instruction on sets, relations, functions, recursion/induction, probability and trees. The few students going on to more advanced courses in logic were usually exposed to such tools in bits and pieces, but without a systematic grounding. Even within logic, the election of material was often quite narrow, focussing almost entirely on natural deduction. But as already remarked, it is difficult for the student to get a clear idea of what is going on in logic without having those other concepts available in the tool-kit. 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 unrooted trees) are equally vital for an adequate course for students of philosophy. With some additional practice on the subtle art of making approximate representations of statements from ordinary language in the language of propositional and predicate logic, the book can also be used as a text for such a course.

The Third Edition

The first edition of this book was published in 2008; the second appeared in 2012 and the many changes made were itemized in the preface for that edition. This third edition adds Chap. 11, *Sticking to the point: relevance in logic*, which introduces procedures of syntactic control for logical consequence; it should be good fun for students of computing as well as those from philosophy.

In previous editions, only about half of the exercises were accompanied by a solution and some readers using the text for self-study let the author know their unhappiness with that situation. This edition provides solutions to all exercises, mostly in full but occasionally in outline. It is hoped that, as a result, the text is friendlier than before to readers working alone and takes some of the burden off instructors. To be honest, the task of writing out solutions in full also helped better appreciate the student's situation facing the exercises, leading to many modifications. Together, the solutions and new chapter increase the size of the book by about a third.

Even more than in the first two editions, the third takes the time to inform readers of shorthand ways of speaking and useful 'abuses of language'. Such information is more important than often realized. In the author's experience, when a student has trouble assimilating a point, as often as not it is a result of misunderstanding the language in which it is made.

The notation chosen for the text is the same as in the previous editions except for two items, both in the chapters on logic. For classical logical consequence, we now use the standard double turnstile $\models$ rather than $\vdash$, reserving the latter sign for consequence relations in general. For the substitution of a term t for all free occurrences of a variable x in a formula α we now write $\alpha_{x:=t}$ instead of the more common notation $\alpha(t/x)$ used in the previous editions; this is to ensure that there can be no hesitation about which way round the substitution goes when t is itself a variable.

To bring out its general structure, the book has now been divided into three broad parts: *Sets* (including mathematical objects built with them, such as relations and functions), *Math* (in the traditional quantitative sense focusing on counting and finite probability), and *Logic*. Of course, this partition leaves open the question where to put the chapter on induction and recursion, which begins with induction on the natural numbers while going on to general issues of structural and well-founded recursion and induction, as also the chapter on trees which, conversely, begins as part of the theory of relations but ends with a combinatorial flavour. The former was put in *Sets*, while the latter went into *Math*, but the reverse allocation would be almost as appropriate.

As the third edition was being prepared a colleague asked whether, leaving aside all options of presentation, the book has content that is not easily found in other introductory texts. The author's response is relayed here for anyone with the same question. Sections 4.6 (structural recursion and induction), 4.7.3 (defining functions by well-founded recursion on their domains) and 8.4.4 (most modular versions of sets of propositional formulae) bring within the grasp of a beginner important notions that are usually left unexplored. The same is true of most of Chap. 10 (consequence relations and higher-order rules of inference) as well as all of Chap. 11 (relevance in logic).

A section of the author's webpage https://sites.google.com/site/davidcmakinson/ is earmarked for material relating to this edition: typos and other errata, additional comments, further exercises, etc. As the text goes to press these rubrics are empty but, with the help of readers, they will surely soon be populated. Observations should go to david.makinson@gmail.com.

London, UK
February 2020

David Makinson

Acknowledgements The late Colin Howson and ensuing department heads at the London School of Economics (LSE) provided a wonderful working environment and the opportunity to test material in classes. Anatoli Degtyarev, Franz Dietrich, Valentin Goranko, George Kourousias, Abhaya Nayak and Xavier Parent provided helpful comments on drafts of the first two editions; George also helped with the diagrams. For the third edition, Chap. 11 benefitted from valuable remarks by Lloyd Humberstone, Abhaya Nayak, Xavier Parent, Nick Smith and Peter Smith.

A number of readers and LSE students drew attention to typos that remained in the first two editions: Luc Batty, Alex Bendig, Michael Broschat, Panagiotis Dimakis, Niklas Fors, Rick Greer, Herbert Huber, Deng (Joe) Jingyuan, Regina Lai, Daniel Mak, Pascal Meier, Fredrik Nyberg, Daniel Shlomo. Thanks to you all.

Contents

Part I
Sets

Collecting Things Together: Sets

1

Chapter Outline

In this chapter we introduce the student to the world of sets. Actually, only a little bit of it, enough 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: *meet*, *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 that point, the material is all 'flat' set theory, in the sense that it does not look at what else 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 meet 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.

D. Makinson, *Sets, Logic and Maths for Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-42218-9_1

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 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 the *singleton* for that element, 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. At least to begin with, we will need to consider only 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 simpler 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 diagramed, 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 often useful: iff there is no element of A that is not an element of B. Looking at the same relation from the reverse direction, we also say that B *includes* A (B is a *superset* of A) and write $B \supseteq A$. For brevity, we write $A \nsubseteq B$ to mean that A is not included in B.

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 We are, and we could. But life will be easier if for the moment we just use these particles intuitively. We will get around to their exact logical analysis later.

Exercise 1.2.1
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 $C \subseteq B$, $E \subseteq D$, $F \subseteq B$, $B \subseteq F$, $C \subseteq F$. None of the other converses hold. For example, $B \nsubseteq C$, since $7 \in B$ but $7 \notin C$. Note also that $E \nsubseteq B$, since we are not taking 1 to be a prime number.

Warning Box: Subset versus element 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 $\mathbf{N}^+$, i.e. each of its elements 2, 4, 6, … is an element of $\mathbf{N}^+$; but E itself is not an element of $\mathbf{N}^+$. Neglect of this distinction quickly leads to serious confusion.

1.2.2 Identity and Proper Inclusion

A basic principle of set theory, called the axiom (or postulate) of *extensionality*, says that whenever sets A and B have exactly the same elements then they are identical; they are one and the same set, and we write $A = B$. Clearly, A and B have exactly the same elements iff both $A \subseteq B$ and $B \subseteq A$. So we can also say: $A = B$ iff both $A \subseteq B$ and $B \subseteq A$.

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)
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. By the way, in set theory we do not distinguish between identity and equality; they are synonyms for the same concept.

Exercise 1.2.2 (2)
Which among the sets in the preceding exercise, 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
All of B through F are proper subsets of A, with 9 and 8 as witnesses for all of them. Also, $C \subset B$ with witnesses 2,7; $E \subset D$ with sole witness 2. Comment: $F \not\subset B$, since B and F have exactly the same elements.

Exercise 1.2.2 (3)
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) {+} = {'+'}.

Solution

(a) Yes. The order of enumeration makes no difference: the sets have the same elements.
(b) Yes. Repeating an element in the enumeration is inelegant, but it makes no substantive difference: the sets still have the same elements.
(c) Yes. The elements have been named differently, as well as being written in a different order, but they are the same.

(d) No. 7 is a number while {7} is a set with 7 as its only element, i.e. its singleton.
(e) No. 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 left set is thus the singleton of the right one.
(f) Yes. London is the same city as Londres, and Beijing is the same city as Pekin, although they are named differently. So the sets have exactly the same elements and thus are identical.
(g) No. The left set has the operation of addition as its sole element while, under a standard convention for naming by inverted commas, the right set has as its sole element a certain sign serving as the standard *name* of that operation, so the sets are not identical. *End of solution*.

The distinction between an object and its name is particularly important when dealing with symbols, as is often the case in computer science, logic, linguistics and some parts of mathematics; its neglect can lead to confusion. That said, it is also the practice among mathematicians to omit the visually distracting inverted commas whenever the context makes it clear that we are considering the symbol rather than what it names. Philosophers tend to be less tolerant. The line between sloppiness and fussiness is fine.

Exercise 1.2.2 (4)
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 $B \nsubseteq A$.
(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 $A \subset C$.

Solution

(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: $A = \{1\}$, $B = \{1,2\}$, $C = \{2\}$.

(c) True, by essentially the same argument as for (a) repeated sufficiently many times. In more detail: it suffices to show $A_i \subseteq A_j$ for all $i, j \leq n$. Take any x, and suppose $x \in A_i$; by the definition of inclusion, it is enough to show $x \in A_j$. By the given inclusions, we have $x \in A_{i+1}$, $x \in A_{i+2}$, …, $x \in A_j$ following upwards or round in a loop as needed to A_j. Strictly speaking, we are arguing by induction on the positive integers, to be analysed in Chap. 4.

(d) True. First, suppose the RHS, we want to show the LHS. Since $B \not\subseteq A$ there is an x with $x \in B$ and $x \notin A$, so $A \neq B$ so, since $A \subseteq B$, the definition of strict inclusion tells us that $A \subset B$. Next, suppose the LHS, we want to show the RHS. We can run essentially the same argument in reverse. Since $A \subset B$, the definition of strict inclusion tells us that $A \subseteq B$ but $A \neq B$. From the latter, A does not have exactly the same elements as B, but by the former every element of A is an element of B, so there is some $x \in B$ with $x \notin A$, so by the definition of strict inclusion again, $B \not\subseteq A$.

(e) True. First, suppose the LHS, we want to show the RHS. Now, if $A \neq B$ then by the definition of strict inclusion, $A \subset B$ and we are done. For the converse, suppose the RHS. If $A = B$ then A has exactly the same elements as B so by the definition of inclusion, $A \subseteq B$, while if $A \subset B$ we again have $A \subseteq B$ by the definition of strict inclusion.

(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: $A = \{1\}$, $B = \{2\}$.

(g) True. Take any sets A, B, C. Suppose $A \subseteq B$ and $B \subset C$. From the latter by the definition of proper inclusion we have $B \subseteq C$. So by exercise (a) we have $A \subseteq C$. It remains to show that $A \neq C$. Since $B \subset C$ we have by exercise (d) that $C \not\subseteq B$; but $A \subseteq B$ by supposition, so $A \neq C$ as desired.

(h) True, by essentially the same argument as for (g).

Logic Box: Proving conditional statements In Exercise 1.2.2 (4), we needed to show several statements of the form 'if the first then the second', and 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*. For example, in the verification of part (a), we did it 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. In the exercise, there also other logical procedures; for example, in part (a) again, we take any x, and suppose $x \in A$. We will have a little more to say about such steps as we go on, and considerably more in the Chap. 10, Sect. 10.3.

Exercise 1.2.2 (4) 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 $B \subseteq C$, our goal became $A \subseteq C$. We then chose an arbitrary x, supposed $x \in A$, and aimed to show $x \in C$. In just a few 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.

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? Essentially, three kinds of resource. 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 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 (g) 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

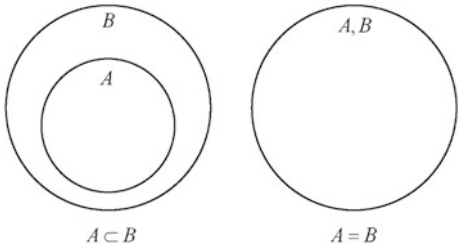

Fig. 1.1 Euler diagrams for proper inclusion and identity

equality, of course, by drawing just one circle and labelling it both A and B. Thus we have the *Euler diagrams* of Fig. 1.1, so named after the eighteenth century mathematician Euler who used them when teaching a princess by correspondence.

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 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, or add some annotation to the diagram for $A \subset B$.

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. $\varnothing$) in an area to indicate that it has no elements, and another kind of mark (say, a cross) to indicate that it does have at least one element. With these conventions, the left diagram of Fig. 1.2 represents $A \subseteq B$ while the right diagram represents $A \subset B$.

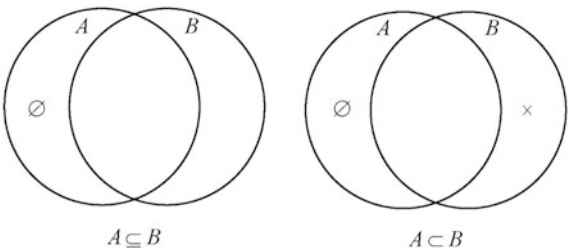

Fig. 1.2 Venn diagrams for inclusion and proper inclusion

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 they can be adapted to represent basic operations on sets as well as relations between them. However, they also have shortcomings. When you have more than two sets to consider, Venn diagrams can become complicated and lose their intuitive clarity—which was, after all, their principal *raison d'être*.

Alice Box: Venn and Euler

Alice So, Venn diagrams are not the same as Euler diagrams?

Hatter As different as chalk and cheese. They are constructed differently and read differently. Unfortunately, people are sometimes rather confused about the names, using the terms interchangeably or even in reverse.

Alice Another question. You said that in Venn diagrams, we indicate that an area is non-empty by putting a cross. But how do we represent non-emptiness in Euler diagrams?

Hatter That's a delicate point. The usual convention for Euler diagrams is that all areas considered are non-empty; under that convention there is no way of representing non-emptiness. But in practice, one occasionally sees people mixing the layout of Euler with the annotations of Venn, adding crosses or $\varnothing$ signs to various parts of their Euler diagram.

Alice Is that legitimate? Can we mix chalk with cheese?

Hatter Well, diagrams are like these are heuristic devices, not rigorous methods of proof. So, it can be legitimate, so long as it helps intuition and provided you make it clear what conventions you are following.

Exercise 1.2.3 (1)

(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?

Solution Outline

(a) Euler is straightforward: put the A circle inside the B circle, and that inside the C circle. For Venn, put $\varnothing$ in the *smallest* areas that must be empty, *then* place crosses in the *smallest* areas that must be non-empty. You will end up with three areas marked $\varnothing$ and two marked by crosses

(b) When constructing Euler diagrams, think of the various possible relations between B and C. You will end up with four diagrams, according as $B = C$, $B \subset C$, $C \subset B$ and none of the above, i.e. where B, C properly intersect with each other. When building the Venn diagram try to proceed as you did for (a). Again, you will end up with three areas marked $\varnothing$ and two marked by crosses, but the crosses will need to be placed in boundary lines rather than inside cells to indicate, in a rather ad hoc manner, that there is something on one or the other side of the boundary.

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 is to *specify all its elements individually*, say by listing them between curly brackets, as we did in Exercise 1.2.2 (3). Evidently, such a specification can be completed only when there are finitely many elements, and it is convenient only when the set is fairly small. The order of enumeration makes no difference to what items are elements of the set, e.g. $\{1,2,3\} = \{3,1,2\}$, 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 most of the sets in Exercise 1.2.1. There is a notation for this. For example, we write the first set as follows: $A = \{x \in \mathbf{N}^+ : x < 10\}$, where $\mathbf{N}^+$ stands for the set of all integers greater than zero (the positive integers). Some texts use a vertical bar in place of a colon.

Exercise 1.2.4 (1)

(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. $D = \{x \in \mathbf{N}^+ : x$ divides all even integers$\}$; $E = \{x \in \mathbf{N}^+ : x$ is less than or equal to every positive integer$\}$. *End of solution.*

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, …}, {2,3,5,7,11, …}. 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, and normally it is the most familiar or simplest is the one that is meant. *Without a specific understanding of that kind, the notation is meaningless*. We can also use the suspended dots notation to abbreviate a finite set. Thus, in the exercise above, we could write as $A = \{1, \ldots, 9\}$. Again, this is an informal notation; it hovers between two formal ones: complete enumeration or by a common property.

These two ways of identifying a set are not the only ones available. In a later chapter we will be looking at another very important way, 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 that is hinted at by writing 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?

$$A = \{x \in \mathbf{N}^+ : x \text{ is both even and odd}\}$$
$$B = \{x \in \mathbf{N}^+ : x \text{ is prime and } 24 \leq x \leq 28\}$$

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. In other words, $A = B$, 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*. It 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)
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 The solution to Exercise 1.3.1 (1) is certainly short, but rather strange. 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 distinct sets X, Y, Z such that X is disjoint from Y and Y is disjoint from Z, but X, Y, Z are not pairwise disjoint.
(f) Show that the empty set is disjoint from every set, including itself.

Solution

(a) D is disjoint from C, E is disjoint from B and from C, F is disjoint from E. And, of course, when one set is disjoint from another, that other is disjoint from the first.

(b) In the Euler diagram you will have two circles quite separate from each other. In the Venn diagram you will put $\varnothing$ in the common part of the two circles.
(c) Put a cross in the common part of the two circles.
(d) If we follow the standard convention for Euler diagrams that all areas are non-empty, then we need four diagrams to represent four alternative ways in which the overlapping can occur. If we don't follow that convention and don't allow mixed diagrams, we can't represent overlapping by a Euler diagram at all—indeed, we can't even diagram the non-emptiness of a single set.
(e) One simple example: put $X = \{1,2\}$, $Y = \{3\}$, $Z = \{1,4\}$.
(f) Since the empty set has no elements, it has no elements in common with any set, not even with itself.

1.4 Boolean Operations on Sets

We now define some operations on sets, that is, ways of constructing new sets out of old beginning, in this section, with three basic ones: meet, union, and relative complement.

1.4.1 Meet

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

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

Of the two terms, 'meet' and 'intersection', the former is short while the latter is perhaps more graphic. Mathematicians and computer scientists tend to favour *meet*, and we will do the same.

Exercise 1.4.1 (1)
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 meet.

Solution

For (a): To show $A \cap B \subseteq A$, notice that whenever $x \in A \cap B$ then immediately by the definition of meet we have $x \in A$ and we are done. Similar reasoning shows $A \cap B \subseteq B$.

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 $x \in A$; similarly, since $x \in X$ and $X \subseteq B$ we have $x \in B$. So, by the definition of meet, $x \in A \cap B$ as desired.

For (c): Take any x and suppose $x \in A \cap B$. Then $x \in A$ and $x \in B$ so $x \in B$ and $x \in A$, so $x \in B \cap A$. This shows $A \cap B \subseteq B \cap A$. The converse is verified similarly. It can also be seen as saying the same thing with the choice of variables A, B reversed, and so is already established by the first verification.

For (d): Take any x and suppose $x \in A \cap (B \cap C)$. Then $x \in A$ and $x \in B \cap C$, so in turn $x \in B$ and $x \in C$. Thus $x \in A$ and $x \in B$, also $x \in A$ and $x \in C$, so $x \in A \cap B$, also $x \in A \cap C$, and hence $x \in (A \cap B) \cap C$. The converse is verified similarly.

For (e): Clearly $x \in A$ iff both $x \in A$ and $x \in A$.

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 $A \cap B = \varnothing$. *End of solution*.

Property (a) in Exercise 1.4.1 (1) 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. These notions will be useful in later chapters on relations.

Heuristics Box: Unpack, reason, repack The solutions to the above exercises exhibit a pattern that reappears whenever we carry out simple verifications of basic properties. We unpack the notions involved in the problem to be solved, using their definitions, rearrange and reason with the information thus obtained, and pack up again using the same definitions. Of course, less simple verifications will require more, but the heuristic works well for basic ones.

Meet 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: αtrue with β false, α false with βtrue, α false with βfalse. All this can be put in the form of a *truth-table*, presented in the logic box for conjunction.

Logic Box: Conjunction

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

In each row, the left two entries represent 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 noticed in Exercise 1.4.1 (1), the behaviour of meet (as an operation on sets) reflects that of conjunction (as a connective between statements), because of the direct way in which the latter is used in the definition of the former. For example, the commutativity of meet 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 meet 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:

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

The contrast with meet may be illustrated by Venn diagrams, but they are read differently 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 the diagrams of 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 meet 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 present.

Exercise 1.4.2 (1)
Show the following:

(a) $A \subseteq A \cup B$ and $B \subseteq A \cup B$.
(b) Whenever $A \subseteq X$ and $B \subseteq X$ then $A \cup B \subseteq X$.
(c) $A \cup B = B \cup A$ (commutation).
(d) $A \cup (B \cup C) = (A \cup B) \cup C$ (association).
(e) $A \cup A = A$ (idempotence).
(f) $A \cup \varnothing = A$ (bottom).

Solution Outline
Re-run 'upside down' the solution to Exercise 1.4.1 (1).

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

Solution
For (a): upper bound, for (b): least upper bound. *End of solution.*

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 way of making a disjunction false. The truth-table is given in the logic box for disjunction.

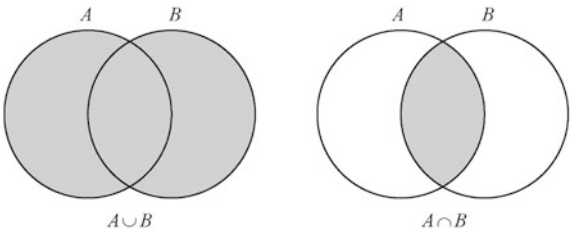

Fig. 1.3 Venn Diagrams for union and meet

Just as with meet and conjunction, the properties of union between sets reflect those of disjunction as a connective between statements, essentially because of the direct way in which the logical connective is used to define the set-theoretic operation.

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.

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?

Solution
The same table as for inclusive disjunction except that in the top row $\alpha \vee \beta$ is given the value 0.

We have seen some of the basic properties of meet 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)
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 meet over union)
(d) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ (distribution of union over meet)
(e) $A \subseteq B$ iff $A \cup B = B$, and likewise iff $A \cap B = A$.

Solution
In the following answers we use the familiar abbreviations LHS, RHS for the left, right sides respectively of a statement of inclusion or of identity.

(a) If $x \in$ LHS then $x \in A$ and $x \in B$, so x is an element of at least one of A, B, so $x \in$ RHS.
(b) We already know from the exercises on meet and on union taken separately that $A \cap (A \cup B) \subseteq A$ and that $A \subseteq A \cup (A \cap B)$. It remains to show the two converse inclusions. For $A \subseteq A \cap (A \cup B)$, suppose $x \in A$. Then also $x \in A \cup B$, so $x \in A \cap (A \cup B)$. For $A \cup (A \cap B) \subseteq A$, suppose $x \in A \cup (A \cap B)$. Then $x \in A$ or $x \in A \cap B$, and in both cases $x \in A$.
(c) 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. Suppose alternatively 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. Thus, both cases give us $\in (A \cap B) \cup (A \cap C) =$ 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 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.
(d) We give only the hint: work in the same spirit as for (c).
(e) We show $A \subseteq B$ iff $A \cup B = B$. Suppose first that $A \subseteq B$; we need to show $A \cup B = B$. Now $B \subseteq A \cup B$ always holds, as shown in an earlier exercise, so we need only show that $A \cup B \subseteq B$. But if $x \in A \cup B$ then either $x \in A$ or $x \in B$. But since by supposition $A \subseteq B$, $x \in B$ and we are done.

Logic Box: Breaking into cases In several parts of Exercise 1.4.2 (4) we used a technique known as 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 $\beta$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, disjunctive proof holds the goal unchanged throughout. Sometimes, the arguments carried out under the two suppositions resemble each other, sometimes they are quite different. But the suppositions

themselves must be made quite separately: we cannot use one of them in the argument for the other.

Closely related to disjunctive proof is a procedure known as *proof by cases*. We have a set A of premises and we want to get a conclusion γ. It can sometimes be useful to articulate a suitable proposition β, first add the supposition that β is true, then the supposition that β is false. If we can get γ in each of these two cases separately, then we know that it follows from the premises in A considered alone. There are subtle differences between disjunctive proof and proof by cases but in informal practice, as in this chapter, they are almost indistinguishable; we leave the fine points of discussion to Chap. 10, Sect. 10.4.1, where they are both studied in detail.

Sometimes we need to break cases into subcases, and so on—in principle without limit. But when the case-depth becomes too large, the proof becomes rather inelegant, and it can be a good idea to search for a neater one.

Exercise 1.4.2 (5)
As 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.

Solution
Suppose for the first case that n is even. Then $n = 2k$ for some positive integer k, so $n^2 = 2k \cdot 2k = 4k^2$, which is even. Suppose for the other case that n is odd. In the sub-case that $n = 1$ we have $n^2 = 1$, which is odd. In the sub-case that $n > 1$, we have $n = 2k + 1$ for some positive integer k, so $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1$, which is odd.

Alice Box: Overlapping cases

Alice What if *both* cases are true? For example, in the solution to exercise 1.4.2 (4) (c), 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, while in others 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 within A* (also called *A less B*),

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)

(a) Draw a Venn diagram for $A\backslash B$.
(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 $A\backslash(B\backslash C) \subseteq (A\backslash B) \cup C$.

Solution Outline

(a) It is like the right diagram in Fig. 1.3 but with just the left crescent shaded.
(b) For a simple example where neither of A, B includes the other, put $A = \{1,2\}$, $B = \{2,3\}$. Then $A\backslash B = \{1\} \neq \{3\} = B\backslash A$. For another kind of example, takes any sets A, B where $\varnothing \neq A \subset B$, e.g. $A = \{1\}$, $B = \{1,2\}$; then $\varnothing = \text{LHS} \neq \text{RHS} = \{2\}$.
(c) For (i), $x \in A\backslash A$ iff $x \in A$ and $x \notin A$, which is impossible, showing that $A\backslash A = \varnothing$. For (ii), $x \in A\backslash\varnothing$ iff $x \in A$ and $x \notin \varnothing$, which holds just if $x \in A$.
(d) For (i), suppose $A \subseteq A'$ and $x \in A\backslash B$. From the latter supposition, $x \in A$ and $x \notin B$ so, using the former supposition, $x \in A'$ and $x \notin B$, so $x \in A'\backslash B$. For (ii), reason along similar lines.
(e) For (i), we first show LHS $\subseteq$ RHS. Suppose $x \in$ LHS. Then $x \in A$ and $x \notin B \cup C$, so $x \notin B$ and $x \notin C$, so both $x \in A$ and $x \notin B$ and $x \in A$ and $x \notin C$, so that $x \in$ RHS as desired. For the converse LHS $\subseteq$ RHS, run the same argument backwards. For (ii) argue in the same fashion.
(f) We first show LHS $\subseteq$ RHS. Suppose $x \in$ LHS. Then $x \in A\backslash B$ and $x \notin C$ so, from the former, $x \in A$ and $x \notin B$. Putting two of these together, $x \in A\backslash C$ and so, combining with $x \notin B$, we have $x \in$ RHS as desired. For the converse, argue in similar fashion.
(g) Here is a very simple example: take A to be any non-empty set, e.g. $\{1\}$, and put $C = B = A$. Then LHS $= A\backslash(A\backslash A) = A\backslash\varnothing = A$ while RHS $= (A\backslash A)\backslash A = \varnothing\backslash A = \varnothing$.
(h) Suppose $x \in$ LHS. Then $x \in A$ and $x \notin B\backslash C$. From the latter, either $x \notin B$ or $x \in C$. In the former case, $x \in A\backslash B$, so in both cases $x \in (A\backslash B) \cup C$ and we are done. *End of solution*.

Logic Box: Negation You will have noticed that in our discussion of difference there are 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	α
1	0
0	1

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

The difference operation acquires particular importance in a special context. Suppose that we are carrying out an extended investigation into the elements of a 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 that set, which we therefore consider as 'fixed'. Then it is customary to refer to the fixed set as a *local universe* writing it, say, 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 U).

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)

(a) Taking the case that A is a local universe U, rewrite equations (e) (i) and (ii) of the preceding exercise using the simplest of the complementation notations mentioned above.
(b) Show that (i) $-(-B) = B$, (ii) $-U = \varnothing$, (iii) $-\varnothing = U$.

Solution

(a) (i) $-(B \cup C) = -B \cap -C$; (ii) $-(B \cap C) = -B \cup -C$. To be strict, the right-hand sides should have more brackets with $(-B) \cap (-C)$, $(-B) \cup (-C)$ respectively, but a convention about the scope of the complementation operation, like the minus sign in arithmetic, permits us to omit the brackets, reducing clutter.
(b) (i) We need to show that $-(-B) = B$, in other words that $U\backslash(U\backslash B) = B$ whenever $B \subseteq U$ (as is 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 $x \in B$. Putting these together we have $x \in B =$ 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.

(b) (ii) It suffices to show that nothing is an element of the LHS, so it suffices to suppose that $x \in$ LHS and get a contradiction. Suppose that $x \in$ LHS. Then $x \in U \backslash U$, so $x \in U$ and $x \notin U$, which is impossible as required.
(b) (iii) Suppose $x \in$ LHS. Then $x \in U \backslash \varnothing$ so $x \in U =$ RHS. Suppose conversely that $x \in U$. Now we know that $x \notin \varnothing$, so $x \in U \backslash \varnothing =$ LHS. *End of solution.*

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 assumption that $B \subseteq U$.

Exercise 1.4.3 (3)
Show that if $B \nsubseteq U$, then $U \backslash (U \backslash B) \neq B$.

Solution
Suppose $B \nsubseteq U$. Then there is an x with $x \in B$ and $x \notin U$, and from the latter we have $x \notin U \backslash (U \backslash B)$, so RHS $\neq$ LHS.

Alice Box: Complementation

Alice There is something about complementation 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 $U - B$ for a given set B. But why not define it in absolute terms? Put the absolute complement $-B$ of B to be the set of all x that are not in B. Then, if you need a complement relative to some B, put $U - B = U \cap -B$.

Hatter A natural idea indeed! Unfortunately, it leads to contradiction, so we cannot follow it.

A laconic reply: Alice deserves more than that. You may have heard of *Russell's paradox*, which considers the set of all those sets that are not elements of themselves, i.e. $\{x: x \text{ is a set and } x \notin x\}$. Call this set S. Question: do we have $S \in S$ or $S \notin S$? Suppose first that $S \in S$. Then, by the very definition of S, we get $S \notin S$, giving a contradiction. Suppose on the other hand that $S \notin S$. Since S is a set, the definition of S gives us $S \in S$, again a contradiction. We are forced to conclude that there is no such set S—contrary to initial expectations. Axiomatic set theory developed as a way of generating the sets that we need for mathematical work while avoiding those that lead to such paradoxes; the standard axiom system in use today is Zermelo-Fraenkel set theory, ZF for short, or ZFC when the axiom of choice is included.

How does this relate to absolute complement, as defined by Alice? Suppose that we admit absolute complements. Then, given the empty set $\varnothing$, there is a set

$-\varnothing = \{x: x \notin \varnothing\}$. Since $\varnothing$ has no elements, $-\varnothing$ has everything as an element: it is an 'absolutely universal' set U with $x \in U$ for all x whatsoever. Now one of the principles of Zermelo-Fraenkel set theory is that given any set A and any property expressible in purely set-theoretic terms, there is a set consisting of exactly those items that are elements of A and have the property. This is called the *axiom of separation*. So, choosing A to be the absolutely universal set U and taking the property of 'being a set that is not an element of itself', we get a set whose elements are just those sets $x \in U$ such that $x \notin x$. By the definition of U, this is just Russell's set $S = \{x: x \text{ is a set and } x \notin x\}$, and we can apply the same argument as before to get a contradiction.

In summary, any set theory that, like ZF, admits the principle of separation must reject the absolutely universal set and so, if it also admits the empty set, must reject absolute complementation.

To be sure, there are some versions of set theory that, unlike ZF, *do* admit the universal set and absolute complementation. The best known is called NF (short for 'New Foundations') devised by the logician W.V.O. Quine. To avoid contradiction, they give up the principle of separation, which in turn leads to losing many important theorems, especially about infinite sets; there are also many important results that can be retained but with much more complex formulations and proofs. For this reason, such systems are not in general favoured by working mathematicians or computer scientists.

Our presentation of the 'nuts and bolts' of set theory follows the ideas of ZFC, but there is no need for the reader to study its axiomatization to understand what is going on. Moreover, the lack of a universal set and absence of an operation of absolute complementation are not troublesome for our work. 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, using it as your 'local universe' for relative complementation.

If, however, your curiosity has been aroused, and you would like to read more on absolute complementation, the universal set, ZF and NF, you can follow up with references mentioned at the end of this chapter.

1.5 Generalised Union and Meet

It is time to go a little beyond the cosy world of 'flat' set theory and look at some sets whose elements are themselves sets. We begin with the operations of *generalised union and meet*.

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 $A_1 \cup A_2$ 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, i.e. $(A_1 \cup A_2) \cup A_3 = A_1 \cup (A_2 \cup A_3)$. 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 as $A_1 \cup A_2 \cup A_3$.

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, … 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 $\cup\{A_i\colon i \in \mathbf{N^+}\}$ or more compactly as $\cup\{A_i\}_{i\in\mathbf{N+}}$ or as $\cup_{i\in\mathbf{N+}}\{A_i\}$. The sign is the same as that for binary union; often it is written rather larger than the binary operation but we write it in the same font except when a larger size makes easier reading.

Quite generally, if we have a collection $\{A_i\colon i \in I\}$ 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 meet 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, classes, families

Alice One moment! Why did you refer to $\{A_i\colon i \in I\}$ as a *collection* while its elements A_i, and its union $\cup_{i\in I}\{A_i\}$ and meet $\cap_{i\in I}\{A_i\}$, are *sets*? Shouldn't you say that $\{A_i\colon i \in I\}$ is a *set* of 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 with 'set of sets of sets'; 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 you should be a little careful. Sometimes the term 'family' is used rather loosely to refer to a set of sets. But strictly speaking, it means something a little different, a certain kind of function. So better not to use that term until it is explained in Chap. 3.

Exercise 1.5 (1)
From the definitions of general union and meet, 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 meet 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 meet)
(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
These are simple 'unpack, rearrange, repack' verifications. For each of them we show the inclusion LHS $\subseteq$ RHS; for the converse inclusion RHS $\subseteq$ LHS, you should check that the same arguments can be run backwards (in other words, that the steps all correspond to iffs).

(a) 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. For the converse, check that the same argument works backwards.
(b) Suppose $x \in$ LHS. Then either $x \in A$ or $x \in \cap_{i\ \in I}\{B_i\}$. By the definition of general meet, the latter implies that $x \in B_i$ for all $i \in I$. So, in both cases, $x \in A \cup B_i$ for all $i \in I$, and thus again by the definition of general meet, $x \in$ RHS as desired. For the converse, check that the same argument works backwards.
(c) Suppose $x \in$ LHS. Then $x \notin \cup_{i\in I}\{A_i\}$ so $x \notin A_i$ for all $i \in I$, so $x \in -A_i$ for all $i \in I$, so $x \in$ RHS.
(d) Suppose $x \in$ LHS. Then $x \notin \cap_{i\in I}\{A_i\}$ so $x \notin A_i$ for some $i \in I$, so $x \in -A_i$ for some $i \in I$, so $x \in$ RHS.

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, $\mathcal{P}(A) = \{B\colon B \subseteq A\}$. 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)
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 with warning
The elements of $\mathcal{P}(A)$ are as follows, beginning with the smallest and working our way up: $\varnothing$, {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 $\varnothing$, 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, the curly brackets may seem like pedantic points of punctuation, but if they are missed then you can easily get into a dreadful mess. *End of solution*.

All 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)
Draw Hasse diagrams for the power sets of each of $\varnothing$, {1}, {1,2}, {1,2,3,4} and count how many elements does each of these power sets has.

Solution Outline
They have 1, 2, 4 and 16 elements respectively. *End of solution*.

There is a pattern here. Quite generally, if A is a finite set with n elements, its power set $\mathcal{P}(A)$ 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 as a function of, say, the number of input items. It can happen that this measure increases in proportion to 2^n, i.e. is of the

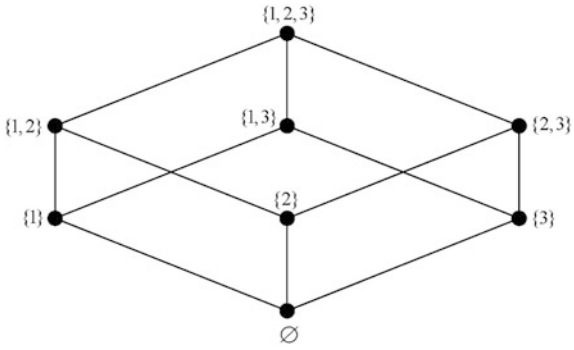

Fig. 1.4 Hasse diagram for $\mathcal{P}$(A) when A = {1,2,3}

form $k \cdot 2^n$ for some fixed k. This is known as *exponential growth* and it is to be avoided whenever possible, as it quickly leads to unfeasibly expensive calculations. For example, suppose that such a process is dealing with an input of 10 items. Now $2^{10} = 1024$, 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, as well as *iff* (alias *biconditional*) ones. It is time to fulfill 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 of the two to grasp; the logic box for 'iff' gives its truth-table.

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

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

The table for 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, namely the 'disastrous combination' that α is true and β is false.

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

From the table one can see that $\alpha \rightarrow \beta$ is always true except in the situation described in the second row. Comparing the two tables, it is easy to check 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 use *if…then…* in subtle ways that are more complex than any truth-table. But in mathematics, one uses the conditional in the simplest possible manner, which is that given by the truth-table. Moreover, it turns out that this way of understanding the conditional underlies all the other ones, in the sense that it is an important ingredient in their analysis.

Once again, we should say a bit more than the Hatter. Here is an example that may not entirely convince you, but it should make the truth-table rather 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*, is 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 corresponds to the third row (α false, β true); the last corresponds to the fourth row (both components false)—while in each of these three cases, as we noted, the conditional is true. The following exercise, due to Dov Gabbay, also helps 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 purchase something and report back on what happened. Agent 1 buys a computer and gets a free printer. Agent 2 buys a computer but is not offered a free printer. Agents 3 and 4 buy wi-fi modems, and while agent 3 does not get a free printer, in a burst of generosity agent 4 does. Which of these configurations would justify charging the manager with false advertising?

Solution
Evidently, only the second. The first fulfills the promise, while in the third and fourth the condition of the promise is not satisfied. *End of solution*.

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.

$\mathbf{N} = \mathbf{N}^+ \cup \{0\}$, i.e. the set consisting of zero and all the positive integers. This is usually 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*.
$\mathbf{Q} = \{p/q: p, q \in \mathbf{Z} \text{ and } q \neq 0\}$, 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).

1.7 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 $A + B \subseteq A \cup B$. Give a simple counter-example to the converse.
(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 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, meet, complement.
(h) We have seen that each of meet, 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.

Solution

(a) Suppose first that $x \in A + B$. Then $x \in (A \backslash B) \cup (B \backslash A)$ so either $x \in (A \backslash B)$ or $(B \backslash A)$. In the former case, x is an element of exactly one of A, B, and the same holds in the latter case. For the converse, run the same argument backwards.

(b) Take the diagram for $A \cup B$ in Fig. 1.3 and de-shade the cell in the middle.

(c) Suppose $x \in$ LHS. Then $x \in (A \backslash B) \cup (B \backslash A)$ so either $x \in (A \backslash B)$ or $(B \backslash A)$. In the former case, $x \in A$ so $x \in A \cup B$, while in the latter case, $x \in B$ so $x \in A \cup B$. For a counter-example to the converse, put $A = \{1,2\}$, $B = \{2,3\}$. Then, $2 \in A \cup B$ but $2 \notin A + B$.

(d) Suppose $x \in A \cap B$. Then $x \in A$ and $x \in B$, so $x \notin A \backslash B$ and $x \notin B \backslash A$ so $x \notin (A \backslash B) \cup (B \backslash A) = A + B$.

(e) Suppose first that $x \in$ LHS. Then $x \in (A \backslash B) \cup (B \backslash A)$ so either $x \in (A \backslash B)$ or $(B \backslash A)$. In both cases $x \in A \cup B$, and in both cases $x \notin A \cap B$, so $x \in$ RHS. For the converse, suppose $x \in$ RHS. Then $x \in A \cup B$ and $x \notin A \cap B$. From the former, either $x \in A$ or $x \in B$. If $x \in A$ then since $x \notin A \cap B$ we have $x \in$ LHS; similarly when $x \in B$.

(f) (i) Yes. We need to show that $A + B = B + A$. But LHS $= (A \backslash B) \cup (B \backslash A) = (B \backslash A) \cup (A \backslash B) =$ RHS and we are done.

(g) (ii) Yes. We need to show that $(A + B) + C = A + (B + C)$. For one direction, suppose $x \in$ LHS. Then $x \in (A + B) \backslash C$ or $x \in C \backslash (A + B)$. In the first case, $x \in A + B$ and $x \notin C$. Since $x \in A + B$ we have either $x \in A$ while $x \notin B$ or $x \in B$ while $x \notin A$. In the first of these two sub-cases we have $x \in A$ and $x \notin B + C$ so $x \in$ RHS. In the second of the two sub-cases we have $x \notin A$ and $x \in B + C$ so again $x \in$ RHS as desired. The converse can be verified in the same spirit.

(e) (iii) Yes. We need to show that $A \cap (B + C) = (A \cap B) + (A \cap C)$. For one direction, suppose $x \in$ LHS. Then $x \in A$, $x \in B + C$ so x is in exactly one of B, C, so x is in exactly one of $A \cap B$, $A \cap C$, so $x \in$ RHS. For the converse, re-run the argument in the reverse direction.

(f) (iv) No! We need to find a counter-example to $A + (B \cap C) = (A + B) \cap (A + C)$. Put $A = B = \{1\}$ and $C = \varnothing$. Then $1 \in$ LHS but $1 \notin A + B$ so $1 \notin$ RHS.

(g) $-(A + B) = (A \cap B) \cup (-A \cap -B)$.

(h) Exclusive disjunction. The table is like that for inclusive disjunction given in Sect. 1.4.2 except that in the top row we have 0 for $\alpha \vee \beta$.

Alice Box: Running in the reverse direction

Alice Hold on! Before we do any more exercises, I have a question.

Hatter Fire away!

Alice In several of the exercises so far, the proposed solution said that the converse can be verified by running the same argument in the

reverse direction. But, how can I know *in advance* that this can be done without hitting a snag?

Hatter With experience one can anticipate this quite well, but you can't know *for sure* in advance; you need to do the reverse run and check that there are no hidden obstacles. And sometimes there are obstacles. For example, you can easily verify the inclusion $(A + B) \cap (A + C) \subseteq A + (B \cap C)$ but, as we have just seen in (f) (iv), the converse fails.

Alice So, strictly speaking, such answers are incomplete, with pointers for completion?

Hatter Exactly.

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) $\#(A \cup B) \leq \#(A) + \#(B)$, but sometimes $\#(A \cup B) < \#(A) + \#(B)$.
(b) $\#(A \cup B) = \#(A) + \#(B) - \#(A \cap B)$.
(c) When A, B are disjoint then $\#(A \cup B) = \#(A) + \#(B)$.
(d) $\#(A \cap B) \leq min(\#(A), \#(B))$, but sometimes $\#(A \cap B) < min(\#(A), \#(B))$. Here, $min(m, n)$ is whichever is the lesser of the integers m, n.
(e) Formulate a necessary and sufficient condition for the equality $\#(A \cap B) = min(\#(A), \#(B))$ to hold.

Solution

(a) If you count the elements of A, then count the elements of B, you will certainly have counted all the elements of $A \cup B$. But if A, B have any common elements, you will have counted them twice, making $\#(A) + \#(B) > \#(A \cup B)$. A simple example: put $A = \{1\} = B$. Then $\#(A \cup B) = 1 < 2 = \#(A) + \#(B)$.
(b) We can calculate $\#(A \cup B)$ by counting $\#(A)$, adding $\#(B)$, and taking away the number of items that we counted twice, that is, subtracting $\#(A \cap B)$.
(c) When A, B are disjoint then $\#(A \cap B) = 0$, so (b) gives us the desired equality.
(d) Since $A \cap B \subseteq A$ and $A \cap B \subseteq B$ we have $\#(A \cap B) \leq \#(A)$ and $\#(A \cap B) \leq \#(B)$ so $\#(A \cap B) \leq min(\#(A), \#(B))$. For a simple example of the two sides not being equal, put $A = \{1\}$, $B = \{2\}$. Then $\#(A \cap B) = 0 < 1 = min(\#(A), \#(B))$.
(e) Either $A \subseteq B$ or $B \subseteq A$.

Exercise 1 (3) Generalized union and meet

(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 $\cap\{A_i\}_{i\in\mathbf{N}}$ empty.
(c) Why can't (b) be done for some *finite* collection $\{A_i\}_{i\in I}$ of non-empty sets?

Solution

(a) (i) For the left-to-right implication, suppose $\cup\{A_i\}_{i\in I} \subseteq B$. Let $i \in I$ and take any $x \in A_i$. We need to show $x \in B$. But since $x \in A_i$ we have $x \in \cup\{A_i\}_{i\in I}$ and so by our supposition $x \in B$. For the converse, suppose $A_i \subseteq B$ for every $i \in I$. Let $x \in \cup\{A_i\}_{i\in I}$. We need to show $x \in B$. But since $x \in \cup\{A_i\}_{i\in I}$ we have $x \in A_i$ for some $i \in I$ so, by the supposition, $x \in B$.
(ii) For the left-to-right implication, suppose $B \subseteq \cap\{A_i\}_{i\in I}$. Let $x \in B$ and choose any $i \in I$. We need to show $x \in A_i$. But since $x \in B$ the supposition tells us that $x \in \cap\{A_i\}_{i\in I}$ and so $x \in A_i$. For the converse, suppose $B \subseteq A_i$ for every $i \in I$. Let $x \in B$. We need to show $x \in \cap\{A_i\}_{i\in I}$. But since $x \in B$ the supposition tells us that $x \in A_i$ for every $i \in I$ so $x \in \cap\{A_i\}_{i\in I}$ as desired.
(b) Put $A_0 = \mathbf{N}$, $A_1 = A_0\backslash\{0\}$ and quite generally $A_{n+1} = A_n\backslash\{n\}$. Then the collection $\{A_i\}_{i\in\mathbf{N}}$ has the desired properties. In particular, $\cap\{A_i\}_{i\in\mathbf{N}}$ is empty because each $n \notin A_{n+1}$.
(c) If the collection $\{A_i\}_{i\in\mathbf{N}}$ is finite and each $A_i \supset A_{i+1}$ then $\cap\{A_i\}_{i\in I} = A_k$ which by supposition is non-empty.

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? $\mathcal{P}(A \cup B) = \mathcal{P}(A) \cup \mathcal{P}(B)$. Proof or counterexample.

Solution

(a) Suppose $A \subseteq B$ and $X \in \mathcal{P}(A)$. Then $X \subseteq A$ so $X \subseteq B$ so $X \in \mathcal{P}(B)$.
(b) True. $X \in$ LHS iff $X \subseteq A \cap B$, iff $X \subseteq A$ and $X \subseteq B$, iff $X \in \mathcal{P}(A)$ and $X \in \mathcal{P}(B)$, iff $X \in$ RHS.
(c) False. Put $A = \{1\}$, $B = \{2\}$. Then $\{1,2\} \in$ LHS but $\{1,2\} \notin$ RHS.

1.8 Selected Reading

For a very gentle introductions to sets, which nevertheless takes the reader up to an outline of the Zermelo-Fraenkel axioms for set theory, see E. D. Bloch *Proofs and Fundamentals: A First Course in Abstract Mathematics*, Springer 2011 (second edition), chapter 3.

For more on the axiomatizations ZF and NF, as well as on the universal set and absolute complementation, consult also the articles on those topics, as well as on Russell's paradox, on the websites of *The Stanford Encyclopedia of Philosophy* and *Wikipedia*.

A classic and beautiful exposition but presupposing a certain mathematical sophistication, is Paul R. Halmos *Naive Set Theory*, Springer, 2001 (new edition), chapters 1–5, 9. It is short on exercises and readers should, instead, verify systematically all claims made in the text.

The present material is also covered, with lots of exercises, in Seymour Lipschutz *Set Theory and Related Topics*, McGraw Hill Schaum's Outline Series, 1998, chapters 1 and 2 as well as chapters 5.1–5.3.

Comparing Things: Relations

2

Chapter Outline

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 more or more sets, from which one may define the notion of a relation. 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 task of classifying 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 ordering, we look at several kinds of *reflexive order*, and then at their *strict parts*.

2.1 Ordered Tuples, Cartesian Products, 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; a set being a subset of another; a student exchanging one book for another with a friend.

D. Makinson, *Sets, Logic and Maths for Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-42218-9_2

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.

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, <7,9>. It is distinct from the ordered pair (9,7) although they have the same elements. In other words, (7,9) ≠ (9,7) because the elements are considered in a different order.

Abstracting from this example, the *criterion for identity* 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 identity criterion for 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 $x_2 = y_1$).

Solution
From right to left: suppose either ($x_1 = y_1$ and $x_2 = y_2$) or ($x_1 = y_2$ and $x_2 = y_1$). We consider the case that $x_1 = y_1$ and $x_2 = y_2$; the other is similar. By the condition of the case, $\{x_1, x_2\} \subseteq \{y_1, y_2\}$ and also $\{y_1, y_2\} \subseteq \{x_1, x_2\}$ so $\{x_1,x_2\} = \{y_1,y_2\}$. From left to right: suppose $\{x_1,x_2\} = \{y_1,y_2\}$ and $x_1 \neq y_1$ or $x_2 \neq y_2$; we need to show $x_1 = y_2$ and $x_2 = y_1$. Case 1: suppose $x_1 \neq y_1$. Then since $\{x_1, x_2\} \subseteq \{y_1, y_2\}$ we have $x_1 = y_2$, and since $\{y_1, y_2\} \subseteq \{x_1, x_2\}$ we also have $y_1 = x_2$ and this case is done. Case 2: suppose $x_2 \neq y_2$. We argue in similar fashion.

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! 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 relations and order in general.

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: $A \times B = \{(a,b)\colon a \in A \text{ and } b \in B\}$.

In English, this says that $A \times B$ 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) Let A = {John, Mary} and B = {1,2,3}, $C = \varnothing$. What are A B, $B \times A$, $A \times C$, $C \times A$, A^2, B^2? How many elements in each?

Solution

$$A \times B = \{(\text{John}, 1), (\text{John}, 2), (\text{John}, 3), (\text{Mary}, 1), (\text{Mary}, 2), (\text{Mary}, 3)\}$$
$$B \times A = \{(1, \text{John}), (2, \text{John}), (3, \text{John}), (1, \text{Mary}), (2, \text{Mary}), (3, \text{Mary})\}$$
$$A \times C = \varnothing = C \times A$$
$$A^2 = \{(\text{John}, \text{John}), (\text{John}, \text{Mary}), (\text{Mary}, \text{John}), (\text{Mary}, \text{Mary})\}$$
$$B^2 = \{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)\}$$
$$\#(A \times B) = 6 = \#(B \times A), \#(A \times C) = 0 = \#(C \times A), \#(A^2) = 4, \#(B^2) = 9.$$

Comment: Note that we also have, for example: $A \times B$ = {(John,1), (Mary,1), (John,2), (Mary,2), (John,3), (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), where the order is vital. *End of solution*.

From Exercise 2.1.2 (1) you may already have guessed a general counting principle for 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. So, there are $n + n + \ldots + n$ (m times), i.e. $m \cdot n$ 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)

(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

(a) Suppose $A \subseteq A'$ and $B \subseteq B'$ and let $(a,b) \in A \times B$. Then $a \in A$ and $b \in B$ so $a \in A'$ and $b \in B'$ so $(a, b) \in A' \times B'$.
(b) Suppose $A \neq \varnothing$ and $B \neq \varnothing$. We need to show that $A \times B = B \times A$ iff $A = B$. 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 its *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. *End of solution*.

The solution to part (b) of Exercise 2.1.2(2) 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 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 is not always the most transparent way. Sometimes it is better to suppose *not*-β and head for *not*-α. Why is proof by contraposition legitimate? 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? Sometimes, one supposition gives us more to 'grab hold of' than the other. In our example, the supposition $A \neq B$ tells us that there is an a with $a \in A$, $a \notin B$ (or conversely); that is not yet very concrete, but we can then consider such an a, and start reasoning about it.

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: $m = n$ 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.

Solution

(a) The left-to-right conditional is immediate, since m divides m. Right-to-left, if each of m, n divides the other then $n = a \cdot m$ and $m = b \cdot n$ for some natural numbers a, b so $n = a \cdot (b \cdot n) = (a \cdot b) \cdot n$ so $a \cdot b = 1$, so $a = 1 = b$ so $m = n$. As often, one direction is easier than the other, and it is usually good manners towards the reader to give the easier one first.

(b) Suppose that x, y are both rational. Then $x = a/b$ and $y = c/d$ for integers a, b, c, d with $b \neq 0 \neq d$. Then $x \cdot y = (a \cdot c)/(b \cdot d)$ where numerator and denominator are integers with the latter non-zero, so $x \cdot y$ is rational.

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 $A \times B$. In other words, it is any set of ordered pairs (a,b) such $a \in A$ and $b \in B$. Thus, it is fully determined by the ordered pairs that it has as elements, and it makes no difference to the its identity how those 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 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 generalised 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, that is what will be meant.

Exercise 2.1.3 (1) Let $A = \{\text{John, Mary}\}$ and $B = \{1,2,3\}$ as 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 smallest relation from A to B? What is the largest?
(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, if you read the brackets carefully, it is an ordered pair of sets, not a set of ordered pairs.
(b) $\varnothing$ is the smallest, in the sense that $\varnothing \subseteq R \subseteq A \times B$ for every relation R from A to B. $A \times B$ is the largest relation A to B; it is often called *the total relation* over $A \times B$ or, when $B = A$, the total relation over A.
(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 $\mathscr{P}(A \times B)$, and by a counting principle in the chapter on sets, $\#(\mathscr{P}(A \times B)) = 2^{\#(A \times B)} = 2^6 = 64$. *End of solution*.

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 caution 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 least possible 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 Box: Codomain You may occasionally see the term 'codomain' contrasting with 'domain'. But care is again needed, as the term is sometimes used broadly for 'target', at other times 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 (also written $=_A$) over a set A is defined by putting $I = \{(a,a): a \in A\}$. Identify $dom(I_A)$ and $range(I_A)$.
(c) Identify $dom(A \times B)$, $range(A \times B)$.

Solution

(a) $dom(R) = \{1,3,13\}$, $range(R) = \{7,3,11\}$, $dom(S) = \{1,3,13,15\}$, $range(S) = \{1,11,12\}$.
(b) $dom(I_A) = A = range(I_A)$.
(c) $dom(A \times B) = A$, $range(A \times B) = B$. *End of solution.*

Since relations are sets (of ordered pairs or tuples) we can apply to them all the concepts that we have 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 non-empty disjoint relations over a small finite set A.

Solution

(a) The argument is the same as for showing that the empty set is a subset of every set, in an exercise of Chap. 1. In detail: we need to show that for all (x,y), if $(x,y) \in \varnothing$ then $(x,y) \in R$. In other words: there is no (x,y) with $(x,y) \in \varnothing$ but $(x,y) \notin R$. But by the definition of $\varnothing$, there is no (x,y) with $(x,y) \in \varnothing$ and we are done.
(b) $dom(\varnothing) = \varnothing = range(\varnothing)$.
(c) R is disjoint from S iff there is no pair (x,y) with both $(x,y) \in R$ and S; that is, iff $R \cap S = \varnothing$. Example: Put $A = \{1,2\}$, $R = (1,1)$, $S = \{2,2\}$.

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 especially arrow diagrams are helpful.

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 it by a table?

We need two rows and three columns, for the elements of A and of B respectively. Each cell in this table is uniquely identified by its coordinate (a,b) where by convention 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 the Table 2.1.

Tabular representations are particularly useful when dealing with relations in 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.

Solution We give the solution for (i), with Table 2.2; its diagonal consists solely of 0 s; if we fold along the diagonal, every entry on or below the fold is a 0 while every entry strictly above it is a 1.

Table 2.1 Table for a relation

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

Table 2.2 Table for the relation < over {1,2,3,4}

<	1	2	3	4
1	0	1	1	1
2	0	0	1	1
3	0	0	0	1
4	0	0	0	0

2.2.2 Digraphs

Another way of representing binary relations, less amenable to software implementation but friendlier to humans, is by 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 Table 2.1, the digraph comes out as in Fig. 2.1.

The Hasse diagram between the subsets of {1, 2, 3} that we displayed 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 $B \subset X \subset C$. To streamline, the Hasse diagram uses plain lines rather than arrows but with direction indicated by the convention that subsets are below and supersets above.

Given a diagram for a relation, we can use it to read off its reflexive and transitive closure. For reflexive closure, think of adding a little loop from each node to itself; for transitive closure, think of following the arrows; for the reflexive-and-transitive closure, do both. Evidently, writing in all these arrows creates a lot of clutter, so usually the additions are not actually made, but are signaled as understood by a note attached to the diagram. For example, 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-point path). To appreciate the economy of the Hasse representation, try filling in the arrows thus understood in Fig. 1.4 of Chap. 1.

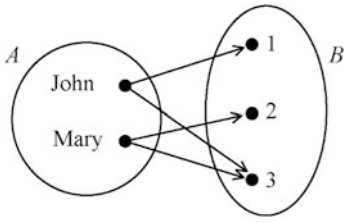

Fig. 2.1 Diagram for a relation

Warning Box: Diagrams are not proofs 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 suggest or illustrate a proof, conveying its basic idea and making it much easier to follow. However, they have their limitations: (1) most important, *a diagram cannot itself constitute a proof of a general claim*; (2) an overcrowded diagram defeats its main purpose of creating intuitive transparency; (3) a poorly labelled diagram can be ambiguous and even meaningless in the absence of standard conventions for reading it.

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 how we specify the source A and target B as well as on R itself.

As well as such 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 Box: Converse versus complement Do not confuse converse with complement! 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 loved by*.

Exercise 2.3.1 (1)

(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.

Solution

(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. *End of solution*.

This exercise illustrates some interesting points. In (a) we see 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. In (b), 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 needed 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, as do their exact meanings also differ in some cases.

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.

Solution

(a) In a digraph: reverse the directions of all arrows. In a table: rotate the table anti-clockwise by 90°.
(b) $a \in dom(R^{-1})$ iff there is a b with $(a,b) \in R^{-1}$, iff there is a b with $(b,a) \in R$, iff $a \in range(R)$.

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 maintaining 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 (say, 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 must associate exactly one identity number with each name and conversely, but S may legitimately associate more than one telephone extension to some names of high-ranking employees, and more than one name to some telephone extensions of low-ranking ones. We will not bother with these special properties at present but will return to them in the chapter on functions.

The manager may decide to merge these two databases into one, thus economising 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 generalised 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 $(b_1, \ldots, b_n, c_1, \ldots, c_p) \in S$.

Warning Box: Join Another case of terminological difference between cultures: while database theorists (and this book) use the word 'join' for this this operation, set theorists and algebraists sometimes use that word as a synonym for the 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, say, 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 $(a_1, \ldots, a_m, b_1, \ldots, b_n) \in R$. We could equally well project onto any other non-empty subset of the $m + n$ places.

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, say, $A_1, \ldots, A_m$ 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 $(c_1, \ldots, c_m, b_1, \ldots, b_n) \in R$ such that each $c_i \in C_i$. Again, we could equally select from any other subset of the $m + n$ places, even the empty subset although this will just be the identity operation. 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 the arity and join increases it. By combining them one can carry out a lot of manipulations.

Exercise 2.3.2

(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 a subset A' of A.
(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 $A \times C$.

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) 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) $\in R$. The projection of R to B is the set of all $b \in B$ such that there is an $a \in A$ with (a,b) $\in R$.
(c) The set of all pairs (a,b) $\in R$ with $a \in A'$.
(d) (i) R = {(2,12), (3,12), (3,15)}; S = {(12,16), (15,11), (15,21), (17,11), (17,21)}. (ii) *join*(R,S) = {(2,12,16), (3,12,16), (3,15,11), (3,15,21)}. (iii) {2, 3}, C itself, {(2,16), (3,16), (3,11), (3,21)}.

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 closely related operation that children learn at a very young age—as soon as they can recognise 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 in its most commonly used case; the composition of two relations, each two-place.

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 two, one less than that of the join, which is three; in fact, $S \circ R$ is the projection of *join*(R, S) onto the first and last of its three places.

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) $\in P$, i.e. 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, it seems to be written in reverse. In the example, wouldn't it be better to write the composition 'father of a parent of' as $F \circ P$? The letters would then occur in the same order as they do in

the English. That is also the order of the predicates when we say, in the definition, '$(a,x) \in F$ and $(x, c) \in P$'—which, with x in the middle, is easier to digest than '$(x, c) \in P$ and $(a,x) \in F$'.

Hatter Indeed, that would correspond more directly with the way that ordinary language handles composition, and early authors, like de Morgan, 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 later, 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 was felt best to use the same notation in the two contexts, and the one in use for functions won out.

If, like Alice, you keep on mixing up the orders, you should write the definition down before each exercise involving composition, and read $(a,c) \in S \circ R$ as "a is an R of something which is an S of c".

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 $(x,c) \in S$ is written by set-theorists as $\exists x((a,x) \in R \wedge (x,c) \in S)$ or more briefly, by logicians, as $\exists x(Rax \wedge Sxc)$.

The existential quantifier $\exists$ has its own logic, which we will describe in Chap. 9 along with the logic of its companion, the universal quantifier $\forall$. In the meantime, we simply adopt the notation $\exists x(\ldots)$ as a convenient shorthand for longer phrase 'there is an x such that … holds', and likewise use $\forall x(\ldots)$ to abbreviate 'for every x, … holds'.

Exercise 2.3.3 (1)
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. Identify the following relative products, in each case giving a reason for the identification: (a) $P \circ P$, (b) $M \circ F$, (c) $S \circ P$ (d) $B \circ B$. Be careful about order. In some cases there will be a handy word in English for the relation, but in others it will have to be described in a more roundabout (but still precise) way.

Solution and Comments

(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: Two

common errors here. (1) Getting the order wrong. (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: 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 'parent of a sister of'; we can do no better than use the entire phrase.

(d) $B \circ B$ = 'brother of a brother of'. Comments: 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'. *End of solution*.

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, i.e. whether you regard the wives of the brothers of your parents as your aunts. If you *do* include them under the term 'aunt', then $P \circ S$ is a proper subrelation of the aunt relation. If you *don't* include them, then it is the whole relation, i.e. $P \circ S$ = 'aunt'.

Exercise 2.3.3 (2)

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

Solution

These are sufficiently simple to do by a chain of *iffs*. But if you are uncomfortable about that, do them by showing LHS $\subseteq$ RHS and RHS $\subseteq$ LHS separately.

(a) $(a,c) \in$ LHS iff there is an x with $(a,x) \in T$ and $(x,c) \in S \cup R$, iff (for that x) either $(a,x) \in T$ and $(x,c) \in S$ or $(a,x) \in T$ and $(x,c) \in R$, iff either $(a,c) \in (S \circ T)$ or $(a,c) \in (R \circ T)$, iff $(a,c) \in$ RHS.
(b) $(a,c) \in$ LHS iff $(c,a) \in S \circ R$, iff there is an x with $(c,x) \in R$ and $(x,a) \in S$, iff (for that x) we have $(x,c) \in R^{-1}$ and $(a,x) \in S^{-1}$, iff $(a,c) \in$ RHS.

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)

(a) Let A = {John, Mary, Peter} and let $B = \{1, 2, 3\}$. Let R be the relation {(John,1), (John,3), (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

(a) R(John) = {1}, R(Mary) = {2,3}, R(Peter) = $\varnothing$.
(b) Go back to Fig. 2.1, add a point for Peter in the left ellipse, draw a kidney around the points 1,3 on the right with the label R(John), a small ellipse around the points 2, 3 with the label R(Mary), and a small circle on the right containing none of the marked points, labelled R(Peter).
(c) $\leq(9) = \{n \in \mathbf{N}: 9 \leq n\}$. *End of solution.*

It is often useful to 'lift' the notion 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, $R(X) = \{b \in B: \exists x \in X, (x,b) \in R\}$.

Warning Box: Ambiguous notation for image 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 of itself. For example, when $A = \{1, 2, \{1,2\}\}$, $B = \{1, 2\}$, $R = \{(1,1), (2,2), (\{1,2\}, 1)\}$, what is $R(X)$ for $X = \{1,2\}$? When X is understood as an element of A, then $R(X) = \{1\}$, but when X is understood as a subset of A, then $R(X) = \{1,2\}$.

For this reason, some authors instead adopt some other way of writing the lifted image of a subset, for example $R''(X)$. However, we will rarely be dealing with sets of the kind that can generate ambiguity and so, unless signalled explicitly, it will be safe to stick with the simpler notation.

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 over the natural numbers 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.

Solution

(a) $R(\varnothing) = \varnothing$, {1}, $R(\{\text{John}\}) = \{1\}$, $R(\{\text{Mary}\}) = \{2,3\}$, $R(\{\text{Peter}\}) = \varnothing$, $R(\{\text{John, Mary}\}) = \{1,2,3\}$, $R(\{\text{John, Peter}\}) = \{1\}$, $R(\{\text{Mary, Peter}\}) = \{2,3\}$, $R(A) = B$.
(b) (i) $\leq(\{3,12\}) = \{n \in \mathbf{N} : 3 \leq n\}$, (ii) $\leq(\{0\}) = \mathbf{N}$, (iii) $\leq(\varnothing) = \varnothing$, (iv) $\leq(\{n \in \mathbf{N} : n \text{ is even}\}) = \{n \in \mathbf{N} : 2 \leq n\}$, (v) $\leq(\{n \in \mathbf{N} : n \text{ is odd}\}) = \mathbf{N}^+$, (vi) $\leq(\mathbf{N}) = \mathbf{N}$.
(c) (i) $P(\{3, 12\}) = \{2, 11\}$, (ii) $P(\{0\}) = \varnothing$, (iii) $P(\varnothing) = \varnothing$, (iv) $P(\{n \in \mathbf{N}: n \text{ is even}\}) = \{n \in \mathbf{N}: n \text{ is odd}\}$, (v) $P(\{n \in \mathbf{N}: n \text{ is odd}\}) = \{n \in \mathbf{N}: n \text{ is even}\}$, (vi) $P(\mathbf{N}) = \mathbf{N}$.
(d) $\leq^{-1} = \geq$, $(a,x) \in P^{-1}$ iff $a = x + 1$.

Exercise 2.3.4 (3)

(a) Identify a set A with just one element, such that the element 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?

Solution

(a) $\{\varnothing\}$.
(b) $\{\varnothing, \{\varnothing\}\}$.
(c) Put $A_n = \{a_1, \ldots, a_n\}$, where $a_1 = \varnothing$ and $a_{i+1} = \{a_1, \ldots, a_i\}$ for each $i < n$. *End of solution.*

The definition of the sets A_n in Exercise 2.3.4(3) is an instance of a general procedure known as *definition by cumulative recursion*, which will be studied in Chap. 4.

At first glance, the specific sets A_n generated by that definition are rather perverse: who could be interested in such weird objects? In turns out, however, that they play an important role in the philosophy of mathematics, specifically in the reconstruction of numbers as special sets. In the von Neumann account of natural numbers, 0 is defined as $\varnothing$, 1 as $\{\varnothing\}$, 2 as $\{\varnothing, \{\varnothing\}\}$ and generally n as A_n above, and it is shown how to continue this construction beyond the natural numbers to define what are known as the transfinite ordinals. But all that is beyond the limits of this book.

2.4 Reflexivity and Transitivity

In the next three sections we will look at special properties that relations may possess. Some of these properties are desirable for relations expressing similarity between items, in particular, when we want to consider them as *equivalent* in a certain respect or *classify* them. Some are needed when we want to express an *order* among items. Two of the properties, reflexivity and transitivity, are central 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)

(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

(a) For example, identity and the relation that holds when a divides b without remainder are both reflexive. The complements of these two relations are irreflexive. The relation that holds between two numbers iff their sum is a square, is neither reflexive nor irreflexive.
(b) Yes, in the limiting case of the empty relation over the empty set, and in no other case.
(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. *End of solution*.

A subtle point needs attention. Let R be a relation over a set A, i.e. $R \subseteq A^2$ and let $B \subset A$. It can happen that R is reflexive over B, but not over the entire set A. This is because we may have $(b,b) \in R$ for all $a \in B$ but $(a,a) \notin R$ for some $a \in A - B$. Because of this 'domain dependence', when we say that a relation is reflexive or irreflexive, we should always be clear for what set we are making the claim. 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 taken in any particular context.

Exercise 2.4.1 (2)

(a) Give a small finite example of the domain dependence mentioned above.
(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.

Solution

(a) The smallest possible example is the empty relation, which is reflexive over the set $B = \varnothing$, but fails reflexivity over any larger set A. For a slightly less degenerate example, put $A = \{1,2\}$, $B = \{1\}$, $R = \{(1,1), (1,2)\}$.
(b) Suppose that $(a,a) \in R$ for all $a \in A$. Then if $B \subseteq A$, we have $(a,a) \in R$ for all $a \in B$.
(c) Suppose that for all $a \in A$, $(a,a) \in R$ and $(a,a) \in S$. Then $(a,a) \in R \cap S$ for all $a \in A$.
(d) Clearly, $(a,a) \in R$ iff $(a,a) \in R^{-1}$.

2.4.2 Transitivity

Another important property for relations is transitivity. We say that R is *transitive* over a set A iff for all a, b, $c \in A$, if $(a,b) \in R$ 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. As a rule of thumb, when a relation is defined using an existential quantifier rather than a universal one, there can be a serious risk that it fails transitivity.

Another relation over the natural numbers failing transitivity is that of being the immediate predecessor of: 1 is the immediate predecessor of 2, which is the 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 above relation of sharing a common prime factor, likewise 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.

Just as for reflexivity, the transitivity of $R \subseteq A$ can depend on the choice of A. Let R be a relation over a set A, i.e. $R \subseteq A^2$ and let $B \subset A$. It can happen that R is

transitive over B, but not over the entire set A. In particular, every relation is transitive over the empty set. Admittedly, that is very much a limiting-case counter-example, but the following is little less so: put $A = \{1,2\}$, $R = \{(1,1), (1,2)\}$, $B = \{1\}$. Then R is transitive over B but not over A.

Evidently, the notion is not the same that that which you may have encountered in grammar, where a verb is said to be transitive if, in its primary use, it admits a direct object. Thus, the verb 'loves' is transitive in the grammatical sense since we may speak of one person loving another, but not in the sense defined here, since one person may love another, who loves a third, without the first loving the last.

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: (i) sister of, (ii) parent of, (iii) ancestor of. In each case, give reasons.
(c) Show that the inverse of a transitive relation is transitive.
(d) Is the intersection of two transitive relations always transitive? Give a verification or a counter-example.
(e) Is the union of two transitive relations always transitive? Give a verification or a counter-example.
(f) Is the composition $S \circ R$ of transitive relations R, $S \subseteq A^2$ always transitive? Give a verification or a counter-example.

Solution

(a) Yes. This happens when (and only when) there are no a, b, c such that both $(a, b) \in R$ and $(b,c) \in R$.
(b) (i) Sister of: neither transitive nor intransitive, at least as the term is ordinarily understood. We usually have that when a is a sister of b and b is a sister of c then a is a sister of c, so the relation is not intransitive. But when a and b are sisters of each other we do not say that a is a sister of herself, as transitivity would require. Note that the definition of transitivity requires something for *all* a, b, c, even when they are not all mutually distinct. (ii) Parent of: one is tempted to say that is intransitive, and that would be the case if incest never occurred; in fact, it is neither transitive nor intransitive. (iii) Ancestor of: transitive.
(c) Suppose that R is transitive. Let $(a,b), (b,c) \in R^{-1}$; we need to show that $(a,c) \in R^{-1}$. But since $(a,b), (b,c) \in R^{-1}$ we have $(b,a), (c,b) \in R$ so, by the transitivity of R, $(c,a) \in R$ and thus $(a,c) \in R^{-1}$.
(d) Yes. Let R, S be transitive relations, we want to show that $R \cap S$ is transitive. Suppose both $(a,b) \in R \cap S$ and $(b,c) \in R \cap S$; we need to show that $(a,c) \in R \cap S$. But from the suppositions we have $(a,b) \in R$ and $(b,c) \in R$ so $(a,c) \in R$; likewise for S; so that $(a,c) \in R \cap S$ as desired.
(e) No. Consider $A = \{1,2,3\}$, $R = \{(1,2), (2,3), (1,3)\}$, $S = \{(2,3), (3,1), (2,1)\}$. These relations are transitive. But $(2,3) \in R \cup S$ and $(3,1) \in R \cup S$ whilst $(2,1)$

$\notin R \cup S$. A smaller but more degenerate example: $A = \{1,2\}$, $R = \{(1,2)\}$, $S = \{(2,1)\}$.

(f) No. Consider $A = \{1,2,3,4,5\}$, $R = \{(1,2), (3,4)\}$, $S = \{(2,3), (4,5)\}$. Then R, $S \subseteq A \times A$ and are both vacuously transitive. But $(1,3), (3,5) \in S \circ R$ whilst $(1,5) \notin S \circ R$.

You may wonder how one can cook up examples like those in (e) and (f) above. One strategy is to focus on the property to be falsified and look for the smallest and simplest configuration in which it can be made to fail.

2.5 Equivalence Relations and Partitions

We now look at properties that are of interest when we want to express notions of similarity and, more strongly, equivalence between items.

2.5.1 Symmetry

We say that R is *symmetric* over a set A iff for all $a, b \in A$, iff $(a,b) \in R$ 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 $m \leq n$ but sometimes $m \not\leqslant n$ (the former when $m = n$, the latter when $m \neq n$), 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.

Just as for reflexivity and transitivity, the symmetry of $R \subseteq A$ can depend on the choice of A.

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: (i) sister of, (ii) parent of, (iii) 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 over A always symmetric? Give a verification or a counter-example.
(f) Give a small finite example to illustrate how the symmetry of $R \subseteq A$ can depend on the choice of A.

Solutions

(a) If we fold the table along the diagonal, entries coincide.
(b) Yes. The one and only such relation is the empty relation.
(c) (i) Sister of: Neither. In particular, a may be a sister of b while b is a brother of a. On the other hand, the relation 'being a sibling of' is symmetric (although not quite transitive). (ii) Asymmetric. (iii) Asymmetric.
(d) For converse: suppose R is symmetric and $(a,b) \in R^{-1}$. Then $(b,a) \in R$ so by the first supposition $(a,b) \in R$ so $(b,a) \in R^{-1}$ as needed. For intersection: suppose R, S are symmetric and $(a,b) \in R \cap S$. Then $(a,b) \in R$ so by symmetry of R we have $(b,a) \in R$; likewise for S; so $(b,a) \in R \cap S$ as needed. For union: suppose R, S are symmetric and $(a,b) \in R \cup S$. Then $(a,b) \in R$ or $(a,b) \in S$. In the former case $(b,a) \in R$ while in the latter case $(b,a) \in S$, so in each case $(b,a) \in R \cup S$ as needed.
The essential reason why union satisfies symmetry but not transitivity is that the definition of symmetry has only one clause in its antecedent whilst that of transitivity has two; when given that (a,b), $(b,c) \in R \cup S$ we may have $(a,b) \in R$ while $(b,c) \in S$, blocking application of the suppositions that each of those two relations is transitive.
(e) No. Consider $A = \{1,2,3\}$, $R = \{(1,2), (2,1)\}$, $S = \{(2,3), (3,2))\}$. Then R, S are both symmetric. But $(1,3) \in S \circ R$ whilst $(3,1) \notin S \circ R$. Try to articulate the essential reason why we have this negative result.
(f) We can recycle the example already used for transitivity. Put $A = \{1,2\}$, $R = \{(1,1), (1,2)\}$, $B = \{1\}$. Then R is symmetric over B but not over A.

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—reflexivity, symmetry, and transitivity—it is called an *equivalence relation*.

Equivalence relations are often written as 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* $(a,b) \in \approx$ or by *prefixing* as in $\approx(a,b)$.

Exercise 2.5.2

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

Recall that we write the identity relation over A as I_A or as $=_A$. When it is clear what set A is intended we often drop the subscript, as in the following solution.

(a) (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) Let $\approx$ be any equivalence relation $\approx$ over a set A; we need to show that $= \subseteq \approx$, that is, whenever $a = b$ then $a \approx b$. But this is given by the reflexivity of $\approx$. (iii) The largest equivalence relation over A is A^2.
(b) 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 for both domains.
(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.
(d) Clusters of points where, within each cluster, all points are linked to each other as well as to themselves.

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 and set theorists 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 identity, only a 'tight' equivalence relation, perhaps also satisfying certain 'congruence' conditions. But more on this later.

There is a very important logical difference between identity and other equivalence relations. When $a = b$, the items a, b are elements of exactly the same sets and have exactly the same mathematical 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*. In high-school algebra you may have learned this under the rather elliptic dictum that 'substitution of equals gives equals'.

2.5.3 Partitions

Your correspondence is in a mess—one big heap of paper or undifferentiated computer file. You need to classify it, putting items into mutually exclusive categories that together exhaust all the items; 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.

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

- Recall from Sect. 1.3.2 that the collection is said to be *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 $B_i \cap B_j = \varnothing$. In logical notation: $\forall i, j \in I\big(B_i \neq B_j \rightarrow B_i \cap B_j = \varnothing\big)$.

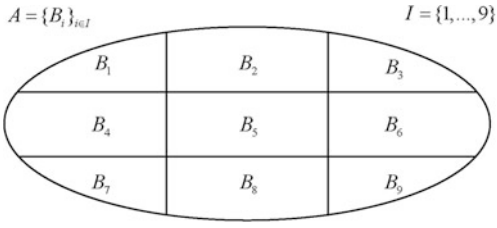

Fig. 2.2 Diagram for a partition

- 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: $\forall a \in A \, \exists i \in I (a \in B_i)$.

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 as in Fig. 2.2.

Exercise 2.5.3

(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) $\{\{1,2\}, \{3,4,5\}\}$ (v) $\{\{1,4\}, \{3,2\}\}$.

(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 some 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 elements ($n \geq 1$), how many cells in its finest and least fine partitions?

Solution

(a) Only (v) 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; in (iv) the set $\{3,4,5\}$ has an element that is not in A.

(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 elements ($n \geq 1$) then its finest partition has n singleton cells, while its least fine partition has only one cell, A itself.

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. Moreover, the round trip, in either direction, takes you back to your staring point. The precise articulation and verification of this is rather abstract, and hence challenging, but you should be able to follow it.

We begin with the partition-to-equivalence-relation 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, i.e. iff $\exists i \in I$ with $a,b \in B_i$. We need to show that it is an equivalence relation.

For reflexivity 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, $a \in B_i$.

For symmetry: 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 $(b,a) \in R$.

For transitivity: Suppose (a,b), $(b,c) \in R$; 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 $B_i \cap B_j = \emptyset$. Since $b \in B_i \cap B_j$ the latter is not the case, so $B_i = B_j$. Hence both $a,c \in B_i$, which gives us $(a,c) \in R$ as desired.

Now for the equivalence-relation-to-partition direction, Let $\approx$ be an equivalence relation over a non-empty set A. For each $a \in A$ we consider its image under the relation $\approx$, as defined earlier in this chapter: $\approx(a) = \{x \in A: a \approx x\}$. 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 in what follows. Clearly, each set $|a|$ is a subset of A. We take the collection of these sets $|a|$ for all $a \in A$ and verify that it has the properties needed to be a partition.

For non-emptiness: each set $|a|$ is non-empty, because $\approx$ is reflexive over A.

For exhaustion: The sets $|a|$ together exhaust A, that is, $\cup\{|a|\}_{a\in A} = A$, again because $\approx$ is reflexive over A.

For pairwise disjointedness: We argue contrapositively. Let a, $a' \in A$ and suppose $|a| \cap |a'| \neq \emptyset$; 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 $x \in |a|$, so $a \approx x$; we need to show that $x \in |a'|$, i.e. that $a' \approx x$. By the initial supposition $|a| \cap |a'| \neq \emptyset$, there is a y with $y \in |a|$ 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.

When a partition is constructed from the corresponding equivalence relation, its cells are often called *equivalence classes*.

Alice Box: Similar verifications

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

Hatter Strictly speaking, you don't! But when two cases under consideration have the same structure, and the arguments are 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 verification so as to focus on essentials.

Alice Isn't that dangerous?

Hatter Indeed, there can be small differences that turn out to be unexpectedly important, so care is needed. One should check all cases before omitting any of them and present things in such a way that the reader can always reconstruct missing parts in a routine way.

There is still part of the partition-equivalence-class correspondence that we haven't checked: that the round trip, in either direction, takes us back to our starting point. The verifications are straightforward from the definitions but students without much experience in mathematics tend to become lost in the notation. So, we omit them.

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. (i) Write this partition by enumeration as a collection of sets, then (ii) describe the corresponding equivalence relation in words.

(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?

Solution

(a) (i) {{2,4,6,8,10}, {1,3,5,7,9}}. (ii) $\approx$ is the set of all pairs (a,b) such that either a, b are both even or a, b are both odd.

(b) {2,4,8,16}, {3,9}, {5}, {6,12}, {7}, {10}, {11}, {13}, {14}, {15}}.

(c) Finest partition: the identity relation over A; least fine partition: the total relation over A, i.e. A^2.

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 also 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 further 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, used for classifying rather than ordering. 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, transitive, and also antisymmetric over A 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 a 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 (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 $\mathbf{Z}$, i.e. 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 $\mathbf{Q}^+$, i.e. 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.

Solution

(a) For reflexivity, it suffices to note that $a = 1a$. For transitivity, if $b = ka$ and $c = jb$ then $c = (jk)a$. For antisymmetry, suppose $b = ka$ and $a = jb$. Then $b = kjb$ so, since $b \neq 0$, $kj = 1$ so $k = 1$ so $b = a$.
(b) No. For a counter-example to antisymmetry, put $a = 1$, $b = -1$, $k = -1 = j$.
(c) No. Put $a = 1$, $b = ½$, $k = ½$, $j = 2$ to get a counter-example to antisymmetry,
(d) Yes.
(e) Recall from Exercise 2.5.3 (b) that one partition of a set is at least as fine as another iff every cell of the former is a subset of some cell of the latter. Since the subset relation is a partial ordering, this relation is too. *End of solution*.

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 $b \in A$. These notions may at first sight appear to be the same, but by working carefully through the following exercises you will begin to appreciate the subtle and important differences between them. It may help to visualize what is going on by making rough heuristic diagrams as you proceed.

Exercise 2.6.1 (2)
Let A be a set and $\leq$ a partial ordering of A. Show the following.

(a) Whenever a is a least element of A then it is a minimal element of A.
(b) The converse can fail (give a simple example).
(c) A can have zero, one, or more than one minimal elements (give an example of each).
(d) A can have at most one least element, i.e. if a least element exists then it is unique.
(e) If A is finite and non-empty then it must have at least one minimal element.

Solution

(a) Suppose a is a least element of A. Suppose for *reductio* that there is a $b \in A$ with $b < a$, that is, $b \leq a$ and $b \neq a$. By the leastness of a we have $a \leq b$ so, by antisymmetry, $a = b$ giving us a contradiction.
(b) The simplest example puts $A = \{1,2\}$ and $\leq_A = I_A$, so we have two minimal elements and no least element.
(c) The above example has two minimal elements. For exactly one minimal element, consider the set of natural numbers under its usual partial ordering, where 0 is the unique minimal element. For no minimal elements, consider the set of all integers, positive, negative and zero. Of course, small finite examples are also readily available.
(d) Suppose a, b are least elements of A. Then both $a \leq b$ and $b \leq a$ so, by antisymmetry, $a = b$.

(e) Suppose for *reductio* that A is non-empty but does not have a minimal element. By non-emptiness it has an element a_0, which is not a least element of A, so there is an $a_1 \in A$ with $a_1 < a_0$. But a_1 is not a least element of A, so there is an $a_2 \in A$ with $a_2 < a_1$, and so on. Thus, there is an infinite sequence $a_0 > a_1 > a_2 \ldots$ of mutually distinct elements of A (using transitivity here), so A is infinite. Remark: our reasoning here is a little rough. It is really a proof by induction, and its structure will become clearer in Chap. 4. *End of solution*.

Dual notions of maximal and greatest elements are defined just as one would expect. Let A be a set and $\leq$ a partial ordering of A. An element $a \in A$ is said to be a *maximal* element of A (under $\leq$) iff there is no $b \in A$ with $a < b$. On the other hand, an element $a \in A$ is said to be a *greatest* element of A (under $\leq$) iff $b \leq a$ for every $b \in A$.

Also as one would expect, maximal and greatest elements behave dually to minimal and least ones. In detail, they have the following properties as counterparts to those listed in Exercise 2.6.1 (2). (a) Whenever a is a greatest element of A then it is a maximal element of A but (b) the converse can fail. (c) A can have zero, one, or more than one maximal elements, but (d) it can have at most one greatest element. (e) If A is finite and non-empty then it must have at least one maximal element. The verifications also routinely dualize.

With these notions in hand, we can define the concepts of lower and upper bounds. As some readers will know already, they play a strategic role in the theory of continuity for functions on the real numbers; they are also very useful in the more general context of any partial order of an arbitrary set.

Let A be a set, $\leq$ a partial ordering of A, $X \subseteq A$. An element $a \in A$ is called a *lower bound* of X (in A, under $\leq$) iff $a \leq x$ for all $x \in X$. When $a \in A$ is a greatest element of the set of all lower bounds of X in A, i.e. it is greatest among all the lower bounds of X (in A, under $\leq$) then it is said to be a *greatest lower bound* of X (in A, under $\leq$). 'Greatest lower bound' is commonly abbreviated to *glb* and is sometimes called *infimum*. When the identity of the set A and its partial order $\leq$ are clearly identified and fixed, it is common to omit mention of one or both, leaving them as understood.

Exercise 2.6.1 (3)

(a) Let A be a set, $\leq$ a partial ordering of A, $X \subseteq A$. Show that if X has a greatest lower bound (in A, under $\leq$) then it is unique.

(b) Let A be a set, $\leq$ a partial ordering of A. Let $X \subseteq A$ and suppose that X has a glb in A. Show that for all $a \in A$, a is a lower bound of X iff $a \leq glb(X)$.

Solution

(a) Suppose a, b are both glbs of X in A under $\leq$. Then they are both lower bounds of X in A and so by the definition of glb we have both $a \leq b$ and $b \leq a$ so, by antisymmetry of $\leq$, $a = b$ as desired. The application of antisymmetry here is vital; there is no decent theory of glbs for orderings that lack that property.

(b) From left to right is given explicitly by the definition of glb. For right to left, $\text{glb}(X) \leq x$ for all all $x \in X$ so if $a \leq glb(X)$ then by transitivity we have $a \leq x$ for all $x \in X$ so that a is a lower bound of X. Transitivity is thus also vital for a good theory of glbs.

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 ordering* or *chain*.

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 $\mathscr{P}(A)$ is not complete. To show this, take any two distinct $a, b \in A$, and consider the singletons $\{a\},\{b\}$; they are both elements of $\mathscr{P}(A)$, but neither $\{a\} \subseteq \{b\}$ nor $\{b\} \subseteq \{a\}$ because $a \neq b$.

Exercise 2.6.2 (1)

(a) *Give two more linear orderings of* $\mathbf{N}$.
(b) Which of the following are linear orderings over an arbitrary set of people? (i) is at least as old as, is (ii) identical to or a descendent of.
(c) Give an example of an infinite linearly ordered set without any minimal element.

Solution

(a) There are plenty, but here are two: (i) the relation $\geq$, i.e. 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.
(c) The simplest example is the set of all negative integers (or the set of all integers positive, negative or zero).

Warning Box: total ordering vs total relation A linear ordering over a set A is sometimes called a 'total ordering'. It should not be confused with the unique 'total relation' over A in the sense of exercises 2.1.3 (1) and 2.5.4 (c), which is A^2 itself. Evidently, for sets A with more than one element, A^2 is not a linear order, since it is not antisymmetric.

Exercise 2.6.2 (2)
Show that for linear orderings: (i) A can have at most one minimal element, and (ii) if a minimal element exists then it is the least element of A.

Solution
Let $\leq$ be a linear ordering of A. (i) Let a, b be minimal elements of A; we want to show that $a = b$. By minimality, $a \not< b$ and $b \not< a$ so either $a \not\leqslant b$ or $a = b$ and either $a \not\leqslant b$ or $b = a$. But completeness requires that either $a \leq b$ or $a \leq b$ so we have either $a = b$ or $b = a$, thus $a = b$. (ii) Let a be a minimal element of A and let $x \in A$; we need to check that $a \leq x$. By minimality $x \not< a$ so either $x = a$ or $x \not\leqslant a$. The former gives $a \leq x$ immediately, and the latter gives it by linearity.

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)

(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 $a < a$. Hence, 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 $\leq$ gives $a \leq c$; it remains to check that $a \neq c$. Suppose $a = c$; we get a contradiction. Since $b \leq c$ and $a = c$ we have $b \leq a$, so by the antisymmetry of $\leq$ using also $a \leq b$ 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. Twice, 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 newfangled 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* and this name is still used, sometimes abbreviated to *reductio* or the acronym RAA.

Reductio ad absurdum is a universal proof method, in the sense that it is always possible to use it. Some people like to use 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, which is often the case. We will have more to say about it in the chapters on logic.

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.

Solution
For transitivity, we begin in the same way. 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 $\leq$ gives $a \leq c$; it remains to check that $a \neq c$. At this point the reasoning diverges. Since $a \leq b$ and $a \neq b$ the antisymmetry of $\leq$ gives us $b \not\leq a$. Combining that with $b \leq c$ gives us $c \neq a$, combining that with $a \leq c$ finally tells us that $a < c$. Your preference is your privilege; the author's is for explicit *reductio* here. *End of solution*.

In the preceding chapter, Fig. 1.4, we observed that the inclusion relation between finite sets may be represented by a Hasse diagram. The links in that diagram, read from below to above, represent the relation of one subset B of A

being an *immediate proper subset* of another subset C of A, in the sense that $B \subset C$ and there is no subset X of A with $B \subset X \subset C$.

Evidently, we can generalize this notion to apply to any partial order $\leq$ over a set A: b is an *immediate predecessor of* c iff $b < c$ and there is no x with $b < x < c$. When A is finite, we can represent the partial ordering by a diagram for the immediate predecessor part of $<$. Once we have such a diagram for immediate predecessors we can read off the entire relation: $a \leq b$ iff there is an ascending path, with at least one node on it, from a to b; for $<$ the path must contain at least two distinct nodes. Evidently, this is a much more economical representation than drawing the digraph for the entire partial ordering. We loosely call it a Hasse-style diagram of the partial ordering itself.

Exercise 2.6.3 (3)

(a) Enumerate the pairs in the immediate predecessor part of the relation of being an exact divisor of, over the set A of positive integers from 2 to 13.
(b) Draw a Hasse-style diagram for the relation.

Solution

(a) The pairs are: (2,4), (2,6), (2,10), (3,6), (3,9), (4,8), (4,12), (5,10), (6,12).
(b) Your diagram should contain nodes for each integer from 2 to 13 inclusive, and upwards links corresponding to just the ordered pairs (a). It will be most elegant to organize by rows: place the primes 2, 3, 5, 7, 11, 13 in a row at the bottom, then 4, 6, 9 10 in the next row up, 8 the sole node in the following row, 12 the only one in the top row, and inset links from the ordered pairs in the list. Note that 7, 11, 13 have no links to or from them. *End of solution.*

We have seen in 2.6.3 (1)(b) 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 above, 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 $<$.

Solution

(a) Reflexivity: immediate since $a = b$. Antisymmetry: suppose $a \leq b$ and $b \leq a$. Then either $a < b$ or $a = b$, also either $b < a$ or $b = a$. So, either $a = b$ or both $a < b$ and $b < a$. The latter is ruled out by the asymmetry of $<$, so $a = b$. Transitivity: suppose $a \leq b$ and $b \leq c$; we need to show $a \leq c$. By the suppositions, either $a < b$ or $a = b$, and also either $b < c$ or $b = c$. Three cases arise. If $a = b$ then since $b < c$ we have $a < c$ so $a \leq c$ as desired. If $b = c$ then since $a < b$ we again have $a < c$ so $a \leq c$. If $a \neq b$ and $b \neq c$ then we have $a < b$ and $b < c$ so $a < c$ by the transitivity of $<$.

(b) One way of doing this is by first showing that $\leq'$ is a subrelation of $\leq$, then showing the converse; another way is by a series of *iffs*. In both presentations, each step to the right is justified by the definition of a concept on the left. We answer this exercise in the first way, the next exercise in the other way. To show that $\leq' \subseteq \leq$, suppose $a \leq' b$. Then either $a < b$ or $a = b$. In the case that $a < b$ we have $a \leq b$ and $a \neq b$, so in particular $a \leq b$. In the case that $a = b$ we again have $a \leq b$, and this direction is done. For the converse, suppose $a \not\leq' b$. Then both $a \not< b$ and $a \neq b$. Since $a \not< b$ we have either $a \not\leq b$ or $a = b$. So, since $a \neq b$ we have $a \not\leq b$ as desired.

(c) $a <' b$ iff $a \leq b$ and $a \neq b$, iff $a \neq b$ and ($a < b$ or $a = b$), iff $a < b$.

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. They are cousins, but not the same; try not to confuse one with the other.

2.7.1 Transitive Closure of a Relation

Suppose you are given a relation R over A that is not transitive, but which you want to 'make' transitive. Of course, you cannot literally 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).

Why bother with three definitions of the same thing? One reason is that they provide three quite different ways of looking at transitive closure and three tools for working with it. Sometimes one suggests a more transparent way of proving a property of R^*, sometimes another. The top-down definition often gives particularly elegant proofs, but the finite-path and bottom-up definitions, which are quite close to each other, are usually better for computations. The other reason, which we will explain in Chap. 4, is that the same three modes of presentation arise for any recursively defined concept; you need all three of them in your toolkit.

The *finite-path* definition of the transitive closure R^* of R 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 $1 \leq i < n$.

The '*bottom-up*' definition, begins by defining the relations R_0, R_1, … as follows. The base of the definition puts $R_0 = R$, while the recursion step of the definition puts $R_{n+1} = R_n \cup \{(a,c)$: $\exists x$ with $(a,x) \in R_n$ and $(x,c) \in R\}$. We then put R^* to be their union: $R^* = \cup \{R_n : n \in \mathbf{N}\}$. 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.

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

Exercise 2.7.1 (1)

(a) Identify the transitive closures of the following relations: (i) parent of, (ii) mother of, (iii) descendant of.
(b) Use the finite-path definition to show that the transitive closure of a symmetric relation is symmetric.
(c) Give a simple example to show that the transitive closure of an asymmetric relation need not be asymmetric.
(d) Use the bottom-up definition to show that the converse of the transitive closure of a relation equals the transitive closure of the converse of that relation.
(e) Use the top-down definition to show that R^* is the least transitive relation that includes R, where 'least' means' least under set inclusion.

Solution

(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).
(b) Let R be a symmetric relation. We want to show that R^* is also symmetric. Suppose $(a,b) \in R^*$; we need to show that $(b,a) \in R^*$. Since $(a,b) \in R^*$, 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 $1 \leq i < n$. Since R is symmetric, each $x_{i+1} R x_i$ in the reverse sequence $x_n, \ldots, x_1$, so $(b,a) \in R^*$.

(c) Put $A = \{1,2,3\}$ and $R = \{(1,2), (2,3), (3,1)\}$. Then both $(1,2), (2,1) \in R^*$, so R^* is not asymmetric (in fact, $R^* = A^2$).

(d) We need to show that $(R^*)^{-1} = (R^{-1})^*$. For LHS $\subseteq$ RHS, suppose $(a,b) \in$ LHS. Then $(b,a) \in R^* = \cup\{R_n : n \in \mathbf{N}\}$ so $(b,a) \in R_k$ for some $k \in \mathbf{N}$, so $(a,b) \in R_k^{-1} \subseteq \cup\{R_n^{-1} : n \in \mathbf{N}\} = (R^{-1})^*$. For te converse, run the argument backwards.

(e) It suffices to show that (i) R^* with $R^* \supseteq R$, (ii) R^* is transitive, and (iii) $R^* \subseteq S$ for every transitive relation S with $R \subseteq S$. For (i): The intersection of any collection of relations, each of which includes R, is clearly a relation and includes R. For (ii): In Exercise 2.4.2 (c) we checked that the intersection of any two transitive relations is transitive; essentially the same argument shows that the intersection of any collection of transitive relations is transitive. In detail, let $\{R_i\}_{i\in I}$ be any collection of transitive relations. Suppose (a,b), $(b,c) \in \cap\{R_i\}_{i\in I}$. Then for every $i \in I$, (a,b), $(b,c) \in R_i$, so for every $i \in I$, $(a,c) \in R_i$, so $(a,c) \in \cap\{R_i\}_{i\in I}$ as needed. For (iii): Let R' be any transitive relation over A with $R \subseteq R'$; we need to check that $R^* \subseteq R'$. Let $(a,b) \in R^*$; we need to check that $(a,b) \in R'$. Now $R^* = R^* = \cap\{S: R \subseteq S \subseteq A^2$ and S is transitive$\}$, so since R' is a transitive relation over A with $R \subseteq R'$, it is one of the sets S in the collection, so $R^* \subseteq R'$. *End of solution*.

As already mentioned, the definitions are equivalent. In principle, that could be shown now; but it is perhaps better to postpone the rather abstract verifications until we have seen more examples of the three ways of presenting a recursive definition and discussed them in general terms in Chap. 4.

2.7.2 Closure of a Set Under a Relation

We now look at a closely related construction, the closure of an arbitrary set under an arbitrary relation. It is quite general, and the construction of R^* out of R can be conceived as a special instance of it, but it is probably best to think of them separately.

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 y such that $(x,y) \in R$ for some $x \in X$; briefly, $R(X) = \{y: \exists x \in X, (x,y) \in R\}$.

Now, suppose we are given a relation R (not necessarily transitive) over a set A, and a subset $X \subseteq A$. We define the *closure* $R[X]$ *of* X *under* R to be the least subset of A that both includes Y and includes $R(Y)$ whenever it includes a set Y. Again, this is a top-down definition, with 'least' understood as the result of intersecting. It can also be expressed bottom-up, with basis setting $X_0 = X$, the recursion step putting $X_{n+1} = X_n \cup R(X_n)$ for each natural number n, finally gathering the results together by taking $R[X] = \cup\{X_n : n \in \mathbf{N}\}$.

The closure $R[X]$ of X under R is thus constructed by beginning with X itself, and at each stage adding in the elements of the image of whatever set has so far been constructed. The closure of a set under a relation thus differs from its image under that relation, in two important respects: the closure always includes the initial set, and we apply the relation not just once but over and over again.

When the relation R is understood, we sometimes write the closure $R[X]$ briefly as X^+, and say that R *generates* the closure from X. Once again, notation differs somewhat from author to author.

Exercise 2.7.2
In the set $\mathbf{N}$ of natural numbers, what is the closure $R[X]$ of the set $X = \{2,5\}$ under the following relations over $\mathbf{N}^2$: (a) $(m, n) \in R$ iff $n = m + 1$, (b) $(m, n) \in R$ iff $n = m\text{-}1$, (c) $(m, n) \in R$ iff $n = 2\ m$, (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) $\{n \in \mathbf{N}: n \geq 2\}$. if you wrote $\{1\}$ as your answer to (d) or gave $\{n \in \mathbf{N}: n > 2\}$ as your answer to (f), it is because you forgot that $B \subseteq R[B]$.

2.8 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 $A \times (B \cup C) = (A \times B) \cup (A \times C)$.

Solution

(a) $(a,x) \in$ LHS iff $a \in A$ and $x \in B \cap C$, iff $a \in A$, $x \in B$, $x \in C$, iff $(a,x) \in A \times B$ and $(a,x) \in A \times C$, iff $(a,x) \in$ RHS.
(b) $(a,x) \in$ LHS iff $a \in A$ and $x \in B \cup C$, iff $a \in A$ and either $x \in B$ or $x \in C$, iff either $(a,x) \in A \times B$ and $(a,x) \in A \times C$, iff $(a,x) \in$ RHS.

If you dislike such chains of *iffs*, feel free to show the two inclusions separately, first supposing $(a,x) \in$ LHS to get $(a,x) \in$ RHS, then conversely.

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 for $X = \varnothing$.
(d) Explain how to carry out composition $S \circ R$ by means of join and projection.

Solution

(a) $R \cap S = \{(1,7)\}$ so $dom(R \cap S) = \{1\}$, $range(R \cap S) = \{7\}$. $R \cup S = \{(1,1), (1,7), (3,3), (3,11), (13,11)\}$ so $dom(R \cup S) = \{1,3,13\}$, $range(R \cup S) = \{1,3,7,11\}$.
(b) $join(R,S) = \{(a,b,c)$ such that $(a,b) \in R$, and $(b,c) \in S\} = \{(3,3,11)\}$. On the other hand, $join(S,R) = \{(a,b,c)$: $(a,b) \in S$, and $(b,c) \in R\} = \{(1,1,7), (15,1,7)\}$. For composition, $S \circ R = \{(a,c)$: $(a,x) \in R$ and $(x,c) \in S$ for some $x\} = \{(3,11)\}$; $R \circ S = \{(a,c)$: $(a,x) \in S$ and $(x,c) \in R$ for some $x\} = \{(1,7), (15,7)\}$; $R \circ R = \{(a,c)$: $(a,x), (x,c) \in R$ for some $x\} = \varnothing$; $S \circ S = \{(a,c)$: $(a,x), (x,c) \in S$ for some $x\} = \{(1,7)\}$.
(c) When $X = \{1,3,11\}$ then $R(X) = \{7,3\}$, $S(X) = \{1,11\}$ and when $X = \varnothing$ then $R(X) = \varnothing = S(X)$.
(d) Given R, S, form $join(R,S)$, and then project into initial and terminal parts. More precisely, 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 $B_1 \times \ldots \times B_n \times C_1 \times \ldots \times C_p$. Form $join(R,S)$ and then project each of its tuples $(a_1, \ldots, a_m, b_1, \ldots, b_n, c_1, \ldots, c_p)$ to get $(a_1, \ldots, a_m, c_1, \ldots, c_p)$.

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) A relation R is said the be *acyclic* over a set A iff there are no $a_1, \ldots, a_n \in A (n \geq 2)$ 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 asymmetric (and so also irreflexive). (iii) Give an example of an acyclic relation that is not transitive.

Solution

(a) R is reflexive over A iff for all $a \in A$, $(a,a) \in R$, iff $I_A \subseteq R$.
(b) R is reflexive over A iff for all $a \in A$, $(a,a) \in R$, iff for all $a \in A$, $(a,a) \in R^{-1}$, iff R^{-1} reflexive over A.
(c) Left-to-right: suppose that R is transitive, and let $(a,b) \in R \circ R$. Then there is an x with (a,x), $(x,b) \in R$, so by transitivity $(a,b) \in R$. Right-to-left: suppose $R \circ R \subseteq R$, and let (a,x), $(x,b) \in R$. Then $(a,b) \in R \circ R$ so by the supposition $(a,b) \in R$.

(d) (i) Suppose that R is not acyclic but is transitive; we show that it is not irreflexive. By the first supposition, there are $a_1, \ldots, a_n\ (n \geq 2)$ such that each $(a_i, a_{i+1}) \in R$ and also $a_n = a_1$. By n-1 applications of transitivity, $(a_1, a_n) \in R$, that is, $(a_1, a_1) \in R$ and we are done.

(d) (ii) Suppose that R is not asymmetric; we show that it is not acyclic. By the supposition, there are a, b with (a,b), $(b,a) \in R$, giving us a cycle of length 2.

(d) (iii) For example, put $A = \{1,2,3\}$ and $R = \{(1,2), (2,3)\}$.

Exercise 2 (4) Symmetry, equivalence relations and partitions

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

(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.

(e) Give an example showing that the union of two equivalence relations need not be an equivalence relation.

(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.

Solution

(a) It suffices to show that (i) implies (ii), implies (iii), implies (i). First, suppose (i) and let $(a,b) \in R$. By (i), $(b,a) \in R$ so $(a,b) \in R^{-1}$ as needed for (ii). Next, suppose (ii); we need to show $R^{-1} \subseteq R$. Let $(a,b) \in R^{-1}$. Then $(b,a) \in R$ so by (ii) $(b,a) \in R^{-1}$ as needed. Finally, suppose (iii). Let $(a,b) \in R$. Then $(b,a) \in R^{-1} = R$ and the round trip is complete. Of course, any other circuit around the three conditions would do as well.

(b) Let R be a reflexive relation over a non-empty set A. Choose any $a \in A$. By reflexivity, $(a,a) \in R$ and, to repeat ourselves, $(a,a) \in R$. Intransitivity would then

(c) Suppose that R is reflexive and transitive over A, and put $(a,b) \in S$ iff both $(a, b) \in R$ and $(b,a) \in R$. Symmetry of S is immediate from the definition, and reflexivity of S is immediate from that of R. For transitivity, suppose (a,b), $(b, c) \in S$. Then (a,b), $(b,c) \in R$ so $(a,c) \in R$, also (b,a), $(c,b) \in R$ so $(c,a) \in R$. Putting these together, (a,c), $(c,a) \in R$ so $(a,c) \in S$ as desired.

(d) Let R, S be equivalence relations over A. Then they are both reflexive over A so clearly $R \cap S$ is reflexive over A. Likewise for symmetry, and we have already checked in Exercise 2.4.2 that the same holds for transitivity.

(e) In Exercise 2.4.2 we gave an example of two transitive relations whose union is not transitive. But that example does not work here, as its relations are neither reflexive nor symmetric, so let's try again. Let $A = \{1,2,3\}$ again and put $(a, b) \in R$ iff $a = b$ or $a,b \in \{1,2\}$ while $(a,b) \in S$ iff $a = b$ or $a,b \in \{2,3\}$. To visualize this, draw an arrow diagram with two colours for the arrows. Clearly each of R, S is an equivalence relation over A (and each corresponds to a partition of A into two cells, with one cell a singleton and the other a pair). But the union $R \cup S$ is not transitive since $(1,2), (2,3) \in R \cup S$ while $(1,3) \notin R \cup S$.

(f) Let $\{B_i\}_{i\in I}$ and $\{C_j\}_{j\in J}$ be partitions of A, and let R, S be their associated equivalence relations. Now, $\{B_i\}_{i\in I}$ is at least as fine as $\{C_j\}_{j\in J}$ iff (1) for every $i\in I$ there is a $j\in J$ with $B_i \subseteq C_j$. On the other hand, $R \subseteq S$ iff (2) for all $a, b \in A$, if there is an $i\in I$ with $a,b \in B_i$ then there is a $j\in J$ with $a,b \in C_j$. So, it suffices to show that (1) is equivalent to (2). Suppose (1) and let $a, b \in A$. Suppose $a, b \in B_i$. Then by (1) there is a $j\in J$ with $B_i \subseteq C_j$, so that $a, b \in C_j$ as needed. For the converse, suppose (2) and let $i\in I$. Now B_i is non-empty so there is a $b \in B_i$ so we have $b, b \in B_i$ so by (2) there is a $j\in J$ with $b \in C_j$ and we are done.

Exercise 2 (5) 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.

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

(d) (i) Define dually *upper bounds* and *least upper bounds* (aka lubs, suprema) dually to the lower bounds and greatest lower bounds in the text. (ii) State and verify the counterparts of properties (a), (b) of Exercise 2.6.1 (3).

Solution

(a) We need to show that S is reflexive, transitive and antisymmetric. Reflexivity is immediate from the definition, as is antisymmetry. There are many ways of presenting the verification of transitivity; here is one. Suppose $(a,b), (b,c) \in S$; we want to show $(a,c) \in S$. Divide the argument into two cases. *Case* 1. Suppose $a = b$ or $b = c$. Then the suppositions immediately give us $(a,c) \in S$. *Case* 2. Suppose $a \neq b$ and $b \neq c$. Then, by the definition of S, both (a,b), $(b,c) \in R$ and both (b,a), $(c, b) \notin R$. By the transitivity of R, the former pair gives us $(a,c) \in R$. It remains to show that either $a = c$ or $(c, a) \notin R$. We cannot show $a = c$ since $(b,c) \in R$, $(b, a) \notin R$ together imply $a \neq c$. But we can show $(c, a) \notin R$. For suppose $(c,a) \in R$; we get a contradiction. Since (c,a), $(a,b) \in$

R the transitivity of R tells us that that $(c,b) \in R$ so, since $(b,c) \in S$, we have by the definition of S that $b = c$ contradicting one of the conditions of Case 2.

(b) Let $\leq$ be a linear order, and $\leq^{-1}$, written briefly as $\geq$, its converse. We need to check that $\geq$ is reflexive, antisymmetric, transitive and complete. Each of these is immediate from the supposition of the corresponding property for $\leq$ and the definition of converse.

(c) We already know that $=_A$ is an equivalence relation. But it is also trivially antisymmetric, so it is a partial order. Conversely, let $\approx_A$ be any equivalence relation over A that is also an equivalence relation. By the reflexivity of $\approx_A$ we have $=_A \subseteq \approx_A$. For the converse, let $a \approx_A b$. By the symmetry of $\approx_A$ we have $b \approx_A a$ so, by the antisymmetry of $\approx_A$ we conclude $a =_A b$.

(d) (i) Let A be a set, $\leq$ a partial ordering of A, $X \subseteq A$. We call an element $a \in A$ an *upper bound* of X iff $x \leq a$ for all $x \in X$. When $a \in A$ is a least element of the set of all upper bounds of X in A, i.e. it is least among all the upper bounds of X, then it is said to be a *least upper bound* of X (in A, under $\leq$). (ii) Counterpart of property (a) of Exercise 2.6.1 (3): If X has a lub then it is unique. Counterpart of (b): If X has a lub in A, then for all $a \in A$, a is a upper bound of X iff $lub(X) \leq a$. The verifications routinely dualize those for Exercise 2.6.1 (3).

Exercise 2 (6) Strict orderings

(a) Give examples of relations that are (i) transitive but neither symmetric, asymmetric nor antisymmetric (ii) asymmetric but not transitive.

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

Solution

(a) (i) For example, $A = \{1,2,3\}$, $R = \{(1,2), (1,3)\} \cup \{2,3\}^2$. (ii) For example, $A = \{1,2,3\}$, $R = \{(1,2), (2,3)\}$.

(b) Note that we are asked to show this for relations in general, not just for partial orderings. Let A be a set, $R \subseteq A^2$, $R_0 = \{(a,b) \in R : a \neq b\}$. Suppose first that R is antisymmetric; show that R_0 is asymmetric. Let $(a,b) \in R_0$; show that $(b,a) \notin R_0$. Then $(a,b) \in R$ and $a \neq b$ so by antisymmetry of R we have $(b,a) \notin R_0$ as needed. For the converse, suppose that R_0 is asymmetric; show that R is antisymmetric. Let (a,b), $(b,a) \in R$; show $a = b$. But if $a \neq b$ then (a, b), $(b,a) \in R_0$ contradicting its asymmetry, and we are done.

Exercise 2 (7) Closure

(a) Does the intersection of the transitive closures of two relations always equal the transitive closure of their intersection?

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

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

Solution

(a) No. While it is easy to check that always $(R \cap S)^* \subseteq R^* \cap S^*$, the converse can fail. For an example, let A be any set with at least two elements, let R be the identity relation $=_A$ over A, and let S be its complement, i.e. the relation $\neq_A$ of non-identity over A. For the LHS, since $=_A$ is transitive, $(=_A)^* = (=_A)$ and it is easy to check that $(\neq_A)^* = A^2$, so LHS $= (=_A) \cap A^2 = (=_A)$. On the other hand, for the RHS we have $R \cap S = \varnothing$ and clearly $\varnothing^* = \varnothing$. Thus LHS $\neq$ RHS, as desired.

(b) Using the bottom up definition: already $X_1 = X_0 \cup R(X_0) = X \cup R(X)$ and clearly $X_1 \subseteq X_k$ for all $k \geq 1$, so $X_0 \cup R(X_0) \subseteq \cup \{X_n : n \in \mathbf{N}\} = R[X]$. If asked to justify the word 'clearly', we would carry out a simple inductive argument with induction step from $X_1 \subseteq X_n$ to $X_1 \subseteq X_{n+1}$ for every n. (c) Reflexivity of R gives us $X_n \subseteq R(X_n)$ for all $n \geq 0$, so $X_{n+1} = X_n \cup R(X_n) = R(X_n)$. Transitivity of R implies that $R(X_n) = R(R(X_n))$ for all $n \geq 0$. Thus $\cup \{X_n : n \in \mathbf{N}\} = R(X_0) = R(X)$. Spelling this out in detail is again an inductive argument. We will have more to say about inductive arguments in Chap. 4.

2.9 Selected Reading

The books of Bloch, Halmos and Lipschutz listed at the end of Chap. 1 also have good chapters on relations, with the same attractions as for sets. Bloch chapter 5 pays careful attention to underlying logical and heuristic matters; Halmos chapters 6 and 7 is a model of insightful exposition but short on exercises and with no solutions to the few that there are; Lipschutz has all the exercises one could possibly desire.

Out of the many other texts available, we mention Daniel J. Velleman *How to Prove It: A Structured Approach*, Cambridge University Press 2006 (second edition), Chap. 4, written in the same spirit as Bloch, and James L. Hein *Discrete Structures, Logic and Computability*, Jones and Bartlett 2002 (second edition), Chap. 4, written specifically for computer science students.

Associating One Item with Another: Functions

3

Chapter Outline

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 general concepts for relations, 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 principles of *equinumerosity*, *comparison* and the surprisingly versatile *pigeonhole* rule. Finally, we identify some very simple kinds of function that appear over and again (*identity*, *constant*, *projection*, *characteristic* and *choice* 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, or tuple of numbers, called the *arguments* of the function with another number, called the *value* of the function. The concept has for long been used in science and engineering 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 or gasoline consumption). In accord with such applications, the argument of the function was often called the 'independent variable', and the value of the function termed the 'dependent variable'. The idea was that each choice of a value for the independent variable causally determines a value for the dependent variable.

D. Makinson, *Sets, Logic and Maths for Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-42218-9_3

Over the last two hundred years, the concept of a function has evolved in the direction of greater abstraction and higher generality. The argument and value of the function need not be numbers—they can be items of any kind whatsoever. The function need 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 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, i.e. 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 that *for all* $a \in A$ there is *exactly one* $b \in B$ with $(a,b) \in R$. The italicised 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)
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; since $4 \neq 5$, 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. *End of solution*.

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 terminology and notation that will be familiar from school mathematics. 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 $f: A \to B$ to mean that f is a function from A into B. In the case that $B = A$, we have a function $f: A \to A$ from A into itself and we often 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 $a_i \in A_i$ 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 $n = 1$. The index is often called the *arity* of the function.

However, the additional generality in the notion of n-place functions may 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 in arithmetic, are usually thought of as two-place functions f and g with $f(x,y) = x + y$ and $g(x,y) = x \cdot y$, but they may also be treated 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, to keep notation simple, we formulate principles in terms of one-place functions.

Alice Box: Brackets and arity

Alice Typo! Too many brackets in $f((x,y))$ and $g((x,y))$.

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), rather than two items x,y.

Alice So, we can drop the inner brackets if we read addition or multiplication as two-place functions, but need them if we understand them as one-place? In that respect, using n-place functions can simplify our notation.

Hatter Indeed. And in computing, one must keep eyes open for brackets. But let's get back to more substantial things.

In mathematics there are many occasions like this, where a concept or construction may be seen in either of two ways, each with its costs and benefits in notation, intuitive transparency, formal elegance, rapidity of calculation. One perspective can be taken as more general or more 'basic' than another, but also vice versa. Alice saw this for the specific concepts of one-place and many-place functions, but it also comes up in many other contexts, notably for the very concepts of

relation and function themselves. In this book, we are taking relations as basic with functions as a special case, but it is also possible to do the reverse, as 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)

(a) Which of the functions in Exercise 3.1.1 is a partial function from A to B?
(b) State which of the following relations from $\mathbf{Z}$ into $\mathbf{Z}$ (see Chap. 1 Sect. 7 to recall its definition) are functions on $\mathbf{Z}$. For those that are not so, state whether they are nevertheless partial functions on $\mathbf{Z}$ into $\mathbf{Z}$? (i) $\{(a, |a|): a \in \mathbf{Z}\}$, (ii) $\{(|a|, a): a \in \mathbf{Z}\}$, (iii) $\{(a, a^2): a \in \mathbf{Z}\}$, (iv) $\{(a^2, a): a \in \mathbf{Z}\}$, (v) $\{(a, a + 1): a \in \mathbf{Z}\}$, (vi) $\{(a + 1, a): a \in \mathbf{Z}\}$, (vii) $\{(2a, a): a \in \mathbf{Z}\}$.

Solution

(a) All of the functions from A to B are, of course, partial functions from A to B. Of the other functions in the exercise, $\{(a,1), (b,2), (c,3)\}$ is the only one that is nevertheless 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 on $\mathbf{Z}$.
(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 on $\mathbf{Z}$: 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 on $\mathbf{Z}$ into $\mathbf{Z}$. *End of solution*.

When f is a function from A into B and we wish to note the identity of those sets, we write it as $f\colon A \to B$. By an abuse of English grammar, it is also common to write, say, "when $f\colon A \to B$ …" to mean "when f is a function from A into B …".

3.2 Operations on Functions

Since functions are relations, all the operations that we introduced in the theory of relations in Chap. 2 apply to them. However, in the context of functions, some of those operations become especially useful, 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 $dom(R) = \{a \in A: \exists b \in B\ ((a,b) \in R)\}$. When R is in fact a function f, this is expressed as $dom(f) = \{a \in A: \exists b \in B\ (f(a) = b)\}$. Thus, when f: $A \to B$, then $dom(f) = A$.

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

Exercise 3.2.1

(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?

Solution

(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, that is, the unique function f_X: $X \to B$ with $f_X(a) = f(a)$ for all $a \in X$.

Notations vary: often a vertical bar or similar sign is inserted to produce the sign $f|_X$. Because restriction is such a trivial operation and rarely at the centre of attention, it is also common to omit even the subscript X, leaving it to context to make it clear that the domain has been reduced from A to its subset X.

Again, let f: $A \to B$ and $X \subseteq A$. 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 $\{f(a): a \in X\}$. 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 nearly but 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 that notation. Just as for relations, however, 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, i.e. both $X \in A$ and $X \subseteq A$. Then the expression $f(X)$ can be read in two ways: 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, for example in the theory of transfinite ordinal numbers defined as sets, they sometimes avoid ambiguity by using an alternative notation like $f'(X)$ for the image.

Just as for relations, image should not be confused with *closure*. The definition is the same as for relations in Chap. 2 Sect. 2.7.2 but conveniently expressed in the notation for functions. Let $f: A \to A$ and $X \subseteq A$. Expressed in 'top down' form, 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. Equivalently, in 'bottom up' form, we define $X_0 = X$ and $X_{n+1} = X_n \cup f(X_n)$ for each natural number n, and put $f[X] = \cup\{X_n: n \in \mathbf{N}\}$.

Warning Box: Two details in the denition of closure
Note that the closure of X under f is defined only when f is a function on A into A (rather than into some unrelated set B). Note also that in the top-down definition, just as in the corresponding definition for relations in Chap. 2, the word 'includes' means 'is a superset of', not 'contains as an element'.

Exercise 3.2.2 (1)
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*, *predecessor* respectively. Now recall the set $\mathbf{N} = \{0, 1, 2, 3, \ldots\}$ from Sect. 1.7.

(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 **N** is the predecessor function, since $0 - 1 = -1 \notin \mathbf{N}$. It is thus a function on **N** into $\{-1\} \cup \mathbf{N}$.

(c) $Mod(A) = \{0,1,2\}$ but $mod[A] = A \cup \{0,1,2\} = A$; $square(A) = \{0,1,4\}$ but $square[A] = A \cup \{4, 16, 16^2, \ldots\}$; $successor(A) = \{0, 1, 2, 3\}$ but $successor[A] = \{-1, 0, 1, 2, \ldots\} = \{-1\} \cup \mathbf{N}$; $predecessor(A) = \{-2, -1, 0, 1\}$ but $predecessor[A] = \{\ldots, -2, -1, 0, 1, 2\} = \mathbf{Z}^- \cup \{0, 1, 2\}$. Remember that is always included in its closure $f[A]$ even though it is not always included in its image $f(A)$.

3.2.3 Composition

Perhaps the most common operation applied to functions is *composition*. Recall the notion as it appeared in Chap. 2, for relations in general. Suppose we are given two relations $R \subseteq AXB$ and $S \subseteq BXC$, so that the target of R is the source of S. 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 R is in fact a function f from A to B and S is a function g 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 $c = g(f(a))$.

In brief, 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 $g \circ f(a) = g(f(a))$.

Sometimes, to avoid any possible confusion, one writes $(g \circ f)(a)$ instead of $g \circ f(a)$ as above. To be sure, the brackets are normally unnecessary, since an expression like $g \circ (f(a))$ is devoid of meaning except when $f(a)$ is itself a function with suitable domain; but a little bit of redundant bracketing can sometimes facilitate comprehension.

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 $a \in A$, $(h \circ (g \circ f))(a) = h(g(f(a))) = ((h \circ g) \circ f)(a)$. Verifying this equality 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, $(h \circ g): B \to D$ 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 in general $f \circ g$ is not a function on B unless $range(g) \subseteq A$. Even when $g \circ f$ 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 answer 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:

$B \to C$ respectively, then $S \circ R$ becomes $g \circ f$, which corresponds nicely to the order on the right-hand side in the definitional equality $(g \circ f)(a) = g(f(a))$.

Exercise 3.2.3

(a) Give an example of familiar functions f, g: $\mathbf{N} \to \mathbf{N}$ such that $f \circ g \neq g \circ f$.
(b) 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: $A \to B$, g: $B' \to C$ and $range(f) \subseteq B'$.
(c) Why can (b) fail when $range(f) \not\subseteq B'$?

Solution

(a) Put $f(n) = n + 1$, $g(n) = 2n$. Then $f \circ g(1) = 3$, $g \circ f(1) = 4$.
(b) Let $A \to B$, g: $B' \to C$, $B \subseteq B'$. As before, define $g \circ f$ to be the set of all ordered pairs (a,c) such that there is some x with both $x = f(a)$ and $c = g(x)$. Clearly, the assumptions imply that $g \circ f$: $A \to C$.
(c) If $range(f) \not\subseteq B'$ then there is an $a \in A$ with $f(a) \notin B'$ so that there is no c with $c = g(f(a))$, so that $g \circ f$ can, at best, be understood only as a *partial* function from A into C. In that situation, one says simply that $g \circ f$ is not well-defined.

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

(a) Let $A = \{1,2,3\}$ and $B = \{a,b,c,d\}$. Let $f = \{(1,a), (2,a), (3,b)\}$ and $g = \{(1,a), (2,b), (3,c)\}$. Explain why neither of the relations f^{-1}, g^{-1} is a function from B to A.
(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

(a) 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.
(b) (i) Put $f(n) = |n|$. (ii) Put $f(n)$ to be the least even number greater than n.

Alice Box: Converse or inverse?

Alice In Chap. 2 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. The term 'converse' grew up in the relations community, 'inverse' in the functions culture.

Alice In school, examples such as those in Exercise 3.2.4 (a) were described by saying 'the inverse does not exist'. Why don't you use the same terms?

Hatter We might have done so if we were interested in functions alone, with no concern for either relations or partial functions, but the term can be misleading when working in the broader context. One could still say that f^{-1}, g^{-1} in Exercise 3.2.4 (a) do not 'exist as functions', but do exist 'as a relation' and 'as a partial function' respectively. But it is clearer simply to say that they are not functions although one is a relation and the other is a partial function.

3.3 Injections, Surjections, Bijections

The notion of the inverse of 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 other 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 (a), 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)

(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 $\mathbf{Z}$, i.e. from $\mathbf{Z}$ to $\mathbf{Z}$, called *mod*, *square*, *successor*, *predecessor* respectively. Which of them are injective?

(b) What does injectivity mean in terms a digraph for the function?
(c) Show that the composition of two injective functions is injective.

Solution

(a) Over **Z**, *mod* is not injective since e.g. $|1| = |-1|$, likewise for *square* since e.g. $3^2 = (-3)^2$. 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 $a + 1 = a' + 1$.
(b) It means that no point in the target circle is hit by two arrows coming from different points in the source circle.
(c) Let $f: A \to B$, $g: B \to C$ be injective; we want to check that $g \circ f$ is injective. Suppose $a \neq a'$. Then $f(a) \neq f(a')$ so $g(f(a)) \neq g(f(a'))$, that is, $g \circ f(a) \neq g \circ f(a')$. *End of solution.*

Part (a) of Exercise 3.3.1 (1) brings out the importance of always keeping clearly in mind the domain of the function when checking whether it is injective. Take, for example, the function of squaring. As the exercise shows, when understood with domain **Z**, the function is not injective. But when it is 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 $a = b$. In connection with part (c) of the exercise, it is easy to give examples of how failure of injectivity in either of the two component functions can lead to failure of injectivity for the composition.

The injectivity of a function from A to B is not enough to guarantee that its inverse is a function from B to A. This is already illustrated by the function of squaring on **N** (rather than on **Z**): as just noted, it is injective, but its inverse is not a function on **N**, since there are elements of **N**, e.g. 5, that are not in its domain, not being the square of any natural number.

Nevertheless, injectivity gets us part of the way: it ensures that the inverse relation f^{-1} of a function $f: A \to B$ is indeed 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 check that for all $b \in B$ there is at most one $a \in A$ such that $(b,a) \in f^{-1}$, i.e. at most one $a \in A$ with $(a,b) \in f$. But this is exactly what is given by the injectivity of f. This argument can also be run backwards, with the result that we have the following equivalence: *a function* $f: A \to B$ *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?

Solution
Let $f: A \times B \to C$ be a function of two arguments. The natural way of defining injectivity for it, which fits in with the reduction of two-argument functions to one-argument one that we described in the text, is to require: for all a, a', b, b', if $f(a,$

$b) = f(a',b')$ then $a = a'$ and $b = b'$. However, we can also define weaker notions of left injectivity and right injectivity. For example, left injectivity: for all a, a', b, if $f(a,b) = f(a',b)$ then $a = a'$. This corresponds to injectivity of all the left projections of f.

Even when a function $f: A \to B$ is not injective, so that its inverse relation f^{-1} is not a function, there is still a closely related 'inverse' 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 fully as $f^{-1}: B \to \mathcal{P}(A)$, and is defined by putting $f^{-1}(b) = \{a \in A: f(a) = b\}$ 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.

3.3.2 Surjectivity

Let $f: A \to B$. 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)
Consider the function $f(a) = a^2$ over $\mathbf{N}$, $\mathbf{N}^+$, $\mathbf{Z}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{R}^+$ respectively and determine in each case whether it is surjective.

Solution
For $\mathbf{N}$, $\mathbf{N}^+$, $\mathbf{Z}$, $\mathbf{Q}$ the answer is negative, since e.g. 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 $\mathbf{R}$, the answer is negative, since for example -1 does not have a square root, but for $\mathbf{R}^+$ the answer is positive again: any $b \in \mathbf{R}^+$ there is an $a \in \mathbf{R}^+$ with $a^2 = b$.

Alice Box: What is f(x)?

Alice I'm a bit puzzled by your notation in Exercise 3.3.2 (1). 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 the statement of Exercise 3.3.2 (1), 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?

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. To let's just apologise on behalf of the community for this abuse of notation and carry on.

The Hatter's 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.

Exercise 3.3.2 (2)

(a) What does surjectivity mean in terms a digraph for the function?
(b) Show that the composition of two surjective functions is surjective.
(c) What would be the natural definition of surjectivity for a function of two arguments?

Solution

(a) It means that every point in the target circle is hit by at least one arrow coming from a point in the source circle.
(b) Let $f\colon A \to B$, $g\colon B \to C$ be surjective; we want to check that $g \circ f$ is surjective. Let $c \in C$. Then, since g is onto C, there is a $b \in B$ such that $c = g(b)$. Since f is onto B, there is an $a \in A$ such that $b = f(a)$. Thus $c = g(f(a)) = g \circ f(a)$ and we are done.
(c) Let $f\colon A \times B \to C$ be a function of two arguments. The natural way of defining surjectivity for it would be as follows: for all $c \in C$, there are $a \in A$, $b \in B$ with $c = f(a,b)$. Again, this fits in with the reduction of two-argument functions to one-argument ones that we described earlier in the text. *End of solution*.

Each of the terms 'onto' and 'surjective' is in common use, but the former is snappier since it makes it easier for us to say onto *what*: we can say simply 'onto B', whereas 'surjective with respect to B' is quite a mouthful. 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!

3.3.3 Bijective Functions

A function f: $A \to B$ 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: $A \to B$ is a bijection between A and B iff its inverse f^{-1} is a function from B onto A.*

Exercise 3.3.3 (1)
Prove the claim just made.

Solution
From left to right: suppose f is a bijection between A and B. Then f is both injective and onto B. Since f: $A \to B$ is injective, then as we saw in Sect. 3.3.1, f^{-1} is function on $range(f) \subseteq B$ onto A. Moreover, since f is onto B, then as we saw at the beginning of this section, $range(f) = B$. Putting these together, f^{-1} is a function from B onto A.

From right to left: suppose f^{-1} is a function from B onto A. Since f^{-1} is a function, f must be injective, and since f^{-1} has domain B, f must be onto B. Putting these together, f is a bijection between A and B.

Exercise 3.3.3 (2)

(a) What is a bijection in terms of a digraph?
(b) Show that the inverse of a bijection from A to B is a bijection from B to A.
(c) Show that any injective function on a finite set into itself is a bijection.

Solution

(a) There are arrows from the source circle to the target circle such that every point in the source circle has exactly one outgoing arrow and every point in the target circle has exactly one incoming arrow.
(b) Let f: $A \to B$ be a bijection from A to B. We know from Exercise 3.3.3 (1) that f^{-1} is a function from B to A, so we only need check out injectivity for f^{-1}. Suppose $f^{-1}(b) = f^{-1}(b')$; we need to show that $b = b'$. But since $f^{-1}(b) = f^{-1}(b')$ we have $f(f^{-1}(b)) = f(f^{-1}(b'))$. But since f^{-1} is a function, the definition of f^{-1} gives us $f(f^{-1}(b)) = b$ and $f(f^{-1}(b')) = b'$, so $b = b'$ as desired.
(c) Let f: $A \to A$ be an injection from A into A where A has n elements for some natural number n. Suppose for *reductio* that f is not a bijection. Then it is not onto A, so there is an $x \in A$ with $x \notin range(f)$. Since A has n elements, we can list its elements as $a_1, \ldots, a_n$ and so the elements of $range(f)$ may be listed as $f(a_1), \ldots, f(a_n)$. Thus $range(f)$ has n elements, so $range(f) \cup \{x\}$ has $n + 1$ elements, so A has more than n elements, giving us the desired contradiction.

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 would also cover 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 sets A, B have the same number of elements? The *equinumerosity principle* supplies a necessary and sufficient condition: $\#(A) = \#(B)$ showed that there is no bijection between *iff there is some bijection* f: $A \to B$.

Proof for finite sets. Let A, B be finite sets. 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. *End of solution*.

In general set theory, developed from its own axioms that do not mention anything from arithmetic, the existence of a bijection between two sets (whether finite or infinite) is often used as a *definition* of when they have the same cardinality. But in this book, we consider only its application to finite sets and make free use of elementary arithmetic as we go.

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

Alice How can you prove that?
Hatter The proof is ingenious and very short. It uses what is known as a '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, e.g. Paul Halmos' *Naïve Set Theory* and discussed in more general terms in resources such as the internet *Stanford Encyclopedia of Philosophy.*

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 allows us to express the problem in a more abstract but more amenable form.

Exercise 3.4.1

(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 onto B and injective. 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. Consider 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, so the equinumerosity principle tells us that A has the same number of elements as B. It remains to work out the number of elements in B. Now, there must be $n.(n-1)$ *ordered* pairs of distinct people in the reception, for 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. *End of solution.*

Don't feel bad if you couldn't work out the second example for yourself; we will be getting back to this sort of thing in Chap. 5 on counting. But for the present, we note that the sought-for bijection can 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 at a reception) with a rather more

'abstract' one (like the set of all unordered pairs $\{x,y\}$ of distinct elements of a given set), helping us to forget irrelevant information concerning the former and apply standard 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 A, B: $\#(A) \leq \#(B)$ *iff there is some injective function* f: $A \to B$.

To prove this for finite sets, we use the same sort of argument as for the equinumerosity principle. Let A, B be finite sets. 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: $A \to B$ 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: $A \to B$ 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, i.e. $\#(A) \leq \#(B)$.

Once again, abstract 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: $A \to B$ is injective. In other words: if $\#(A) > \#(B)$ *then for every function* f: $A \to B$ *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: $A \to B$ *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 be appreciated only by looking at examples. We begin with a very straightforward one.

Exercise 3.4.3 (1)
A village has 400 inhabitants. Show that at least two of them have the same birthday, and that there are at least 34 among them who 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: $A \to B$ associate each villager with day of birth, while g: $A \to C$ associates each villager with 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 $a \in A$, answering 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. *End of solution*.

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

Exercise 3.4.3 (2)
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: $A \to B$ associates with each member a pair as specified in the club ruling. 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 $a, a' \in A$. *End of solution*.

When solving problems by the pigeonhole principle, always begin by choosing carefully the sets A, B as well as the function f: $A \to B$. 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. The problem is thus one of finding a suitable abstract pattern in concrete data.

Exercise 3.4.3 (3)
A multiple-choice exam has 5 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?

Solution
Choose A to be the set of students taking the exam, B the set of all possible answers to the exam (under the constraints indicated) and f: $A \to B$ associating with each student their answer. Now B clearly has $2^5 = 32$ elements. So, if A has at least $3 \cdot 32 + 1 = 97$ elements, the pigeonhole principle guarantees that at least four distinct elements a of A receive the same value in B under f.

3.5 Some Handy Functions

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

3.5.1 Identity Functions

Let A be any set. The identity function over A is the function $f\colon A \to A$ 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, an everyday tool to be used almost without thinking.

Exercise 3.5.1

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

Solution

(a) To show that i_A is a bijection we need to check that it is injective and onto A. For injectivity, if $i_A(a) = i_A(a')$ then $a = i_A(a) = i_A(a') = a'$ and we are done. For surjectivity, let $a \in A$. Then $a = i_A(a)$ as needed.
(b) $f \circ i_A(a) = f(i_A(a)) = f(a)$ for all $a \in A$. Likewise, $i_B \circ f(a) = i_B(f(a)) = f(a)$ for all $a \in A$.
(c) Suppose $A \neq B$. Then either there is an $a \in A$ with $a \notin B$ or vice versa. Consider the former case, the latter is similar. Then $a \in dom(i_A)$ but $a \notin dom(i_B)$, so $i_A \neq i_B$.

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, as we showed in Exercise 3.5.1 (c).
Alice Why not define a great big identity function 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 or, failing that, the set of all sets—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 set is serving as a 'local universe' to work 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 whose domain includes A! 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 Chap. 9 (Sect. 9.4.1) we will set out systematically the logical relations between the eight different forms $\mathcal{Q}x\mathcal{Q}'y(\ldots)$, $\mathcal{Q}y\mathcal{Q}'x(\ldots)$ where $\mathcal{Q}$ and $\mathcal{Q}'$ are quantifiers (universal or existential), x and y are their attached variables, while 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: $A \to B$ are there?

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

Solution

(a) There are n of them, one for each element of B.

(b) Thinking visually with an arrow diagram, this is pretty obvious; but let's spell it out in full detail. For (i), suppose $\#(A) > 1$. Then A has at least two distinct elements a, a'. But if f: $A \to B$ is a constant function then $f(a) = f(a')$, failing injectivity. For (ii), suppose that $\#(B) > 1$ and let f: $A \to B$ be a constant function. If $A = \emptyset$ then f must be the empty function, and since B is not empty, f is not onto B. On the other hand, if $A \neq \emptyset$ then there is an $a \in A$ so, since $\#(B) > 1$, there is a $b \in B$ with $b \neq f(a)$. But since f is a constant function, $f(a') = f(a)$ for all $a' \in A$, so $b \neq f(a')$ for all $a' \in A$, so that f is not onto B.

3.5.3 Projection Functions

Let f: $A \times B \to C$ 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: $B \to C$ 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: $A \to C$ 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: $\mathbf{N} \times \mathbf{N} \to \mathbf{N}$ 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 (i) the right projection of the multiplication function $x \cdot y$: $\mathbf{N} \times \mathbf{N} \to \mathbf{N}$ at the value $x = 3$, (ii) the left projection at the value $y = 3$. Are they the same function?

Solution

(a) $f_2(n) = 2^n$. $f_3(m) = m^3$.

(b) (i) f: $\mathbf{N} \to \mathbf{N}$ with $f(y) = 3 \cdot y$, (ii) g: $\mathbf{N} \to \mathbf{N}$ with $g(x) = x \cdot 3$. Yes, $f = g$ because multiplication is commutative.

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: $U \to \{1,0\}$ 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), sometimes called its *indicator function*. Thus, in terminology from logic, the characteristic function f_A specifies the truth-value of the statement that $u \in A$.

Conversely, when f: $U \to \{1,0\}$, we can define the *associated subset* of U by putting $A_f = \{u \in U: f(u) = 1\}$. Clearly, there is a bijection between the subsets of U and the functions f: $U \to \{1,0\}$, 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 (i) the set of all prime numbers, in the local universe $\mathbf{N}^+$ of positive integers? (ii) Of the composites (non-primes)? (iii) Of the empty set, and (iv) 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?

Solution

(a) (i) It is the function f: $\mathbf{N}^+ \to \{1,0\}$ that puts $f(n) = 1$ when n is prime, otherwise 0. (ii) The function f: $\mathbf{N}^+ \to \{1,0\}$ that puts $f(n) = 0$ when n is prime, otherwise 1. (iii) The function f: $\mathbf{N}^+ \to \{1,0\}$ that puts $f(n) = 0$ for all $n \in \mathbf{N}^+$. (iv) The function f: $\mathbf{N}^+ \to \{1,0\}$ that puts $f(n) = 1$ for all $n \in \mathbf{N}^+$.

(b) $f_{A \cap B}(x) = 1$ iff $f_A(x) = 1$ and $f_B(x) = 1$; $f_{A \cup B}(x) = 1$ iff $f_A(x) = 1$ or $f_B(x) = 1$; $f_{U \backslash A}(x) = 1$ iff $f_A(x) = 0$, for all $x \in U$. They correspond respectively to the truth-tables for conjunction, disjunction and (approximately) negation as applied to the propositions $(x \in A) \wedge (x \in B)$, $(x \in A) \vee (x \in B)$, $(x \in U) \wedge (x \notin A)$.

3.5.5 Choice Functions

Let $\mathcal{A}$ be any collection of non-empty sets. Then there is a function f: $\mathcal{A} \to \cup\mathcal{A}$ such that $f(A) \in A$ for all $A \in \mathcal{A}$. In other words, there is a function that picks out, from each set in the collection, one of its elements. They are called *choice functions* and are very useful in both workaday and abstract set theory.

The principle has considerable notoriety because of its 'non-constructive' nature: it asserts the existence of at least one choice function without at the same time providing a way of specifying any one such function uniquely. A famous illustration of this, devised by Bertrand Russell for a non-mathematical audience, imagines that we have an infinite set $\mathcal{A}$ of pairs of socks. A choice function for $\mathcal{A}$ would pick out one sock from each pair. Whereas shoes come in left and right pairs, socks have no built-in features for such a choice function to be fully specified yet, by the axiom of choice, at least one choice function exists.

Because of this non-constructive aspect, when $\mathcal{A}$ is infinite the principle cannot in general be proven from more elementary principles of set theory; it must instead be assumed as a postulate, called the *axiom of choice*. Actually, a more adequate name would be the 'axiom for carrying out infinitely many choices all at once' since that is what the function does when $\mathcal{A}$ is infinite; but that is quite a mouthful, so the shorter name is standard. On the other hand, when $\mathcal{A}$ is finite a choice function for it can be specified constructively, if only by giving a long but finite list of all the ordered pairs in the function.

As also noted by Russell, the axiom of choice can be formulated equivalently without speaking of functions at all: for any collection A of disjoint non-empty sets, there is a set that contains exactly one element of each set in $\mathcal{A}$. Note the qualification 'disjoint' in this formulation, which we call *Russell's version*.

The axiom of choice has a great many useful consequences, some of which turn out to be equivalent to it. Perhaps the most famous among the equivalent principles is the *well-ordering theorem*: for every non-empty set A there is a relation R that well-orders A (in a sense to be defined in Chap. 4). Its derivation from the axiom of choice is quite difficult (although the converse is easy) and is suited for a more advanced course in set theory. Another equivalent principle, very useful in abstract algebra and advanced logic, is known as *Zorn's Lemma*. Using notions and terminology from Chap. 2, it says: every partially ordered set A each of whose sub-chains, has an upper bound in A has a maximal element.

A much more immediate consequence of the axiom of choice that is handy in everyday applications of set theory is that for every relation R there is a set $f \subseteq R$ with the same domain. That in turn permits a way of reformulating multiply-quantified statements of the kind "for all $x \in A$ there is a $y \in B$ such that $(x, y) \in R$" as "there is a function f: $A \to B$ such that for all $x \in A$, $(x, f(x)) \in R$". We will be doing this, as an essential step in a construction, in Chap. 4 Sect. 4.7.1 and as a stylistic simplification of another one in Chap. 11, Sect. 11.4.

Exercise 3.5.5

(a) Give a simple finite example that shows why the condition of disjointedness is needed in Russell's version of the axiom of choice.

(b) Show that the standard, Russell, and '$f \subseteq R$' versions of the axiom of choice are mutually equivalent.

Solution

(a) Let $X = \{1,2,3\}$ and put $\mathcal{A} = \{\{1,2\}, \{2,3\}, \{1,3\}\}$; the sets in $\mathcal{A}$ are non-empty but $\mathcal{A}$ is not disjoint. We claim that there is no set $C \subseteq X$ containing exactly one element from each element of $\mathcal{A}$. Suppose there is such a set C. It must contain 1 or 2. If it contains 1 then it doesn't contain 2 so it must contain 3 so it contains both elements of $\{1,3\}$. Similarly, if it contains 2 then it doesn't contain 1 so it must contain 3 so it contains both elements of $\{2,3\}$.

(b) We show that the standard version implies Russell's, which implies the '$f \subseteq R$' one, which implies the standard one. Of course, one could cycle around the three versions in a different order, or choose one of the three and show that it is equivalent to each of the other two.

Assume the standard version of the axiom of choice. Let $\mathcal{A}$ be any disjoint collection of non-empty sets; we want to show that there is a set that contains exactly one element of each set in A. By the standard version, there is a function f: $\mathcal{A} \to \cup\mathcal{A}$ such that $f(A) \in A$ for all $A \in \mathcal{A}$. Put $C = \{f(A)$: $A \in \mathcal{A}\}$. By definition, C contains at least one element from each element of $\mathcal{A}$ and, since $\mathcal{A}$ is disjoint, C contains no more than one from each.

Assume now the Russell version. Let R be any relation with A its domain and B its range. For each $a \in A$, put $S_a = \{(a,b)$: $b \in R(a)\}$ and put $\mathcal{A} = \{S_a$: $a \in$

$A\}$. Then the elements of $\mathcal{A}$ are non-empty and disjoint—it is to ensure disjointedness that we did not simply put $S_a = R(a)$—so, by Russell, there is a set C consisting of exactly one element (a,b) from each S_a. Define f as follows: for each $a \in dom(R)$, $f(a)$ is the unique b in $range(R)$ such that $(a,b) \in S_a$. Then f is a function on the domain of R into the range of R with $f \subseteq R$.
Assume now the '$f \subseteq R$' version. Let $\mathcal{A}$ be any collection of non-empty sets. We need to show that there is a function f: $\mathcal{A} \to \cup\mathcal{A}$ such that $f(A) \in A$ for all $A \in \mathcal{A}$. Put $R = \{(A,a)$: $a \in A \in \mathrm{A}\}$. Note that since the elements of $\mathcal{A}$ are non-empty, the domain of R is just $\mathcal{A}$ and the range of R is $\cup\mathcal{A}$. By the assumption, there is a function f with the same domain with $f \subseteq R$, that is, f is a function on $\mathcal{A}$ into $\cup\mathrm{A}$ with $f(a) \in A$ for all $A \in \mathcal{A}$ as desired.

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 handle, 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 mental processing.

We now introduce the subtly different concept of a *family* of sets. This refers to any *function* on a domain I (again 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 $\{A_i$: $i \in I\}$ referred to in the preceding paragraph.

The difference is fine, and in some contexts sloppiness does not matter and the collection notation $\{A_i$: $i \in I\}$ may then be used indiscriminately for both the collection and abused, for the family. But in certain situations, notably applications of the pigeonhole principle and other counting rules of Chap. 5, the distinction between a collection of sets and a family of sets becomes 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 $f(1) = f(2)$, in other words, $A_1 = A_2$. 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 than n 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 $i \in \mathbf{N}^+$', we are inviting consideration of the function p: $\mathbf{N}^+ \to \mathbf{N}^+$ such that each $p(i)$ is the ith prime number, as well the collection that is its range—which, in this instance, is just the set of all prime numbers.

The moral of this story is that when you see or use a notation like $\{A_i: i \in I\}$ you should pause to ask yourself two questions: is this referring to the function (family) or its range (collection), and does it matter for the issue under discussion?

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

Solution
(a) When f is a constant function, the collection has just one element. (b) The family has exactly n elements, whatever function is in question, so long as its domain has n elements. (c) When f is injective, then both the collection and the family have n elements.

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, steps in its execution and so on. 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 with 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 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 $i \leq n \in \mathbf{N}^+$ and with $f(n + m) = f(n)$ for all $m \in \mathbf{N}^+$, 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: $\{i \leq n \in \mathbf{N}^+\} \to A$, i.e. as a partial function on $\mathbf{N}^+$ whose domain is the initial segment $\{i \leq n \in \mathbf{N}^+\}$ of $\mathbf{N}^+$. Formally, it doesn't really matter which way we do it; in either case one has made a rather vague notion sharper by explaining it in terms of functions. In what follows we will follow the former definition.

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: $\{i \leq n \in \mathbf{N}^+\} \to A$ 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 Chap. 2 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…

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 $(a_1, a_2, \ldots, a_n)$, 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?

Hatter Let's have some more tea.

We should try to help Alice and to get the Hatter off the hook. 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 that it satisfies 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: $\{i \leq n \in \mathbf{N}^+\} \to A$. So defined, sequences 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, in particular strings of symbols, this tool is the operation of *concatenation*, which consists of taking any strings s and t and forming a string $con(s,t)$. 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 thus 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 more limited in its power. 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 *con* to the alphabet set A. The 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 is not in general satisfied by concatenation. 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. Finite sequences may be defined as functions on initial segments of the positive integers, so 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, on the other, a restricted form of concatenation that ensures unique decomposability. These tools are vital to the computer scientist for constructing, manipulating and proving things about composite symbols.

3.7 End-of-Chapter Exercises

Exercise 3.7 (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.

Solution

(a) (i) In terms of tables: a partial function from A to B is a relation $R \subseteq A \times B$ whose table has at most one 1 in each row. In terms of digraphs: every point in the A circle has at most one arrow going out from it to the B circle.
(b) Let R be a relation from A to B. For the left-to-right implication, suppose that R is a partial function from A to B. Then it is a set of ordered pairs (a,b) with $a \in A$, $b \in B$ such that for each $a \in A$ there is at most one $b \in B$ with $(a,b) \in R$. But $dom(R)$ is the set of all $a \in A$ such that there is at least one $b \in B$ with $(a,b) \in R$. So R is a set of ordered pairs (a,b) with $a \in dom(R)$, $b \in B$ such that for each $a \in dom(R)$ there is exactly one $b \in B$ with $(a,b) \in R$; that is, R is a function from $dom(R)$ to B.

For the right-to-left part, the verification is essentially the same in reverse, but we give it in full. Suppose that R is a function from $dom(R)$ to B. Then R is a set of ordered pairs (a,b) with $a \in dom(R)$, $b \in B$ such that for each $a \in dom(R)$ there is exactly one $b \in B$ with $(a,b) \in R$. Since $dom(R) \subseteq A$, we can say that R is a set of ordered pairs (a,b) with $a \in A$, $b \in B$ such that for each $a \in A$ there is at most one $b \in B$ with $(a,b) \in R$, that is, R is a partial function from A to B.

Exercise 3.7 (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 (i) $[0,1] = \{x \in \mathbf{R}: 0 \leq x \leq 1\}$, (ii) $[0,1) = \{x \in \mathbf{R}: 0 \leq x < 1\}$, (iii) $(0,1] = \{x \in \mathbf{R}: 0 < x \leq 1\}$, (iv) $(0,1) = \{x \in \mathbf{R}: 0 < x < 1\}$?

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

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

(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 $f: A \to B$ be a function from set A into set B. Recall the 'abstract inverse' function $f^{-1}: B \to \mathcal{P}(A)$ defined at the end of Sect. 3.3.1 by putting $f^{-1}(b) = \{a \in A: f(a) = b\}$ for each $b \in B$. (i) Show that the family 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 family the sets $f^{-1}(b)$ for $b \in B \backslash f(A)$?

Solution

(a) (i) $\{0,1\}$, (ii) $\{0\}$, (iii) $\{0,1\}$, (iv) $\{0\}$.

(b) (i) Using the top-down definition, it suffices to show that every set $Y \subseteq A$ such that both $X \subseteq Y$ and $f(Y) \subseteq Y$, includes $f(X)$. But if $X \subseteq Y$ then clearly $f(X) \subseteq f(Y)$, so $f(X) \subseteq Y$ and we are done. Using the bottom-up definition by unions, we note that $f[X] = \cup\{X_n: n \in \mathbf{N}\}$ where $X_0 = X$ and $X_{n+1} = X_n \cup f(X_n)$ so that $f(X) \subseteq X \cup f(X) = X_1 \subseteq f[X]$. (ii) For an example of the failure of the converse inclusion, put $A = \mathbf{N}$, $f(n) = n + 1$, $X = \{0\}$ so $f(X) = \{1\}$ while $f[X] = \mathbf{N}$.

(c) It is convenient to use the bottom-up definition. Suppose $f(X) \subseteq X$. Since $f[X] = \cup\{X_n: n \in \mathbf{N}\}$, it suffices to show that $X_n = X$ for every $n \in \mathbf{N}$. Now by definition $X_0 = X$; and whenever $X_k = X$ then $X_{k+1} = X_k \cup f(X_k) = X_k = X$. This kind of argument is known as proof by induction and we will have more to say about it in the next chapter.

(d) Let $\{B_i\}_{i \in I}$ be a partition of A and, for each $a \in A$, let $f(a)$ be the unique B_i with $a \in B_i$. This is indeed a function with domain A and, since each $B_i \subseteq A$, it is into the powerset $\mathcal{P}(A)$ of A. Since the cells B_i are non-empty and together exhaust A, $range(f) = \{B_i\}_{i \in I}$.

(e) Let f: $A \to B$ be a function from set A into set B. For (i), we need to show that the sets $f^{-1}(b)$ for $b \in f(A)$ are non-empty, disjoint, and exhaust A. For non-emptiness, let $b \in f(A)$; we need to check that $f^{-1}(b)$ is not empty. Now $b = f(a)$ for some $a \in A$ and by the definition of the abstract inverse f^{-1} we have $a \in f^{-1}(f(a)) = f^{-1}(b)$. For exhaustion, let $a \in A$; we need to check that $a \in f^{-1}(b)$ for some $b \in f(A)$. Put $b = f(a)$. Then $a \in f^{-1}(f(a)) = f^{-1}(b)$ as desired. For disjointedness, suppose $a \in f^{-1}(b) \cap f^{-1}(b')$; we need to check that $b = b'$. By the supposition, $f(a) = b$ and $f(a) = b'$ so, since f is a function, $b = b'$ as needed. For (ii), suppose we include in the family the sets $f^{-1}(b)$ for $b \in B \backslash f(A)$. In the case that f is not onto B, there is at least one such b. By the definition of abstract inverse, $f^{-1}(b) = \varnothing$, contrary to the requirement for a partition that each cell be non-empty.

Exercise 3.7 (3) Injections, surjections, bijections

(a) (i) Is the floor function from $\mathbf{R}$ into $\mathbf{N}$ injective? (ii) Is it onto $\mathbf{N}$?
(b) Is the composition of two bijections always a bijection?
(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: $A \to B$ and g: $B \to A$, then there is a bijection from A to B.

Solution

(a) (i) No since, for example $\lfloor 0.1 \rfloor = 0 = \lfloor 0.2 \rfloor$. (ii) Yes, since for every $n \in \mathbf{N}$ we have $\lfloor n \rfloor = n$.
(b) Yes, since by Exercise 3.3.1 (1) (c) it is injective and by Exercise 3.3.2 (2) (b) it is surjective.
(c) Let A be a finite set, B a proper subset of A. Then if A has n elements, B must have $m < n$ elements, and for an injective function f: $B \to A$, if B has m elements then also $f(B)$ has m elements, so $f(B) \neq A$ and f is not onto A.
(d) Let E be the set of all even natural numbers. Then $E \subset \mathbf{N}$, but the function f: $\mathbf{N} \to E$ defined by putting $f(n) = 2n$ is clearly a bijection between $\mathbf{N}$ and E. This simple fact was surprisingly difficult to assimilate in the history of mathematics before the nineteenth century. It is sometimes known as Galileo's paradox, although it is not a contradiction but a behaviour that is in contrast with that for finite sets, as shown by (c).
(e) Let A, B be finite sets and suppose if there are injective functions f: $A \to B$ and g: $B \to A$. The principle of comparison tells us that $\#(A) \leq \#(B)$ and $\#(B) \leq \#(A)$, so $\#(A) = \#(B)$. we may thus write $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ for the same natural number n. Define h: $A \to B$ by putting $h(a_i) = a_i$ for all $i \leq n$; clearly h is both injective and onto B. For infinite sets the same

principle holds, but it cannot be proven by the pigeonhole principle, which applies only to finite sets. It is known as the Cantor-Schröder-Bernstein theorem and can be proven by a very beautiful argument that is suited for a more advanced course in set theory.

Exercise 3.7 (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 $A = \{1,2,3,4,5,6,7,8\}$. How many distinct numbers must be selected from A to guarantee that there are two of them that sum to 9?

Solution

(a) Let P be a set of seven propositions, $V = \{0,1\}$ the set of the two truth-values, and f: $P \to V$ the function that attributes to each proposition in P its truth-value. Since $\#(P) > 3 \cdot \#(V)$ the general form of the pigeonhole principle tells us that there is a truth-value $v \in V$ such that $v = f(p)$ for at least $3 + 1 = 4$ propositions $p \in P$.
(b) The answer may be intuitively obvious, but we give full details carefully. When we select k distinct elements of A (the qualification 'distinct' is important here), we are specifying a subset X of A with k elements. So, the question amounts to: What is the smallest integer k such that for every subset X of A where $\#(X) = k$, at least two elements of X are in the same cell? With this reconceptualization in mind, let A be a set, $X \subseteq A$, $\mathcal{B}$ a partition of A into n cells, and f: $X \to \mathcal{B}$ the function that attributes to each element of X its cell. The pigeonhole principle tells us immediately that if $\#(X) > n$ then there is a $B \in \mathcal{B}$ with $B = f(x)$ for at least two distinct elements x of X. Thus $n + 1$ does the job. No smaller integer can do the job: for any $m \leq n$ we can take m of the n disjoint cells and select one element from each, getting a subset $X \subseteq A$ with $\#(X) = m$ and no two elements of X are in the same cell.
(c) Let $A = \{1, \ldots, 8\}$. We set things up so that we can apply (b). Let $\mathcal{B}$ be the set of all unordered pairs $\{x,y\}$ with $x, y \in A$ and $x + y = 9$. Then $\mathcal{B} = \{\{1,8\}, \{2,7\},\{3,6\},\{4,5\}\}$, which is clearly a partition of A into four cells. So, by part (b) of this exercise, selecting five distinct elements of A will guarantee that there are two of them that are in the same cell, i.e. that sum to 9.

We can also give a solution that does not appeal to partitions or even to the pigeonhole principle. Consider the subsets $X \subseteq A$ such that no two distinct elements of X sum to 9. How big can X be? Write X as $\{x_1, \ldots, x_8\}$. There are 8 ways of choosing x_1 with the constraint unviolated. Note that for each $x \in A$ there is a unique $y \in A$ such that $x \neq y$ and $x + y = 9$. Hence there are $7 - 1 = 6$ ways of choosing a distinct x_2 so as not to sum to 9, $6 - 2 = 4$ ways of choosing a distinct x_3

so that no two of $x_1, \ldots, x_3$ sum to 9, $5 - 3 = 2$ ways of choosing a distinct x_4 so that no two of $x_1, \ldots, x_4$ sum to 9, and $4 - 4 = 0$ ways of choosing a distinct x_5 so that no two of $x_1, \ldots, x_5$ sum to 9. So, selecting five distinct elements of A will guarantee that there are two of them that sum to 9.

This illustrates how a single problem can often be solved by quite different methods. That can lead us to think more about the methods themselves, and sometimes we can see that at bottom, they are not quite so different as they seemed.

Exercise 3.7 (5) Handy functions

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

(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: $A \times B \to C$ be a function of two arguments. Verify or give counter-examples to the following. (i) If f is injective in the strongest of the three senses of Exercise 3.3.1 (2), then every (left and right) projection of f is also injective. (ii) If f is surjective then every projection of f is also surjective.

Solution

(a) (i) Let f: $A \to B$ and g: $B \to C$. Suppose that there is a $c \in C$ such that g $(b) = c$ for all $b \in B$. Choose such a c, call it c_0. Then since always $f(a) \in B$, we have that for all $a \in A$, $g \circ f(a) = g(f(a)) = c_0$. Suppose alternatively that there is a $b \in B$ such that $f(a) = b$ for all $a \in A$. Choose such a b, call it b_0. Then, for all $a \in A$, $g \circ f(a) = g(f(a)) = g(b_0)$.

(a) (ii) No; draw a diagram as you read this counter-example. Put $A = \{1,2\}$, $B = \{3,4,5\}$, $C = \{6,7\}$. Define f by $f(1) = 3, f(2) = 4$ and define g by $g(3) = g(4) = 6$ but $g(5) = 7$. Then neither f nor g is a constant function but $g \circ f$ is a constant function with value 6 for all its arguments.

(b) In general terms, we hold the ith argument fixed at some value and consider the $(n - 1)$-argument function obtained by allowing the other arguments to vary. In full notational glory, let f: $A_1 \times \ldots \times A_n \to C$ be a function of n arguments and let $a \in A_i$. We define f_a: $A_1 \times \ldots \times A_{i-1} \times A_{i+1} \times \ldots \times A_n \to C$ by putting $f_a(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n) = f_a(a_1, \ldots, a_{i-1}, a, a_{i+1}, \ldots, a_n)$ for all $a_j \in A_j$ with $j \neq i$.

(c) (i) True. Verification: suppose f is injective in the strongest of the three senses of Exercise 3.3.1 (2). Then, for the right projection, if $f_a(b) = f_a(b')$ we have $f(a,b) = f(a,b')$ so $b = b'$ as needed. Similarly, for the left projection. (ii) False; draw a diagram as you read this counter-example. Put $A = \{1,2\}$, $B = \{3\}$, $C = \{4,5\}$. Define f by $f(1,3) = 4, f(2,3) = 5$. Then f is onto C but neither of its right projections f_1, f_2 is onto C.

3.8 Selected Reading

The texts of Bloch, Halmos, Hein, Lipschutz and Velleman, listed at the end of chapters 1 and 2, also have good chapters on functions with the same attractions as their presentations of sets and relations. The specific chapters are 4, 8–10, 2, 4–5 and 5 respectively.

If you are tempted to explore the axiom of choice or the Cantor-Schröder-Bernstein theorem, a good place to begin is Paul R. Halmos *Naive Set Theory*, Springer, 2001 (new edition) chapters 15–17 and 22 respectively.

Recycling Outputs as Inputs: Induction and Recursion

4

Chapter Outline

This chapter introduces induction and recursion, which are omnipresent in computer science and logic. 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 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 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 by their domains.

The broadest and most powerful kind of induction/recursion may be formulated for any set at all, provided it is 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 briefly 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 inductive proofs and recursive definitions. But it should not be thought that they

D. Makinson, *Sets, Logic and Maths for Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-42218-9_4

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 long-standing rigidity. From the time of Aristotle onwards it was a basic tenet of philosophy, logic and 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 acceptable. But it was also gradually realized that definitions, proofs, and procedures may 'call upon themselves', in the sense that later steps may systematically appeal to the outcome of earlier steps. For example, the value of a function for a given value of its argument may be defined in terms of its value for smaller arguments; a proof of a fact about an item may assume that we have already proven the same fact about lesser items; 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 been performed. 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 $\mathbf{Z}$ of all integers, negative as well as positive, has no starting point under its natural ordering. The set $\mathbf{Q}$ of rational numbers not only lacks a starting point, but it also has the wrong kind of order: no rational number has an immediate rational successor; likewise for the set $\mathbf{R}$ of reals.

Conversely, 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 italicised requirements above. Such non-numerical applications are very important for computer science but are often neglected in accounts written from a traditional mathematical perspective, where numbers take front stage. 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 $\Sigma\{i: 1 \leq i \leq n\}$ or $\Sigma_i\{1 \leq i \leq n\}$ or just $f(n)$ for the sum of the first n 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 hear 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 $f(1) = 1$, while 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 $f(n) = n \cdot (n + 1)/2$ 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. Notice 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 possibly 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* $f(k) = k \cdot (k + 1)/2$. We need to *show* that $f(k + 1) = (k + 1) \cdot (k + 2)/2$. Now:

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

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 italicized 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 $(k + 1) \in A$.

Checking the basis is usually a matter of trivial calculation. Establishing the induction step is 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.

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, i.e. that $k + 1$ has the property, is called the *induction goal*. We will often need to use these two terms in what follows.

Exercise 4.2.2 (1)
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, i.e. for $1 + 3 + \ldots + (2n - 1)$. 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 $n = k$, i.e. suppose that $f(k) = k^2$. We need to show (induction goal) that it holds when $n = k + 1$, i.e. that $f(k + 1) = (k + 1)^2$. Now:

$$\begin{aligned} f(k+1) &= 1+3+\ldots+(2k-1)+(2(k+1)-1) && \text{by definition of } f \\ &= (1+3+\ldots+(2k-1))+(2(k+1)-1) && \text{arranging brackets} \\ &= f(k)+2(k+1)-1 && \text{also by definition of } f \\ &= k^2+2(k+1)-1 && \text{by the induction hypothesis} \\ &= k^2+2k+1 && \text{by elementary arithmetic} \\ &= (k+1)^2 && \text{ditto. } \textit{End of solution} \end{aligned}$$

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, say, 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 interested only in the positive integers, 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 an 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 the first two of the displayed formulations of the induction step in contrapositive form, stating them with subtraction rather than addition.

Solution
For every positive integer $k > 1$, if k lacks the property than so does $k - 1$.

For every positive integer $k > 1$, if $k \notin A$, then $k - 1 \notin A$.

Exercise 4.2.2 (3)
In Exercise 3.4.1 (b) of 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 $n \cdot (n - 1)/2$ handshakes. Show this again, by using the simple principle of induction over the positive integers.

Solution

Basis: This is 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 $k + 1$ people then 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 that 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. *End of solution.*

Exercise 4.2.2 (3) 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 $2s = (1 + \ldots + n) + (n + \ldots + 1) = n \cdot (n + 1)$ by adding the corresponding terms n times, so $s = n \cdot (n + 1)/2$. When a 'direct' proof like this is available, people often find it more illuminating and, in some elusive sense, more explanatory, than an inductive proof, but the two are equally conclusive.

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. 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 the whole procedure 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) we 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 classroom experience again warns that doing so irrevocably confuses those who are already struggling. So, to emphasise that the proposition to be proven by induction and the proposition in the proof serving as induction hypothesis 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, instead, to show by induction that we can express the sum of the first n even positive integers in some

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) Do what 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.

Solution

(a) We want to show that $\Sigma\{2i\colon 1 \leq i \leq n\}$, that is, the sum of the first n even positive integers, is $n(n + 1)$. *Basis*: When $n = 1$, then LHS = 2 while RHS = $1(1 + 1) = 2$. *Induction step*: Suppose (induction hypothesis) that $\Sigma\{2i\colon 1 \leq i \leq k\} = k(k + 1)$. We want to show (induction goal) that $\Sigma\{2i\colon 1 \leq i \leq k + 1\} = (k + 1)(k + 2)$. Now, LHS = $\Sigma\{2i\colon 1 \leq i \leq k\} + 2(k + 1) = k(k + 1) + 2(k + 1) = k^2 + 3k + 2$ = RHS using the induction hypothesis for the second equality and multiplying out the RHS for the last equality.
(b) Begin by observing that the first $2n$ positive integers are precisely the first n odd ones and the first n even ones. So, the sum of the first n odd integers plus the sum of the first n even integers equals the sum of the first $2n$ integers. In other words, the sum of the first n even integers equals the sum of the first $2n$ integers minus the sum of the first n odd integers. We already have the formulae for the RHS: $2n \cdot (2n + 1)/2 - n^2 = n \cdot (2n + 1) - n^2 = n^2 + n = n(n + 1)$ as desired. *End of solution.*

Exercise 4.2.2 (4) shows that we sometimes have the choice between proving something from first principles or proving it from facts already established. Usually, the latter is quicker, but not always; in this instance they are about the same length.

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 $1 + 2 + 3 + \ldots + n$ 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, as is often said, an 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:

$$\textit{Basis of definition} : f(1) = 1$$
$$\textit{Recursive step of definition} : \textit{when } n \geq 1 \textit{ then } f(n+1) = f(n) + (n+1).$$

This way of expressing the recursive step defines $f(n + 1)$ in terms of $f(n)$ (and other functions taken as given, in our example addition). Another way of writing it defines $f(n)$ in terms of $f(n - 1)$. In our example, this puts:

$$\textit{Recursive step of definition} : \textit{ when } n > 1 \textit{ then } f(n) = f(n-1) + n.$$

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

$$\textit{If } n = 1 \textit{ then } f(n) = 1$$
$$\textit{If } n > 1 \textit{ then } f(n) = f(n-1) + n.$$

This can be abbreviated to the popular *if-then-else* form:

$$\textit{If } n = 1 \textit{ then } f(n) = 1$$
$$\text{Else } f(n) = f(n-1) + n.$$

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

$$f(n) = \text{if } n = 1 \text{ then } 1 \text{ else } f(n-1) + n,$$

which can look like mumbo-jumbo to the uninitiated. *All of these formulations say the same thing*. You will meet each of them in applications and 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

(a) Define recursively the following functions $f(n)$ on the positive integers: (i) The sum $2 + 4 + \ldots + 2n$ of the first n even integers, (ii) The sum $1 + 3 + \ldots + (2n - 1)$ 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 $n!$, is pronounced 'n factorial'.
(d) Use the recursive definitions of the functions concerned to show that for all $n \geq 4$, $n! > 2^n$. *Hint*: Here the basis will concern the case that $n = 4$.

Solution

(a) (i) Basis: $f(1) = 2$. Recursive step: when $n > 1$ then $f(n + 1) = f(n) + 2(n + 1)$. (ii) Basis: $f(1) = 1$. Recursive step: when $n > 1$ then $f(n + 1) = f(n) + 2n + 1$.
(b) Basis: $2^0 = 1$. Recursive step: $2^{n+1} = 2 \cdot 2^n$.
(c) Basis: $1! = 1$. Recursive step: $(n + 1)! = (n + 1) \cdot n!$
(d) Basis of proof: We need to show that $4! > 2^4$. This is immediate by calculating $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24 > 16 = 2^4$. Inductive 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:

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

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 (as measured by, say, number of steps required, time taken or memory required) as inputs increase in size. In Chap. 1 we saw that exponentiation already gives unmanageable rates of growth. Exercise 4.3 shows that factorial is worse.

4.4 Evaluating Functions Defined by Recursion

Go back yet again to the first function that was 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 $f(7)$ using the recursive definition and without appealing to a general formula such as $f(n) = n \cdot (n + 1)/2$. There are basically two ways of doing it.

An 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:

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

Here we have made one application of the base clause followed by six applications of the recursion clause. Each application is followed by arithmetic simplification as needed. Each of the seven steps fully eliminates the function sign f on the right-hand side 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:

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

In this calculation, we begin by writing the expression $f(7)$ whose value we are seeking, then 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 purely numerical expression in which f does not appear. From that point on, 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).

Solution
See Table 4.1.

Table 4.1 Calculating 6! bottom-up and top-down

Bottom-up	Top-down
$1! = 1$ $2! = 1! \cdot 2 = 1 \cdot 2 = 2$ $3! = 2! \cdot 3 = 2 \cdot 3 = 6$ $4! = 3! \cdot 4 = 6 \cdot 4 = 24$ $5! = 4! \cdot 5 = 24 \cdot 5 = 120$ $6! = 5! \cdot 6 = 120 \cdot 6 = 720$	$6! = 5! \cdot 6$ $= (4! \cdot 5) \cdot 6$ $= ((3! \cdot 4) \cdot 5) \cdot 6$ $= (((2! \cdot 3) \cdot 4) \cdot 5) \cdot 6$ $= ((((1! \cdot 2) \cdot 3) \cdot 4) \cdot 5) \cdot 6$ $= ((((1 \cdot 2) \cdot 3) \cdot 4) \cdot 5) \cdot 6$ $= (((2 \cdot 3) \cdot 4) \cdot 5) \cdot 6$ $= ((6 \cdot 4) \cdot 5) \cdot 6$ $= (24 \cdot 5) \cdot 6$ $= 120 \cdot 6$ $= 720$

4.5 Cumulative Induction and Recursion

We now turn to some rather more sophisticated forms of recursive definition and inductive proof. Ultimately, the forms introduced in this section can be derived from the simple one, 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 (that is, 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 $n \geq 2$.

This function illustrates how a top-down evaluation by unfolding, if not pruned somehow, can lead to great inefficiencies in computation due to repetition of calculations already made. Beginning a top-down computation of $F(8)$ we have to make many calls, as represented in Table 4.2. To read it, notice that each cell in a row is split into two in the row below, following the recursive rule for the 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 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 (if we include the one still to come in the next row). Unless partial calculations are saved and recycled in some manner, the inefficiency of the procedure is very high.

Table 4.2 Calls when computing F(8) top-down

$F(8)$															
$F(7)$								$F(6)$							
$F(6)$				$F(5)$				$F(5)$				$F(4)$			
$F(5)$		$F(4)$		$F(4)$		$F(3)$		$F(4)$		$F(3)$		$F(3)$		$F(2)$	
$F(4)$	$F(3)$	$F(3)$	$F(2)$	$F(3)$	$F(2)$	$F(2)$	$F(1)$	$F(3)$	$F(2)$	$F(2)$	$F(1)$	$F(2)$	$F(1)$	$F(1)$	$F(0)$
etc															

Exercise 4.5.1

(a) Rewrite the definition of the Fibonacci function in *if-then-else* language.
(b) Express the recursion clause for the Fibonacci function by defining $F(n + 2)$ in terms of $F(n + 1)$ and $F(n)$. Be explicit about the bound of your quantification.
(c) Carry out a bottom-up evaluation of $F(8)$ and compare it with the (incomplete) top-down evaluation in the text.

Solution

(a) If $n = 0$ then $F(n) = 0$, if $n = 1$ then $F(n) = 1$, else $F(n) = F(n - 1) + F(n - 2)$.
(b) When $n \geq 2$ then $F(n + 2) = F(n + 1) + F(n)$.
(c) $F(0) = 0$, $F(1) = 1$, $F(2) = 1 + 0 = 1$, $F(3) = 1 + 1 = 2$, $F(4) = 2 + 1 = 3$, $F(5) = 3 + 2 = 5$, $F(6) = 5 + 3 = 8$. *End of solution*.

On the other hand, there are situations 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 together with the basis. But if we were to calculate $f(8)$ bottom-up without devising shortcuts, we would be doing a lot of unnecessary work calculating $f(n)$ for odd values 3,5,7 of n. We will see a more dramatic example of the same phenomenon shortly, when considering what is known as the Ackermann function.

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 argument that permits us to do the same when proving things about $f(n)$. 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)

(a) Express 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. This way of expressing the principle is sometimes known as *descending induction*.
(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 with $2 \leq a, b < k$. So the induction hypothesis tells us that each of a, b has the property, i.e. is the product of (one or more) prime numbers. Hence their product is too, and we are done. *End of solution*.

Part (b) of Exercise 4.5.2 (1) 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 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; when $n < 2$ then, as we saw in Sect. 1.6, the conditional is vacuously true. But when the induction begins at a specified number $a > 0$, it is usually more transparent to formulate its induction step in a form that mentions a explicitly as a bound on the quantifier. So expressed, cumulative induction starting at a is as follows.

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.

Don't forget the $a \leq j$ in the induction step, as the property may not hold for smaller numbers!

The third comment is that, strictly speaking, for cumulative induction the basis is 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 the redundancy in the cumulative case is most easily seen when the induction starts from zero. There are no natural numbers less than zero so, vacuously, every natural number $j < 0$ has whatever property is under consideration so that, given the induction step, we may apply it to conclude that zero has the property and the basis holds.

It looks like magic, but it is logic. Nevertheless, even for cumulative induction it is quite common to state the basis explicitly and even to check it out separately, in effect double-checking that we did not carelessly overlook some peculiarity of the case $k = 0$ when establishing the induction step in its generality. You are free to 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, we would give the proof, now but it isn't, so we don't. The important point for us is that although cumulative induction 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)

(a) Use cumulative induction to show that for every natural number n, $F(n)$ is even iff n is divisible by 3, where F is the Fibonacci function.
(b) Adapt the remarks above to explain why the basis for cumulative induction is redundant when it starts from $a > 0$.

Solution

(a) Although we are carrying out a cumulative induction, we give the basis explicitly. Consider first the case $n = 0$, which is divisible by 3 while $F(n) = 0$, which is even. Consider next the case $n = 1$, which is not divisible by 3 while $F(n) = 1$, which is not even.

For the induction step, let k be any natural number ≥ 2 and suppose (induction hypothesis) that for every $j < k$, $F(j)$ is even iff j is divisible by 3; we need to show that $F(k)$ is even iff k is divisible by 3. We show the two implications separately.

*S*uppose that k is divisible by 3; we need to check that $F(k)$ is even. By the condition of the case, neither of $k - 1$, $k - 2$ is divisible by 3 so by the induction hypothesis $F(k - 1)$, $F(k - 2)$ are both odd, so $F(k - 1) + F(k - 2) = F(k)$ is even.

*S*uppose that k is not divisible by 3; we need to check that $F(k)$ is odd. By the condition of the case, exactly one of $k - 1$, $k - 2$ is divisible by 3 so by the induction hypothesis one of $F(k - 1)$, $F(k - 2)$ is even while the other is odd, so $F(k - 1) + F(k - 2) = F(k)$ is odd.

(b) There is no natural number j less than a with $a \leq j$ so, vacuously, every natural number j less than a with $a \leq j < a$ has the property (whatever that property is) so, given the induction step, a itself also has the property.

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, we can often 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:

$$\begin{aligned} &\textit{Basis}: \ m \cdot 0 = 0 \\ &\textit{Recursion step}: \ m \cdot (n+1) = (m \cdot n) + m \end{aligned}$$

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 intuitively 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, which is how things are done in what is called second-order arithmetic. 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.

Solution

(a) Basis: $m^0 = 1$; recursion step: $m^{n+1} = m^n \cdot m$.
(b) In both cases the basis puts $f(m,0) = a$ where a is a specified natural number, while the recursion step puts $f(m,n + 1) = g(f(m,n), m)$ where g is a two-place function assumed to be already available. *End of solution*.

When a two-argument function is defined in this way, then inductive proofs for it will tend to follow the same lines as before, 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 can be given the following definition, due to Rósza Péter.

$$A(0,n) = n+1$$
$$A(m,0) = A(m-1,1) \text{ for } m > 0$$
$$A(m,n) = A(m-1,A(m,n-1)) \text{ for } m,n > 0$$

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 which are, roughly speaking, functions that can be defined by recursions of a fairly simple syntactic kind. 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 us here us is the way in which the second and third clauses of the definition makes the value of the Ackermann function $A(m,n)$ depend on its value for the first argument diminished by 1, but paired with a value of the second argument *larger than n*—larger by one in the case of the second clause and, after a few steps, very much larger for the third. 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 in Sect. 4.7.

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 distinguishing feature that marks them both as parts of the recursion step is the reappearance of A on the right. The second clause is recursive in so far as the first argument is concerned, although it is a basis for the second argument; the third clause is recursive for both arguments. Quite a subtle pattern.

The Ackermann function illustrates dramatically the difference that can arise between calculating bottom-up or top-down, in the sense explained at the end of Sect. 4.4. For the first few values of m, n the two procedures are of about the same efficiency, but after a while the top-down approach does better and better. The reason is that a bottom-up calculation of $A(m,n)$ without clever short-cuts will in general require calculating values of the function for many values of the arguments that are not really needed for the task.

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 the versions for the natural numbers but may also be used, very conveniently, without ever mentioning any kind of number. 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 of 'unique decomposability' is satisfied.

4.6.1 Defining Sets by Structural Recursion

In earlier chapters we introduced the notions of the image and the closure of a set under a relation (Chap. 2) or function (Chap. 3). We now make intensive use of those notions. We begin by recalling their definitions, generalizing a little from binary (i.e. two-place) relations to relations of any finite number of places.

Let X 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 X under an $(m + 1)$-place relation R (where $m \geq 1$) is defined by putting $y \in R(X)$ iff there are $x_1, \ldots, x_m \in X$ with $(x_1, \ldots, x_m, y) \in R$. In the case that R is an m-argument function f, this is equivalent to saying: $y \in f(X)$ iff there are $x_1, \ldots, x_m \in X$ with $y = f(x_1, \ldots, x_m)$. This definition is not recursive; that comes with the closure $R[X]$. We saw that it can be defined in either of two ways which we recall, leaving mention of the local universe implicit.

The *way of union* (bottom up) defines $R[X]$ recursively a sequence of sets indexed by the natural numbers and then takes their union:

$$X_0 = X$$
$$X_{n+1} = X_n \cup R(X_n), \text{for each natural number } n$$
$$R[X] = \cup \{X_n : n \in \mathbf{N}\}.$$

The *way of intersection* (top down) dispenses with numbers altogether. It puts $R[X] = \cap \{Y: X \subseteq Y \supseteq R(Y)\}$. In other words, it defines $R[X]$ as the intersection of the collection of all those sets Y such that both $X \subseteq Y$ and $R(Y) \subseteq Y$. When $R(Y) \subseteq Y$ holds, it is often convenient to say that Y is *closed under* R; in that terminology, the top-down definition says that $R[X]$ is the intersection of the collection of all those sets $Y \supseteq X$ that are closed under R.

Exercise 4.6.1 (1) asks you to check out some important facts about closure that were stated without proof in Chap. 2 Sect. 2.7.2. They are quite challenging, but well worth the effort of mastering.

Exercise 4.6.1 (1)

(a) Verify that the intersection of all sets $Y \supseteq X$ that are closed under R is itself closed under R.

(b) Writing $X^{\cup}$ as temporary notation for $R[X]$ defined bottom-up and $X^{\cap}$ as temporary notation for $R[X]$ defined top-down, show that $X^{\cup} = X^{\cap}$.

Solution

(a) Let $x_1, \ldots, x_m \in \cap \{Y: X \subseteq Y \supseteq R(Y)\}$ and suppose $(x_1, \ldots, x_m, y) \in R$; we need to show that $y \in \cap \{Y: X \subseteq Y \supseteq R(Y)\}$. Take any set Y with $X \subseteq Y \supseteq R(Y)$; it suffices to show that $y \in Y$. But since $x_1, \ldots, x_m \in Y$ and $(x_1, \ldots, x_m, y) \in R$ we have $y \in R(Y)$ and so, since $R(Y) \subseteq Y$, we have $y \in Y$ as desired.

(b) To show that $X^{\cup} = X^{\cap}$ it is best to check the two inclusions $X^{\cup} \subseteq X^{\cap}$ and $X^{\cap} \subseteq X^{\cup}$ separately.

(i) For $X^{\cup} \subseteq X^{\cap}$ it suffices to show that each $X_n \subseteq A^{\cap}$, which we do by induction on n. For the basis, we need to check that $X_0 \subseteq X^{\cap}$. But $X_0 = X$, and $X^{\cap}$ is an intersection of sets all of which include X so that $X_0 = X \subseteq X^{\cap}$. For the induction step, suppose that $X_k \subseteq X^{\cap}$; we need to show that $X_{k+1} \subseteq X^{\cap}$. Now, $X_{k+1} = X_k \cup R(X_k)$, so it suffices to show that $R(X_k) \subseteq X^{\cap}$, and so in turn it suffices to show that $R(X_k) \subseteq Y$ for each of the sets Y that are intersected to form $X^{\cap}$. Take any one such set Y. We know by the induction hypothesis $X_k \subseteq X^{\cap}$, so $X_k \subseteq Y$ so clearly $R(X_k) \subseteq R(Y)$. We also know from the definition of $X^{\cap}$ that $R(Y) \subseteq Y$. Putting these together gives $R(X_k) \subseteq Y$ as desired.

(ii) For $X^{\cap} \subseteq X^{\cup}$, it suffices to show that $X^{\cup}$ is one of the Y that are intersected to form $X^{\cap}$; in other words, it suffices to show that $X \subseteq X^{\cup}$ and $R(X^{\cup}) \subseteq X^{\cup}$. The first is immediate from $X = X_0 \subseteq \cup\{X_n : n \in \mathbf{N}\} = X^{\cup}$. For the second, suppose $x_1, \ldots, x_m \in X^{\cup}$ and $(x_1, \ldots, x_m, y) \in R$; we need to show that $y \in X^{\cup}$. Since $x_i \in X^{\cup} = \cup\{X_n: n \in \mathbf{N}\}$ we know that for each x_i with $1 \leq i \leq m$ there is a natural number $n(i)$ with $x_i \in X_{n(i)}$. Let k be the greatest of the numbers $n(1), \ldots, n(m)$. Since the sets $X_0, X_1, X_2, \ldots$ form a chain under inclusion, this implies that $x_i \in X_k$ for all i with $1 \leq i \leq m$. Hence $y \in R(X_k) \subseteq X_{k+1} \subseteq \cup\{X_n: n \in \mathbf{N}\} = X^{\cup}$. *End of solution.*

Having shown that $X^{\cup} = X^{\cap}$, we can now leave aside the temporary notation and refer to the set again as $R[X]$, or even more briefly as X^{+} when the identity of the relation R is understood. Evidently, the definition of X^{+} can also be extended to cover an arbitrary collection of relations, rather than just one. The *closure* X^{+} of X under a *collection* $\{R_i\}_{i \in I}$ of relations is defined to be the intersection of all sets $Y \supseteq X$ that are closed under all the relations in the collection.

This brings us, at last, to the notion of definition by structural recursion. A set is *defined by structural recursion* whenever it is introduced as the closure X^{+} of some set X (referred to as the *basis*, or initial set of the definition) under some collection $\{R_i\}_{i \in I}$ of relations (often called the *constructors* or *generators* of X^{+}).

In this context, it is sometimes intuitively helpful to think of each $(m + 1)$-place relation R as a *rule* that authorizes us to pass from items $a_1, \ldots, a_m$ in X^{+} to an item $y \in X^{+}$ whenever $(a_1, \ldots, a_m, y) \in R$. In this vein, X^{+} is also described as the closure of A under a *collection of rules*.

We illustrate the idea of definition by structural recursion with five examples, the first two drawn from computer science, the next two from logic, and finally one from abstract algebra.

Example 1 The notion of a *string*, already mentioned informally Chap. 3 Sect. 3.6.2, is of central importance for formulating many ideas of computer science, for instance the theory of finite state machines. Let X be any alphabet, consisting of elementary signs. Let λ be an abstract object, distinct from all the letters in X, which is understood to serve as the empty string. The set of all strings over X, conventionally written as X^*, is the closure of $X \cup \{\lambda\}$ under the rule of concatenation, that is, under the operation of taking two strings s, t and forming their concatenation by writing s immediately followed by t.

Example 2 We can define certain *specific kinds of string* by structural recursion. For instance, a string over an alphabet X 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 X 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 of palindromes to be the *closure* of the set $\{\lambda\} \cup X$ under the rule permitting passage from a string s to a string xsx for any $x \in X$. In other words, it is the least set Y including $\{\lambda\} \cup X$ such that $xyx \in Y$ for any $y \in Y$ and $x \in X$.

Example 3 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 X under some operations. In this case, X 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 Y^* of all strings in the alphabet $Y = X \cup \{\neg,\wedge,\vee\} \cup \{(,)\}$. In case you have trouble reading the last bit, $\{(,)\}$ is the set consisting of the left and right parentheses.

Example 4 The set of *theorems* of a formal logical system is also defined by structural recursion. It is the closure X^+ of some initial set X 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 that is very often used in this context is modus ponens (also known as *detachment* or *$\rightarrow$-elimination*), permitting passage from formulae α and $\alpha \rightarrow \beta$ to the formula β.

Example 5 Algebraists also use this kind of definition, even though they are not, in general dealing with strings of symbols. For example, if X is a subset of an algebra, then the *subalgebra generated by X* has as its carrier (i.e. underlying set) the *closure* X^+ of X 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 X_0, X_1, X_2, … in the generation of the closure by the way of union, and replace the structural definition by one that makes a cumulative recursion on those indices. Indeed, that is a fairly common style of presentation, and in some contexts has its advantages. But in other cases, it is more convenient to dispense with numerical indices and carry out the recursive definition in perfectly set-theoretic terms using the top-down formulation.

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 certain designated sets. There are many ways of doing it. For example, we can take **N** to be the least collection that contains the empty set $\varnothing$ and is closed under the operation taking each set Y to $Y \cup \{Y\}$. 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 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)
Define by structural recursion the set of even palindromes over an alphabet X, i.e. the palindromes with an even number of letters.

Solution
The set of even palindromes over X is the closure of $\{\lambda\}$ under the same rule as used for palindromes in general, i.e. passage from a string s to a string xsx for any $x \in X$.

4.6.2 Proof by Structural Induction

We have seen that the procedure of defining a set by structural recursion, i.e. as the closure of a set under given relations or functions, is pervasive in computer science, logic and abstract algebra. Piggy-backing on that mode of definition is a mode of demonstration that we will now examine—proof by structural induction.

Let X be a set of any items whatsoever, let X^+ be the closure of X under a collection $\{R_i\}_{i\in I}$ of relations. Consider any property that we would like to show holds of all elements of X^+. We say that a relation R *preserves the property* iff whenever $x_1, \ldots, x_m$ have the property and $(x_1, \ldots, x_m, y) \in R$, then y also has it. When R is a function f, this amounts to requiring that whenever $x_1, \ldots, x_m$ has the property then $f(x_1, \ldots, x_m)$ also has the property.

The *principle of proof by structural induction* may now be stated as follows. Again, let X be a set, and X^+ the closure of X under a collection $\{R_i\}_{i\in I}$ of relations. To show that every element of X^+ has a certain property, it suffices to show two things:

Basis: Every element of X 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 X^+. Let P be the set of all items that have the property in question. Suppose that both basis and induction step hold. By the basis, $X \subseteq P$. Since the induction step holds, P is closed under the relations R_i. Hence by the definition of X^+ as the *least* set with those two features, we have $X^+ \subseteq P$, i.e. every element of X^+ 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 from the last exercise, that the set of even palindromes over a set X of letters is the closure of $\{\lambda\}$ under passage from a string s to a string xsx where $x \in X$. So we need only show two things: that λ has the property in question, and that whenever s has it then so does xsx for any $x \in X$. 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.

The next exercise will be meaningful to those who have already done a little bit of propositional logic. Others should return to it when doing the end-of-chapter exercises for Chap. 8.

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 on a numerical measure of the depth of formulae, what would be a suitable measure of depth?

(c) Let X be any set of formulae of propositional logic, and let R be the derivation rule of detachment defined in Example 4 above. Use structural induction to show that every formula in the closure X^+ of X under R is a sub-formula of some formula in X.

(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.

Solution

(a) It suffices to show (basis) that the set of proposition letters has the property, and (induction step) that application of the connectives to build formulae $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$ out of formulae α, β preserves the property. The basis is immediate from the fact that proposition letters contain no brackets. For the induction step, suppose that the number of left brackets in α equals the number of right brackets in α, and likewise for β. Then the formulae $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$ each add one more left bracket and one more right bracket so that the number of left brackets continues to equal the number of right ones.

(b) Put $depth(p) = 0$ for every proposition letter p, $depth((\neg\alpha)) = depth(\neg\alpha) + 1$, $depth((\alpha \wedge \beta)) = max(depth(\alpha), depth(\beta)) + 1$. In effect, we are here defining *depth* as a function on the set of formulae of propositional logic into the set of the natural numbers, by structural recursion on the domain. You might have been tempted to use addition in the recursion step; that cannot be said to be wrong, but it is less simple and less useful than using *max*. By the way, *depth* is also often called *length*.

(c) For the basis we need only observe that every formula is a sub-formula of itself. For the induction step, suppose that each of the formulae α, $\alpha \rightarrow \beta$ is a sub-formula of some formula $\chi \in X$. We need to show that β is a sub-formula of some formula in X. But β is a sub-formula of $\alpha \rightarrow \beta$, and any sub-formula of a sub-formula of χ is itself a sub-formula of χ, so we are done.

(d) Let v be the assignment that puts $v(p) = 1$ for every sentence letter p. We verify that $v^+(\alpha) = 1$ for every formula α constructed from sentence letters using only the connectives $\wedge,\vee,\rightarrow,\leftrightarrow$. *Basis*. If α is a sentence letter p, then $v^+(\alpha) = v(p) = 1$ by the definition of the assignment v. *Induction Step*. Suppose $v^+(\beta) = 1 = v^+(\beta)$; we need to show that $v^+(\alpha \wedge \beta) = v^+(\alpha \vee \beta) = v^+(\alpha \rightarrow \beta) = v^+(\alpha \leftrightarrow \beta) = 1$. But this is immediate from the truth-tables for those connectives (recall the boxes in Chap. 1).

Alice Box: What property should I induce on?

Alice In Exercise 4.6.2 (d) we are asked to show that no formula of the kind specified is a contradiction, in other words, that every such formula has the property that *there is some* assignment v such that v

(α) = 1. But the solution is not carried out in terms of that property; instead, it worked with the property that $v(\alpha) = 1$ for a *certain specific* assignment, namely the one putting $v(p) = 1$ for sentence letters p. Why?

Hatter Because the inductive argument would have snagged in the induction step, due to the existential nature of the property under consideration.

Alice I don't understand.

Hatter Look at the case for conjunction. Suppose, as induction hypothesis, that *there is* an assignment v with $v^+(\alpha) = 1$ and that *there is* an assignment u with $u^+(\beta) = 1$. Nothing tells us that $v = u$, so we have no guarantee that $v^+(\alpha \wedge \beta) = 1$, nor that $u^+(\alpha \wedge \beta) = 1$. To unblock the argument we must work with the same assignment throughout the induction step.

Alice So we are actually proving something a bit stronger than what was requested?

Hatter Indeed we are. For inductive proofs, proving more can sometimes be easier than proving less. It happens quite often when we want to establish an existential property by induction, as in the exercise. It sometimes happens when we want to establish a conditional property; we may need to articulate a suitable biconditional that implies it and establish the latter by induction.

4.6.3 Defining Functions by Structural Recursion on Their Domains

There is an important difference between the examples in parts (a), (c) of Exercise 6.6.2. In the recursive definition of the set of formulae of propositional logic, we begin with a set X of proposition letters, and the closing functions (forming negations, conjunction s, disjunctions) always produce strictly longer formulae, so always giving us something fresh. But an application of the closing three-place relation (which is, indeed, a two-place function) of detachment to input formulae α, $\alpha \rightarrow \beta$ gives us the formula β, which is shorter than one of the two inputs. As a result, detachment *is not guaranteed always to give us something fresh*: β may already be in the initial set X or already available at an earlier stage of the closing process.

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 very frequently.

Suppose that we want to give a recursive definition of a function f whose domain is the closure X^+ of a set A under a function g. For example, we might want to define $f\colon X^+ \to \mathbf{N}$, as some kind of measure of the depth or complexity of the elements of the domain X^+, by setting as the basis $f(x) = 0$ for all $x \in X$ and, as the recursion step, putting $f(g(x)) = f(x) + 1$ for all $x \in X^+$. Is this legitimate?

Unfortunately, not always! If the function g is not injective, then we will have two distinct x, x' with $g(x) = g(x')$ yet it may be that the recursion has already given us $f(x) \neq f(x')$ so that $f(x) + 1 \neq f(x') + 1$, in which case the recursion step does not give $f(g(x))$ a unique value. Even if g is injective, we may still be in trouble. For $g(x)$ may already be an element $x' \in X$, so that $f(g(x))$ is defined as 0 by the basis of the definition but as $f(x) + 1 > 0$ by the recursion clause, again lacking a unique value so that f fails to be well-defined.

To illustrate this second eventuality, let $g\colon \mathbf{N} \to \mathbf{N}$ 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 $X = \{0\}$. Then the closure X^+ of X under g is the set $\{0, \ldots, 99\}$. Now suppose that we try to define a function $f\colon X^+ \to \mathbf{N}$ by structural induction putting $f(0) = 0$ and $f(g(n)) = f(n) + 1$ for all $n \in X^+$. The recursion step is fine for values of $n < 99$, indeed we get $f(n) = n$ for all $n < 99$. But it breaks down at $n = 99$. It tells us that $f(g(99)) = f(99) + 1 = 99 + 1 = 100$, 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 X be a set and X^+ its closure under a collection $\{g_i\}_{i \in I}$ of functions. An element x of X^+ is said to be *uniquely decomposable* iff either: (1) $x \in X$ and is not in the range of any of the functions g_i, or (2) $x \notin X$ and $x = g_i(x_1, \ldots, x_m)$ for a unique function g_i in the collection and a unique tuple $x_1, \ldots, x_m \in X^+$. Roughly speaking, these two conditions together guarantee that there is a unique way in which x can have got into X^+. When the elements of X^+ are symbolic expressions of some kind, this property is also called *unique readability*.

Unique decomposability/readability suffices to guarantee that a structural recursion on X^+ succeeds, i.e. that it defines a unique function with X^+ as domain. To be precise, we have the following *principle of structural recursive definition*: let X be a set, $\{g_i\}_{i \in I}$ a collection of functions, and X^+ the closure of X under the functions. Suppose that every element of X^+ is uniquely decomposable. Let V be any set (we use this letter because it is often convenient to think of its elements as

values) and let f: $X \to V$ be a given function on X into V. Then, for every collection $\{h_i\}_{i \in I}$ of functions h_i on powers of V into V with arities corresponding to those of the functions g_i, there is a unique function f^+: $X^+ \to V$ satisfying the following recursively formulated conditions.

Case	Conditions
Basis: $x \in X$	$f^+(x) = f(x)$
Recursion step: $x = g_i(x_1, \ldots, x_k)$ is the unique decomposition of x	$f^+(x) = h_i(f^+(x_1), \ldots, f^+(x_k))$

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: $X \to V$ homomorphically to a function f^+: $X^+ \to V$ if every element of X^+ 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 we close a singleton under a single 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 Fig. 4.1.

In the diagram, the left set is generated from the singleton $A = \{a\}$ by the single one-place function g, so we write it as A^+. 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 at most one arrow (and hence, since A^+ is the closure of $\{a\}$ under g, by exactly one arrow). With these conditions satisfied, the function f^+ in the diagram is well-defined.

Is the unique decomposition/readability condition an obstacle, in practice, to defining functions recursively on recursively defined domains? The good news is that when we look at examples in computer science and logic, particularly those

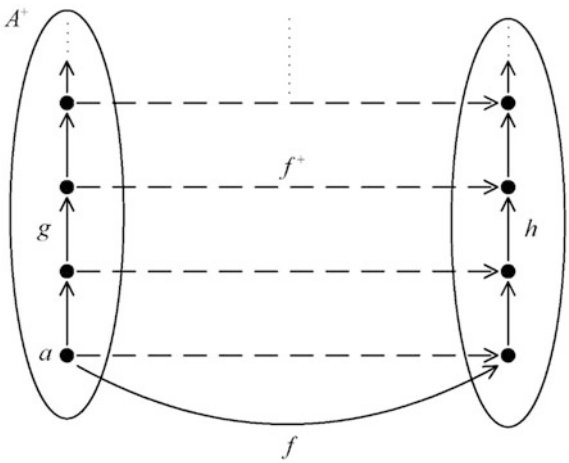

Fig. 4.1 A recursive structural definition

dealing with symbolic expressions, the condition is often satisfied. Moreover, the applications of the principle 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 X, and let X^+ 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 Chap. 3, Sect. 3.6.2). We might define the *length* of a list recursively as follows, where $\langle\rangle$ is the empty list:

Basis: $length(\langle\rangle) = 0$
Recursion 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 we could get around the failure of unique readability by showing that nevertheless whenever $ts = t's'$ then $length(t) + length(s) = length(t') + length(s')$ even when $length(t) \neq length(t')$ and $length(s) \neq length(s')$. But that is a bit of a bother and in other examples such a patch may not be available.

For another example of appeal to the unique readability condition, consider the definition of the *depth* of a formula of propositional logic. Suppose our formulae are built up using just negation, conjunction and disjunction. We proceed as in the solution to Exercise 4.6.2 (a), where we took it for granted that the depth function, introduced by the following recursion, is well defined. *Basis*: $depth(p) = 0$ for any elementary letter p. *Recursion step*: If $depth(\alpha) = m$ and $depth(\beta) = n$ then $depth(\neg\alpha) = m + 1$ and $depth(\alpha \wedge \beta) = depth(\alpha \vee \beta) = max(m,n) + 1$. The principle of structural recursive definition guarantees that the function is indeed well-defined. This is because the formulae of propositional logic have unique decompositions under the operations of forming negations, conjunctions, and disjunctions, *provided* they are written with suitable bracketing. Indeed, we can say that the mathematical purpose of bracketing is to ensure unique decomposition.

Exercise 4.6.3

(a) Suppose we forget brackets from formulae of propositional logic, without introducing any conventions about the cohesive powers of connectives. Show that the depth of the expression $\neg p \wedge \neg q$ is not well defined.

(b) In propositional logic, to substitute a formula φ for a sentence letter p in a formula α is, intuitively, to replace all the occurrences of p in α by φ. Fix a formula φ and a proposition letter p, and define this operation by recursion on α, as a function σ: $F \rightarrow F$, where F is the set of formulae. For simplicity, assume that the primitive connectives of the propositional logic are just $\neg$, $\wedge$.

Solution

(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, one as the conjunction $(\neg p \wedge \neg q)$ of two negated propositional letters, the other as the negation $\neg(p \wedge \neg q)$ of the conjunction of a letter with the negation of a letter. The depth of the former is 2 but the depth of the latter is 3. So, brackets are needed for unique readability of formulae of propositional logic (as also for expressions of elementary algebra). Nevertheless, in Chap. 7 Sect. 7.4 we will see a trick for writing formulae in a different way that restores unique readability without brackets!

(b) The basis is $\sigma(q) = \alpha$ when q is a proposition letter. The recursion step has two cases. Case 1: $\alpha = \neg\beta$: put $\sigma(\alpha) = \neg\sigma(\beta)$. Case 2: $\alpha = \beta \wedge \gamma$: put $\sigma(\alpha) = \sigma(\beta) \wedge \sigma(\gamma)$. Or just say briefly: $\sigma(\neg\beta) = \neg\sigma(\beta)$, $\sigma(\beta \wedge \gamma) = \sigma(\beta) \wedge \sigma(\beta)$.

4.7 Recursion and Induction on Well-Founded Sets

The notion 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 that domain. 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, i.e. that the relation is also complete, cf Chap. 2 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.

- We recall from Chap. 2 Sect. 2.6.1 that 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 at least one minimal element. The '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. There must be no non-empty subset of W that lacks a minimal element under $<$. Thus, the requirement is very demanding.

The definition can also be put in a quite different but nevertheless equivalent way. Again, 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 not 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. The argument is short, but at the same time subtle. The second part of it, which looks so simple, implicitly calls on the axiom of choice (Chap. 3 Sect. 3.5.5), appeal to which cannot be avoided.

To help appreciate the contours of the notion of a well-founded set, 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, Chap. 2 Sect. 2.6.1 again). This contrasts with the set $\mathbf{Z}$ of all integers, which is not well-founded under the customary ordering since it has subsets (for instance, the set of negative integers, or indeed the whole of $\mathbf{Z}$) that do not have a minimal element.

Exercise 4.7.1 (1)

(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 $\mathrm{P}(A)$ is well-founded under the relation of proper set inclusion.
(e) Can you find a subset of $\mathbf{N}$ that is not well-founded under the usual relation $<$?

Solution

(a) No. The whole set does have a minimal element, namely 0, but it has plenty of non-empty subsets without a minimal element. Consider, for example, the subset consisting of all rationals greater than zero.
(b) Yes, vacuously, since it has no non-empty subsets.
(c) We induce on the size of the set. We know from (b) that the empty set is well-founded under any such relation, and it is immediate that any unit set is too. For the induction step, let $k > 2$ and suppose that every finite set of size less than k is well-founded under any transitive irreflexive relation over it. Let A be a set with $\#(A) = k$ and let $<$ be a transitive irreflexive relation over it; we want to show that A is well-founded under $<$. Let B be any non-empty subset of A; we need to show that B has a minimal element under $<$. If $B \neq A$ then $\#(B) < k$ and so by the induction hypothesis has a minimal element. It remains only to show that A itself has a minimal element under $<$. Choose any element $a \in A$ and consider $B = A\backslash\{a\}$. Then $\#(B) = k - 1 \neq 0$ since $k > 2$, so by the induction hypothesis B has a minimal element b. On the one hand, if $a \not< b$ then b is also a minimal element of A. On the other hand, if $a < b$ then, using

the transitivity and irreflexivity of < (we need both), a must be a minimal element of A.

(d) Since A is finite, so is P(A). Thus, since proper inclusion is both transitive and irreflexive, 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. Quite generally, any subset of a well-founded set is well-founded under the same relation.

Exercise 4.7.1 (2)

(a) Show that the collection P(**N**) of all subsets (finite or infinite) of **N** is not well-founded under proper inclusion. *Hint*: Use the definition in terms of infinite descending chains.

(b) Let A be any infinite set. Show that (i) the collection $P_f(A)$ of all its *finite* subsets is infinite, but (ii) is well-founded under proper inclusion.

Solution

(a) Consider for example the infinite descending chain $\mathbf{N} \supset \mathbf{N}\backslash\{0\} \supset \mathbf{N}\backslash\{0,1\} \supset \mathbf{N}\backslash\{0,1\} \supset \ldots$ of subsets of **N**, i.e. elements of P(**N**).

(b) For (i) consider for example all the sets $A_n = \{0, \ldots, n\}$ for $n \in \mathbf{N}$. There are infinitely many such sets, even though each of them is finite, and they are all subsets of **N** so are all elements of $P_f(A)$. For (ii), consider any descending chain $\ldots \subset X_2 \subset X_1 \subset X_0$ of elements of $P_f(A)$. Now X_0 is finite, with say k elements. Since the relation is proper inclusion, each step down the chain loses at least one element. Hence the chain can have at most k elements, so is not infinite. *End of solution*.

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.

N is clearly well-ordered under the usual < relation. However, a well-ordered set can be 'longer' than **N** in a natural sense. 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 $\{0, 1, 2, \ldots; -1, -2, -3, \ldots\}$, then this is well-ordered in the order of listing, notwithstanding the fact that there is evidently a bijection between this set (considered without the ordering) and **N** itself. Clearly the 'lengthening' operation can be repeated as many times as we like. It opens the door to the theory of transfinite ordinal numbers, but we stop just short of going through the portal to that heavenly domain.

Exercise 4.7.1 (3)

(a) Re-order **N** itself in such a way as to obtain a relation that well-orders it that is longer than that that given by $<$, in the same natural sense as in the text. Don't do anything complicated.
(b) Show how, from any non-empty well-ordered set we may form one that is not well-ordered but is still well-founded, by adding a single element.

Solution

(a) Take the odds in their usual order, followed by the evens in their usual order, giving us the set $\{1, 3, 5, \ldots; 2, 4, 6, \ldots\}$.
(b) Let W be any non-empty set that is well ordered by a relation $<$. Add to W one new element a while keeping $<$ unchanged, so that a does not have the relation $<$ to anything in W, nor to itself. Irreflexivity and transitivity still hold for $<$ over the enlarged set $W \cup \{a\}$ and there are no infinitely descending chains, so $<$ well-founds $W \cup \{a\}$. But it does not well-order $W \cup \{a\}$, because it does not satisfy the completeness (aka linearity) condition that for all $x, y, \in W \cup \{a\}$ either $x = y$ or $x < y$ or $y < x$.

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 following *principle of induction over well-founded sets*. 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 an abstraction on the principle of cumulative induction, from the specific order that we are familiar with over the natural numbers, to any well-founded relation over any set whatsoever. Of course, we could write in a basis, which would say "every minimal element of W itself has the property in question" but it would be redundant and is customarily omitted.

Exercise 4.7.2 (1)

(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 $b < a$. *End of solution.*

The proof of the principle of induction over well-founded sets is remarkably brief, considering its power; the principle is, in effect, an immediate consequence of the definition of well-founded sets. In detail, 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 under $<$. 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, but there is something about it 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 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 change anything. That is because irreflexivity is implied, anyway, by the minimal-element condition: if $a < a \in A$ then the singleton $\{a\} \subseteq A$ 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. Since applications of the principle almost always work with a transitive relation, we rest content with the formulation given above.

Exercise 4.7.2 (2)
Show that a well-founded relation is always acyclic (defined in Chap. 2, end-of-chapter Exercise 2.3).

Solution
Let W be a set and $<$ a well-founded relation over W. Suppose for *reductio* that $<$ is not acyclic. Then there are $a_1, \ldots, a_n \in W$ $(n \geq 2)$ such that each $a_i < a_{i+1}$ $(i < n)$ and $a_n = a_1$. From here we can argue in either of two different ways. *First way*: By transitivity, $a_1 < a_n = a_1$ contradicting irreflexivity of $<$. *Second way*: Form an infinite descending chain $\ldots\ a_1 < \ldots < a_n = a_1 < \ldots < a_n$ by repeating $a_1 < \ldots < a_n$ infinitely many times downwards, contradicting well-founding.

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 (Sect. 4.5) is well-defined. Those not so confident may skip directly to the next section. We recall the recursive definition of the Ackerman function.

$$A(0,n) = n+1$$
$$A(m,0) = A(m-1,1) \text{ for } m > 0$$
$$A(m,n) = A(m-1, A(m,n-1)) \text{ for } m,n > 0$$

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 $A = \{n: 0 \leq n \leq 25\}$ and identify these with the letters of the English alphabet. The lexicographic order of A^2 then corresponds to its dictionary order.

Now, the lexicographic relation $<$ is irreflexive, transitive, and then that it well-founds $\mathbf{N^2}$ (Exercise 4.7.3 (a) below). Knowing that, we can 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 $\mathbf{N^2}$.

Solution

(a) For irreflexivity, we need to show that $(m,n) \not< (m,n)$. This is immediate from the definition since both $m \not< m$ and $n \not< n$.

For transitivity, suppose $(m,n) < (p,q) < (r,s)$; we need to show that $(m,n) < (r, s)$. We break the argument into four cases. *Case 1*: Suppose $m < p$ and $p < r$. Then $m < r$ and so immediately $(m,n) < (r,s)$. *Case 2*. Suppose $m < p$ but $p \not< r$. Using the latter, $p = r$. So $m < r$ and again $(m,n) < (r,s)$. *Case 3*. Suppose $m \not< p$ but $p < r$. By the former, $m = p$. So $m < r$ and again $(m,n) < (r,s)$. *Case 4*. Suppose $m \not< p$ and $p \not< r$. From the former, $m = p$ and $n < q$, while using the latter, $p = r$ and $q < s$. So $m = r$ and $n < s$, again yielding $(m,n) < (r,s)$.

For well-founding, consider any non-empty subset S of $\mathbf{N^2}$; we need to show that it has a minimal element. Since S is non-empty, the set A of all $m \in \mathbf{N}$ such that there is an $n \in \mathbf{N}$ with $(m,n) \in S$, is non-empty. Since $\mathbf{N}$ is well-founded under $<$, A has a minimal element a_0. Then the set B of all $n \in \mathbf{N}$ with $(a_0,n) \in$

S, is non-empty, and so has a minimal element b_0. Clearly, (a_0,b_0) is a minimal element of S under the lexicographic ordering.

(b) To show linearity, let (m,n), $(p,q) \in \mathbf{N^2}$. Suppose $(m,n) \not< (p,q)$ and $(p,q) \not< (m,n)$; we need to show that $(m,n) = (p,q)$. By the former supposition, $m \not< p$ and either $m \neq p$ or $n \not< q$. By the latter supposition, $p \not< m$ and either $m \neq p$ or $q \not< n$. Since $m \not< p$ and $p \not< m$ we have by the linearity of $<$ over $\mathbf{N}$ that $m = p$. So $n \not< q$ and $q \not< n$ and thus $n = q$. Putting this together, $m = p$ and $n = q$ so $(m,n) = (p,q)$ as desired.
That $<$ well-orders $\mathbf{N}^2$ is immediate from the facts, just established, that it is irreflexive, transitive, well-founded and linear over $\mathbf{N}^2$.

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 precise 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. The definition 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 will 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 definite program, i.e. when does it 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 itself may suffice to guarantee that the function is well-defined. Hopefully, those who designed the language will have proven that to be the case, using inductive arguments and before making the language available to the user. If, on the other hand, we are sketching the program informally, the question of whether we have succeeded in specifying a unique function needs to be analysed by expressing it in an explicitly recursive form and verifying that it always terminates after a finite number of steps with a unique value. Typically, this will involve constructing suitable relations over appropriate sets of items, verifying that they are well founded, and checking that the definitions satisfy the conditions for applying the principle of recursive definition on well-founded sets.

When the recipe does give us a function, other questions remain. One vital question is whether the unique output fulfils requirements, in other words, is what it should be. Another important question is of course how feasible it is, that is, how much in time and other resources does it require.

Answers to all these questions require demonstration, and the proofs will typically involve inductive arguments of one kind or other, sometimes of great complexity.

4.8 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, $\Sigma\{2^i: 0 \leq i \leq n\} = 2^{n+1} - 1$.

(d) Use simple induction to show that any postage cost of four or more pence can be covered by two-penny and five-pence stamps.

Solution

(a) *Basis*. We need to check that $5^1 - 1$ is divisible by 4, which holds since $5^1 - 1 = 4$. *Induction step*. Suppose $5^k - 1$ is divisible by 4, i.e. $5^k = 4n + 1$ for some natural number n. We need to show that $5^{k+1} = 4m + 1$ for some natural number m. But $5^{k+1} = 5 \cdot 5^k = 5(4n + 1)$ (by the induction hypothesis) $= 20n + 5 = 20n + 4+1 = 4(5n + 1) + 1 = 4m + 1$, taking $m = 5n + 1$.

(b) *Basis*. We need to check that $1^3 - 1$ is divisible by 3, which holds since $1^3 - 1 = 0$. *Induction step*. Suppose $k^3 - k$ is divisible by 3; we need to show that $(k + 1)^3 - (k + 1)$ is divisible by 3. But $(k + 1)^3 - (k + 1) = k^3 + 3k^2 + 3k + 1 - k - 1 = (k^3 - k) + 3(k^2 + k)$. The induction hypothesis tells us that the first summand is divisible by 3; evidently so is the second one.

(c) For the basis, we need to show that $2^0 = 2^1 - 1$, which is immediate since both sides equal 1. For the induction step, suppose that $2^0 + \ldots + 2^k = 2^{k+1} - 1$. We need to show that $2^0 + \ldots + 2^{k+1} = 2^{k+2} - 1$. But LHS $= (2^0 + \ldots + 2^k) + 2^{k+1} = 2^{k+1} - 1 + 2^{k+1}$ (by induction hypothesis) $= 2 \cdot 2^{k+1} - 1 = 2^{k+2} - 1$ as desired.

(d) One can prove this by a fairly straightforward cumulative induction if we begin by noting that every natural number $k \geq 4$ can be expressed in the form $k = 4n + m$, where $1 \leq n$ and $0 \leq m \leq 3$, and carrying out a simple induction on n.

Basis. Suppose $n = 1$. Then $k = 4 + m$ for some m with $0 \leq m \leq 3$. For $m = 0$ use two two-penny stamps; for $m = 1$, use a single 5-pence stamp; for $m = 2$, use three two-penny stamps; for $m = 3$, use a five-pence stamp and a two-penny stamp. *Induction step*. Suppose the property holds for n; we need to show it for $n + 1$, that is, when $k = 4(n + 1) + m$ with $0 \leq m \leq 3$. Now $4(n + 1) + m = (4n + m) + 4$. So we take the stamps needed to cover $4n + m$ and add two more two-penny stamps.

As remarked by Jorge de la Cal in class, problem 4 (1) (d) can also be solved without using induction. Let k be any positive integer with $k \geq 4$. Now k is either even or odd. If it is even then it is a multiple of 2, i.e. $k = 2n$ for some $n \geq 2$, so we can simply use n two-penny stamps. If k is odd then $k = 5 + 2m$ for some $m \geq 0$, so we can use a five-penny stamp plus n two-penny stamps.

Which way is better? In this example, the direct argument seems more elegant; in others it may be the reverse, while in yet further cases we may have one kind of proof available but not the other.

Exercise 4 (2) Definition by simple recursion

(a) Let f: $\mathbf{N}^+ \to \mathbf{N}$ 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.

(b) Consider the following attempted recursive definition of a function f: $\mathbf{N} \to \mathbf{Q}$ (where $\mathbf{Q}$ is the set of all rational numbers). *Basis*: $f(0) = 5/12$; *recursion step*: $f(n + 1) = [1/f(n)] - 2$. (i) Explain why one should be suspicious that this may not succeed in defining a function f: $\mathbf{N} \to \mathbf{Q}$. (ii) Confirm the suspicion for a small value of n.

Solution

(a) *Basis*: Put $f(1) = 0$. *Recursion step*. If $n + 1$ is not a power of 2, put $f(n + 1) = f(n)$. If $n + 1$ is a power of 2, put $f(n + 1) = f(n) + 1$.

(b) (i) One should be suspicious because if it ever happens that $f(n) = 0$ then $f(n + 1)$ is not well-defined since one cannot divide by zero. (ii) This happens, indeed, for the recursion step from $n = 3$ to $n = 4$.

Exercise 4 (3) Proof by cumulative induction

(a) Use cumulative induction to show that for every natural number n, $F(n) \leq 2^n - 1$, where F is the Fibonacci function.

(b) Express each of the numbers 14, 15, and 16 as a sum of 3 s and/or 8 s. 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.

(c) Show by induction that for every natural number n, $A(1,n) = n + 2$, where A is the Ackermann function.

Solution

(a) *Basis*: This holds for 0, 1 since $F(0) = 0 = 2^0 - 1$, $F(1) = 1 = 2^1 - 1$. *Induction step*: Let $n \geq 2$, and suppose that the property holds for all natural numbers $m < n$; we want to show that it holds for n. Now, $F(n) = F(n - 1) + F(n - 2) \leq (2^{n-1} - 1) + (2^{n-2} - 1)$ (by the induction hypothesis) $= (2 \cdot 2^{n-2}+2^{n-2}) - 2 = 3 \cdot 2^{n-2} - 2 < 4 \cdot 2^{n-2} - 2 = 2^n - 2 < 2^n - 1$ and we are done.

(b) $14 = 8 + 3 + 3$, $15 = 3 + 3 + 3 + 3 + 3$, $16 = 8 + 8$. Let $k \geq 14$, and suppose that all i with $14 \leq i < k$ may be expressed as the sum of 3 s and 8 s; we want to show that the same property holds of k. If $k = 14$, 15, or 16 we are done. If $k \geq 17$ then $14 \leq k\text{-}3 < k$, so by the induction hypothesis $k- 3$ has the desired property, so adding 3 we see that k also has that property.

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) Explain why the verification used for Exercise 4.6.2 (d) does not work if $\neg$ or + (exclusive disjunction) is allowed as a primitive connective.

(d) Show, by an argument similar in spirit to that for Exercise 4.6.2 (d), that in classical propositional logic, no formula built using propositional letters and connectives from the set $\{\wedge,\vee, +\}$, where + is exclusive disjunction, is a tautology.

Solution

(a) *Basis of the definition*. Every $a \in A$ (i.e. every element of the alphabet) is an odd palindrome. *Recursion step*. If x is an odd palindrome and $a \in A$ then the string axa is an odd palindrome.

(b) Recall the definition of the even palindromes. *Basis of the definition*. The empty string is an even palindrome. *Recursion step*. If x is an even palindrome and $a \in A$ then the string axa is an even palindrome. The proof piggy-backs on the definition. Call a string 'good' iff it is either empty or contains two contiguous occurrences of the same letter. We want to show that every even palindrome is good. *Basis*. The empty string is good, by the definition of goodness. *Induction step*. Suppose that x is good, and let $a \in A$; we want to show that the string axa is good. If x is the empty string, then $axa = aa$ which contains two contiguous occurrences of the same letter and so is good. If x is not the empty string, then since it is good it contains two contiguous occurrences of the same letter, so the extended string axa also contains two contiguous occurrences of the same letter and the induction is complete.

(c) The induction step snags in the cases for those connectives. Specifically, $v(\neg\alpha) = 0$ when $v(\alpha) = 1$, and $v(\alpha + \beta) = 0$ when $v(\alpha) = v(\beta) = 1$; in words, neither of the two connectives preserves truth.

(d) Let v be the assignment that puts $v(p) = 0$ for every sentence letter p. We can now show that $v^+(\alpha) = 0$ for every formula α constructed from sentence letters using only the connectives $\wedge,\vee,+$. *Basis*. If α is a sentence letter p, then $v^+(\alpha) = v(p) = 0$ by the definition of the assignment v. *Induction Step*. Suppose $v^+(\beta) = 0 = v^+(\beta)$; we need to show that $v^+(\alpha \wedge \beta) = v^+(\alpha \vee \beta) = v^+(\alpha + \beta) = 0$. But this is immediate from the truth-tables for those connectives (recall the boxes in Chap. 1).

Exercise 4 (5) Definition of functions by structural recursion on their domains
Consider any alphabet X and the set X^* of all strings made up from X. 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 using the operations *cons* and *con* (Chap. 3 Sect. 3.6.2).

Solution
Basis: Put $rev(\lambda) = \lambda$ and $rev(x) = x$ for all $x \in X$. *Recursion step*: Put $rev(cons(x, s)) = con(rev(s),x)$ for all $x \in X$, $s \in X^*$. We need *cons* on the left to ensure unique readability and *con* on the right since the first argument $rev(s)$ is not, in general, an element of the alphabet.

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 counter-example.

(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 sets W_1, W_2, define a well-ordering of their Cartesian product $W_1 \times W_2$.

Solution

(a) Let W be a set well-founded by a relation $<$ and let $V \subseteq W$. Irreflexivity and transitivity for $<$ over V follow immediately from the same properties over W. Since $V \subseteq W$, any non-empty subset of V lacking a minimal element will be a non-empty subset of W lacking a minimal element.

(b) Certainly not! For example, take the set $\mathbf{N}$ of natural numbers under the converse of the usual $<$, that is, under $>$. Then $\mathbf{N}$ itself has no minimal element under that relation, for it would be a maximal element under $<$.

(c) Intuitively, put all elements of W_1 first, under $<_1$, then all elements of W_2, under $<_2$; formally, put $< = <_1 \cup <_2 \cup W_1 \times W_2$. If the sets are not disjoint, put all elements of W_1 first, under $<_1$, then all elements of $W_2 \backslash W_1$, under $<_2$; formally, put $< = <_1 \cup (<_2 \cap (W_2 \backslash W_1)^2 \cup W_1 \times (W_2 \backslash W_1)$.

(d) The lexicographic ordering does the job. In Sect. 4.7.3 we defined this ordering for the special case of $\mathbf{N}^2$, and its natural generalization to the context of $W_1 \times W_2$ puts $(m,n) < (m',n')$ iff either $m <_1 m'$, or $m = m'$ and $n <_2 n'$. To verify that this well-orders $W_1 \times W_2$, use the same argument as for $\mathbf{N}^2$ in Exercise 4.7.3, with suitable editorial adjustments.

4.9 Selected Reading

Induction and recursion on the positive integers. There are innumerable elementary texts, although most are written for students of mathematics and also tend to focus on inductive proof, neglecting recursive definition. One text that is directed to students of computer science is James L. Hein *Discrete Structures, Logic and Computability*, Jones and Bartlett, 2005 (second edition), chapter 4.4. Two others, written to accompany maths courses but proceeding at a leisurely rate, are Carol Schumacher *Chapter Zero: Fundamental Notions of Abstract Mathematics*, Pearson, 2001 (second edition), chapter 3 and Daniel J. Velleman *How to Prove It: A Structured Approach*, Cambridge University Press 2006 (second edition), chapter 6.

Well-founded induction and recursion. Again, introductory accounts tend to be written for students of mathematics and tend to neglect recursive definition. Two presentations accessible to computer science students are Seymour Lipschutz *Set Theory and Related Topics*, McGraw Hill Schaum's Outline Series, 1998, chapters 8–9 and Paul R. Halmos *Naive Set Theory*, Springer 2001 (new edition), 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!

Part II
Maths

Counting Things: Combinatorics

5

Chapter Outline

Up to now, our work has been almost entirely qualitative. The concepts of a set, relation, function, recursion and induction are, in themselves, 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 directly quantitative matters, and specifically to problems of counting. We 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 sets? 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? This question is less straightforward than the first one as there are several different kinds of selection, and they give us very different outcome numbers. We will untangle *four basic modes of selection*, give arithmetical formulae for them, and practice their application.

5.1 Basic Principles for Addition and Multiplication

In earlier chapters, we already saw some principles for calculating the number of elements in a set, given how many in another set. In particular, in Chap. 3 we noted the *principle of equinumerosity*: 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,
https://doi.org/10.1007/978-3-030-42218-9_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 pairwise disjoint sets: Let $A_1, \ldots, A_n$ be pairwise disjoint finite sets. Then $\#(\cup\{A_i\}_{i \leq n}) = \#(A_1) + \ldots + \#(A_n)$.

Exercise 5.1.1 (1)
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 $B \subseteq A$. Verification: 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 $\#(A) < \#(A) + 1 = \#(A \cup \{b\}) \leq \#(A \cup B)$. *End of solution*.

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 for difference, we get:

$$\begin{aligned}\#(A \cup B) &= \#(A \cap B) + (\#(A) - \#(A \cap B)) + (\#(B) - \#(A \cap B)) \\ &= \#(A \cap B) - \#(A \cap B) - \#(A \cap B) + \#(A) + \#(B) \\ &= \#(A) + \#(B) - \#(A \cap B).\end{aligned}$$

We have thus shown the:

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) Illustrate the addition principle for two disjoint sets in terms of Venn diagrams.
(b) Do the same for the non-disjoint version.
(c) Check the claim that $A \cup B = (A \cap B) \cup (A \backslash B) \cup (B \backslash A)$.

Solution

(a) Go to the Venn diagram for union in Fig. 1.3 of Chap. 1. $A \cup B$ is the entire shaded area. If A is disjoint from B then the central lens is empty, so $\#(A)$ is the number of elements in the left crescent, $\#(B)$ is the number of elements in the right crescent, $\#(A \cup B)$ is the number of elements in those two crescents together.
(b) If A is not disjoint from B then $\#(A)$ is the number of elements in the left circle, $\#(B)$ is the number of elements in the right circle, $\#(A \cup B)$ is the number of elements in those two circles together. But if we add $\#(A)$ and $\#(B)$ we are counting the central lens $\#(A \cap B)$ twice, so we must subtract it once.
(c) Suppose $x \in$ LHS. Then $x \in A$ or $x \in B$. In the former case $x \in (A \cap B)$ or $x \in (A \backslash B)$ according as x is, or is not, in B; in the latter case we have similarly that $x \in (A \cap B)$ or $x \in (B \backslash A)$. So, in both cases $x \in [(A \cap B) \cup (A \backslash B)] \cup [(A \cap B) \cup (B \backslash A)] =$ RHS. For the converse, suppose $x \in$ RHS. Then $x \in (A \cap B)$ or $x \in (A \backslash B)$ or $x \in (B \backslash A)$, and each of these sets is included in $A \cup B$.

Exercise 5.1.1 (3)

(a) The classroom contains seventeen male students, eighteen female students, and the professor. How many altogether in the classroom?
(b) The logic class has twenty students who also take calculus, nine who also take a philosophy unit, eleven who take neither unit, and two 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.

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 until Sect. 5.2.

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. Verification: 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)
The menu at our restaurant allows choice: for first course one can choose either soup or a 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 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 incompletely specified. 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 or weaken 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 tool-kit 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 tool-kit. 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?

Solution

(a) There are 10 digits 0 to 9, and so the desired number is $10 \cdot \ldots \cdot 10$ (eight times), that is, 10^8.
(b) $23^2 \cdot 10^4$.

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 into two disjoint subsets—those four-digit numbers 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 $10^3 = 1000$ possibilities. In the second case, beginning with 73, we have two digits to fill in, hence $10^2 = 100$ possibilities. So the total is 1100 four-digit numbers beginning with 5 or with 73.

This kind of problem 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.

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)

(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

(a) The set of possible outfits is the union of two sets: $S \times J$ and $S \times J \times T$. Notice 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.
(b) The set of possible tags is the union of two sets D, L. All elements of D begin with 9, followed by one of 26 letters, then one of 10 digits, finally one of 26 letters. Thus $\#(D) = 26 \cdot 10 \cdot 26 = 6760$. All elements of L begin with the letter m, followed by one of 10 digits, then one of 26 letters, finally one of 10 digits. Thus $\#(L) = 10 \cdot 26 \cdot 10 = 2600$. Notice that these two sets are disjoint. So $\#(D \cup L) = 6760 + 2600 = 10,360$.

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 six 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.

5.2.1 Order and Repetition

Expressed in more detail, the two preliminary questions that we need to consider 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 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 days needing volunteers and, to keep the diagram small, a total of five club members a_1, a_2, a_3, a_4, a_5. If we select, say, volunteers a_2 for the first day and a_4 for the second day, then this is represented in Table 5.1 by the cell for row a_2 and column a_4. The table thus contains $5 \cdot 5 = 25$ cells for the 25 pairs (x,y) of members, x as a candidate for the first day and y for the second day.

The decision we make concerning repetition will determine which cells are considered impossible, and the decision about order will determine which cells are taken as really the same. Thus both decisions determine *which of the 25 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, since that would be to allow the same person to serve on both days; 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 are treated as being the same as

Table 5.1 Cells left uncounted when repetition is not allowed or order is disregarded

	a_1	a_2	a_3	a_4	a_5
a_1	R−				
a_2	O−	R−			
a_3	O−	O−	R−		
a_4	O−	O−	O−	R−	
a_5	O−	O−	O−	O−	R−

their images in the bottom left. For example, (a_2, a_4) represents the same selection as (a_4, a_2). In this case we have only $25 - 10 = 15$ possible selections, subtracting the ten 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 figure for mode O−R + , the non-permitted cells of the diagonal leaving those to the upper right, thus with only $15 - 5 = 10$ selections. Equivalently, we could subtract the ten duplicating cells to the bottom left of the diagonal from the figure for mode O+R−, likewise leaving $20 - 10 = 10$ selections. In other words, we subtract from the total 25 both those marked R− (not permitted, if repetition is not allowed) and those marked O− (duplicates, if ordering does not matter).

In summary: according to the way in which we understand the selection mode, we get four different answers: 25, 20, 15, or 10 possible selections! There are two morals to the story: when working on practical problems we must be clear about which mode is intended and, when articulating general principles, we should treat the four modes of selection separately while keeping an eye open for links between them.

5.2.2 Connections with Functions

It is helpful to notice connections between the above and what we learned about functions in Chap. 3. In the first two modes, where order matters, 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 A may be represented by an ordered k-tuple $(a_1, \ldots, a_k)$ of elements of A. As we already know, such an ordered k-tuple may be understood as a function on the set $\{1, \ldots, k\}$ into A. Thus, counting the selections of k items from an n-element set A according to the O+R+ mode is the same as counting *all the functions* from the k-element set $\{1, \ldots, k\}$ into A.

For mode O+R−: When order matters but repetition is not allowed, then each selection of k items from an n-element set A corresponds to an *injective function* from the k-element set $\{1, \ldots, k\}$ into A. Injective, because when $i \neq j$, $f(i)$ is not allowed to be the same as $f(j)$.

We now look at the modes where order does not matter. It is convenient to consider first the mode O−R− where repetition is not allowed. There, it turns out that we are no longer counting functions but other, closely related, items that are already familiar from Chap. 3.

For mode O−R−: When order does not matter and repetition is not allowed, then each selection of k items from an n-element set A corresponds to the *range of an injective function* from a k-element set into A. If you think about it, these ranges correspond in turn to the k-element *subsets* of A. Clearly, every such range is a k-

element subset of A and, conversely, every k-element subset of A 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 A. They are, conversely, functions from A into $\{1, \ldots, k\}$ satisfying a certain condition.

For 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 is the same as the number of functions f from A into $\{1, \ldots, k\}$ satisfying the condition that $\Sigma\{f(a)\colon a \in A\} = k$. At first sight, this condition may appear rather mysterious, but the reason for it is quite simple. The function f tells us how many times each element of A actually appears in the selection. It can be zero, one or more up to k, 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, I would even say devious.

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 you have not seen before—and won't see again in this book.

Alice Anyway, 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 gained some popularity in certain areas of computer science, and also in 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

When $k > n$, that is, when the number of items to be selected is greater than the number of items in the selection-pool A, 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. In contrast, the two modes permitting repetition (O+R+ and O−R+) still allow selection when $k > n$. *End of solution*.

So far, we have been calling the four modes by their order/repetition acronyms; it is time to mention more familiar 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, i.e. 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 as $P(n,k)$ or nPk, ${}^{n}P_{k}$ or similar.
- When repetition is not allowed and order does not matter, i.e. 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 nCk, ${}^{n}C_{k}$ or similar. A common two-dimensional notation, 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 that goes back to the sixteenth century. They are also the most common in computer science practice.

For the modes allowing repetition, terminology is distressingly variable from author to author. The most straightforward names piggy-back on those for their counterparts without repetition:

- When repetition is allowed and order matters, i.e. 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, i.e. when we are in the mode O−R+, we will speak of *combinations with repetition allowed*.

Table 5.2 summarizes the nomenclature. It lists the four modes in a sequence different from that which was convenient for the conceptual analysis above, but which is more useful for a numerical analysis, for it corresponds to the order in which we will 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*.

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

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 that were distinguished. We begin with the two in which repetition is not allowed: permutations (O+R−) and combinations (O−R−). Table 5.3 states the number of permutations and combinations (without repetition) of k items from an n-element set. Both formulae make central use of the factorial function defined in Exercise 4.3 (c) of Chap. 4.

At this stage, it is tempting to plunge into a river of examples to practice applying the two counting formulae. Indeed, that is necessary if one is really to master the material but, by itself, it is not enough. We also need to see *why* each of the formulae does its job. That also helps 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? If we apply the formula in Table 5.3 for permutations, we get $P(n,k) = n!/(n-k)! = 10!/(10-6)! = 10!/4! = 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 = 151,200$. How do we know that this is the right formula to apply?

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	$n!/k!(n-k)!$	$k \leq n$

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 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 $1 = (n-k)!/(n-k)!$ gives us $n!/(n-k)!$ in agreement with the counting formula in the table.

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 A is the same as the number of injections on A onto A, and is equal to $n!$

Exercise 5.3.1 (1)

(a) Use the formula to calculate each of $P(6,0)$, …, $P(6,6)$.
(b) You have 15 ties, and you want to wear a different one each day of the working week (five days). For how long can you go on without ever repeating a weekly sequence?

Solution

(a) $P(6,0) = 6!/6! = 1$; $P(6,1) = 6!/5! = 6$; $P(6,2) = 6!/4! = 30$; $P(6,3) = 6!/3! = 120$; $P(6,4) = 6!/2! = 360$; $P(6,5) = 6!/1! = 720$; $P(6,6) = 6!/0! = 720$. Thus, $P(6,5) = P(6,6)$ and in general $P(n,n\text{-}1) = P(n,n) = n!$ since $1! = 0! = 1$.
(b) Our pool A is the set of ties with $n = 15$ elements, and we are selecting $k = 5$ elements 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: fifteen is far too many ties!

Exercise 5.3.1 (2)

(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?

Solution

(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 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), $P(n,0) = 1$. Clearly, $P(n,k)$ takes its largest possible value when $k = n$, and its smallest when $k = 0$.

5.3.2 Counting Combinations

To guide ideas, 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 selecting 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 tells us that $C(n,k) = n!/(k!(n - k)!)$ 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 the latter divided by 6!, that is, one 720th. Order matters!

How can we prove the general formula for combinations? Notice that it says, in effect, that $C(n,k) = P(n,k)/k!$ so it suffices to prove that. Now, as explained in Sect. 5.2.2, $C(n,k)$ 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)

(a) (i) Use the formula to calculate each of $C(6,0)$, …, $C(6,6)$. (ii) Do you notice any interesting pattern in the results? (iii) Express the pattern in general terms and prove it.
(b) 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?
(c) 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?

Solution

(a) (i) Use the value of $P(6,k)$, …, $P(6,k)$ for $0 \leq k \leq 6$ already given by Exercise 5.3.1 (1) and divide by $k!$ This gives $C(6,0) = P(6,0)/0! = 1/1 = 1$; $C(6,1) = P(6,1)/1! = 6/1 = 6$; $C(6,2) = P(6,2)/2! = 30/2 = 15$; $C(6,3) = P(6,3)/3! = 120/6 = 20$; $C(6,4) = P(6,4)/4! = 360/24 = 15$; $C(6,5) = P(6,5)/5! = 720/120 = 6$; $C(6,6) = P(6,5)/6! = 720/720 = 1$.

(ii) There is an interesting symmetry as the values climb and fall again: the values are the same going from either end.

(iii) In general, $C(n,k) = C(n,(n - k))$ for any $k \leq n$. Verification: LHS = $n!/k!(n - k)!$ while RHS = $n!/((n - k)!(n - (n - k))!) = n!/(n - k)!k!$ and we are done.

(b) 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: $C(8,3) = 8!/(3!5!) = (8 \cdot 7 \cdot 6)/(3 \cdot 2) = 4 \cdot 7 \cdot 2 = 56$.

(c) In this question it is again implicitly 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 $P(8,3) = 8!/5! = 8 \cdot 7 \cdot 6 = 336$. *End of solution.*

Parts (b) and (c) of Exercise 5.3.2 (1) illustrate the importance of understanding what kind of selection we are supposed to be making *before* applying a counting formula. Implicit assumptions about whether repetition is allowed and whether order is significant need to be articulated explicitly. Sometimes the question may be ambiguous.

It is also instructive to consider an example that looks quite like the last ones, 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)

(a) State and check a necessary and sufficient condition on n,k for $C(n,k) < P(n,k)$.
(b) Show that for $j < k \leq m$ we have $C(p,j) < C(p,k)$ for any $p \geq 2m$.

Solution

(a) Since $C(n,k) = P(n,k)/k!$ we always have $C(n,k) \leq P(n,k)$, and clearly $C(n,k) < P(n,k)$ iff $k! > 1$, thus iff $k > 1$.
(b) Let $j < k \leq m$ and $p \geq 2m$; we want to show that $C(p,j) < C(p,k)$. By the counting formula for combinations, it suffices to show that $p!/j!(p-j)! < p!/k!(p-k)!$ i.e. that $j!(p-j)! > k!(p-k)!$ i.e. that $(p-j)!/(p-k)! > k!/j!$ But the LHS is the product of $k-j$ terms $p-j, p-j-1, \ldots, p-k+1$ each of which is greater than m since $p \geq 2m$ and $j < k \leq m$, while the RHS is the product of $k-j$ terms $k, k-1, \ldots, k-j+1$ each of which is less than or equal to m, so LHS > RHS as desired. *End of solution*.

This is a good moment to fulfil a promise that the 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 first counted the elements of A and B separately and added them; but as the two sets may not be disjoint, that procedure 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 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 as follows (you may find it useful to identify the terms of this rule in a three-set Venn diagram for three sets):

$$\#(A \cup B \cup C) = \#(A) + \#(B) + \#(C) - \#(A \cap B) - \#(B \cap C) - \#(A \cap C) + \#(A \cap B \cap C).$$

For the general case, let $A_1, \ldots, A_n$ be any sets with $n \geq 1$. Then $\#(A_1 \cup \ldots \cup A_n)$ equals the result of the following additions and subtractions:

\+ (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, this rule can be written in a much more compact mathematical notation, but for our purposes we can leave that aside. Traditionally it is called the *principle of inclusion and exclusion*, because we first include too much, then exclude 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 term $\#(A_i \cap A_j)$ 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) Write out the principle of inclusion and exclusion in full for the case $n = 4$.
(b) Calculate $C(4,k)$ for each $k = 1, \ldots, 4$ to check that at each stage you have summed the right number of terms.

Solution

(a)

$$\begin{gathered}\#(A_1 \cup A_2 \cup A_3 \cup A_4) = \\ \#(A_1) + \#(A_2) + \#(A_3) + \#(A_4) \\ -\#(A_1 \cap A_2) - \#(A_1 \cap A_3) - \#(A_1 \cap A_4) - \#(A_2 \cap A_3) - \#(A_2 \cap A_4) - \#(A_3 \cap A_4) \\ +\#(A_1 \cap A_2 \cap A_3) + \#(A_1 \cap A_2 \cap A_4) + \#(A_1 \cap A_3 \cap A_4) + \#(A_2 \cap A_3 \cap A_4) \\ -\#(A_1 \cap A_2 \cap A_3 \cap A_4).\end{gathered}$$

(b) $C(4,1) = 4!/(1!(4-1)!) = 4$; $C(4,2) = 4!/(2!(4-2)!) = 6$; $C(4,3) = 4!/(3!(4-3)!) = 4$; $C(4,4) = 4!/(4!(4-4)!) = 1$.

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 Table 5.3 that we had for the non-repetitive selections, giving us Table 5.4.

5.4.1 Permutations with Repetition Allowed

We are looking at the number of ways of choosing of k elements from a n-element set, this time both 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 to 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 permutations where repetition is not allowed and dwarfs the 210 for combinations. What is the reasoning?

As for permutations without repetition, we apply the multiplication principle. Clearly, there are n ways of choosing the first item. But, since repetitions are allowed, there are again n ways of choosing the second item, and so on, thus giving us $n \cdot \ldots \cdot n$ (k times) i.e. 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 A, so that the set of all the selections corresponds to the Cartesian product of A 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.

We recall that this operation makes perfectly good sense even when $k > n$, since repetition is allowed. For example, even if only two digits are available, we can construct six-digit telephone numbers out of them, and there will be 2^6 of them.

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 allowing repetition	n^k	none	1,000,000
O−R+		Combinations allowing repetition	$(n+k-1)!/k!(n-1)!$	$n \geq 1$	5,005

Exercise 5.4.1 (1)
You have 15 ties. You want to wear one each day of the working week (five days), but you don't care whether they are the same or not within any given week. For how long can you go on without ever repeating a weekly sequence?

Solution
The situation contrasts with that of. You have 15^5 ways of constituting a weekly sequence, and so can go on for that many weeks without repeating such a sequence. This is much more than in the situation considered in Exercise 5.3.1 (1) (b), where repetition was not allowed and we had only $15!/10! = 15 \cdot 14 \cdot 13 \cdot 12 \cdot 11$ possible selections.

Exercise 5.4.1 (2)

(a) Does the figure in the formula for permutation with repetition allowed (O+R+) square with intuition in the case $n \geq 1$, $k = 1$?
(b) What about the case $n \geq 1$, $k = 0$? Does the figure in the formula make intuitive sense then?
(c) And the case $n = 0$ while $k = 1$?
(d) Finally, the case $n = 0$ and $k = 0$?

Solution

(a) Yes. When $n \geq 1$, $k = 1$, the formula gives us $n^1 = n$ selections. This accords with intuition, for there are just n ways to choose a single element when there are $n \geq 1$ items to choose from.
(b) When $n \geq 1$, $k = 0$, the formula gives us $n^0 = 1$ selection. This makes some kind of sense: there is just one way of choosing no item out of a pool of $n \geq 1$ items—namely by doing nothing.
(c) When $n = 0$ while $k \geq 1$, the formula gives us $0^k = 0$ selections. Again, this is what we might expect: no selection can be made since there is nothing to select from.
(d) In the ultra-limiting case where $n = 0$, $k = 0$, the formula provides $n^k = 0^0 = 1$, not 0. At first, this seems quite odd. But one can accommodate it 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 the situation in case (c), where we were asked to select *something* from nothing, for there is no way of doing that—not even by doing nothing! *End of solution*.

From Exercise 5.4.1 (2) it appears that, when carefully cultivated, intuition can accord with limiting cases of the formula for permutations with repetition allowed. Similar reflection leads to the same moral for the other modes of selection.

That said, it should also be remarked that, in mathematics, standard definitions quite often give rather odd-looking outcomes in limiting cases; for example, already

in the definition of exponentiation, the equality $n^0 = 1$ can seem bizarre when first met in school. In such instances, it can be helpful and instructive to try to make intuitive sense of the limiting case but, even if one doesn't succeed in doing so in a convincing manner, there is not necessarily a flaw in the definition. So long as the defined notion behaves as desired in the principal cases, the mathematician is pretty much free to stipulate, for the limiting ones, whatever permits the smoothest formulation and proof of general principles.

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 gives us the figure $(n + k - 1)!/k!(n - 1)! = (10 + 6 - 1)!/6!(10 - 1)! = 15!/6!9! = (15 \cdot 14 \cdot 13 \cdot 12 \cdot 11 \cdot 10)/(6 \cdot 5 \cdot 4 \cdot 3 \cdot 2) = 7 \cdot 13 \cdot 11 \cdot 5 = 5{,}005$. More than the 210 for combinations without repetition, but still much less than the 151,200 for permutations without repetition, not to speak of the 1,000,000 for permutations with repetition allowed.

This mode, O−R+, was the trickiest to interpret in terms of functions in Sect. 5.2.2, and its counting formula is likewise the trickiest to prove. The basic idea of the argument is to re-conceptualize 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) *but with a larger pool of elements to choose from*.

Recall that, as mentioned in Sect. 5.2.2, in the O−R+ mode we are in effect counting the number of functions f from an n-element set $A = \{a_1, \ldots, a_n\}$ into $\{0, \ldots, k\}$ such that $\Sigma\{f(a)\colon a \in A\} = f(a_1) + \ldots + f(a_n) = k$. Consider any such function f. To reconceptualise 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 the label + in the next slot, then 2 in each of the next $f(a_2)$ slots, then + in the next one, and so on up to the $(n - 1)$th plus sign, 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 $f(a_1) + \ldots + f(a_n)$. The construction will thus need $k + (n - 1)$ slots, since we are requiring that $f(a_1) + \ldots + f(a_n) = k$ and we insert $n - 1$ plus-signs.

Now the slot pattern we have created is fully determined by the total number $k + (n - 1)$ of slots and the choice of slots for plus-signs. 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.

Now, 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, because there is a trivial one-one correspondence between the two. So, we are in effect making a selection, where order does not matter and repetition is *not* allowed, of $n - 1$ items from a larger pool of $k + (n - 1)$ elements. We have thus 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, is the same as the number of selections when choosing $n - 1$ items from a pool of $k + n - 1$ elements using the O−R− mode of selection, namely $C(k + n - 1, n - 1)$.

We already know that $C(k + n - 1, n - 1) = (k + n - 1)!/(n - 1)!\cdot((k + n - 1) - (n - 1))!$ and the RHS simplifies to $(n + k - 1)!/k!(n - 1)!$—which is the counting formula for the O−R+ mode in Table 5.4. Our proof of that formula is complete.

Exercise 5.4.2
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
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 or piled on top of each other in a big wafer cone, 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 using the formula n^k for this mode. 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 to each subclass separately the counting formula $(n + k - 1)!/k!(n - 1)!$ for this mode. 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 $(5 + 2 - 1)!/2!(5 - 1)! = 6!/2!4! = 3 \cdot 5 = 15$ 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. *End of solution*.

Exercise 5.4.2 illustrates the general point that it is not always immediately obvious from the verbal formulation of a counting problem what exactly is sought after, in particular, whether repetition is allowed and whether order is taken to make a difference. Sometimes little hints in the presentation need to be highlighted in order to settle on one interpretation rather than another; occasionally the formulation may be irremediably ambiguous, leaving open more than one reading. Conceptual analysis of a problem should always precede blind application of a counting formula to it, for correct application of the wrong formula is just as bad as incorrect application of the right formula.

5.5 Rearrangements

We have merely scratched the surface of the subject of counting. There are many other kinds of counting problem of increasing levels of sophistication, but they all build, in subtle and sometimes complex ways, on the basic operations explained in this chapter: the principles of addition and multiplication, for unions and Cartesian products respectively, applied to four kinds of selection—with/without repetition allowed, with/without discrimination according to order.

To end the chapter, we look at one more kind of counting problem. How many ways of arranging the letters in *banana*? This sounds very simple, but its analysis is quite subtle. At first sight, it looks like a case of permutation 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, one *b*. Operations of this kind are usually called *rearrangements*.

It is important to approach rearrangement problems with the right *gestalt*. In the example of *banana*, don't think of yourself as selecting from a three-element set consisting of the letters *b*,*a*,*n*. Instead, focus on a six-element set consisting of the *six occurrences of letters* in the word or, alternatively, on *six slots* ready for you to insert those occurrences.

These two conceptualizations, letter-occurrences and slots, lead to two ways of analysing the problem. They give the same numerical result, but provide an instructive contrast, for one uses permutations while the other uses combinations. We describe them in turn. For each we begin with an analysis of the example, followed by a statement of the general rule for words with k letters and n letter-occurrences. For both methods we need to distinguish the six letter-occurrences in *banana* from the three letters, by rewriting the word as $b_1a_1n_1a_2n_2a_3$.

Using the first method for analysing 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, if at all, 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 further need to divide. Thus the number of rearrangements of the letters (including, of course, the original arrangement *banana*) is $720/6 \cdot 2 = 60$.

The general rule emerges clearly from the example. Suppose we are given a word with n letter-occurrences, made up of k letters, with those letters occurring m_1, …, m_k times. Then there are $n!/m_1! \ldots m_k!$ rearrangements of the letters in the word.

Now turning to the second method of analysis, we think in terms of six *slots* s_1, …, s_6 ready to receive letters. The rearrangements of the letters in *banana* can then be seen as functions f that take elements of the set of letters $\{a,n,b\}$ 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 the set $\{s_1, \ldots, s_6\}$. So, it suffices to count these functions. Now, 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, we get the same figure as by the first method: $C(6,3) \cdot C(3,2) \cdot C(1,1) = [6!/3!(6 - 3)!] \cdot [3!/2!(3 - 2)!] \cdot [1!/1!(1 - 1)!] = 6!/3!2! = 60$.

The general rule emerging from this method of analysing the example may be articulated as follows. 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 $\{f(a_i)\colon i \leq k)\}$ partitions $\{s_1, \ldots, s_n\}$. 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 the number of ways in which this can be done is $C(n, m_1) \cdot C(n - m_1, m_2) \cdot \ldots \cdot C(n - m_1 - m_2 - \ldots - m_{k-1}, m_k)$. This 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

Apply the counting formula $n!/m_1! \ldots m_k!$ to determine how many ways there are of rearranging the letters in the word *Turramurra*.

Solution

Rewrite the word *Turramurra* as $T_1u_1r_1r_2a_1m_1u_2r_3r_4a_2$ to highlight the letter-occurrences. We have a set $A = \{T_1,u_1,r_1,r_2,a_1,m_1,u_2,r_3,r_4,a_2\}$ of $n = 10$ letter-occurrences, the five letter-types occurring respectively $m_1 = 1$, $m_2 = 2$, $m_3 = 4$, $m_4 = 2$, $m_5 = 1$ times. So, by the rule, there are $n!/m_1! \ldots m_k! = 10!/1!2!4!2!1! = 37{,}800$. Much more than for *banana*.

5.6 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 represent the magnitude. Fix a value of $n > 1$. How many distinct integers can we represent with n binary digits?

(b) 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.

(c) I want to take two books with me on a holiday. I have two logic books, three mathematics books, and two novels. I want the two books that accompany me to be of different types. How many possible sets of two books can I take with me?

(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 per day 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?

Solution

(a) Be careful here: for $n > 1$, every n-tuple of 0 s and 1 s will represent an element of **Z**, but there are two of them that represent the number zero, namely 0 followed by $n - 1$ 0s and 1 likewise followed by $n - 1$ 0s. So, the answer is not 2^n but rather $2^n - 1$.

(b) The set of possible identifiers is the union of four disjoint sets, of 3-letter identifiers beginning with K, 3-letter identifiers beginning with W, 4-letter identifiers beginning with K, 4-letter identifiers beginning with W. By the multiplication principle, these sets have respectively 26^2, 26^2, 26^3, 26^3 elements, so their disjoint union has $2 \cdot 26^2 + 2 \cdot 26^3$ elements.

(c) Let L be the set of two logic books, M the set of three maths books, N the set of two novels. We want to count the number of two-element sets $\{x,y\}$ with x, y in distinct sets from L, M, N. Fix an order, say L, M, N between those sets and consider the *ordered* pairs (x,y) with x, y in distinct sets from L, M, N and written in the order of the sets that they are in. Recall from Sect. 5.4.2 that for counting purposes, holding the order fixed is the same as taking the order to be of no significance, since there is a one-one correspondence between the two. So the problem reduces to one of counting the elements of $(L \times M) \cup (L \times N) \cup (M \times N)$. By the multiplication principle, $L \times M$ has $2 \cdot 3 = 6$ elements, $L \times N$ has $2 \cdot 2 = 4$ elements, $M \times N$ has $3 \cdot 2 = 6$ elements. Since L, M, N are disjoint, $L \times M$, $L \times N$, $M \times N$ are disjoint. So by the addition principle, their union has $6 + 4 + 6 = 16$ elements.

(d) By the multiplication principle there are $5 \cdot 2 \cdot 13 = 130$ flight sequences on the Paris-Biarritz-London-Paris circuit and $13 \cdot 3 \cdot 5 = 195$ on the Paris-London-Biarritz-Paris one. These two sets of flight sequences are disjoint, so we can sum to get their union, giving us 325 flight sequences.

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?

Solution
Write A, B, C for the sets of people telephoned by Alice, Bertrand, Clarice respectively. We want to find $\#(A \cup B \cup C)$. By the principle of inclusion and exclusion, $\#(A \cup B \cup C) = \#(A) + \#(B) + \#(C) - \#(A \cap B) - \#(B \cap C) - \#(A \cap C) + \#(A \cap B \cap C) = 5 + 10 + 7 - 3 - 2 - 2 + 1 = 16$.

Exercise 5 (3) Four basic modes of selection

(a) In 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 the same states but one state at a time. In how many ways may the 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?

Solution

(a) The first contestant faces a set of $2^7 = 128$ possible ways of responding, while the second faces only $7^2 = 49$ possibilities. If they guess blindly, the first has a far more daunting task.
(b) From the formulation of the problem, it appears that we are not concerned with the order in which the tests are conducted, only the choice of states. So we are in the O−R− case, looking at the number of 4-element subsets of a 6-element set and should use the formula for combinations: $C(6,4) = 6!/(4!(6-4)!) = 6!/(4!2!) = 15$.
(c) The word 'test-run' suggests that we want to count the number of 4-element sequences, without repetitions, from a set of 6 elements. So we are in the O+R − case and should use the formula for permutations: $P(6,4) = 6!/2! = 360$

(d) Our set is the union of the number of 4-element sequences, 3-element sequences, 2-element sequences, 1-element sequences of elements of the 6-element set (we do not include the zero-element sequence because the test-run does not stop until either a failure is registered or four states have been covered). These sets of sequences are disjoint, so by the addition principle we sum their respective sizes: $P(6,4) + P(6,3) + P(6,2) + P(6,1) = 360 + 120 + 30 + 6 = 515$.

Exercise 5 (4) Rearrangements

(a) The candidate wins a free holiday in the place whose name admits the greatest number of arrangements of its letters: *Mississippi*, *Ouagadougou*, *Woolloomooloo*. Make the calculations to know where the candidate will be going.
(b) How many arrangements of the letters in Woolloomooloo make all occurrences of each letter contiguous?

Solution

(a) *Mississippi* = $M_1 i_1 s_1 s_2 i_2 s_3 s_4 i_3 p_1 p_2 i_4$ so by the formula $n!/m_1! \ldots m_k!$ we have 11!/4!4!2! = 34,650 rearrangements. *Ouagadougou* = $O_1 u_1 a_1 g_1 a_2 d_1 o_2 u_2 g_2 o_3 u_3$ yielding 11!/3!3!2! = 554,400 rearrangements. *Woolloomooloo* = $W_1 o_1 o_2 l_1 l_2 o_3 o_4 l_3 o_5 o_6$ giving 13!/6!2! = 4,324,320 rearrangements. The candidate thus wins a holiday to *Woolloomooloo*.
(b) If all occurrences of a given letter are contiguous then we can treat them as a block, in effect as a single letter. So we are looking at the set of all rearrangements of the four-letter word *Wolm* which, by the formula has 4! = 24 elements.

5.7 Selected Reading

Every text on discrete mathematics has a chapter on counting finite sets, sometimes under the grander name of 'finite combinatorics'. Most of them cover more ground than we have done; we have considered only the most basic tools with attention to the conceptual fabric behind them.

A clear beginner's exposition, covering just a little more than here, is Rod Haggarty *Discrete Mathematics for Computing*, Pearson 2002, chapter 6. More detailed presentations include Richard Johnsonbaugh *Discrete Mathematics* (sixth edition), Pearson 2005, chapter 6 and Seymour Lipschutz and Marc Lipson *Discrete Mathematics*, McGraw Hill, Schaum's Outline Series, 1997 (second edition), chapter 6.

For more advanced texts going deeply into the subject, see for example Titu **Andreescu** et al. *Counting Strategies: A Path to Combinatorics for Undergraduates*, Springer/Birkhauser, 2004, and **Jiri Herman** et al. *Counting and Configurations: Problems in Combinatorics, Arithmetic and Geometry*, Springer, CMS Books in Mathematics, 1997.

As mentioned by the Hatter, a good place to begin reading about multisets is the relevant article in *Wikipedia.*

Weighing the Odds: Probability

6

Chapter Outline

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*, outlining its behaviour, connections with *independence*, and deployment in *Bayes' rule*. An interlude reflects on a 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 set theory as we know it today 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 of communication for both writers and readers of textbooks 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 that may be couched in the traditional terminology.

D. Makinson, *Sets, Logic and Maths for Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-42218-9_6

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.

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 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 name 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: $S \to [0,1]$ 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, i.e. $[0,1] = \{x \in \mathbf{R}\colon 0 \leq x \leq 1\}$. Actually, when the sample space S is finite, 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)
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, since the values do not add up to one. (b) Yes: values are in the real interval and add to one. (c) Yes: same reason. (d) No, since not all values are in the real interval. (e) Yes: values are in the real interval and add to one. *End of solution*.

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: $S \to [0,1]$, and it puts $p(s) = 1/n$ for all $s \in S$. It is conceptually simple and has played an important role in the history of probability theory, but one should keep in mind that it is a special case. Many problems involve unequal distributions and we cannot confine ourselves to equiprobable ones.

Given a distribution p: $S \to [0,1]$, 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 $\mathcal{P}(S)$ or, as one also says, *over* S.

The pair made up of the sample space S and the probability function p^+: $\mathcal{P}(S) \to [0,1]$ is often called a *probability space*. Traditionally, the elements of $\mathcal{P}(S)$, i.e. the subsets of S, are called *events*, while the elements of S itself are variously called *sample points*, *points*, or *elementary events*.

Exercise 6.1.1 (2)

(a) Show that every probability function p^+ is into $[0,1]$ as claimed.
(b) Show that whenever $\#(S) \geq 2$ then there are infinitely many distributions on S.
(c) Consider the probability function p^+: $\mathcal{P}(S) \to [0,1]$ determined by the equiprobable distribution p: $S \to [0,1]$. Explain why $p^+(A)$ measures the proportion of elements of S that are in A, for every $A \subseteq S$.
(d) Explain why there are no equiprobable distributions over infinite sample spaces.

Solution

(a) Let p: $S \to [0,1]$ be a probability distribution, p^+: $\mathcal{P}(S) \to [0,1]$ the generated probability function, and $A \subseteq S$. We need to show that $p^+(A) \in [0,1]$. If $A = \varnothing$ then $p^+(A) = 0 \in [0,1]$ as desired, so suppose $A \neq \varnothing$. Since $p(s) \in \mathbf{R}$ for each $s \in A$, $\Sigma\{p(s)\colon s \in A\} \in \mathbf{R}$. It remains to check that $0 \leq \Sigma\{p(s)\colon s \in A\} \leq 1$. Clearly $p^+(A) \geq p(s) \geq 0$ for an arbitrarily chosen $s \in A \neq \varnothing$, and since $A \subseteq S$ we have $p^+(A) \leq \Sigma\{p(s)\colon s \in S\} = 1$.
(b) Suppose $\#(S) \geq 2$, and let s_1, s_2 be two distinct elements of S. There are infinitely may (in fact, uncountably many, for those readers familiar with the notion) real numbers in $[0,1]$. For each $r \in [0,1]$ we have a probability distribution p that puts $p(s_1) = r$, $p(s_2) = 1 - r$, and $p(s) = 0$ for all other $s \in S$. Clearly these distributions are mutually distinct and so the probability functions that they generate are also mutually distinct.
(c) Consider the probability function p^+: $\mathcal{P}(S) \to [0,1]$ determined by the equiprobable distribution p: $S \to [0,1]$, where $\#(S) = n$. Then $p(s) = 1/n$ for all $s \in S$. So for any non-empty subset of S, $p^+(A) = \Sigma\{p(s)\colon s \in A\} = (1/n)\cdot\#(A) = \#(A)/\#(S)$ as desired.

(d) This will be answered in Sect. 6.2.3 below. *End of solution.*

Now that we are clear about the difference between p: $S \to [0,1]$ and p^+: $\mathcal{P}(S) \to [0,1]$, we will usually 'abuse notation' by dropping the superscript from p^+, calling it just p in contexts where one can easily see that we mean the probability function rather than the underlying distribution.

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.

Exercise 6.1.2 (1)
Let p^+: $\mathcal{P}(S) \to [0,1]$, 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. The verifications tend to be cumulative, that is, you will find that earlier items in the list can be useful for checking later ones.

(a) $p(\varnothing) = 0$.
(b) $p(S) = 1$.
(c) When $A \subseteq B \subseteq S$ then $p(A) \leq p(B)$.
(d) $p(A \cup B) = p(A) + p(B) - p(A \cap B)$.
(e) When A and B are disjoint then $p(A \cup B) = p(A) + p(B)$.
(f) $p(S\backslash A) = 1 - p(A)$.
(g) $p(A \cap B) = p(A) + p(B) - p(A \cup B)$.
(h) When $A \cup B = S$ then $p(A \cap B) = p(A) + p(B) - 1 = 1 - [p(S\backslash A) + p(S \backslash B)]$.

Solution

(a) This is 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 probability function (for the first equality) and the definition of a distribution (for the second one).
(c) If $A = \varnothing$ then $p(A) = 0 \leq p(B)$. If $A \neq \varnothing$ then $p(A) = \Sigma\{p(s)\colon s \in A\} \leq \{p(s)\colon s \in B\} = p(B)$.
(d) $p(A \cup B) = \Sigma\{p(s)\colon s \in A \cup B\} = \Sigma\{p(s)\colon s \in A\} + \Sigma\{p(s)\colon s \in B\} - \Sigma\{p(s)\colon s \in A \cap B\} = p(A) + p(B) - p(A \cap B)$, using the principle of inclusion and exclusion from Chap. 5 Sect. 5.3.2.
(e) Immediate from (d), noting that when $A \cap B = \varnothing$ then $p(A \cap B) = p(\varnothing) = 0$. Of course, one can also check it directly without going through (d).
(f) $S = A \cup (S\backslash A)$ and the sets A, $S\backslash A$ are disjoint, so by (e) we have $p(S) = p(A) + p(S\backslash A)$ and thus $p(S\backslash A) = p(S) - p(A) = 1 - p(A)$ using (b).
(g) This can be shown from first principles, but it is quicker to get it by a straightforward arithmetic manipulation of (d).

(h) The left equality is immediate from (g) with (b). For the right equality, (e) tells us that $p(A) + p(B) - 1 = [1 - p(S\backslash A)] + [1 - p(S\backslash B)] - 1$, which simplifies to $1 - [p(S\backslash A) + p(S\backslash B)]$. *End of solution.*

In Exercise 6.1.2 (1), 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 the set operation and the arithmetic one are intimately related as recorded in part (f) of the exercise, they are of course different. However, trusting that there will be no confusion, from now on we will take the liberty of writing the set operation more briefly as $-A$.

Here's a hint for simplifying your scratch-work when doing exercises like the above 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, write $p(A)$ as $\underline{A}$ with the underlining representing the probability function p. For example, write the equation $p(A \cup B) = p(A) + p(B) - p(A \cap B)$ as $\underline{A} \cup \underline{B} = \underline{A} + \underline{B} - \underline{A} \cap \underline{B}$.

But be careful! When the problem involves more than one probability function, this shorthand notation cannot be used, since it cannot keep track of which function is which. To be sure, if there are just two probability functions in play one could in principle use two different kinds of underlining, but the expedient quickly becomes 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 (2)

(a) Show that $p(A \cap B) \leq p(A)$ and $p(A \cap B) \leq p(B)$.
(b) Let $S = \{a,b,c,d\}$ and let p be the equiprobability distribution on S. Give examples of events, i.e. sets $A, B \subseteq S$, such that respectively (i) $p(A \cap B) < p(A) \cdot p(B)$, (ii) $p(A \cap B) = p(A) \cdot p(B)$, (iii) $p(A \cap B) > p(A) \cdot p(B)$.
(c) Show that if $p(B) = 1$ then $p(A) = p(A \cap B)$.

Solution

(a) In each case LHS $\subseteq$ RHS so we can apply Exercise 6.1.2 (1) (c).
(b) Here are some simple examples; of course there are many others. For (i), put $A = \{a\}$, $B = \{b\}$ so $p(A \cap B) = p(\emptyset) = 0 < 0.125 = p(A) \cdot p(B)$. For (ii), put $A = \{a,b\}$, $B = \{b,c\}$ so $p(A \cap B) = p(\{b\}) = 0.25 = p(A) \cdot p(B)$. For (iii), put $A = \{a,b\} = B$ so $p(A \cap B) = p(A) = 0.5 > 0.25 = p(A) \cdot p(B)$.
(c) Given (a) above we need only show that if $p(B) = 1$ then $p(A) \leq p(A \cap B)$. Now $A = (A \cap B) \cup (A \cap -B)$ so $p(A) = p((A \cap B) \cup (A \cap -B)) = p(A \cap B) + p(A \cap -B)$ using Exercise 6.1.2 (1) (e), so we need only show that if $p(B) = 1$ then $p(A \cap -B) = 0$. But $p(A \cap -B) \leq p(-B) = 0$, using Exercise 6.1.2 (1) (f) recalling that $-B$ is shorthand for $S\backslash B$, and we are done. *End of solution.*

Table 6.1 Two languages for probability

	Traditional probabilistic	Modern set-theoretic
1	Sample space S	Arbitrary non-empty set S
2	Sample point	Element of S
3	Probability distribution	Same name, defined as any function p: $S \to [0,1]$ 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	$p(A\|B)$	$p(A\|B)$ or $p(A,B)$
10	Random variable X	(Payoff) function f: $S \to \mathbf{R}$
11	Range space $\mathbf{R}_X$	Range $f(S)$ of a function f: $S \to \mathbf{R}$

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 in two respects from any of the topics that we have studied in previous chapters. One is philosophical while the other concerns practical application.

6.2.1 Philosophical Interpretations

The notion of probability is quite slippery even in commonplace applications. 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 beliefs of any agent. This is known as the *objectivist* conception of probability. It comes in several flavours. For example, the probability of an event is often 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 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 entirely a subjective matter; properly unpacked, the subjectivist perspective may end up being not so subjective after all!

On the other hand, the frequentist view leaves us with the difficult problem of explaining what exactly is meant by 'contexts similar to the one within 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 initial equiprobability assumptions.

It is often felt by practitioners—that is, those needing to make calculations to answer real-life questions—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 is 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 it underlies the testing processes by which new medicines are cleared for safety.

As this is not a text in the philosophy of science, we will not take such issues further. We simply note that they are difficult even to formulate meaningfully, tricky to handle, and have not obtained any consensual resolution despite the centuries with which probability theory has been studied mathematically as well as applied to everyday affairs. The amazing thing is that we are able to carry on with

both the mathematics of probability theory and many of its applications to the real world without resolving, or even taking a position on, the philosophical quandaries 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 is that, unfortunately, you cannot get probabilities out of nothing. You need 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 determining probabilities, two steps should be made before any calculation, or at least checked out after a draft of the calculation.

- *Identify the sample space of the problem*. What is the set S that, in the problem, we take as domain of the probability distribution?
- *Specify the values of the distribution*. Can it reasonably be assumed to be an equiprobable distribution? If so, and if 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 whether there is any other distribution that is reasonable to assume.

Many students are prone to premature calculation, and training is needed to overcome the habit. 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 sub-section is only for the curious; others may skip it. What differences arise in the infinite case?

Exercise 6.1.1 (2) (d) asked for a proof that when the sample space S is infinite, it cannot have an equiprobable distribution, that is, one where $p(s) = p(s')$ for all s, $s' \in$ S. This is a good moment to give the promised answer. If the distribution is equiprobable, then either all points in S get the value zero, or else they all get the same value r 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) such that $n \cdot r > 1$. 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 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 as arguments some (but not necessarily all) subsets A of the (possibly infinite, even uncountable) sample space S, to give values $p(A)$ in [0,1]. The domain of the functions is required to be a *field of subsets* of S, that is, a non-empty collection $\mathrm{C} \subseteq \mathrm{P}(S)$ that is closed under complement (with respect to S) and binary union (and thus 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 cover arbitrary infinite unions of subsets of the sample space. Nevertheless, a restricted generalization, known as *σ-additivity*, may consistently be accepted, and often is so. 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 into what is usually known as *measure theory*.

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

Finite probability is an area where it pays to do many exercises in order to gain practice in applying basic principles. In this section we run through a series of them; the first concerns equiprobable distributions and the others become progressively more sophisticated as they bring in additional devices.

Exercise 6.3.1

(a) Throw a fair die. What is the probability that the number (of dots on the uppermost surface) is divisible by 3?
(b) Select a playing card at random from a standard European pack. What is the probability that it is a heart? 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

(a) 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* tells us that the questioner is assuming an equiprobable distribution, putting $p(n) = 1/6$ for all positive $n \leq 6$. The event we are asked to consider is $A = \{3,6\}$ and so $p(A) = 1/6 + 1/6 = 1/3$.

(b) Sample space: the set S of 52 cards in the standard European pack. To specify the distribution, we note that in this context, the term *random* again means that we are asked to consider an equiprobable one. 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 $H \cap C$, $H \backslash C$, $H \cup C$. 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 $p(H \cup C) = 22/52 = 11/26$.

Exercise 6.3.2

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 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 $p(G) = p(4) + p(5) + p(6) = 0.2 + 0.1 + 0.3 = 0.6$. Hence it is slightly more probable (difference 0.1) that the die falls even than greater than 3.

Exercise 6.3.3

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 $\{H,T\}^3$ 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 presumed 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. You seem to be making a hidden assumption in Exercise 6.3.3. We are told that the coin is fair, so $p(H) = 0.5 = p(T)$. But how do you know that, say, $p(\text{HHH}) = p(\text{HTH}) = 1/8$? Perhaps 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. Another way of putting it is that we are assuming that $p(H)$ *always* has the value 0.5 no matter what happened on earlier tosses; in other words, that 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

Exercise 6.3.4 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, W be the sets of the four male, female candidates respectively. 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 O + R−) of this seven-element set. As we know from Chap. 5, there are $P(7,7) = 7!$ such permutations; we assume that they are equiprobable, as is 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 $P(3,3)\cdot P(4,4)$ of these. So the probability of such an interview pattern is $P(3,3)\cdot P(4,4)/P(7,7) = 3!4!/7! = 1/35$. *End of solution*.

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.

Exercise 6.3.5
Of the 10 envelopes in my letterbox, 2 contain bills. If I pick 3 envelopes at random to open today, leaving the others for tomorrow, what is the probability that none of the 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.

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 then cancel down, producing 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.

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 die is a 3. Your sample space is the set of all 36 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 what is known as *conditional probability*.

In effect, we are changing the sample space. We are restricting it from the set S of all 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 conditional probability; it is easier to keep the sample space fixed and obtain the same result in another 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 $p(A|B) = (2/36)/(4/36) = 1/2$—which is 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 $p(A|B)$.

Solution
$p(B|A) = p(B \cap A)/p(A) = (2/36)/(11/36) = 2/11$, very much smaller than $p(A|B) = 1/2$. *End of solution*.

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 $Y \subseteq X$. However, it is surprising how often such confusion does occur in unreflecting discourse, in all three contexts.

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 $p(B) = 0$ —which will be the case when B is empty set, and can sometimes hold even when B is non-empty? Then 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. More tea?

Yet again, the Hatter is being rather laconic; let's say a bit more. The approach that he describes, which is the most common, treats conditional probability—like division itself—as a *partial function* of two arguments (Sect. 3.1 of Chap. 3). Its domain is not $\mathcal{P}(S)^2$ where S is the sample space, but rather $\mathcal{P}(S)$ x $\{X \subseteq S: p(X) \neq 0\}$. 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 exclude some other subsets since, as Alice remarked, it can also happen that $p(X) = 0$ for a non-empty $X \subset S$. This option gives us what is called 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.

But that is not the only possible option. An alternative is to treat conditional probability as a full function on $\mathcal{P}(S)^2$ but giving $p(A|B)$ a constant value for all B such that $p(B) = 0$—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 as much a difference of style as of content.

A third approach 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 when $p(B) = 0$, thus taking the domain to be $\mathcal{P}(S)^2$ and giving us a full rather than a partial function. But unlike both, it allows the value of $p(A|B)$ to vary when $p(B) = 0$, under the constraint that it satisfies the *product rule* $p(A \cap B|C) = p(A|C) \cdot p(B|A \cap C)$. This may be called *revisionary conditional probability*. We will not go into it any further, except to note three points. It has several main variants, corresponding to the presence or absence of further constraints for the 'critical zone' where $p(B) = 0$ but $B \neq \varnothing$. It is quite popular among philosophers of probability, has occasionally been employed by some economists using game theory, but is rarely used by statisticians, mathematicians and physicists.

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 where S is the sample space and assuming that $p(B) \neq 0$: (i) $p(\varnothing|B) = 0 = p(-B|B)$, (ii) $p(S|B) = 1 = p(B|B)$, (iii) $p(A|S) = p(A)$.

(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 $p(A \cap C) \neq 0$.

Solution

(a) Noting that $p(C) \neq 0$, we have $p(H|C) = p(H \cap C)/p(C) = 0.1/0.3 = 1/3$.

(b) Noting that $p(H) \neq 0$, we have $p(C|H) = p(C \cap H)/p(H) = 0.1/0.2 = 1/2$.

(c) For (i): $p(\varnothing|B) = p(\varnothing \cap B)/p(B) = 0/p(B) = 0 = p(-B \cap B)/p(B) = p(-B|B)$, using the assumption that $p(B) \neq 0$. For (ii): $p(S|B) = p(S \cap B)/p(B) = p(B)/p(B) = 1$ using the assumptions, and similarly for $p(B|B)$. For (iii): $p(A|S) = p(A \cap S)/p(S) = p(A)/1 = p(A)$.

(d) Assume $p(B) \neq 0$. Then, for (i): $p(A|B) = p(A \cap B)/p(B) = p(A \cap B \cap B)/p(B) = p((A \cap B)|B)$. For (ii): Suppose also that $A \subseteq A'$. Then $A \cap B \subseteq A' \cap B$ so, by Exercise 6.1.2 (1) (c) we have $p(A \cap B) \leq p(A' \cap B)$ so $p(A|B) = p(A \cap B)/p(B) \leq p(A' \cap B)/p(B) = p(A'|B)$ as desired.

(e) Suppose $p(A \cap C) \neq 0$ so also $p(C) \neq 0$. Then RHS $= [p(A \cap C)/p(C)] \cdot [p(A \cap B \cap C)/p(A \cap C)] = p(A \cap B \cap C)/p(C) =$ LHS.

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 Exercise 6.4.1 (2) (a), we may be interested in the extent to which high cholesterol leads to heart attack and use conditional probabilities when analysing the data. But great care is needed when linking probability and causality. Three points need to be borne in mind.

In the first place, conditional probability may make sense even when questions of time and causality do not arise. For instance, 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.

Secondly, when time and causality do arise, 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 in Exercise 6.4.1 (2) (b). Conditional probability makes sense in both directions of causality, both directions of time.

Thirdly, in the presence of time and causality, discovery of a value of $p(A|B)$ (even the extreme value $p(A|B) = 1$) 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, deductively speaking, 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 bristles with practical complications. Even inference in the reverse direction, from a causal relationship to a conditional probability, is not as straightforward as one might imagine. 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) In this exercise, understand all percentage figures given as *exact* (rather than at least, at most, or approximate). Use the acronyms N,S,P,L, for those members of the association who write novels, songs, poetry, and about love.

(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 song-writers write about love. But none of the novelists write about love. What is the probability that: (i) an arbitrary poet writes songs; (ii) an arbitrary song-writer writes poetry; (iii) an arbitrary novelist writing poetry and writing about love also writes songs; (iv) an arbitrary song-writer writes novels?

Solution
The words 'arbitrarily chosen' let us know that in this exercise probabilities follow proportions.

(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 $p(S|P) = p(S)/p(P) = 0.15/0.3 = 0.5$.
(b) (ii) $p(P|S) = p(P \cap S)/p(S)$. Again, since $S \subseteq P$ we have $S \cap P = S$, so $p(P|S) = p(S)/p(S) = 1$.
(b) (iii) $p(S|N \cap P \cap L) = p(S \cap N \cap P \cap L)/p(N \cap P \cap L)$. But by hypothesis, $N \cap L = \emptyset$, so $N \cap P \cap L = \emptyset$ so $p(S|N \cap P \cap L)$ is undefined, i.e. *has no value* on the ratio definition of conditional probability.
(b) (iv) $p(N|S) = p(N \cap S)/p(S)$. But the data given in the problem do not suffice to calculate the value of $p(N \cap S)$, so the problem has no definite solution. *End of solution*.

The last part of Exercise 6.4.2 (1) illustrates an important lesson. In real life, the information available is often insufficient to determine probabilities. One can sometimes get an answer 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 is dangerous. If made, they must be articulated explicitly so that they may be challenged and justified.

Nevertheless, when the data are insufficient to determine an exact 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 last part of Exercise 6.4.2 (1), 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 write songs and exactly 80% of song-writers write about love, we know that exactly 12% of the association are song-writers who write about love. Since none of the novelists write about love, *at least* 12% of the association (but not necessarily exactly 12%) are song-writing non-novelists. But recalling that exactly 15% of the association are song-writers in the first place, we may conclude that *at most* 3% of the song-writers are novelists, i.e. $p(N \cap S) \leq 0.03$. So, $p(N|S) = p(N \cap S)/p(S) \leq 0.03/0.15 \leq 0.2$. Rather tortuous reasoning that did not get us an equality, but it got us something.

Exercise 6.4.2 (2)
Show that the data in Exercise 6.4.2 (1) are compatible with $N \cap S = \emptyset$. What does that tell us about a *lower* bound on $p(N|S)$?

Solution
We specify an example verbally and suggest that you visualize it by drawing a diagram breaking a line into segments. Suppose there are 100 members of the

association, whom we call by the numerals 1 to 100. The novelists are just members 1 through 60 (so 60%) while the song-writers are just members 86 through 100 (so 15%). Thus $N \cap S = \varnothing$. It remains to give values to P and L that satisfy the remaining conditions of the exercise. The poets are members 1 through 10 and 81 through 100, which ensures that 30% write poetry, that all the song-writers write poetry, and that exactly 10% write both novels and poetry. Finally, suppose members 89 through 100 (and no others) write about love, which ensures that 80% of the song-writers write about love and that none of the novelists write about love. All conditions of Exercise 6.4.2 (1) are thus satisfied while $N \cap S = \varnothing$. When this happens, the conditional probability of N given S is zero, which thus serves as a trivial lower bound. *End of solution.*

When working with conditional probabilities it is easy to forget the fact that they are only partial functions, proceeding as if they were 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 such limiting cases while trying to solve a 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 to 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, i.e. reason as if all the conditional probabilities are defined. When finished, go back and *check each step* to see that it is correct in those limiting cases. This means verifying that each condition used in a conditional probability really has positive value or, if the value is zero, seeking an alternative argument. The checks are usually quite straightforward, but sometimes there can be nasty surprises!

Exercise 6.4.2 (3)
Let p: $S \to [0,1]$ be a probability function. For each of the following statements, specify the condition that is needed for all its conditional probabilities to be well-defined, and show that whenever they are well-defined then the statement holds.

(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

(a) Condition needed: $p(B) \neq 0$. Suppose this condition and that $A_1 \cap B$, $A_2 \cap B$ are disjoint. Then LHS = $p((A_1 \cup A_2) \cap B))/p(B) = p((A_1 \cap B)$ $(A_2 \cap B))/p(B) = [p(A_1 \cap B) + p(A_2 \cap B)]/p(B) = p(A_1 \cap B)/p(B) + p$ $(A_2 \cap B)/p(B)$ = RHS. Justifications for the five equalities: unpacking the definition of conditional probability, Boolean distribution in the numerator, addition principle with disjointedness assumption, arithmetic manipulation, repacking with the definition of conditional probability.

(b) Condition needed: $p(B) \neq 0$. Supposing this condition we have LHS = $p(-A \cap B)/p(B)$ while RHS = $1-[p(A \cap B)/p(B)] = [p(B)-p(A \cap B)]/p(B)$, where the first two equalities unpack the definition and the third is by arithmetic using $1 = p(B)/p(B)$. So, it suffices to show that $p(-A \cap B) = p(B)-p(A \cap B)$, i.e. that $p(B) = p(A \cap B) + p(-A \cap B)$, which is immediate by the addition principle.

(c) Condition needed: $p(B) \neq 0$. Suppose this condition and $p(A) \geq p(B)$ so that also $p(A) \neq 0$. We need to show that $p(A \cap B)/p(B) \geq p(A \cap B)/p(A)$. But this is arithmetically immediate by the supposition $p(A) \geq p(B)$.

(d) LHS = $[p(A \cap B)/p(B)]/[p(A \cap B)/p(A)]$ = RHS, where the first equality unpacks the definitions and the second is by arithmetic.

Alice Box: What is $A|B$?

Alice I understand the definitions and have done the exercises. But a question is still bothering me. *What is this object* $A|B$? 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 the definition of $p(A,B)$ is centred on a division. 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 Another question: does an expression like $p(A|(B|C))$ or $p((A|B)|C)$ mean anything?

Hatter Nothing whatsoever! Try to unpack $p((A|B)|C)$, say, using the definition of conditional probability: it would have to mean $p((A|B) \cap C)/p(C)$. The denominator $p(C)$ 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, after Exercise 6.4.2 (4)

Exercise 6.4.2 (4)
Show that the other expression mentioned by Alice, $p(A|(B|C))$, is also meaningless.

Solution
Try to unpack $p(A|(B|C))$ using the definition of conditional probability. It would have to mean $p((A \cap (B|C))/p(B|C)$. The denominator $p(B|C)$ makes sense, but the numerator $p((A \cap (B|C))$ is meaningless, for $p(A \cap X)$ is defined only when X is a subset of the sample set S. *End of solution*.

As we have emphasized, a probability function p: $\mathcal{P}(S) \rightarrow [0,1]$ 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 $\mathcal{P}(S)$ x $\{X \subseteq S: p(X) \neq 0\}$. 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: $\mathcal{P}(S) \rightarrow [0,1]$ and $B \subseteq S$ with $p(B) \neq 0$, we have the one-place function p_B: $\mathcal{P}(S) \rightarrow [0,1]$ 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: $\mathcal{P}(S) \rightarrow [0,1]$ *is also a probability function* over the same sample space S, as is easily verified. 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: $(p_B)_C = p_{B \cap C} = p_{C \cap B} = (p_C)_B$ when all these functions are well-defined, i.e. when $p(B \cap C) \neq 0$.

Exercise 6.4.2 (5)
Let p: $\mathcal{P}(S) \rightarrow [0,1]$ 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: $\mathcal{P}(S) \rightarrow [0,1]$ on a finite set S and any $B \subseteq S$ with $p(B) \neq 0$, the function p_B: $\mathcal{P}(S) \rightarrow [0,1]$ is well-defined and *is also a*

probability function over S. The italicized bit is quite subtle; before attempting it, you should re-read Sect. 6.1.1 carefully so that you can apply the appropriate definitions.

Solution

(a) Noting that $p(S) = 1 \neq 0$, we may apply the definition to give us $p_S(A) = p(A|S) = p(A \cap S)/p(S) = p(A)/1 = p(A)$.

(b) Suppose that $p(B) \neq 0$ and that $A_1 \cap B$, $A_2 \cap B$ are disjoint. Then, using the suppositions and Exercise 6.4.2 (3) (a), LHS $= p((A_1 \cup A_2)|B) = p(A_1|B) + p(A_2|B)$ = RHS as desired.

(c) Suppose that $p(B) \neq 0$. Then, using Exercise 6.4.2 (3) (b), LHS $= p(-A|B) = 1 - p(A|B)$ = RHS.

(d) Let S be a finite set, p: $\mathcal{P}(S) \to [0,1]$ a probability function on S, and $B \subseteq S$ with $p(B) \neq 0$. Then for every $A \subseteq S$, we have $p_B(A) = p(A|B) = p(A \cap B)/p(B)$ which is well-defined under the supposition $p(B) \neq 0$. By the definition of a probability function on S (Sect. 6.1.1), we need to show that (1) $\Sigma\{p_B(\{s\}): s \in S\} = 1$, (2) $p_B(A) = \Sigma\{p_B(\{s\}): s \in A\}$ for any $A \subseteq S$, (3) $p_B(\varnothing) = 0$. It is convenient to carry out the verifications in reverse order.

For (3), $p_B(\varnothing) = p(\varnothing|B) = p(\varnothing \cap B)/p(B) = p(\varnothing)/p(B) = 0/p(B) = 0$.

For (2), LHS $= p(A|B) = p(A \cap B)/p(B)$, while RHS $= \Sigma\{p(\{s\} \cap B)/p(B): s A\} = \Sigma\{p(\{s\} \cap B): s \in A\}/p(B)$, so it suffices to show that $p(A \cap B) = \Sigma\{p(\{s\} \cap B): s \in A\}$. We show this working from the right to the left, claiming that RHS $= \Sigma\{p(\{s\} \cap B): s \in A \cap B\} = \Sigma\{p(\{s\}): s \in A \cap B\}$ = LHS. For the first equality, the reason is that $p(\{s\} \cap B) = 0$ whenever $s \notin B$; for the second one, the reason is that $p(\{s\} \cap B) = p(\{s\})$ whenever $s \in B$; for the third one, the reason is that p is assumed to be a probability function on S, as defined in Sect. 6.1.1.

For (1), we have by (2) that $\Sigma\{p_B(\{s\}): s \in S\} = p_B(S)$, so it suffices to show that $p_B(S) = 1$. But LHS $= p(S|B) = p(S \cap B)/p(B) = p(B)/p(B) = 1$ and we are done.

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, according to historians, 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! So, this section does not introduce new material, but may help you get a better grasp of what we have been doing.

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, to say it in very careful English, 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 also that the same strict inequality holds for the mathematics department, i.e. that:

$$p(H|F \cap - C) > p(H| - F \cap - C)$$

Intuitively, it seems inevitable that the probability that a female candidate is hired, is greater than that for a male candidate, in other words, taking unions of 'given' sets on each side of the inequality, that:

$$p(H|F) > p(H| - F).$$

But it need not be so! Here is a counter-example (based on a lawsuit that was 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 conditional 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

$$p(H|F \cap - C) = p(H \cap F \cap - C)/p(F \cap - C) = 4/5 = 0.8 \text{ while}$$
$$p(H| - F \cap - C) = p(H \cap - F \cap - C)/p(-F \cap - C) = 6/8 = 0.75,$$

so that the second inequality also holds. But we also have:

$$p(H|F) = p(H \cap F)/p(F) = (2+4)/13 = 6/13 < 0.5$$
$$p(H|-F) = p(H \cap -F)/p(-F) = (1+6)/13 = 7/13 > 0.5.$$

Hence the third inequality $p(H|F) > p(H|-F)$ fails—giving rise to a lawsuit!

Of course, the 'paradox' is not a paradox in the strict sense of the term—an explicit 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, lawyers and doctors) can go wrong, especially if they go fast or calculate in their heads rather than on paper.

Mathematically, the probability in Simpson's paradox is merely icing on the cake. The example above is a manifestation, in a probabilistic setting, of an underlying arithmetical configuration: we may have $a_1/x_1 > a_2/x_2$ and $b_1/y_1 > b_2/y_2$ but $(a_1 + b_1)/(x_1 + y_1) < (a_2 + b_2)/(x_2 + y_2)$ for suitable positive integers.

Exercise 6.5
From the example used to illustrate Simpson's paradox, read off positive integers a_1, x_1, a_2, x_2, b_1, y_1, b_2, y_2 that illustrate the arithmetic configuration mentioned above.

Solution

$a_1 = 2$, $x_1 = 8$, $a_2 = 1$, $x_2 = 5$, $b_1 = 4$, $y_1 = 5$, $b_2 = 6$, $y_2 = 8$.

6.6 Independence

There are two equivalent ways of defining the independence of events using probability. One uses conditional probability, the other is in terms of multiplication of unconditional probabilities.

We begin with the definition using conditional probability, which we might call the 'conceptual' definition. 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 $p(A) = p(A|B)$ (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.

The other definition, in terms of multiplication, is shorter: A is independent of B (modulo p) iff $p(A \cap B) = p(A) \cdot p(B)$.

The two definitions are equivalent. Many people find the conceptual definition more intuitive, except for its limiting case, which is a little annoying. But the multiplicative one is often handier to work with; its equivalence with the conceptual definition also helps explain the role of the limiting case $p(B) = 0$ in the latter. After

doing Exercise 6.6 (1) (a) below you can feel free to use whichever of the two definitions is more convenient in a given problem.

Remember 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'. Finally, bear in mind that, whichever of the two definitions we use, we are just doing mathematics; the term 'independent' and 'dependent' should not make you assume the absence or presence of a causal connection.

Exercise 6.6 (1)

(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 $p(B) = 0$, $p(A) = 0$, $p(B) = 1$, $p(A) = 1$.
(d) Show that for disjoint A, B, if A is independent of B then $p(A) = 0$ or $p(B) = 0$.

Solution

(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 $p(A) = p(A \cap B)/p(B)$ so multiplying both sides by $p(B)$ we have $p(A \cap B) = p(A)\cdot p(B)$ and we are done.
(b) This and the following two parts of the are most quickly done using the multiplicative definition. Suppose that A is independent of B, so $p(A \cap B) = p(A)\cdot p(B)$. Then $p(B \cap A) = p(A \cap B) = p(A)\cdot p(B) = p(B)\cdot p(A)$ and we are done. But, just out of curiosity, let's also verify using the definition in terms of conditional probability. Suppose that A is independent of B, so either $p(B) = 0$ or $p(A) = p(A \cap B)/p(B)$. We want to show that either $p(A) = 0$ or $p(B) = p(A \cap B)/p(A)$. Case 1. Suppose $p(B) = 0$. Then either $p(A) = 0$ or $p(A \cap B)/p(A) = 0 = p(B)$. Case 2. Suppose $p(A) = p(A \cap B)/p(B)$. Then by arithmetic, $p(B) = p(A \cap B)/p(A)$.
(c) For the first two limiting cases, when $p(B) = 0$ or $p(A) = 0$ then $p(A \cap B) = 0$ so $p(A \cap B) = p(A)\cdot p(B)$. For the third, when $p(B) = 1$ then $p(A \cap B) = p(A) = p(A)\cdot p(B)$. The fourth is similar.
(d) Suppose that A, B are disjoint and A is independent of B. Then $p(A)\cdot p(B) = p(A \cap B) = p(\varnothing) = 0$, so by arithmetic either $p(A) = 0$ or $p(B) = 0$. *End of solution.*

An advantage of the multiplicative definition is that it generalizes elegantly to the independence of any finite collection of events. Let $\mathcal{C}$ be any finite collection of subsets of a sample space S. We say that the sets in $\mathcal{C}$ are *independent* iff for every non-empty sub-collection $\mathcal{D} \subseteq \mathcal{C}$ the probability of the intersection of the sets in $\mathcal{D}$ is the product of their respective probabilities; that is, $p(\cap\, \mathcal{D}) = \mathrm{X}\{p(A)\colon A \in \mathcal{D}\}$.

Equivalently, using a definition that may at first glance seem circular, but is in fact recursive (Chap. 4): iff every proper non-empty sub-collection of $\mathcal{C}$ is independent and $p(\cap\mathcal{C}) = X\{p(A)\colon A \in \mathcal{C}\}$. 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 counter-example.

When it holds, independence is a very powerful tool for calculating probabilities. We illustrate with an example. Five percent 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, calculate: (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 $\mathcal{C} = \{D_1,D_2,D_3\}$ 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), $p(D_1 \cup D_2 \; D_3) = 1-p(-D_1 \cap -D_2 \cap -D_3) = 1 - 0.857375 = 0.142625$.

Alice Box: A aw 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 (iii) 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 Exercise 6.6 (2)

Exercise 6.6 (2)
Show that the following four conditions are all equivalent (modulo any given probability function): (i) A and B are independent, (ii) A and $-B$ are independent, (iii) $-A$ and B are independent, (iv) $-A$ and $-B$ are independent.

Solution
It suffices to show the equivalence of (i) to (ii) since, by symmetry applied several times, we can apply that equivalence to get the equivalence of (i) to (iii), then the equivalence of the (iii) to (iv). 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 $p(A) = p(A \cap B) + p(A \cap -B) = (p(A)\cdot p(B)) + p(A \cap -B)$ by the supposition, so $p(A \cap -B) = p(A)-(p(A)\cdot p(B))$. Now, p

$(A)\cdot p(B) = p(A)\cdot(1-p(-B)) = p(A)-(p(A)\cdot p(-B))$, so $p(A \cap -B) = p(A)-[p(A)-(p(A)p(-B))] = p(A)\cdot p(-B))$ as desired. *End of solution*.

However, the usefulness of independence for making calculations should not 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 in the same family 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 bulls-eye is 0.2, the probability that A_2 gets a bulls-eye is 0.3. Assuming independence, what are: (i) the probability that they both get a bulls-eye? (ii) That neither gets a bulls-eye? (iii) Assuming also independence for successive tries, that neither gets a bulls-eye in three successive attempts each?
(b) Can an event ever be independent of itself?
(c) Is the relation of independence transitive?

Solution

(a) (i) 0.06, (ii) 0.56, (iii) $(0.8)^3\cdot(0.7)^3 = 0.175616$, which is quite low.
(b) In other words, can we have $p(A)\cdot p(A) = p(A \cap A)$, i.e. $p(A) = p(A)^2$. Clearly, this happens iff either $p(A) = 0$ or $p(A) = 1$.
(c) No. For a counterexample, just take any example of independent events A, B with $p(A) \neq 0$ and $p(A) \neq 1$. For instance, put $S = \{1,\ldots,4\}$, $A = \{2,4\}$, $B = \{2,3\}$. Then $p(A) = 0.5$, $p(B) = 0.5$, $p(A \cap B) = 0.25$ so $p(A|B) = p(A \cap B)/p(B) = 0.5 = p(A)$, verifying independence. By symmetry (Exercise 6.6 (b)), B is independent of A. So, if transitivity holds then A is independent of A which, by (b) of the present exercise, is impossible. There is another example of the failure of transitivity for independence in the end-of-chapter Exercise 6.4 (a).

6.7 Bayes' Rule

We have emphasised 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. Then $p(A|B) = p(A \cap B)/p(B)$, so $p(A \cap B) = p(A|B)\cdot p(B)$. 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, provided we also know the two non-zero unconditional probabilities $p(A)$, $p(B)$.

Indeed, we can go further. Assume that $p(A)$, $p(B)$ and also $p(-A)$ are non-zero. Since, as shown in Exercise 6.7 (1) below, $p(B) = p(B|A)\cdot p(A) + p(B|-A)\cdot p(-A)$ when the three denominators are well-defined, we can substitute in the denominator to get the following equation:

$$p(A|B) = p(B|A) \cdot p(A)/[p(B|A) \cdot p(A) + p(B| - A) \cdot p(-A)].$$

Using this, we can calculate the probability of $p(A|B)$ provided we know the conditional probabilities $p(B|A)$ and $p(B|-A)$ in the reverse direction together with $p(-A)$, provided the three 'non-zero' conditions hold.

Each of the two displayed equalities is commonly referred to as *Bayes' theorem* or *Bayes' rule*. The latter is a better name, for the verification above is so trivial, given the definitions, that it hardly deserves the name of theorem. The rule also has a more general form, as we will see later in this section.

Exercise 6.7 (1)
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.

Solution
$B = (B \cap A) \cup (B \cap -A)$ so $p(B) = p((B \cap A) \cup (B \cap -A)) = p(B \cap A) + p(B \cap -A)$. But $p(B \cap A) = p(B|A)\cdot p(A)$ by the definition of conditional probability using the assumption $p(A) \neq 0$. Similarly, $p(B \cap -A) = p(B|-A)\cdot p(-A)$ under assumption $p(-A) \neq 0$. Putting these together we may conclude the desired equality. *End of solution*.

With the help of Bayes' rule, we can analyse a common kind of error called the *base-rate fallacy*. In the equation $p(A|B) = p(B|A)\cdot p(A)/[p(B|A)\cdot p(A) + p(B|-A)\cdot p(-A)]$, the unconditional probability $p(A)$ is known as the *base rate*. When estimating probabilities mentally, without pencil and paper, it is tempting to simplify life by neglecting the base rate, which can lead to gross error. We illustrate with the following example, due to Keith Devlin.

Suppose we know that the probability of a certain kind of cancer in the population is 0.01, and that we have a new test for detecting it. The test never gives a false negative, that is, those having cancer always react positively to it, while twenty percent of those who do not have cancer also react positively. You take the test and get a positive reaction; what is the probability that you have that kind of cancer?

One is tempted to guess, off-the-cuff, at approximately 0.8. Indeed, that is what one gets if one applies Bayes' rule without attending to the base rate. For mnemonic ease, write the second version of Bayes' rule using the variables C for the presence of cancer and P for a positive outcome to the test:

$$p(C|P) = p(P|C) \cdot p(C)/[p(P|C) \cdot p(C) + p(P| - C) \cdot p(-C)]$$

If we drop the base rate $p(C)$ and its complement rate $p(-C)$ from this equation, we are left with a much simpler one:

$$p(C|P) = p(P|C)/[p(P|C) + p(P| - C)].$$

Using the information $p(P|C) = 1$, $p(P|-C) = 0.2$ given above, this tells that $p(C|P) = 0.83$ (to two decimal places) in accord with the guess above. But if we make the full calculation, taking into account the further information that $p(C) = 0.01$, we get the much lower result $p(C|P) = 0.05$ (to two decimal places).

Why do people so commonly make this kind of error? When doing mental calculation, we can hold only a certain amount in the head at any one time, so we are forced to simplify by focussing on the information that seems to be more specific and hence most relevant to the problem at hand, neglecting other factors. In the example, one is tempted to focus on $p(P|C) = 1$ and $p(P|-C) = 0.2$, which seem more germane to finding $p(C|P)$ than does general information about the value of $p(C)$. When $p(C)$ is low, that can lead us astray.

Bayes' rule is named after the eighteenth-century clergyman-mathematician who is thought first to have noted it. It is surprisingly useful, for it can often happen that we have no information permitting us to determine $p(A|B)$ directly, but do have statistical information, i.e. records of frequencies, that permit us to give figures for $p(B|A)$ together with some kind of estimate of the other probabilities on the right-hand side of one or the other of the two equations. With that information, Bayes' rule tells us how to calculate $p(A|B)$.

However, applications of the theorem can be controversial. Figures coming out of a calculation are, in general, no better than those going into it; as the saying goes, garbage in, garbage out. When wishing to apply the first version of Bayes' rule, we may have good statistics for $p(B|A)$, but only vague theoretical reasons for guessing at $p(A)$ or $p(B)$. Similarly, if seeking to apply the second version, we may have solid data for $p(B|A)$ but less reliable information on $p(A)$ or $p(B|-A)$. A good critical sense is needed to avoid being led into fantasy by the opportunity for quick calculation with dubious inputs.

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 deploying Bayes' rule to get a value for $p(A|B)$ even in contexts where available statistics give us little serious indication of the value of some of the inputs needed for applying the theorem. They are willing to rely on non-statistical reasons for estimating those inputs, or are confident that inaccuracies in their estimation can be 'washed out' in

the course of subsequent conditionalizations, to take place as further information comes in. Their opponents tend to be much more cautious, unwilling to assume any probability that does not have a serious statistical backing for fear of errors that, at bottom, are not so different from the base rate fallacy. 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 presumably can guess where the author's sympathies lie. And it should be remembered that whatever position one takes in the debate on applications, Bayes' rule 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)
15% 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' rule 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. *End of solution.*

Bayes' rule 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). 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' rule*. 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:

$$p(A_i|B) = p(B|A_i)\cdot p(A_i)/\Sigma_{j\leq n}[p(B|A_j)\cdot p(A_j)].$$

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 Clarice?

Solution
The sample set S is the set of all calls. We write A, B, C for the sets of calls dealt with by Alfred, Betty, Clarice respectively, H for the set of successfully handled calls. The sets A, B, C are implicitly assumed to partition S. We are given the probabilities $p(A) = 0.33$, $p(B) = 0.37$, $p(C) = 0.30$, $p(H|A) = 0.60$, $p(H|B) = 0.70$, $p(H|C) = 0.90$ and we are asked to find $p(C|H)$. Bayes' rule tells us that:

$$p(C|H) = p(H|C) \cdot p(C)/[p(H|A) \cdot p(A) + p(H|B) \cdot p(B) + p(H|C) \cdot p(C)].$$

Plugging in the given values we get, to three decimal places:

$$p(C|H) = 0.9 \cdot 0.3/[0.6 \cdot 0.33 + 0.7 \cdot 0.37 + 0.9 \cdot 0.3] = 0.371.$$

6.8 Expectation

Let S be any finite set, and let f: $S \to \mathbf{R}$ be a function associating a magnitude, positive or negative, to 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 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; indeed, in applications its 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 can call f: $S \to \mathbf{R}$ a *payoff function*.

Payoff functions are often called *random variables* and written X: $S \to \mathbf{R}$, 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 and conversely when needed. From a modern point of view, the term 'random variable' can be quite misleading, since we are not talking about a variable in the syntactic sense, now standard among logicians, of a letter used to range over a domain, but rather about a function. Nor is it necessarily random in the sense of being associated with an equiprobable distribution.

Now suppose that we also have a probability distribution p: $S \to [0,1]$ 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 $s \in S$ 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, if $p(s)$ 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* of the payoff function itself. We introduce it through the 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: $S \to \mathbf{R}$ and a probability distribution p: $S \to [0,1]$, we define the *expectation off given p*, written $\mu(f,p)$ or just μ when the two parameters are understood and fixed, by the equation $\mu = \mu(f,p) = \Sigma_{s\in S}\{f(s)\cdot p(s)\}$. Equivalently, in a form that is especially useful when $f(S)$ is small and the probability function p: $\mathcal{P}(S) \to [0,1]$ is independently known, we have $\mu(f,p) = {}_{x\in f(S)}\{x \cdot p(f^{-1}(x))\}$.

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 $mean(f) = \Sigma_{s\in S}\{f(s)\}/\#(S)$. What is the basic reason why they are not the same?

(b) Show that in the special case that p: $S \to [0,1]$ is an equiprobable distribution expectation equals mean.

Solution

(a) We saw that the expectation is 0.3. On the other hand, the mean is 1/3. The difference is due to the different probabilities of a, b, c.

(b) When probabilities of all elements of S are equal, then every $p(s) = 1/\#(S)$, so $\mu(f,p) = \Sigma_{s\in S}\{f(s)\cdot p(s)\} = \Sigma_{s\in S}\{f(s)\}/\#(S) = mean(f)$. *End of solution.*

When the expectation of a gamble comes out negative, then it is not a good idea to place a bet—one is likely to lose money. When it is positive, betting would be advantageous, provided of course the payoff function correctly records *all* your costs (perhaps including, as well as actual monetary outlay, taxes on earnings, time spent and opportunities foregone, stress, reputation,…) and benefits (perhaps including, as well as money won, prestige, the excitement and enjoyment given by the affair). In our horse-race example, if all auxiliary costs and benefits are zero, then $\mu = 0.3$ and betting would be profitable.

Alice Box: Really a good bet?

Alice Hold on! Let's take the horse-race example again, with the probabilities and payoffs that are specified in the text. 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, certainly. 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. So, if you have the opportunity to repeat the same bet as often as you like, it becomes attractive to do so.

Alice Provided I have enough capital to be able to put up with some probable initial losses without undue hardship or inconvenience

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 that we win. One might try to factor this negative feature into the payoff function, but that would be complex and perhaps rather artificial. Once again, we see that while the mathematics of finite probability theory is quite straightforward, its application can be very tricky.

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.

6.9 End-of-Chapter Exercises

Exercise 6.9 (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 possible 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 $s_1,\ldots,s_n$. 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 $A \subseteq S$.

Solution

(a) $S = \{1,\ldots,6\}$, $p(6) = 2/7$, $p(i) = 1/7$ for all $i \leq 5$.
(b) $S = \{1,\ldots,6\}$, $p(i) = 1/12$ for $i \in \{1,3,5\}$, $p(j) = 1/4$ for $i \in \{2,4,6\}$, $p(\{3,5\}) = 1/6$.
(c) There are n such distributions p: each puts $p(s_i) = 1$ for a single $i \leq n$, $p(s_j) = 0$ for other j n. Then $p(A) = 1$ iff $s_i \in A$, where s_i is the unique element of S with $p(s_i) = 1$.

Exercise 6.9 (2) Unconditional Probability

(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?

Solution

(a) The sample space is $\{1, \ldots ,6\}^3$. The simplest procedure is to calculate the probability of not getting any 6, and subtract it from one. Implicitly assuming independence (see the Alice Box in Sect. 6.3) and using the multiplication rule, the former is $(5/7)^3 = 0.36$, so the latter is 0.64, to two decimal places.
(b) Write $p(a_i)$, $p(b_i)$ $(i \leq 6)$ for the probabilities that you, respectively your friend, get the number i. Implicitly assuming independence and using the multiplication rule,, the probability that you both get i is $p(a_i)\cdot p(b_i)$ so, using disjointedness and the addition rule, the probability that there is some $i \leq 6$ such that you both get i is $\Sigma\{p(a_i)\cdot p(b_i)\colon i \leq 6\} = 5/28 = 0.18$ to two decimal places.

Exercise 6.9 (3) Conditional Probability

(a) In a sports club, 70% of the members play football, 30% play basketball, and 20% do both. What are: (i) the probability that a randomly chosen member plays football, given that he/she plays basketball? (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 (towards the end of Sect. 6.4.2) that for any probability function p: $\mathcal{P}(S) \to [0,1]$, if $p(B \cap C) \neq 0$ then $(p_B)_C = p_{B \cap C} = p_{C \cap B} = (p_C)_B$.

Solution

(a) Write F, B for the sets of those who play football, basketball respectively. We are given that $p(F) = 0.7$, $p(B) = 0.3$, $p(F \cap B) = 0.2$. Then (i) $p(F|B) = p(F \cap B)/p(B) = 0.2/0.3 = 2/3$, (ii) $p(B|F) = p(B \cap F)/p(F) = 0.2/0.7 = 2/7$.

(b) Write $p(A)$, $p(B)$ for the probabilities that Alice, Betty, respectively, hit the target. We are given that $p(A) = 1/4$, $p(B) = 1/5$. We want to calculate $p(A|A + B)$ where $A + B$ is the symmetric difference $((A \cap -B) \cup (-A \cap B)$ (see Exercise 1.1 at end of Chap. 1) so that $p(A|A + B) = p(A \cap (A + B))/p(A + B) = p(A \cap -B)/p(A + B)$. By independence, $p(A \cap -B) = (1/4)\cdot(4.5) = 1/5$, while $p(A + B) = p((A \cap -B) \cup (-A \cap B)) = p(A \cap -B) + p(-A \cap B) = (1/4)\cdot(4/5) + (3/4)\cdot(1/5) = 7/20$ using disjointedness for the addition and independence for the multiplication. So $p(A|A + B) = (1/5)/(7/20) = 4/7$, just over half.

(c) Suppose $p(B \cap C) \neq 0$. It will suffice to show the first equality $(p_B)_C = p_{B \cap C}$, for the last one then holds by interchanging variables and the middle one is trivial. Proceeding from LHS to RHS, $(p_B)_C(A) = p_B(A|C) = p_B(A \cap C)/p_B(C) = p(A \cap C)|B/p(C|B) = [p(A \cap C \cap B)/p(B)]/[p(C \cap B)/p(B)] = p(A \cap C \cap B)/p(C \cap B)] = p(A|B \cap C) = p_{B \cap C}(A)$.

Exercise 6.9 (4) Independence

(a) A fair coin is tossed three times, with independence of tosses. Consider the events T_1, T_2, T_3, that the first, second, third tosses respectively give tails, and the event C that we get exactly two tails in a row out of the three tosses. Show that (i) T_1 and C are independent, but (ii) T_2 and C are not independent. (iii) How does this illustrate failure of transitivity for independence?

(b) Show that when A is independent of B and $0 \neq p(B) \neq 1$ then $p(A|B) = p(A|-B)$.

(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 with four elements and three events A, B, C that are pairwise independent but not jointly so.
(e) Show that when $1 \neq p(B) \neq 0$, $p(A) = p(A|B)\cdot p(B) + p(A|{-}B)\cdot p({-}B)$.

Solution

(a) (i) Noting the words 'exactly' and 'in a row', $C = (T_1 \cap T_2 \cap {-}T_3) \cup ({-}T_1 \cap T_2 \cap T_3)$ so $p(C) = 1/8 + 1/8 = ¼$ while $p(T_1 \cap C) = p(T_1 \cap T_2 {-}T_3) = 1/8$. So $p(T_1|C) = p(T_1 \cap C)/p(C) = ½ = p(T_1)$ as required for independence of T_1 from C. (ii) But $C \subseteq T_2$ so $T_2 \cap C = C$ so $p(T_2|C) = p(C)/p(C) = 1 \neq p(T_2)$ showing dependence of C on T_2. (iii) This shows failure of transitivity, since we have: C independent of T_1 by symmetry applied to (i), T_1 independent of T_2 by the statement of the problem, but C is dependent on T_2 by (ii).
(b) Suppose A is independent of B. Then, by Exercise 6.6 (2), A is also independent of $-B$. Suppose also $0 \neq p(B) \neq 1$. Then $p(A|B)$ and $p(A|{-}B)$ are well-defined and by the definition of independence we have $p(A|B) = p(A) = p(A|{-}B)$ as desired.
(c) We use the multiplicative definition. Suppose that $p(A \cap B) = p(A)\cdot p(B)$, $p(A \cap C) = p(A)\cdot p(C)$, and B is disjoint from C. We need to show that $p(A (B \cup C) = p(A)\cdot p(B \cup C)$. Now, using the last supposition, RHS $= p(A)\cdot(p(B) + p(C)) = p(A)\cdot p(B) + p(A)\cdot p(C) = p(A \cap B) + p(A \cap C) = p((A \cap B) \cup (A \cap C))$ = LHS.
(d) Put $S = \{a_1,a_2,a_3,a_4\}$, each $p(a_i) = 1/4$, $X = \{a_1,a_2\}$, $Y = \{a_2,a_3\}$, $Z = \{a_3,a_1\}$. Then $p(X) = p(Y) = p(Z) = 1/2 = p(X|Y) = p(Y|Z) = p(Z|X)$ so that (using symmetry) the sets X, Y, Z are pairwise independent, but $p(X \cap Y \cap Z) = 0$ $p(X)\cdot p(Y)\cdot p(Z) = 1/8$.
(e) Suppose that $1 \neq p(B) \neq 0$ so that the RHS is well-defined. Then RHS $= [p(A \cap B)/p(B)]\cdot p(B) + [p(A \cap {-}B)/p({-}B)]\cdot p({-}B) = p(A \cap B) + p(A \cap {-}B) = p(A)$.

Exercise 6.9 (5) Bayes' rule

(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.

Solution

(a) We are given $p(F) = 0.3$, $p(V) = 0.4$, $p(F|V) = 0.2$; we want to calculate $p(V|F)$. Bayes' rule tells us that $p(V|F) = p(F|V)\cdot p(V)/p(F)$ and we have all the information needed to complete the RHS as $0.2\cdot 0.4/0.3 = 0.27$ to two decimal places.

(b) We are given $p(A) = 0.03$, $p(P|A) = 0.98$, $p(P|-A) = 0.07$ and we want to find (i) $p(A|P)$, (ii) $p(-A|-P)$.

For (i), by Bayes' rule $p(A|P) = p(P|A)\cdot p(A)/p(P)$. We are provided with the values of two components of the RHS, $p(P|A)$ and $p(A)$, but we are not explicitly given the third component, $p(P)$. Nevertheless, we can calculate it using Exercise 6 (4) (e): $p(P) = [p(P|A)\cdot p(A)] + [p(P|-A)\cdot p(-A)] = (0.98\cdot 0.03) + (0.07\cdot 0.97) = 0.0973$. So, $p(A|P) = 0.98\cdot 0.03/0.0973 = 0.302$ to three decimal places.

For (ii), by Bayes' rule, $p(-A|-P) = p(-P|-A)\cdot p(-A)/p(-P)$. We are not given any of the right hand values explicitly, but we can calculate each of them: $p(-A) = 1 - p(A) = 0.97$, $p(-P) = 1 - p(P) = 0.9027$ and, using Exercise 6 (4) (e) again, $p(-P|-A) = 1 - p(P|-A) = 0.93$. Plugging these values into the RHS we get $p(-A|-P) = 0.93\cdot 0.97/0.9027 = 0.999$ to three decimal places.

Exercise 6.9 (6) Expectation

(a) 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?

(b) A loaded coin has $p(H) = 1/4$, $p(T) = 3/4$. It is tossed twice. Let f be the payoff function, on an appropriate sample space, that gives the number of heads that appear in the two tosses. Calculate the expectation $\mu = \mu(f,p)$.

Solution

(a) Call the envelopes e_1, e_2, e_3, e_4, which form our sample space. Their payoffs are $f(e_1) = 1$, $f(e_2) = 10$, $f(e_3) = 100$, $f(e_4) = 1{,}000$ and each $p(e_i) = 1/4$. So, the expected gain in dollars is given by $\mu(f,p) = \Sigma_{s\in S}\{f(s)\cdot p(s)\} = 1{,}111/4 = 277.75$.

(b) The possible outcomes are HH, HT, TH, TT, which form our sample space. Their probabilities are respectively $p(HH) = 1/16$, $p(HT) = 3/16$, $p(TH) = 3/16$, $p(TT) = 9/16$. Their payoffs are respectively $f(HH) = 2$, $p(HT) = 1$, $p(TH) = 1$, $p(TT) = 0$. So, $\mu(f,p) = \Sigma_{s\in S}\{f(s)\cdot p(s)\} = 2 + 3 + 3 + 0 = 8$.

6.10 Selected Reading

Introductory texts of discrete mathematics tend to say rather little on probability, but there is considerable coverage in, for example, Seymour Lipschutz & Marc Lipson *Discrete Mathematics*, McGraw Hill, Schaum's Outline Series, 1997 (second edition), chapter 7 and Kenneth Rosen *Discrete Mathematics and its Applications*, McGraw Hill, 2007 (sixth edition), chapter 6. For those wishing to include the infinite case, Lipschutz and Lipson have also written *Probability*, McGraw Hill Schaum's Outline Series, 2000 (second edition), which is a good place to begin.

If you enjoyed 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, Jason Rosenhouse *The Monty Hall Problem*, Oxford University Press, 2009. For more on the base rate fallacy, also see its *Wikipedia* entry. If you are intrigued by the idea of revisionary conditional probability, mentioned at the end of Sect. 6.4.1, you can find an overview by the present author within the paper 'Conditional probability in the loght of qualitative belief change', *Journal of Philosophical Logic* 40: 2011, 121–153.

In your reading, you may find it useful to use Table 6.1 of the present chapter to coordinate with the more traditional terminology and notation that is sometimes used.

Squirrel Math: Trees

7

Chapter Outline

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 ingredients, varieties and uses—more than proving things about them. The flavour of the chapter, except for its last section, is thus rather different from that of the preceding one on probability: much more use of spatial intuition, rather less in the way of verifications.

We begin by looking at trees in their most intuitive form—*rooted* (aka *directed*) trees, first of all quite naked and then clothed with *labels* and finally *ordered*. 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 fifteen elements a, …, 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 these notions are defined in Chap. 2).
- 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

D. Makinson, *Sets, Logic and Maths for Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-42218-9_7

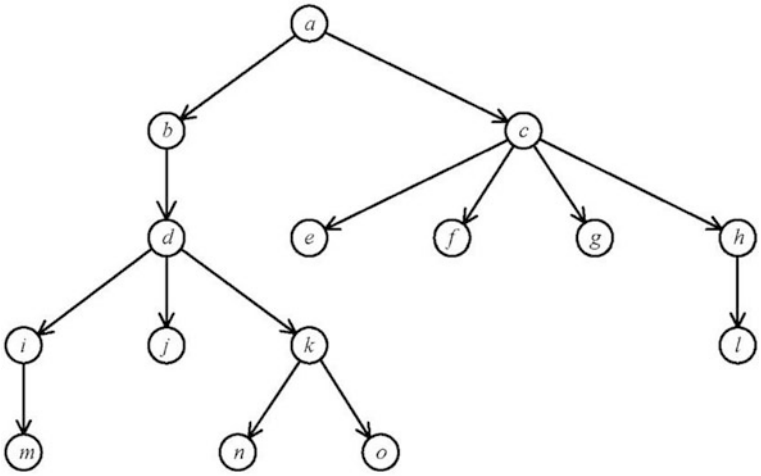

Fig. 7.1 Example of a rooted tree

Chap. 2, all other elements are in the transitive closure of the singleton $\{a\}$ under the relation. This is called the *root* of the tree, and it is unique.

- Paths may fork or continue without forking. Eventually they all stop, since the tree is finite. Thus from the node a we may pass by forking 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 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 path relation is intransitive.

Given these features, it is clear that in our example, and indeed in general, we can leave the arrow-heads off the links, with the direction of the relation understood implicitly from the layout.

Exercise 7.1

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 already extracted from the figure.

(a) How many leaves does the tree have? How many branches?
(b) Why are none of (b,d,i), (a,c,h), (a,b,k,n) branches of the tree?

(c) 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 shortest and longest link-lengths of branches in the tree.
(d) 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?
(e) 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?
(f) 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?
(g) 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.

Solution

(a) There are 8 leaves: m,j,n,o,e,f,g,l and 8 branches. Clearly, there is a bijection between the branches and the leaves at which they terminate.
(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.
(c) The number of nodes in a branch is one greater than its link-length. Shortest link-length: 2. Longest: 4.
(d) 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.
(e) In each case, no: the nodes n, l respectively would have two arrows going to them.
(f) 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, with a new root p.
(g) Nodes n, m are siblings iff they are distinct and have the same parent. Descendant is the transitive closure of child, ancestor is the transitive closure of parent. Node d has no siblings, while its descendants are i, j, k, m, n, o and its ancestors are a, b.

7.2 Rooted Trees

After this intuitive review of the example, we pass to the general definition of rooted trees. There are two ways of doing this. One is *explicit*, giving a necessary and sufficient condition for a structure to be one. The other is *recursive*, defining first the smallest rooted one and a procedure for forming larger ones out of smaller ones. As usual, the recursive approach tends to be better for the computer, though not always for the human working manually with small examples. Each way delivers its particular insights, and problems are sometimes more easily solved using one than the other. In particular, the recursive approach provides support for inductive proofs.

Whichever definition we use, a limiting-case question poses itself. 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 that 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 $(a_i, a_{i+1}) \in R$ for all i with $0 \leq i < n$.

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 be so counted if we wished, but that would tend to complicate formulations down the line, where we usually want to exclude the empty and singleton cases. We do, however, admit the empty tree as a tree.

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 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 uniqueness of paths, as required in the definition of a rooted tree. Recall from Chap. 2 (end-of-chapter Exercise 2.3 (d)) that a relation R over a set is said to be *acyclic* iff there is no path from any element x of the set to itself, i.e. no path $x_0, \ldots, x_n$ ($n \geq 1$) with $x_0 = x_n$.

- For *acyclicity*, Suppose for *reductio* that a tree (T,R) fails acyclicity. Then there is a path from some element $x \in T$ to itself. So the tree is not empty, has a root a, and $x \neq a$. But by the definition of a tree, there is also a unique path from a to x. Form the composite path made up of that path from a to x followed by the one from x to itself. This is clearly another path from a 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 we have one from a to a, giving us a contradiction.
- To show that the root has *no parent*, let $a \in T$ be the root of the tree and suppose 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 x. Now suppose that b has another parent 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, *node-height*(x) = *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, one of these two measures may be more convenient to use than the other. The height (whether in terms of nodes or links) of an entire tree may be defined to be zero in the limiting case of the empty tree, otherwise the greatest height (node/link respectively) of any of its nodes.

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 limiting case of the root node, since we are not counting one-element sequences as paths.

Exercise 7.2.1

(a) What is the (link/node)-height of the tree 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) We have shown that the relation R of a rooted tree (T,R) is acyclic. Show that this immediately implies its irreflexivity and asymmetry.

Solution

(a) The link-height of the tree in Fig. 7.1 is 4, 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, i.e. 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.
(c) Irreflexivity and asymmetry both follow from acyclicity by taking the cases $n = 1$ and $n = 2$. The argument was given in more detail back in Chap. 2, in the solution to end-of-chapter Exercise 2.3 (d).

7.2.2 Recursive Definitions

We turn now to recursive definitions of a rooted tree. There are two main ways of expressing them: the recursion step can bring in a *new leaf*, or a *new root*. The simplest, and most intuitive, is the 'new leaf' definition, but the 'new root' one often lends itself more easily to inductive proofs and computations, and so is more popular in computer science.

We begin with definition via fresh leaves. Basis: The empty tree is a rooted tree (although it has no root) and so too is any singleton $T = (\{a\},\varnothing)$ with a as root. Recursion step: 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. The root of the new tree remains that of the old one. This definition corresponds closely to most

people's thoughts when they draw trees by hand. One usually starts by placing a root at the top of a page and adds new nodes by lengthening or dividing a branch.

The definition via fresh root goes in the reverse direction. The basis is the same: The empty tree is a rooted tree, and so too is any singleton $T = (\{a\},\varnothing)$ with a as root. Recursion step: Whenever we have a finite non-empty disjoint collection of non-empty 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 (so there are no new links when the given tree is empty). Saying the same more 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 $S = (\cup \{R_i\}_{i \leq n}) \cup \{(a,b): b \in B\}_{i \leq n}$ 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, where the T_i are represented schematically by triangles. They are the *immediate (proper) subtrees* of the entire tree.

Exercise 7.2.2

(a) Use the 'new leaf' definition to show that any tree with $n \geq 1$ nodes has $n - 1$ links.
(b) Do the same using the 'new root' recursive definition.

Solution

(a) Basis: If the tree has just one node then it is of the form $T = (\{a\}, \varnothing)$ with a as root and empty set of links, so the property holds trivially. Induction step: Let $k \geq 1$ and suppose the property holds for all trees with k nodes. Consider any tree with $k + 1$ nodes. Then it is formed from a tree with k nodes by taking a fresh item x, choosing a node b in the given tree, and linking b to x. This adds 1 to both the number of nodes and to the number of links, thus preserving the desired property.
(b) Basis: Again, if the tree has just one node then it is of the form $T = (\{a\}, \varnothing)$ with a as root and empty set of links, so the property holds trivially. Induction step: Let $k > 1$ and suppose the property holds for all trees with less than k nodes. Consider any tree with k nodes. Then it is formed from a disjoint

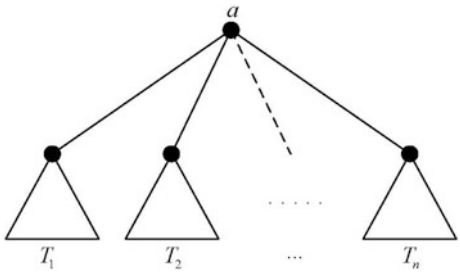

Fig. 7.2 Recursive definition of a tree, adding a new root

collection of $m \geq 1$ non-empty trees, each with less than k nodes, by adding a new root and links to the m given roots. The tree formed has $(n_1 + \ldots + n_m) + 1$ nodes, where each n_i is the number of nodes of the nth given tree, along with $(l_1 + \ldots + l_m) + m$ links. But, by the induction hypothesis, each $n_i = l_i + 1$, so $(n_1 + \ldots + n_m) + 1 = (l_1 + \ldots + l_m) + m + 1$ and we are done. *End of solution.*

A more mathematically oriented presentation would prove the equivalence of the explicit definition and each of the two recursive ones; we will not do so here.

7.3 Working with Trees

Now that we have a definition of rooted trees, we look at some situations in which they can come up and ways of annotating or, one might say, labelling, decorating or clothing them.

7.3.1 Trees Grow Everywhere

What are some typical examples of tree structures in everyday life? The first that springs to mind may 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) and disregard the possibility of same-sex parentage, 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 is the staff organigram, 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, although not always, 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—for example into subject, verb, object, adjectives, adverbs and subordinate clauses. Parsing trees arise in formal languages of computer science, mathematics and logic, when we consider the 'grammatical' structure of a formal expression. For example, the syntactic analysis of a program, a quadratic equation or a formula of logic may take the form of a tree.

In logic, trees arise in further ways. A *formal derivation* can be represented as a tree, with conclusion as root and assumptions as leaves. One popular way for checking whether a formula of propositional logic is a tautology proceeds by constructing what is called its *semantic decomposition tree* or, more briefly, *truth-tree*.

We will look at parsing trees for logical formulae in Chaps. 8 and 9, proof trees in Chap. 10 and truth-trees in Chaps. 8 and 11. But for the present we continue with general notions.

Alice Box: Which way up?

Alice Before going on, why is the tree in Fig. 7.1 upside down, with the root at the top?

Hatter Sometimes we draw trees upside-down, sometimes the right way up. It depends, in part, on how we are constructing it.

Alice what do you mean?

Hatter If we are thinking of the tree as built from the root to the leaves by a 'new leaf' recursion then (given the conventions in most cultures of how to read a page) it is natural to write the root at the top of your page and work downwards. On the other hand, if we are using a 'new root' construction, then it makes sense to put the leaves, which remain leaves throughout the procedure, at the top and work down to the root.

Alice Any examples?

Hatter Parsing trees, whether in logic, mathematics, computing or the grammars of natural languages, usually proceed from the root, which is therefore usually written at the top. The same holds for the truth-trees as well as derivation-trees only using second-level rules (see Chaps. 8 through 11 on logic). But the derivation trees in what is called a Fregean or Hilbertian axiom system (there is an example in Chap. 11) are constructed from the leaves and so written with them at the top.

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 significance; they are thought of as just 'points'.

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, i.e. defined on a

subset of T. Given a node x, $\lambda(x)$ indicates some object or property that is associated with that node. 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 interests us is the labelling function. In a diagram, we write the labels next to the nodes, and may even omit giving 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, you may put labels indicating the date of creation or of last modification. Different folders may thus have the same labels, and in a graphic representation, dots at different positions in the tree may have the same annotations. When that happens, a node cannot be fully identified merely by its label.

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 that is not always convenient. Sometimes, it is useful to label *links* rather than (or in addition to) nodes. For example, we might want to represent some kind of distance, or cost of passage, between a parent node and its children. In some cases, the decision whether to place a label on a node or on a link is just a matter of convention and elegance; 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, …, n to record the ordering. Graphically, we may write these from left to right. 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 7.1: An arithmetic expression: Consider the expression $(5 + (2 \times x)) - ((y - 3) + (2 + x))$. It is formed from two sub-expressions, $5 + (2 \times x)$ and $(y - 3) + (2 + x)$ by inserting a sign for the operation of subtraction (plus brackets). As subtraction 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 sub-expressions. Continuing this process until decomposition is no longer possible, we get the labelled and ordered tree in Fig. 7.3.

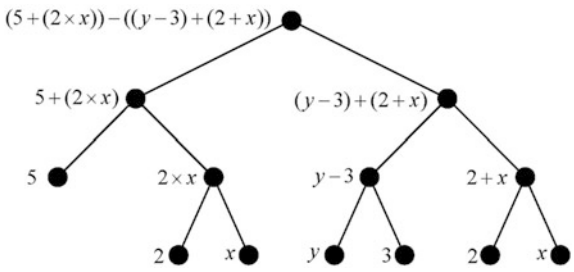

Fig. 7.3 Tree for syntactic structure of an arithmetic expression

Alice Box: Drawing trees

Alice Why do we have to regard the expressions written alongside the nodes as *labels*? Couldn't we just take them to be the *names* of the nodes? Does it make a difference?

Hatter It does. As remarked in the text, a labelling function need not be injective. In Fig. 7.3, we have two different nodes with the label 2, as well as two labelled by the variable x. These two nodes are distinct, with different places in the tree corresponding to different roles in the arithmetic expression, but they are distinguished as nodes, rather than by their labels.

Alice I see. But where, in Fig. 7.3, 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 have used a reading convention to take care of that. To reduce clutter, we wrote the children in the required order from left to right on the paper.

Alice So this tree is different from its mirror image, in which left and right are reversed on the page?

Hatter Yes, since it meant as an ordered tree. If it is meant as an unordered tree, then it and its mirror image are the same.

Actually, the tree in Fig. 7.3 can be written more economically. The interior nodes (that is, the non-leaves) need not be labelled with entire sub-expressions; 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 left-to-right order. It can thus be regarded as an alternative presentation of the same tree.

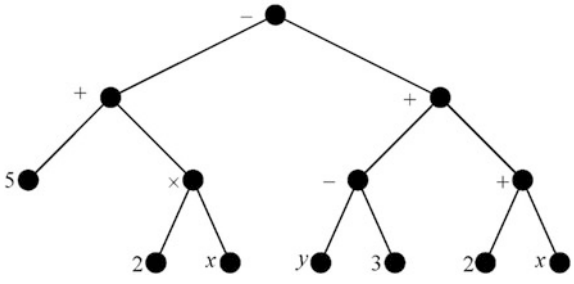

Fig. 7.4 Syntactic structure with compressed labels

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 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 7.2: A logical formula: Syntactic decomposition trees arise not only in arithmetic, but for any kind of 'algebraic' expression in mathematics or logic. For example, the formula $(p \leftrightarrow (q \vee r)) \rightarrow ((s \wedge t) \leftrightarrow (q \leftrightarrow r))$ of propositional logic has a structure that can also be represented by a syntactic decomposition tree.

Exercise 7.3.2
Draw the syntactic decomposition tree for the formula of Example 2, once with full labelling and again with abbreviated labelling.

Solution
It is exactly the same ordered tree as that in Fig. 7.3, compressed in Fig. 7.4, but with different labels: the replace numerals and variables 5, 2, x, y, 3 respectively by the proposition letters p, q, r, s, t and the operations +, $\times$, $-$ by the connectives $\leftrightarrow$, $\vee$, $\rightarrow$. Remark: We are not suggesting that there is a correspondence between the meanings of the operations +, $\times$, $-$ and the connectives $\leftrightarrow$, $\vee$, $\rightarrow$ respectively; we are merely observing that they play analogous roles in the grammatical structure of the two expressions. There are, indeed, interesting connections of content between connectives of propositional logic and certain operations of arithmetic, but they are not the same as the syntactic correspondences in this example.

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 $5 + 2 \times x$ is ambiguous unless brackets are put in or a convention is adopted—as indeed is usually done for multiplication—that it 'binds more strongly than', or 'has priority over' addition. Likewise, the logical expression $p \leftrightarrow q \vee r$ is ambiguous unless parentheses are inserted or analogous conventions employed.

In practice, we frequently adopt such conventions, and also drop brackets whenever different syntactic structures have the same semantic content, as for example with $x + y + z$ and with $p \leftrightarrow q \leftrightarrow r$. This helps prevent visual overload and is part of the endless battle against the tyranny of notation—the latter of which is, in turn, a weapon in the battle against the chaos of muddle and confusion.

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 some of the left or right brackets without ambiguity. The formula $(p \leftrightarrow (q \vee r)) \rightarrow ((s \wedge t) \leftrightarrow (q \leftrightarrow r))$ of Example 2 would be written as $p \leftrightarrow .q \vee r: \rightarrow :s \wedge t. \leftrightarrow .q \leftrightarrow r$. However the system did not catch on and died out.

Is there any systematic way, in arithmetic, logic and other domains, of doing without brackets altogether, without setting elaborate priority conventions for all operations or alternative devices like dots? There is, and it was first noted by the philosopher and logician Jan 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 $(5 + (2 \times x)) - ((y - 3) + (2 + x))$ is written as $- + 5 \times 2x + \ -y3 + 2x$.

To decipher this last sequence, construct a decomposition tree for it but working from the leaves to the root. The leaves of the tree will be labelled by the constants and variables 5, 2, x, y, 3. Then $+2x$ 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 *prefix* or *Polish notation*, the latter term after the nationality of its inventor. It is, effect, the same as what we do in algebra when we write a two-argument function as $f(x,y)$, except that, there, we are more or less redundantly retaining brackets and commas to make the structure more visible to the naked eye. One-place operations are written, as usual, before their argument.

There is also *postfix* (aka *reverse Polish*) notation, where two-place operations are written after their two arguments, and single-place ones are also placed after their arguments. This likewise makes bracketing redundant, and is sometimes used in piecemeal fashion by mathematicians working in category theory. If each operation sign carries with it an indication of its arity, the prefixing and postfixing systems can be extended to cover expressions with operations of more than two arguments.

The ordinary way of writing arithmetical or logical operations of two-places is called *infix* notation, since the operation is written between the arguments.

Exercise 7.4

(a) Put the propositional formula $(p \to \neg q) \to (\neg r \leftrightarrow (s \wedge \neg p))$ in both prefix and postfix notation.
(b) Insert a few redundant brackets into the outputs just obtained, to make them friendlier to non-Polish readers.

Solution

(a) In prefix notation: $\to\to p\neg q\leftrightarrow\neg r\wedge s\neg p$. In postfix, remembering that the one-place operation of negation is placed after its argument: $pq\neg\to r\neg sp\neg\wedge\leftrightarrow\to$.
(b) In prefix notation, but with a few brackets to aid understanding: $\to(\to p\neg q)(\leftrightarrow(\neg r(\wedge s\neg p))$. Postfix with a few brackets,: $(pq\neg\to)(r\neg(sp\neg\wedge)\leftrightarrow)\to$.

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 over 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, perhaps 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 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 understand them as given by slanting the link left or right, never placing 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 (which are, in effect, unary trees). Very efficient algorithms have been devised for working with binary trees, in particular for *searching*, *deleting*, and *inserting* items in 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.

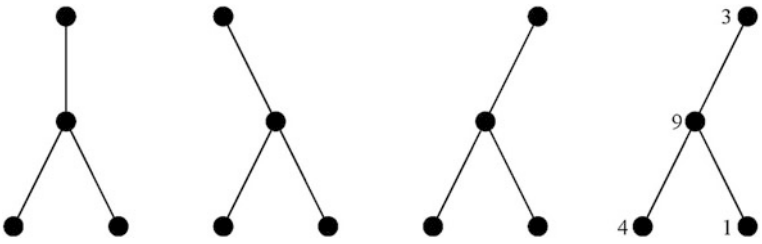

Fig. 7.5 Which are binary trees?

What is a binary *search* tree? Suppose we have a binary tree and a labelling function λ: $T \rightarrow L$ 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 (that is, 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)
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 be enough to disqualify it. Tree (b) isn't either, because $5 > 3$. Nor is (c) a binary

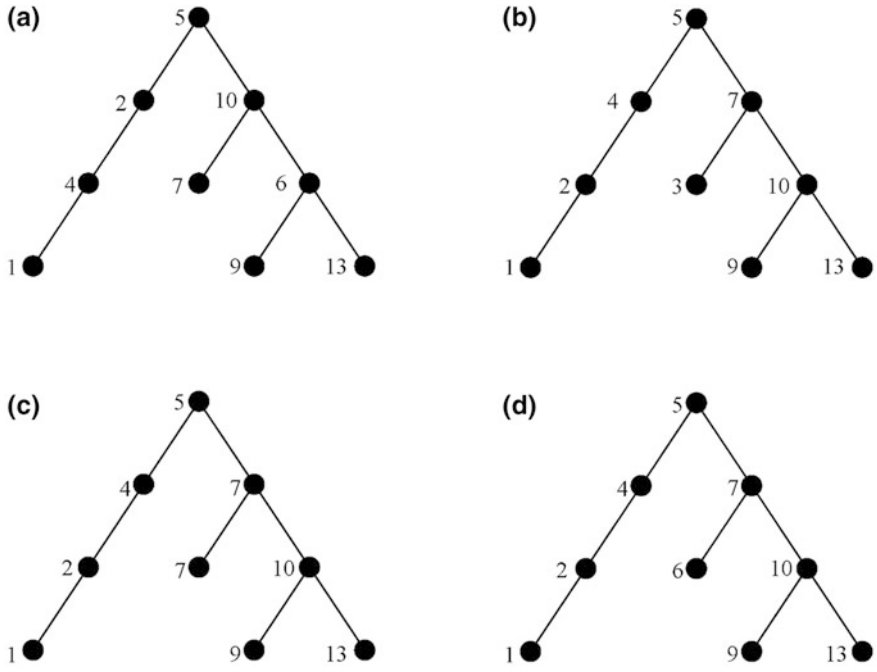

Fig. 7.6 Which are binary search trees?

search tree in our sense of the term since the labelling is not injective. Only (d) is a binary search tree. *End of solution*.

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. As the labelling was, for simplicity, assumed to be injective, we can abuse notation by writing a, x etc. in place of $\lambda(a)$, $\lambda(x)$.

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)

(a) Execute the algorithm manually to search for 6 in the binary search tree (d) of Fig. 7.6.
(b) Do the same to search for 8.

Solution

(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*. *End of solution*.

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, while the one for deletion is rather more complex since it deletes from anywhere in the tree.

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)
Consider again the binary search tree in Fig. 7.6. What is the tree that results from inserting a node 8 by means of the algorithm sketched above?

Solution
Searching for 8 in the tree, in Exercise 7.5 (2) (b), we reached node 9 and noted that 8 should be in its left subtree, but that 9 is a leaf. So we add 8 as left child of 9. *End of solution*.

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.

Evidently, the shape of a tree constructed by this algorithm 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.

Normally, 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 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. 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 in Fig. 7.1, at the beginning of this chapter, and play around with it. Imagine that instead of the figure made of dots and lines on paper, we have physical version of the same, made of beads connected with pieces of string or wire, lying on a table facing us. We can then pull the root a down and push up b, say, so that it becomes the root. Or we can rearrange the beads on the table so that the assembly 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 structure as an unrooted tree.

7.6.1 Definition

Formally, an *unrooted tree* (aka *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 arrow-heads from the diagram of R (if there were any) and omitting any convention for reading a direction into the links. 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), i.e. 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 $(y,x) \in S$.

It appears that the mathematical concept of an unrooted tree was first articulated in the nineteenth century and popularized by Arthur Cayley in 1857 when discussing problems of chemistry. 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 applications 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'.

Alice Box: Unrooted trees

Alice That's a nice, simple definition of unrooted trees—provided we are coming to unrooted trees from rooted ones. 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 in Sect. 7.6.2, after noting some properties emerging from our present definition.

Exercise 7.6.1
In Exercise 7.2.2, we observed 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.

Solution
Consider any unrooted tree (T,S) with $n \geq 1$ vertices. By the above definition of an unrooted tree, S is the symmetric closure of a relation R such that (T,R) is a rooted

tree. Since the carrier set T is the same, (T,R) has n nodes. So, by Exercise 7.2.2 (2), (T,R) has $n - 1$ links. But, since $S = R \cup R^{-1}$, there is a bijection between the links in R and the edges in S, so (T,S) has exactly $n - 1$ edges.

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), i.e. for any two distinct elements x, y of T, there is an S-path from x to y, i.e. 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 and using the assumption that $y \neq x$, we have a R-path from x to y, and this is an S-path. In the case that $a = y$, we have a R-path from y to x and thus, running it in reverse (which is legitimate since S is symmetric), we have a S-path from x to y. Finally, if $a \neq x$ and $a \neq y$ we know, again from the definition of a rooted tree, that there are R-paths from a to each of x, y considered separately. Reversing the path from a to x and composing with the path from a to y, we have an S-path from x through a to y.

Less obvious is the fact that S is a *minimal* connected relation over T. In other words, no proper sub-relation of S is connected over T. The following proof is the most intricate in the entire book and needs to be read carefully—then re-read it after looking at the logic box nearby.

Since any super-relation of a connected relation is connected, it suffices to show that, for any pair $(x,y) \in S$, the relation $S' = S\backslash\{(x,y)\}$ is not connected over T. Choose any $(x,y) \in S$. Then either $(x,y) \in R$ or $(y,x) \in R$. Consider the former case; the argument in the latter case is similar. Since $(x,y) \in R$, we know by the definition of a rooted tree that $y \neq a$ where a is the root of (T,R). So, to show that S' is not connected over T, it will suffice to show that there is no S'-path from a to y.

Suppose for *reductio ad absurdum* that there is such a S'-path, i.e. a finite sequence $\sigma = (a_0, \ldots, a_n)$ $(n \geq 1)$ 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 σ is a shortest such path so that, in particular, no $a_i = a_{i+2}$. 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 alternative is impossible. For, suppose for *reductio* that $(a_{n-1},a_n) \in R$; then since $a_n = y$ we have $(a_{n-1},y) \in R$ and so, since by supposition $(x,y) \in R$ and no node can have two R-parents, we 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 on i from 0 to n, we must have each $(a_{n-i}, a_{n-i-1}) \in R$ for otherwise, using the definition of S, we would have an a_{n-i} with two distinct R-parents, namely a_{n-i-1} and a_{n-i+1}. This gives us an R-path from y to the root a, which is impossible by the definition of a rooted tree.

Logic Box: An interesting proof

From a logical point of view, the proof that S is a *minimal* connected relation over T is interesting, because it has two features that we have not seen in earlier verifications. (1) It uses a proof by contradiction that is within the scope of another one (as well as being in one arm of a proof by cases) as flagged by the phrases 'suppose for *reductio*'. (2) It makes a 'without loss of generality' (abbreviation: wlog) step.

Both features are legitimate. (1) There is no limit to the number of possible layers of proof by contradiction one within another, just as there is no limit to the levels of conditional proof that may be embedded within each other or, for that matter, to the number of levels of proof by cases mixed with conditional proof and proof by contradiction. (2) Wlog steps can serve to simplify presentation, notably by reducing the number of variables deployed. Typically, such steps arise when we are given, or have already shown, that there is an item (in the above example, the finite sequence σ) with a certain property P. When it can easily be checked that if there is such an item then there is also one that has both that property P and another property Q, then we simply say that *we may assume without loss of generality that the item also has the property Q*. This avoids dragging in additional variables in our formulation, in the example a primed variable σ'.

Thus, while feature (1) is a matter of logical power, (2) is a device for notational economy in certain situations. We will see some more examples of the wlog gambit in later exercises.

Another important 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, i.e. 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 nodes.

For example, the cycles a,b,c,b,a and a,b,c,e,c,b,a are not simple, since each contains both links (b,c) and (c,b) so that the edge $\{b,c\}$ appears twice. The cycle a,b,c,e,b,c,a is not simple either, as it repeats the link (b,c), so again the edge $\{b,c\}$ appears twice. On the other hand, the cycle a,b,c,d,e,c,a is simple: despite the repetition of vertices; there is no repetition of edges (i.e. no repetition or reversal of links).

Exercise 7.6.2 (1)
Take the first-mentioned cycle a,b,c,b,a and add c in third place, i.e. after the first b. Is it simple?

Solution
The new cycle is a,b,c,c,b,a. It is not simple, because it still contains links (b,c) and (c,b), so the edge $\{b,c\}$ appears twice. *End of solution*.

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 when we unpack the definition of simplicity we have 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, (T,S) is acyclic?
Hatter	Not so fast! Acyclicity, as we have defined it in Sect. 7.2.1, means that there are no cycles at all. Here we are claiming less: we are saying only that there are no *simple* cycles in the unrooted tree. Indeed, that is all we can ask for; since S is symmetric, the unrooted tree will contain non-simple cycles x,y,x.
Alice	I hope I don't get mixed up with the two negations…
Hatter	If you do, rely on the positive formulation above: 'no simple cycles' means 'every cycle has a repeated edge'.

To prove the claim just made, 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 (wlog again) 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 Sect. 7.2.1 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 bringing us to a contradiction.

Indeed, we can go further and prove a stronger result: 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 $S' = S \cup \{(x,y), (y,x)\}$ is an unrooted tree and we can show that it 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.

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, they 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-1edges, 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 in Sect. 7.6.1: how could we define unrooted trees directly, without reference to rooted ones?

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, one need only take the recursive definition for rooted trees and tweak it a little. But let's stop there.

7.6.3 Spanning Trees

The second of the six characterizations of an unrooted tree in the bullet points at the end of Sect. 7.6.2 gives rise to an important practical problem. A symmetric relation S that minimally connects a set (thus satisfying the condition) is known as a *spanning tree* for the set. 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 will be many S-paths between vertices, and perhaps 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 $n = \#(A)$; in the case of an irreflexive relation every element of A may be related to every other one, still 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 sub-relation $S' \subseteq S$ that still connects A? In other words, is there an algorithm to find a spanning tree for A?

Exercise 7.6.3
Check the figures $n(n+1)/2$ and $n(n-1)/2$ above, using material from Chap. 5.

Solution
In the case of an irreflexive symmetric relation, we are in effect looking at the collection of all two-element subsets of A. this is given by the formula $n!/k!(n-k)!$ of Table 5.4 of Chap. 5 with $n = 2$, that is, $n!/2!(n-2)!$, which simplifies to $n(n-1)/2$. For an arbitrary symmetric relation, we need to add n for the singleton subsets of A, giving us $[n(n-1)/2] + n$, which simplifies to $n(n+1)/2$. *End of solution*.

Clearly there is a 'top down' procedure that does the job of finding a spanning tree for A. 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 start removing 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 in Sect. 7.6.2, S' will also be a *minimal* symmetric relation connecting A—which is what we are looking for.

Both procedures are non-deterministic algorithms, since they allow choice from a finite range of edges at each step; they can be rendered deterministic by specifying an order on the set of all possible edges over the underlying set. In general, the 'bottom up' procedure 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.

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, i.e. minimize *the sum of their weights*. 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.

7.7 End-of-Chapter Exercises

Exercise 7.7 (1) Properties of rooted trees

(a) Show that the relation R of any finite rooted tree (T,R) is intransitive.
(b) Consider the 'new root' definition of a rooted tree. What is the relation between the height (whether understood as node-height or link-height) of the 'output' tree issuing from an application of the recursion step, to the heights of its 'input' immediate subtrees?

Solution

(a) We use the basic properties established in Sect. 7.2.1. Let (T,R) be a finite rooted tree. Suppose for *reductio* that there are $x, y, z \in T$ with (x,y), (y,z), $(x,z) \in R$. Since $(x,y) \in R$ the irreflexivity of R (established in Sect. 7.2.1) tells us that $x \neq y$ so z has two parents, contradicting the uniqueness of parents (also established in Sect. 7.2.1).
(b) It is immediate from the recursive definition that each branch in the output tree has height one more than that of the branch in the input tree from which it is formed, so the height of the output tree itself is one larger than the maximal height of its input trees.

Exercise 7.7 (2) Labelled trees

(a) Construct the syntactic decomposition tree of the arithmetic expression $-(y^3 + (x - 5))$. What is the most salient difference between it and the decomposition trees in Figs. 7.3 and 7.4?
(b) What would be special about the shape of syntactic decomposition tree of an arithmetic expression formed from two-place arithmetic operations alone?
(c) Write the arithmetic expression $-(7 \cdot y + (x - 5))$ in prefix notation and in postfix notation.

Solution

(a) The initial minus sign is a one-place operation, and expressions y^z are shorthand for applications of the two-place operation $exp(y,z)$ of taking y to the power z. The appropriate syntactic decomposition tree is thus as in Fig. 7.7. The most salient difference to the trees of Figs. 7.3 and 7.4 is that the root has only one child.

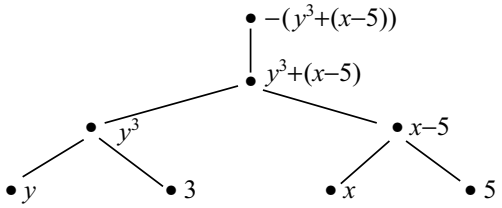

Fig. 7.7 Syntactic decomposition tree for the expression $-(y^3 + (x - 5))$

(b) In such a tree, every non-leaf node has exactly two children.
(c) For unique readability to be guaranteed, each sign should have a unique number of argument places. So, we need to write say $\sim$ for the one-place minus sign to distinguish it from the two-place minus sign $-$. Then we can write $-(7 \cdot y + (x - 5))$ in prefix notation as $\sim + \cdot 7y - x5$ and in postfix notation as $7y \cdot x5 - + \sim$.

Exercise 7.7 (3) Binary search trees

(a) Consider the set of letters $\{g, c, m, b, i, h, a\}$ and its usual alphabetical ordering. Construct a binary search tree for them, such that all branches are of equal height and all non-leaves have two children.
(b) In the tree you have constructed, trace the steps in a search for the letter c, using the search algorithm in the text. Do the same for the letter j.

Solution

(a) See Fig. 7.8.
(b) For c: Compare c with the root g; since $c < g$, compare c with the left child b of g; since $b < c$, compare c with the right child c of b; since $c = c$, print *yes* and stop. For j: Compare j with the root g; since $g < j$, compare j with the right child i of g; since $i < j$, compare j with the right child m of b; since $j < m$ and m has no left child, print *no* and stop.

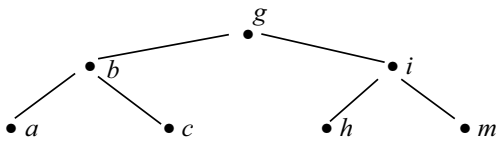

Fig. 7.8 Binary search tree for Exercise 7 (3)

Exercise 7.7 (4) Unrooted trees
Consider the *complete* (irreflexive) graph (G,R) with five vertices 1, …, 5, 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.
(b) Construct a different spanning tree for this by the bottom-up procedure, in such a way that it is determined by a rooted tree in which all non-leaves have exactly two children.

Solution

(a) For visual intuition, draw the complete irreflexive graph over G as a five-pointed star, numbered clockwise and with edges around the perimeter. Following the algorithm, take out successively say the edges {1,2}, {2,3}, {3,4}, {4,5}, {5,1}, {1,3} and stop.
(b) Take the empty set of edges, add to it say {1,2}, {1,3}, {2,4}, {2,5} and stop.

7.8 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 and treating unrooted trees before rooted ones, thus proceeding resolutely from the more general context to the more particular. Richard Johnsonbaugh *Discrete Mathematics*, Pearson 2009 (7th edition) chapter 9 follows this path, while Bernard Kolman et al. *Discrete Mathematical Structures*, Pearson 2009 (6th edition) chapter 7 follows an order closer to the present one. Both texts cover considerably more ground than we have done.

Part III
Logic

Yea and Nay: Propositional Logic

8

Chapter Outline

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 will be the *truth-functional connectives*, the family of concepts around *tautological implication*, the availability of *normal forms* 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 these chapters is only one part of logic, but a very basic part. In a general sense, logic may be understood as the study of *reasoning* or, in even broader 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 given 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

D. Makinson, *Sets, Logic and Maths for Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-42218-9_8

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 carrying out sufficient contraction to maintain consistency of the whole. A closely related form of belief change, of particular 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, recursion, 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 serious limitation becomes apparent if one reflects on the kind of reasoning that is 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 reached are treated as plausible, reasonable, probable, or convincing given the assumptions made; but they are rarely if ever perfectly 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 these chapters. It provides a basis that is needed before trying to tackle uncertain inference, whether that is expressed in qualitative terms or probabilistically, and before analysing other kinds of belief management. Its study goes back two thousand years; in its 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 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 Chap. 1 and you are encouraged to flip back to review those boxes, for we will not repeat all of the comments made there. However, for easy reference, we recall the truth-tables themselves writing α, β, … for arbitrary propositions.

For the one-place connective 'not', see Table 8.1. Here 1 is for truth, and 0 is for falsehood, often written *T*, *F* respectively. A basic assumption made in the table is the *principle of bivalence*: every proposition under consideration is either true, or false, but not both. Another basic assumption is that this is all that matters for determining the truth-value of the compound item, in this case $\neg\alpha$. In other words, negation may be represented by a function with the two-element set {1,0} as both domain and range. The two-place connectives 'and', 'or', 'if' and 'iff' have truth-tables that can be grouped together as in Table 8.2.

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	$\alpha \wedge \beta$	$\alpha \vee \beta$	$\alpha \rightarrow \beta$	$\beta \rightarrow \alpha$	$\alpha \leftrightarrow \beta$	$\alpha + \beta$
1	1	1	1	1	1	1	0
1	0	0	1	0	1	0	1
0	1	0	1	1	0	0	1
0	0	0	0	1	1	1	0

Alice Box: The principle of bivalence

Alice Can we relax the principle of bivalence?

Hatter Yes, we can, if we are willing to depart from the standard account of logic. There are two main ways of going about it. One is to allow that the classical values are not exhaustive, in other words, 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 idea can, moreover, be accommodated within the first one 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 Not in this book.

It is time to look more closely at the concept of a *truth-function* in two-valued logic. A one-place truth-function, such as the one associated with negation, is simply a function on domain {1,0} into {1,0}. A two-place truth-function, such as those associated with conjunction, disjunction, the conditional and the bi-conditional in Table 8.2, is a function on $\{1,0\}^2$ into {1,0}, and so on for those with more than two arguments.

This immediately suggests all kinds of questions. How many one-place truth-functions are there? How many two-place, and generally *n*-place ones? Are there zero-place truth-functions? Are the specific truth functions given in Tables 8.1 and 8.2 sufficient to represent all of them, or do we need further logical connectives for that? 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 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)

(a) Which of these four truth-functions f_i corresponds to negation?
(b) Can you express the other three one-place truth-functions in terms of connectives with which you are already familiar ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$)?

Solution

(a) Obviously, f_3.
(b) f_2 is the identity function, i.e. $f_2(\alpha) = \alpha$, 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. *End of solution.*

Going on to the two-place truth-functions, the Cartesian product $\{1,0\}^2$ 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 two possible values for the function, which gives us $2^4 = 16$ columns to fill in, i.e. 16 truth-functions.

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

For smooth communication we always write the rows of a truth-table in a standard order. In the 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 $f_1, \ldots, f_{16}$.
(b) Identify which of the 16 columns represent truth-functions with which you are already familiar.
(c) Express the others using combinations of familiar ones.

Solution

(a) To fit the 16 columns onto the page, we write each two-place function $f_i(\alpha,\beta)$ simply as f_i. as in Table 8.4.
(b) Familiar: f_4, f_6, f_{13}, f_{11} represent α, β $\neg\alpha$, $\neg\beta$ respectively; $f_8, f_2, f_5, f_3, f_7, f_{10}$ represent $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \rightarrow \beta$, $\beta \rightarrow \alpha$, $\alpha \leftrightarrow \beta$, $\alpha + \beta$ respectively.
(c) Others: f_1 is the constant function with value 1 and represents say $\alpha \vee \neg\alpha$ (or, say, $\alpha \vee \neg\alpha \vee \beta$ if you want to mention the second argument explicitly); f_{16} is the constant function with value 0 and represents, say, $\alpha \wedge \neg\alpha$; f_9 represents $\neg(\alpha \wedge \beta)$; f_{12} represents $\alpha \wedge \neg\beta$; f_{12} represents $\neg\alpha \wedge \beta$; f_{15} represents $\neg(\alpha \vee \beta)$. You could equally well choose formulae that are logically equivalent (in a sense to be defined shortly)—for example, f_{15} also represents $\neg\alpha \wedge \neg\beta$. *End of solution*.

While ¬ is one-place, ∧, ∨, →, ↔, + are all two-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. In Exercise 8.2 (2) (c) we have checked this out for all the two-place truth-functions. We now show, in a more systematic way, that every truth-function of *any finite number of places*, may be represented using at most the three connectives ¬, ∧, ∨. Then we will see how that set of three may be pared down.

Table 8.4 The two-place truth-functions

α	β	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

The idea may be illustrated looking more carefully at any one of the two-place truth-functions f_i in Table 8.2. Take the rows in which $f_i(\alpha,\beta) = 1$; for example, in the case of f_3 there are three of them. For each such row, form the conjunction $\pm\ \alpha \wedge \pm\ \beta$ where $\pm$ is empty or negation according as the argument has 1 or 0. In the case of f_3, 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: for f_3, this will be $(\alpha \wedge \beta) \vee (\alpha \wedge \neg\beta) \vee (\neg\alpha \wedge \neg\beta)$. That formula expresses the truth-function under consideration.

Why? Look again at the example of f_3, 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, i.e. 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 $\neg$, $\wedge$, $\vee$.

(b) Would it have made a difference if we had written exclusive disjunction + instead of inclusive disjunction $\vee$ in the above procedure for devising a formula to express an arbitrary-*n*-place truth function? Why?

Solution

(a) It is the function f_{16}, with constant value 0, since there are no rows in which it comes out true. All other truth-functions come out true in at least one row. But, as we have already seen, f_{16} can still be represented by, say, $\alpha \wedge \neg\alpha$ and many other equivalent formulae.

(b) No, it would not have made any difference, despite that fact that $\vee$ and $\oplus$ have different truth-tables. The reason is that the two tables differ only in their top row, where α and β are both assigned the value 1 (i.e. are both taken as true), and that row cannot be exemplified under the construction described. For example, for f_3 we generated the formula $(\alpha \wedge \beta) \vee (\alpha \wedge \neg\beta) \vee (\neg\alpha \wedge \neg\beta)$ and $(\alpha \wedge \beta) + (\alpha \wedge \neg\beta) + (\neg\alpha \wedge \neg\beta)$ is its counterpart with +. Now, by the truth-tables for $\neg$ and $\wedge$, there is no way in which the disjuncts $(\alpha \wedge \beta)$ and $(\alpha \wedge \neg\beta)$ can both be true under a given assignment to the elementary parts; likewise for the other two disjunct-pairs. Hence exclusive disjunctions of this kind must always get the same truth values as the corresponding inclusive ones. *End of solution*.

Thus, every truth-function can be expressed using only $\neg$, $\wedge$, $\vee$. Are all three connectives really needed, or can we do the job with even fewer ones? We will return to this question in the next section, after clarifying some fundamental concepts that we have already been using intuitively in examples.

8.3 Logical Relationship and Status

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 a certain kind 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 abstract structures that may be used to interpret them. A good deal of logic thus *looks at the language*, seeing how it corresponds to structures in which it is interpreted. This can take quite some time to get used to, since ordinarily in mathematics we look *through* the language to the structures alone. It is like learning to look at the window pane as well as at the landscape beyond it, as in René Magritte's painting *La clef des champs* or the three levels of visual existence in Maurits Cornelis Escher's engraving *Three worlds* (look them up on the web).

8.3.1 The Language of Propositional Logic

Our language be able to should 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,…, 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 as either countably infinite or finite but of unspecified cardinality. As subscripts are a pain, we usually write elementary letters as p, q, r, … bringing in indices only when we run short of letters or they are convenient for formulating general facts.

The *formulae* (US dialect 'formulas', in some texts 'well-formed formulae' abbreviated 'wffs') 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 elementary letters of the language and is closed under the connectives. In other words, whenever α, $\beta \in L$ then so are $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$; 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 such 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. If you did not cover Chap. 7 before coming to the present one, you are advised to read now its Sect. 7.4 and do the exercises there.

We also agreed to make reading easier by omitting brackets when this can be done without ambiguity, for example by omitting the outermost brackets of a formula, which, in fact, we have just done in the above examples and will continue to do. Finally, we reduce brackets a bit further by using a few standard grouping conventions, notably for negation, for example reading bracket-free $\neg p \wedge q$ as $(\neg p) \wedge q$ rather than as $\neg(p \wedge q)$. You will get used to these conventions just as you did in school for expressions using arithmetic operations.

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: $E \to \{1,0\}$ on the set E of elementary letters of the language into the two-element set $\{1,0\}$. Roughly speaking, an assignment corresponds to left-hand part of a row in a truth-table, the part under the elementary letters. Using the fact that the brackets in compound formulae ensure unique readability, we can apply the general principle of structural recursive definition (Chap. 4, Sect. 4.6.3) to conclude 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 familiar 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 function v^+ is called a *valuation* of formulae, specifically, the valuation *determined* by the assignment v. Assured of this unique determination and 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.

We are interested in a group of five notions that are intimately related to each other: the relations of tautological implication, tautological equivalence and inconsistency and the properties being a tautology and satisfiability

8.3.2 Tautological Implication

We begin with the relation of *tautological implication*, whose converse is known as *tautological consequence*. On its most basic level, it is a relation between individual formulae. Let α, β be formulae. We say that α *tautologically implies* β and write $\alpha \models \beta$ iff there is no valuation v such that $v(\alpha) = 1$ but $v(\beta) = 0$.

It is convenient, from the beginning, to consider this in a more general form as 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 $A \models \beta$ 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 $v(\alpha) = 1$, that is, requiring that $\alpha \models \beta$ in the sense of the first definition.

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.

It is very important to understand that the symbol $\models$ is *not* one of the connectives of propositional logic, as are $\neg$, $\wedge$, $\vee$ (or whichever connectives are chosen). Whereas $p \rightarrow q$ is a formula of the language L of propositional logic (or abbreviates a formula such as $\neg p \vee q$ if $\rightarrow$ is not one of the chosen (or, as one says, 'primitive') connectives of the language, the expression $p \models q$ is not a formula. The sign $\models$, known as a *turnstile* or *gate*, is a symbol that we use when *talking about* formulae of propositional logic. In handy jargon, one says that it belongs to our *metalanguage* rather than to the *object language*. This distinction takes a little getting used to, but it is vital; its neglect can lead to inextricable confusion.

Table 8.5 lists some of the more important tautological implications, between a set of either one or two formulae on the left and a single formula on the right. 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 item under the heading LHS (left hand side) tautologically implies the item under the heading 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 with the curly brackets omitted. Thus, for example, the premise set for modus ponens is the pair $\{\alpha, \alpha \rightarrow \beta\}$ and the conclusion is β.

We can check each of these tautological implications by building a truth-table. For example, for modus tollens the four possible truth-value combinations for α, β are given in the left part Table 8.6. The resulting values of $\alpha \rightarrow \beta$, $\neg\beta$, $\neg\alpha$ are

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, $\rightarrow^-$	$\alpha \rightarrow \beta, \alpha$	β
Modus tollens, MT	$\alpha \rightarrow \beta, \neg\beta,$	$\neg\alpha$
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, right and left explosion	$\alpha \wedge \neg\alpha$	β
	α	$\beta \vee \neg\beta$

Table 8.6 Truth-tabular verification of 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

calculated step by step, from the inside to the outside, in other words, from the leaves of the miniature syntactic decomposition trees for the three formulae to their roots. 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; in brief, $\{\alpha \rightarrow \beta, \neg\beta\} \vDash \neg\alpha$.

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*.

Exercise 8.3.2

(a) Draw a truth-table to check out disjunctive syllogism.
(b) (i) Check the first limiting case of tautological implication, called also right explosion. (ii) Explain in general terms why it holds, even when the formulae α and β share no elementary letters with α or β. (iii) Give a similar general explanation for the other limiting case of 'left explosion'.
(c) Show that the relation of tautological implication between individual formulae is reflexive and transitive, but not symmetric.
(d) Show that the relation of tautological implication between individual formulae is not complete in the sense of Chap. 2, Sect. 2.6.2.

Solution

(a) In Table 8.7 for disjunctive syllogism, there is just one row in which both premises come out true, and that row makes the conclusion true. So, the set of the two premises does tautologically imply the conclusion.
(b) (i) In Table 8.8 for right explosion, there is no row that makes the sole premise true so, *a fortiori*, there is no row that makes all premises (i.e. just that one) true and makes the conclusion false. So, the premise does tautologically imply the conclusion. By the way, when drawing up the truth-table, there is no need to repeat the column for β again on the right, although no damage is done if you do so.
(b) (ii) Instead of talking in terms of rows, we now speak more briefly of valuations of truth-values and use the usual notation for functions. Since there is no valuation v of truth-values with $v(\alpha \wedge \neg\alpha) = 1$, there is no valuation that simultaneously does that and makes $v(\beta) = 0$.

Table 8.7 Truth-tabular verification of disjunctive syllogism

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

Table 8.8 Truth-tabular verification of right explosion

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

(b) (iii) Since there is no valuation v with $v(\beta \vee \neg\beta) = 0$, there is no valuation v with both $v(\beta \vee \neg\beta) = 0$ and $v(\alpha) = 1$. We will have more to say about the explosion principles (and disjunctive syllogism) in Chap. 11.

(c) For reflexivity, let α be any formula. Clearly, if $v(\alpha) = 1$ then $v(\alpha) = 1$. For transitivity, suppose for *reductio* that $\alpha \models \beta$ and $\beta \models \gamma$ but $\alpha \not\models \gamma$. By the last, there is a valuation v with $v(\alpha) = 1$, $v(\gamma) = 0$. Since $v(\alpha) = 1$, we know from $\alpha \models \beta$ that $v(\beta) = 1$, so we have from $\beta \models \gamma$ that $v(\gamma) = 1$, contradicting $v(\gamma) = 0$. For failure of symmetry, let p, q be distinct elementary letters. Then $p \wedge q \models p$ but $p \not\models p \wedge q$. Remarks: Although symmetry fails, the relation $\models$ is not asymmetric, witness $\alpha \models \alpha$. And although $\alpha \models \alpha \wedge \beta$ fails for some formulae, there are choices of α, β for which it holds—for example, let α be any formula and choose β to be α.

(d) Take any two distinct elementary letters p, q; clearly neither $p \models q$ nor $q \models p$.

8.3.3 Tautological Equivalence

Let α, β be formulae. Depending on the choice of these formulae, we may have none, just one, or both of $\alpha \models \beta$, $\beta \models \alpha$ holding (see Exercise 8.3.2 (c) for the failure of symmetry). When $\alpha \models \beta$ and $\beta \models \alpha$ both hold, we say that the two formulae are *tautologically equivalent* and write $\alpha \dashv\vDash \beta$. Equivalently: $\alpha \dashv\vDash \beta$ iff $v(\alpha) = v(\beta)$ for every valuation v.

In terms of truth-tables: α is tautologically equivalent to β iff, when we draw up a truth-table that covers (at least) all the elementary letters that occur in them, the column for α comes out 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.

Table 8.9 Some important tautological equivalences using $\neg$, $\wedge$, $\vee$

Name	LHS	RHS
Double negation	α	$\neg\neg\alpha$
Commutation for $\wedge$	$\alpha \wedge \beta$	$\beta \wedge \alpha$
Association for $\wedge$	$\alpha \wedge (\beta \wedge \gamma)$	$(\alpha \wedge \beta) \wedge \gamma$
Commutation for $\vee$	$\alpha \vee \beta$	$\beta \vee \alpha$
Association for $\vee$	$\alpha \vee (\beta \vee \gamma)$	$(\alpha \vee \beta) \vee \gamma$
Distribution of $\wedge$ over $\vee$	$\alpha \wedge (\beta \vee \gamma)$	$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
Distribution of $\vee$ over $\wedge$	$\alpha \vee (\beta \wedge \gamma)$	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
Absorption	α	$\alpha \wedge (\alpha \vee \beta)$
	α	$\alpha \vee (\alpha \wedge \beta)$
Expansion	α	$(\alpha \wedge \beta) \vee (\alpha \wedge \neg\beta)$
	α	$(\alpha \vee \beta) \wedge (\alpha \vee \neg\beta)$
De Morgan	$\neg(\alpha \wedge \beta)$	$\neg\alpha \vee \neg\beta$
	$\neg(\alpha \vee \beta)$	$\neg\alpha \wedge \neg\beta$
	$\alpha \wedge \beta$	$\neg(\neg\alpha \vee \neg\beta)$
	$\alpha \vee \beta$	$\neg(\neg\alpha \wedge \neg\beta)$
Limiting cases	$\alpha \wedge \neg\alpha$	$\beta \wedge \neg\beta$
	$\alpha \vee \neg\alpha$	$\beta \vee \neg\beta$

We list the most important tautological equivalences that can be expressed in up to three elementary letters: Table 8.9 contains equivalences using at most the connectives $\neg$, $\wedge$, $\vee$, while Table 8.10 considers $\rightarrow$, $\leftrightarrow$, by themselves as well as in interaction with the others. Just as for the tautological implications in Table 8.5, they should be committed to memory after being understood. The ones that students have most difficulty memorizing correctly are distribution and expansion, particularly the latter version of each; for some reason that nobody has explained, humans seem to have more difficulty processing a conjunction of disjunctions than a disjunction of conjunctions.

Exercise 8.3.3 (1)

(a) Verify distribution of $\vee$ over $\wedge$ by reasoning verbally about valuations, rather than by drawing a table.
(b) What syntactic feature is shared by the limiting case equivalences, absorption and expansion, but none of the others?
(c) Show from the definition that tautological equivalence is indeed an equivalence relation.

Table 8.10 Some important tautological equivalences using $\rightarrow$, $\leftrightarrow$

Name	LHS	RHS
Contraposition	$\alpha \rightarrow \beta$	$\neg\beta \rightarrow \neg\alpha$
	$\alpha \rightarrow \neg\beta$	$\beta \rightarrow \neg\alpha$
	$\neg\alpha \rightarrow \beta$	$\neg\beta \rightarrow \alpha$
Packing/unpacking (import/export)	$\alpha \rightarrow (\beta \rightarrow \gamma)$	$(\alpha \wedge \beta) \rightarrow \gamma$
Permutation	$\alpha \rightarrow (\beta \rightarrow \gamma)$	$\beta \rightarrow (\alpha \rightarrow \gamma)$
Consequentia mirabilis (miraculous consequence)	$\alpha \rightarrow \neg\alpha$	$\neg\alpha$
	$\neg\alpha \rightarrow \alpha$	α
Commutation for $\leftrightarrow$	$\alpha \leftrightarrow \beta$	$\beta \leftrightarrow \alpha$
Association for $\leftrightarrow$	$\alpha \leftrightarrow (\beta \leftrightarrow \gamma)$	$(\alpha \leftrightarrow \beta) \leftrightarrow \gamma$
$\neg$ through $\leftrightarrow$	$\neg(\alpha \leftrightarrow \beta)$	$\alpha \leftrightarrow \neg\beta$
	$\neg(\alpha \leftrightarrow \beta)$	$\neg\alpha \leftrightarrow \beta$
Translations between two-place 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 two-place connectives	$\neg(\alpha \rightarrow \beta)$	$\alpha \wedge \neg\beta$
	$\neg(\alpha \wedge \beta)$	$\alpha \rightarrow \neg\beta$
	$\neg(\alpha \leftrightarrow \beta)$	$(\alpha \wedge \neg\beta) \vee (\beta \wedge \neg\alpha)$

Solution

(a) We want to show that $\alpha \vee (\beta \wedge \gamma) \dashv\vDash (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$. For LHS $\vDash$ RHS, let v be a valuation and suppose v(LHS) = 1. Then either $v(\alpha) = 1$ or $v(\beta \wedge \gamma) = 1$. In the former case, $v(\alpha \vee \beta) = 1 = v(\alpha \vee \gamma)$, so v(RHS) = 1. For RHS $\vDash$ LHS, suppose v(LHS) = 0. Then $v(\alpha) = 0$, also $v(\beta \wedge \gamma) = 0$ so that either $v(\beta) = 0$ or $v(\gamma) = 0$. In the former case, $v(\alpha \vee \beta) = 0$ so v(RHS) = 0. In the latter case, $v(\alpha \vee \gamma) = 0$ so similarly v(RHS) = 0 and we are done. Suggestion: Draw up a truth-table (8 rows) for this tautological equivalence and compare the two verifications: which is longer, more easily programmed for computer, more boring?

(b) For all other tautological equivalences in Table 8.9, LHS and RHS contain the same elementary letters, while that is not always the case for the absorption, expansion, limiting case equivalences. In absorption and expansion the RHS may gain letters, in explosion LHS and RHS may have no letters in common.

(c) Reflexivity holds since trivially: if $v(\alpha) = 1$ then $v(\alpha) = 1$. Symmetry is immediate from the definition. Transitivity holds by the same reasoning as given for tautological implication in Exercise 8.3.2 (c) but carried out in both directions. *End of solution*.

Many of the equivalences in Table 8.9 have well-known analogues for sets, familiar from Chap. 1. For example, the first two de Morgan principles for propositions have as counterparts the identities $-(A \cap B) = -A \cup -B$ and $-(A \cup B) = -A \cap -B$ for arbitrary sets A, B, where—is complementation with respect to some local universe. Indeed, there is a *systematic correspondence* between tautological equivalences and Boolean identities between sets and, similarly, one between tautological implications and rules of inclusion between sets. This is not surprising, since intersection, union and complementation were defined in Chap. 1 using 'and', 'or' and 'not' respectively: the behaviour of those logical connectives is reflected in the mirror of the corresponding set operations.

The de Morgan equivalences in Table 8.9 answer a question that we posed, at the end of Sect. 8.2, about connectives needed to express all truth-functions. Every truth-function can be expressed using just the two connectives $\neg$, $\wedge$, since from them we can get $\vee$ 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 $\neg$, $\vee$ suffices to express all possible truth-functions, via the third de Morgan equivalence.

Exercise 8.3.3 (2)

(a) Use a truth-table to verify association for $\leftrightarrow$.
(b) Use information from the list of equivalences to show that the pair $\{\neg, \rightarrow\}$ is enough to express all truth-functions.
(c) (i) Use information from Tables 8.9 and 8.10 to sketch the basic idea for a proof that any formula built using at most the connectives $\neg$, $\leftrightarrow$ is equivalent to one in which all occurrences of $\neg$ act on elementary letters. (ii) How might one go about transforming that idea into a rigorous proof by structural induction, following the principles of Chap. 4?

Solution

(a) See Table 8.11. To facilitate comparison, we have put in bold type the columns for $\alpha \leftrightarrow (\beta \leftrightarrow \gamma)$ and $(\alpha \leftrightarrow \beta) \leftrightarrow \gamma$; to reduce clutter we have not repeated the columns for α and γ alone, as they can be read off from the left side.
(b) The last of the listed translations between two-place connectives tells us how $\vee$ may be expressed in terms of $\neg$,$\rightarrow$, and we already know that the pair $\{\neg,\vee\}$ suffices to express all truth-functions.
(c) (i) Use either of the two '$\neg$ through $\leftrightarrow$' equivalences in Table 8.10 to 'push negations inwards' through material equivalences, using also the equivalence in Table 8.9 to delete any double negations that appear in the process, continuing until all negations have been pushed up against sentence letters.

Table 8.11 Truth-tabular verification of association for ↔

α	β	γ	α	↔	(β ↔ γ)	(α ↔ β)	↔	γ
1	1	1		**1**	1	1	**1**	
1	1	0		**0**	0	1	**0**	
1	0	1		**0**	0	0	**0**	
1	0	0		**1**	1	0	**1**	
0	1	1		**0**	1	0	**0**	
0	1	0		**1**	0	0	**1**	
0	0	1		**1**	0	1	**1**	
0	0	0		**0**	1	1	**0**	

(c) (ii) One way is by defining a suitable function on formulae into the natural numbers to measure the distance of a formula as it is from the shape we would like it to be in, and induce on that. Articulating a function that does the job properly is trickier than might at first appear; here is one. Given a formula α using at most the connectives ¬, ↔, construct its syntactic decomposition tree with economical labels (Chap. 7, Sects. 7.3 and 7.4). For each branch B of the tree, define $d(B)$ to be the number of occurrences of ¬, ↔ in the branch such that ↔ is in the scope of ¬, then put $d(\alpha)$ to be the maximum value of $d(B)$ for branches B in the tree. Carry out a cumulative induction on the structure of formulae, noting in the induction step that $d(\alpha \leftrightarrow \neg\beta)$, say, is less than $d(\neg(\alpha \leftrightarrow \beta))$. *End of solution*.

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 {¬,∧,∨}, {∧,¬}, {¬,∨} and {¬,→} are all functionally complete.

Exercise 8.3.3 (3)

(a) Is there any two-place truth-connective that, taken alone, is functionally complete?

(b) Show that if a truth-functional connective (of any arity) is functionally complete by itself, then it must be both (i) *contrarian* (in the sense that it receives value 0 in the top row of its truth-table) and (ii) *dual-contrarian* (receives 1 in the bottom row).

Solution outline

(a) Go through the sixteen two-place truth-functions in Table 8.4. You will find two that do the job: f_9 (known as *not-both*, or *nand*) and f_{15} (known as *neither-nor*). To check them, first express ¬ (hint: identify elementary letters), and then express either ∧ or ∨, whichever you find easier.

(b) Call the connective * and consider formulae built using it alone. For (i), suppose that * is not contrarian. Let v be the valuation that puts $v(p) = 1$ for all elementary letters p. An easy structural induction shows that $v(\alpha) = 1$ for all formulae. Hence, * cannot express any truth-function that has 0 in the top row of its table; for example, it cannot express negation. For (ii), suppose that * is not dual-contrarian and carry out a dual argument. *End of solution*.

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 $\alpha \wedge \beta \dashv\vDash ' \wedge \beta'$ and $\alpha \vee \beta \dashv\vDash \alpha' \vee \beta'$ and likewise for all other truth-functional connectives since they are definable from these two. Moreover, tautological equivalence 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 (4)
Verify that tautological equivalence is a congruence relation.

Solution
For negation, suppose $\alpha \dashv\vDash \alpha'$ and let v be any valuation. By the supposition $v(\alpha) = v(\alpha')$ so by the truth-table for negation, $v(\neg\alpha) = v(\neg\alpha')$. For conjunction, suppose $\alpha \dashv\vDash \alpha'$ and $\beta \dashv\vDash \beta'$ and let v be any valuation. Then $v(\alpha\wedge\beta) = 1$ iff both $v(\alpha) = 1$ and $v(\beta) = 1$, which by the supposition holds iff both $v(\alpha') = 1$ and $v(\beta') = 1$, thus iff $v(\alpha'\wedge\beta') = 1$. *End of solution*.

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 both $A \vDash \beta$ for all $\beta \in B$ and $B \vDash \alpha$ for all $\alpha \in A$. Equivalently: for every valuation v, $v(\alpha) = 1$ for all $\alpha \in A$ iff $v(\beta) = 1$ for all $\beta \in B$. In a concise notation, we could write $v(A) = 1$ to abbreviate '$v(\alpha) = 1$ for all $\alpha \in A$' and say: $A \dashv\vDash B$ iff for all valuations v, $v(A) = 1$ iff $v(B) = 1$. By suitably editing the argument for Exercise 8.3.3 (1)(c) we can verify that $\dashv\vDash$ is, as its name suggests, a genuine equivalence relation between sets of formulae.

8.3.4 Tautologies, Contradictions, Satisfiability

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, that is, 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 terms, iff $v(\alpha) = 1$ and $u(\alpha) = 0$ for some valuations v and u.

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.

Personally, the author prefers the term 'counter-tautology' to 'contradiction', but has never been able to persuade anyone else to use it, so we follow the standard name.

Exercise 8.3.4 (1)
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). *End of solution*.

If you did Exercise 8.3.4 (1) conscientiously, you will have sensed several general lessons that emerge 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 conditional formula $\alpha \rightarrow \beta$ is a tautology iff $\alpha \models \beta$. Indeed, for all $n \geq 1, \{\alpha_1, \ldots, \alpha_n\} \models \beta$ iff the formula $(\alpha_1 \wedge \ldots \wedge \alpha_n) \rightarrow \beta$ is a tautology. Example: formula (v) of the exercise and disjunctive syllogism in Table 8.4.
- A bi-conditional formula $\alpha \leftrightarrow \beta$ is a tautology iff $\alpha \dashv\models \beta$. Example: formula (vi) of the exercise and one of the de Morgan equivalences in Table 8.10.

Exercise 8.3.4 (2)
Verify the third bullet point in its general form.

Solution
Suppose $\{\alpha_1, \ldots, \alpha_n\} \not\models \beta$. Then there is a valuation v with each $v(\alpha_i) = 1$ and $v(\beta) = 0$. Hence $v(\alpha_1 \wedge \ldots \wedge \alpha_n) = 1$ while $v(\beta) = 0$, so $(\alpha_1 \wedge \ldots \wedge \alpha_n) \rightarrow \beta$ is not a tautology. For the converse, run the same argument backwards. *End of solution*.

Another lesson is suggested by parts (x) and (xi) of Exercise 8.3.4 (1). 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, respectively, a tautology or contradiction.

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.

The notion of a substitution in propositional logic can be given a precise mathematical content. A *substitution* is a function σ: $L \rightarrow L$, where L is the set of all formulae, satisfying the following *homomorphism conditions*:

$$\sigma(\neg\alpha) = \neg\sigma(\alpha)$$

$$\sigma(\alpha \wedge \beta) = \sigma(\alpha) \wedge \sigma(\beta)$$

Here, for simplicity, we are assuming that $\neg$, $\wedge$ are the only two primitive connectives in L. Note that, in this definition, = is not just tautological equivalence; it is full identity between formulae, i.e. the left and right sides stand for the very same formula. Since the bracketing in formulae guarantees unique readability (Chap. 4, Sect. 4.6.3) a substitution function is uniquely determined by its values for elementary letters, in other words, the notion is well-defined by those values. Since $\alpha \vee \beta$, $\alpha \rightarrow \beta$, $\alpha \leftrightarrow \beta$ are understood as abbreviations for $\neg\alpha \wedge \neg\beta$, $\neg(\alpha \neg\beta)$, $(\alpha \wedge \beta) \vee (\neg\alpha \wedge \neg\beta)$ respectively, substitution functions satisfy analogous homomorphism conditions for them too.

Exercise 8.3.4 (3)

(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) Write out the analogous homomorphism conditions for $\rightarrow$, $\leftrightarrow$.

Solution

(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, $\sigma(r \vee (q \rightarrow r)) = (p \rightarrow s) \vee (\neg q \rightarrow (p \rightarrow s))$. By now, you will have seen that you can perform substitutions by using the copy-paste facility on your word processor.

(b) $\sigma(\alpha \rightarrow \beta) = \sigma(\alpha) \rightarrow \sigma(\beta)$, $\sigma(\alpha \leftrightarrow \beta) = \sigma(\alpha) \leftrightarrow \sigma(\beta)$. *End of solution*.

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 able to prove the claim that every substitution instance of a tautology is a tautology or, as we also say in the terminology of Chap. 3, Sect. 3.2.2, 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 letters 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 $\alpha \nvDash \beta$ then $\sigma(\alpha) \nvDash \sigma(\beta)$
- 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.

Solution
For the basis, we need to show that $v'(p) = v(\sigma(p))$; but that is given explicitly by the definition of v. For the induction step, suppose that the property holds for φ, ψ; we need to show that it holds for $\neg\varphi$, $\varphi \wedge \psi$. For negation, we need to show that $v'(\neg\varphi) = v(\sigma(\neg\varphi))$. Now, RHS $= v(\neg\sigma(\varphi)) = 1 - v(\sigma(\varphi)) = 1 - v'(\varphi) =$ LHS, using the induction hypothesis for the penultimate equality. For conjunction, we need to show that $v'(\varphi \wedge \psi) = v(\sigma(\varphi \wedge \psi))$. Now, RHS $= v(\sigma(\varphi) \wedge \sigma(\psi)) = min\{v(\sigma(\varphi)), v(\sigma(\psi))\} = min\{v'(\varphi), v'(\psi)\} =$ LHS, again using the induction hypothesis for the penultimate equality.

The last of the inter-connected notions that we need to define is that of the *satisfiability of a set of formulae*. When A is a set of formulae and v is a valuation, we say that v *satisfies* A iff $v(\alpha) = 1$ for all $\alpha \in A$ (briefly, iff $v(A) = 1$). The set A is *satisfiable* iff there is some valuation v that satisfies it, otherwise it is said to be unsatisfiable.

Exercise 8.3.4 (5)
Verify the following:

(a) (i) A singleton $\{\alpha\}$ is satisfiable iff α is not a contradiction, and (ii) $\{\alpha\}$ is unsatisfiable iff $\neg\alpha$ is a tautology,
(a) (i) A finite set of formulae is satisfiable iff the conjunction of all its elements is not a contradiction. (ii) It is unsatisfiable iff the disjunction of the negations of all its elements is a tautology.

Solution outline
Just apply the definitions and, where needed, the relevant truth-tables.

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. The motivation is the same as in school arithmetic when we express polynomials in a standard form, but the way in which this plays out in logic is rather different. In this section we will look briefly at four kinds of normal form for logic. The first two focus on the positioning of connectives in a formula, showing that they may always be applied in a neat order. The next two concern the interplay of elementary letters; one of them is about letters that can be eliminated 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, it is sometimes also referred to as a *canonical form*. 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 forms are *disjunctive normal form*, abbreviated *dnf* and its dual, *conjunctive normal form*, aka *cnf*.

To explain them, we need the concepts of a literal and basic conjunction. A *literal* is simply an elementary letter or its negation. A *basic conjunction* is any conjunction of $n \geq 1$ literals in which no letter occurs more than once. Thus, we do not allow repetitions of an unnegated letter as in $p \wedge q \wedge p$, nor repetitions of a negated 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 $\pm$ indicates the presence or absence of a negation sign. In the limiting case that $n = 1$, a basic conjunction is of the form $\pm p$, without any conjunction sign.

A formula is said to be in *disjunctive normal form* (*dnf*) iff it is a disjunction of $m \geq 1$ basic conjunctions. Again, in the limiting case that $m = 1$, 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 its disjuncts, i.e., in each of the basic conjunctions.

Exercise 8.4.1 (1)

(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) $(p \wedge q) \vee (\neg p \ \neg q \wedge \neg p)$, (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) through (x) all yes. (xi) No: $p \wedge \neg p$ has two occurrences of p, one negated and the other unnegated.

(b) (ix) is not in full dnf because its two disjuncts do not contain exactly the same letters. On the other hand, the remaining dnf formulae (iii), (vi), (vii), (viii), (x) are full. *End of solution*.

How can we find a disjunctive normal form for an arbitrarily given formula α? There are two basic algorithms. One is semantic, via the truth-table for α. The other is syntactic, via successive transformations of α that are justified by tautological equivalences from Tables 8.9 and 8.10.

The semantic construction is the simpler of the two, although not always the shortest to carry out. 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. 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.
- When there are no such rows, i.e., in the limiting case that α is a contradiction, 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) $\neg(p \rightarrow (q \rightarrow p))$, (iv) $(p \vee q) \wedge (\neg p \vee \neg q) \wedge (\neg p \vee q)$.

Solution outline

We give the final results without the intermediate calculations. (i) $(p \wedge q) \vee (\neg p \; \neg q)$. (ii) $(p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (\neg p \wedge q \; \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge \neg q \wedge \neg r)$. (iii) The formula has no dnf, *a fortiori* no full dnf, since it is a contradiction. (iv) $\neg p \wedge q$. *End of solution.*

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. Translate the connectives $\rightarrow$ and $\leftrightarrow$ (and any others in the formula, such as exclusive disjunction) into $\neg$, $\wedge$, $\vee$ using translation equivalences such as those in Table 8.10.

- 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 literals.
- Clean up by deleting repetitions of a literal with the same sign within each basic conjunction, and by deleting those basic conjunctions that contain a letter in both negated and unnegated form.

It is understood that when carrying out these steps the association and commutation principles for each of $\wedge$ and $\vee$ may be applied whenever convenient.

The algorithm gives us as output a formula in disjunctive normal form, except when α is a contradiction; in that case the output comes out as empty as a result of applications of the 'clean-up' step. However, the dnf obtained in this way is rarely full: there will usually be basic conjunctions with less than the full collection of letters occurring in them.

Exercise 8.4.1 (3)

(a) Use the syntactic algorithm to transform the formula $(p \vee \neg(q \vee r)) \rightarrow r$ into disjunctive normal form.

(b) How might you transform the dnf obtained in (a) into a full one, by further basic equivalences and without resorting to truth-tables?

Solution

(a) $(p \vee \neg(q \vee r)) \rightarrow r \dashv\vDash \neg(p \vee \neg(q \vee r)) \vee r \dashv\vDash (\neg p \wedge \neg\neg(q \vee r)) \vee r \dashv\vDash (\neg p \wedge (q \vee r)) \vee r \dashv\vDash (\neg p \wedge q) \vee (\neg p \wedge r) \vee r$. This exercise is interesting since, while the output is in dnf, there is a way in which it can be further simplified. Clearly, $\neg p \wedge r \vDash r$ so our formula is tautologically equivalent to $(\neg p \wedge q) \vee (\neg p \wedge r)$ obtained by dropping the last disjunct and still in dnf.

(b) Apply repeatedly the expansion principle $\alpha \dashv\models (\alpha \wedge \beta) \vee (\alpha \wedge \neg\beta)$ from Table 8.9, then clean out a repeated disjunct, as follows:

$$
\begin{aligned}
(\neg p \wedge q) \vee r &\dashv\models (\neg p \wedge q \wedge r) \vee (\neg p \wedge q \wedge \neg r) \vee r \\
&\dashv\models (\neg p \wedge q \wedge r) \vee (\neg p \wedge q \wedge \neg r) \vee (p \wedge r) \vee (\neg p \wedge r) \\
&\dashv\models (\neg p \wedge q \wedge r) \vee (\neg p \wedge q \wedge \neg r) \vee (p \wedge q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (\neg p \wedge \neg q \wedge r) \\
&\dashv\models (\neg p \wedge q \wedge r) \vee (\neg p \wedge q \wedge \neg r) \vee (p \wedge q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge \neg q \wedge r).
\end{aligned}
$$

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; it is a *full* conjunctive normal form of α iff every letter of α occurs in each of the basic disjunctions. Cnfs may also be constructed in any of three ways: syntactically, semantically, or piggybacking on dnfs.

Syntactically, we can follow the same algorithm as for dnfs, except that we use the dual distribution rule $\alpha \vee (\beta \wedge \gamma) \dashv\vdash (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$ to move all disjunctions inside conjunctions and, in the final 'cleaning stage', we delete any basic disjunction that contains both a letter and its negation.

Semantically, 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.

At first sight, the semantic construction may seem a little mysterious. Why, when constructing the basic disjunctions, do we take letters that were assigned 0 and negations when the letter was assigned 1? Is there an underlying idea? There is, indeed. Instead of allowing the valuations that make the formula α true, we are disallowing those 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 Why should we bother with cnfs when we already have dnfs? They are much less intuitive!

Hatter Indeed they are, with their conjunctions of disjunctions. But they have been found useful in a discipline known as *logic programming*. Such a programme 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 $n = 0$). They are also known as *Horn clauses*, after the mathematician Alfred Horn who made good use of them in abstract algebra. Any such formula may be expressed as a basic disjunction with just one unnegated letter, namely $\neg p_1 \vee \ldots \neg p_n \vee q (n \geq 0)$. A finite collection of such formulae may be treated as behaving like their conjunction, which evidently is in cnf.

Finally, here is how to construct a cnf for a formula α by piggy-backing on a dnf for its negation $\neg\alpha$. Given α, we construct a dnf of its negation $\neg\alpha$ by whichever method you like (truth-tables or successive transformations). This will be a disjunction $\beta_1 \wedge \ldots \wedge \beta_n$ of basic conjunctions β_i. Negate it, getting $\neg(\beta_1 \vee \ldots \vee \beta_n)$, which is tautologically equivalent to $\neg\neg\alpha$ and thus by double negation equivalent to the original α. Then use de Morgan to push all negations inside, getting $\neg(\beta_1 \wedge \ldots \wedge \beta_n)$. 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 elimination as needed, to express it as a disjunction of literals, and we are done.

Exercise 8.4.2

(a) Take again the formula $(p \vee \neg(q \vee r)) \rightarrow r$, and find a cnf for it by all three methods: (i) successive syntactic transformations, (ii) semantic, (iii) the piggy-backing 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 $\neg p_1 \vee \ldots \neg p_n \vee q \dashv\vDash (p_1 \wedge \ldots \wedge p_n) \rightarrow q$.

Solution

(a) (i) $(p \vee \neg(q \vee r)) \rightarrow r \dashv\vDash \neg(p \vee \neg(q \vee r)) \vee r \dashv\vDash (\neg p \wedge \neg\neg(q \vee r)) \vee r \dashv\vDash (\neg p \wedge (q \vee r)) \vee r \dashv\vDash (\neg p \vee r) \wedge (q \wedge r \vee r) \dashv\vDash (\neg p \vee r) \wedge (q \vee r)$.

(a) (ii) Three of the eight rows make the formula false; applying the cnf algorithm to those rows gives us $(\neg p \vee \neg q \vee r) \wedge (\neg p \vee q \vee r) \wedge (p \vee q \vee r)$. This is not quite the same formula as in (a) (i), but they are easily checked to be tautologically equivalent.

(a) (iii) The same three rows make the negation of the formula true, so the dnf of that negation is $(p \wedge q \wedge \neg r) \vee (p \wedge \neg q \vee r) \vee (\neg p \vee \neg q \vee \neg r)$. Negate

that and apply successive de Morganizations and double negation eliminations to get the same output as in (a) (ii). From this example, you can see that the piggy-backing method is very close to the direct semantic one.

(b) When the formula is tautologically equivalent to a basic conjunction or to a basic disjunction.

(c) There is exactly one way of making the LHS false, namely each $v(p_i) = 1$ and $v(q) = 0$, and that is the same as the unique valuation that makes the RHS false.

8.4.3 Least Letter Set

A set A of propositional formulae may contain redundant elements in the sense that $A \dashv\vDash A\backslash\{\beta\}$ for some $\beta \in A$. For example, the element q is redundant in the set $A = \{p, p \to q, q\}$ since $\{p, p \to q, q\} \dashv\vDash \{p, p \to q\}$. Indeed, $\{p, p \to q\}$ is a minimal subset of A that is tautologically equivalent to A, since none of its three proper subsets have that property. Note, however, that it is not the least such subset, since we also have $\{p, p \to q, q\} \dashv\vDash \{p, q\}$.

But there is also another, *quite different* kind of redundancy, this time of elementary letters occurring within a formula (or within some element of a set of formulae): it can happen that we are able to rewrite the formula (resp. set of formulae) in such a way that the elementary letter in question no longer appears in it (resp. no longer appears in any of its elements). That is the notion that will be studied in the present section.

We begin with a simple example. We have already seen in Table 8.9 that p $(p \vee q) \dashv\vDash p \dashv\vDash p \vee (p \wedge q)$; the letter q is thus eliminable from each of the two outlying formulae. The expansion equivalences $(p \wedge q) \vee (p \wedge \neg q) \dashv\vDash p \dashv\vDash (p \vee q) \wedge (p \vee \neg q)$ in the same table give another example. So does the equivalence $(p \to q) \wedge (\neg p \to q) \dashv\vDash q$, not in the table: the letter p on the LHS does not appear on the RHS.

In general terms, we say that an elementary letter p is *eliminable* from a formula α (resp. from a set A of formulae) if there is some formula α' (resp. some set A' of formulae) with $\alpha \dashv\vDash \alpha'$ (resp. $A \dashv\vDash A'$) such that p does not occur in α' (resp. does not occur in any element of A'). The word 'redundant' is sometimes also used for this concept, but that is not a good idea as it invites confusion with the sense of that term given at the beginning of this section.

Notice that while the letter p is redundant in the set $A = \{p, p \to q, q\}$, it is *not* eliminable from that set: there is no set $A' \dashv\vDash A$ such that p does not occur in any element of A'. Notice also that while the formula $p \to q$ is also redundant in the same A, the question of its eliminability does not even arise, as the latter concept applies only to elementary letters. Conversely, while the letter q is eliminable from the formula $(p \vee q) \wedge (p \vee \neg q)$, it is not a redundant element of the singleton $\{(p \vee q) \wedge (p \vee \neg q)\}$, since it is not an element of that set at all.

In the study of eliminability, it helps streamline our formulations if we expand our language a little to admit a zero-ary connective $\bot$ (called *the falsum*, or *bottom*), 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 $p \vee \neg p$ and will also have eliminable letters, since $p \wedge \neg p \dashv\models \bot$ and $p \vee \neg p \dashv\models \neg\bot$.

It is clear that for every formula α containing elementary letters $p_1, .., p_n$ (n 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 $\alpha \dashv\models \alpha'$ 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 discard 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 seen in Exercise 2.5(d) at the end of Chap. 2, 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, given the availability of the falsum. Let α be any formula and let $\{p_1, \ldots, p_m\}$ be a minimal letter-set for α, so that there is a formula $\alpha' \dashv\models \alpha$ containing only the letters $p_1, \ldots, p_m$. We want to show that $\{p_1, \ldots, p_m\}$ is in fact 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. Let σ be the substitution that puts the falsum $\bot$ for p_i leaving all other letters unchanged. Since $\alpha' \dashv\models \alpha''$, we know from Sect. 8.3.4 that $\sigma(\alpha') \dashv\models \sigma(\alpha'')$. But, since p_i does not occur in α'', we also have $\sigma(\alpha'') = \alpha'' \dashv\models \alpha$, so $\sigma(\alpha') \dashv\models \alpha$. Now, $\sigma(\alpha')$ 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 α has a least letter-set, which we know from Chap. 2 to be unique. Any formula α' equivalent to α that is built from those letters is known as a *least letter-set version* of α. Such formulae α' are not *themselves* unique, but they are all tautologically equivalent and they all have the same letter-set. 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 rely on our experience with truth-functional formulae to inspire guesses, which we then settle by checking.

Exercise 8.4.3

(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".

Solution

(a) (i) $\bot$, (ii) $p \wedge r$, (iii) $\neg\bot$ (which, for obvious reasons, is sometimes written as $\top$). (b) (i) $p \vee r$, (ii) $\neg p$. (c) True: A finite set A of formulae has the same elementary letters as the conjunction $\wedge A$ of all its elements; clearly the two are tautologically equivalent and so whatever is equivalent to one is equivalent to the other.

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 the deployment of the letters as 'modular' or 'compartmentalized' as possible. Its definition makes essential use of the notion of a partition and, before going further, you are advised to review the basic theory of partitions in Chap. 2 (Sect. 2.5.3).

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 eliminable – 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 necessarily 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 E for A.

In general terms, here is the definition. Let A be any set of formulae, with E_A the set of elementary letters occurring in it. If $E_A \neq \varnothing$, a *splitting* of E_A for A is defined to be a partition of E_A such that A is tautologically equivalent to some set A' of formulae with $E_{A'} = E_A$ but with each formula in A' taking all its letters from a single cell of the partition.

Now, as we saw in Exercise 2.5.3 (b) of the chapter on relations, partitions of a given set can be compared according to their fineness: 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: reflexive, transitive, antisymmetric. As we also saw in the end-of-chapter Exercise 2.4 (f), one partition is at least as fine as another iff the equivalence relation corresponding to the former is a sub-relation 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 $A' = \{\neg p \wedge \neg q,\ r \rightarrow s\}$ that is also 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 for any set A of formulae, E_A has a *unique finest splitting* for A. In our running example, it is the three-cell partition $\{\{p\}, \{r,s\}, \{q\}\}$ of $E_A = \{p, q, r, s\}$. The four-cell partition $\{\{p\}, \{r\}, \{s\}, \{q\}\}$ of E_A is finer – but it is not a splitting for 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 while $r \rightarrow s$ is not equivalent to any of $\bot$, $\neg\bot$, r, $\neg r$, s, $\neg s$.

Let A be a set of formulae with letter-set E_A and let $\mathscr{F}$ be the unique finest splitting of E_A for A. By a *most modular version* of A, we mean any set A' of formulae that 'witnesses' this finest splitting; that is, any set A' with $A' \nVdash A$ such that for each $\alpha \in A'$, all the letters of α are taken from a single cell of $\mathscr{F}$. Thus a most modular version of A disentangles, as much as possible, the roles that are played by the different elementary letters in elements of A. It makes the presentation of the set as compartmentalized as possible: we have reached the finest way of partitioning the letters such that no formula in our presentation A' of A contains letters from two distinct cells

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 we can, if we wish, take that as the most modular version of A.

Whereas the idea of the least letter-set of a set of formulae is quite old, perhaps dating back into the nineteenth century, that of the finest splitting is surprisingly new. It was first formulated and verified for the finite case by Rohit Parikh only in 1999; a proof of uniqueness for the infinite case was given by the author in 2007.

Just as for least letter-set versions, any general algorithm for finding most modular versions is highly exponential, and very laborious to execute by hand. But in very simple examples, we can use our experience with truth-functional formulae to inspire a guess and then check it out.

Exercise 8.4.4 (1)

Find the finest splitting and a most modular version for each of the following sets of formulae: $A = \{p \wedge q,\ r\}$, $B = \{p \rightarrow q,\ q \rightarrow r,\ r \rightarrow \neg p\}$, $C = \{(\neg p \wedge q \wedge r) \vee (q \neg s \wedge \neg p)\}$, $D = \{p \vee q,\ q \vee r,\ r \vee \neg p,\ \neg q \vee r\}$.

Solution

Remember, we are not eliminating letters—in fact, in these examples no letters are eliminable. We are splitting the set of letters that occur in formulae of the set.

The finest splitting of the letter-set $E_A = \{p, q, r\}$ for A partitions E_A into three singleton cells $\{p\}$, $\{q\}$, $\{r\}$, with $A' = \{p, q, r\}$ a most modular version of A.

The finest splitting of the letter-set $E_B = \{p, q, r\}$ for B partitions E_B into two cells $\{p\}$, $\{q,r\}$, with $B' = \{\neg p, q \rightarrow r\}$ a most modular version of B.

The finest splitting of the letter-set $E_C = \{p, q, r, s\}$ for C partitions E_C into two cells $\{p,q\}$, $\{r,s\}$, with $C' = \{\neg p \wedge q, r \vee \neg s\}$ a most modular version of C.

The finest splitting of the letter-set $E_D = \{p, q, r\}$ for D partitions E_D into two cells $\{p,q\}$, $\{r\}$, with a most modular version $D' = \{p \vee q, r\}$. *End of solution.*

We have described three kinds of letter management: constructing a disjunctive or conjunctive normal form (full or otherwise), finding the least letter-set, getting a most modular representation. We have given algorithms for dnfs and cnfs; although algorithms for finding the least letter-set and a most modular version do exist, we have not attempted to articulate them, merely working out some simple examples by guess-and-check. There is nothing to stop us from combining the three kinds of manipulation with each other. In particular, we can often get a better understanding of the logical import of a set A of formulae by finding a most modular version A_2 of a least letter-set version A_1 of A.

Exercise 8.4.3 (2)
Consider the set $A = \{p \rightarrow q, \neg q \wedge (p \vee u), (u \rightarrow s) \wedge (r \rightarrow u)\}$ of formulae. Find a most modular version A_2 of a least letter-set version A_1 of A.

Solution
The first two elements of A tautologically imply u, which tautologically implies $r \rightarrow u$, so $A \nexists\models A_1 = \{p \rightarrow q, \neg q \wedge (p \vee u), u \rightarrow s\}$ where r is eliminated. The letter-set is $\{p, q, u, s\}$ of A_1 is indeed a least letter-set for A, as one can quickly convince oneself by unsuccessfully trying to eliminate more letters, although it would be quite tedious to verify it rigorously. A most modular presentation of A_1 is $A_2 = \{\neg p, \neg q, u, s\}$, which splits $\{p, q, u, s\}$ into singleton subsets $\{p\},\{q\},\{u\},\{s\}$. The elements of A_2 are literals, which are already in dnf, so no further transformation is useful.

8.5 Truth-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 implication and equivalence, tautologies and contradictions). In this section, we describe a another procedure, called the method of *truth-trees*, also known as *semantic decomposition trees* or *semantic tableaux*. It is usually rather faster, and certainly less boring, than the method of truth-tables. Moreover, it provides a useful support for some theoretical investigations; when we consider the issue of relevance in Chap. 11, it will be exploited intensively.

It is *semantic*, in the sense that it is formulated in terms of truth-values; *algorithmic*, in the sense that it its steps can be carried out by a computer; and *two-sided*, in that it gives a way of determining, in a finite time, whether or not a formula is a

tautology, thus supplying us with a *decision procedure*. In these respects, it is just as powerful as the truth-table method.

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 verify it by a *reductio ad absurdum* argument without constructing a truth-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 material implication 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 order.

To see what is going on, notice the following features of the way in which this tree is built.

- 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, i.e. whether it 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 sub-formulae*. Never in the opposite direction. The information is supplied by the truth-table for whatever connective that is being decomposed at that point.

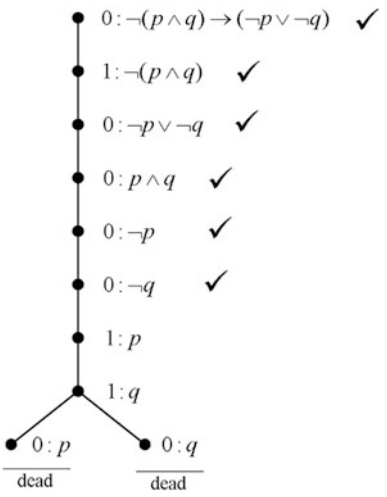

Fig. 8.1 Truth-tree for $\neg(p \wedge q) \rightarrow (\neg p \vee \neg q)$

- When we get *definite* information about the immediate sub-formulae 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.
- When we get *disjunctive* information about the immediate sub-formulae of φ, we *divide* our branch into two sub-branches, with one of the alternatives on one branch and the other alternative on the other.

Having constructed the decomposition tree, we need to be able to read off our answer from it. We make sure that we have decomposed each node in the tree whenever 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. We locate the *crash-pairs* in the tree (also commonly referred to as *explicit contradictions*), that is, pairs (1: φ and 0: φ) of nodes on a common branch, labelled by the same formula but with opposite signs. Then, we read the completed tree as follows:

- If *every* branch contains a crash-pair, then the label of the root is impossible. This is what happens in our example: there are two branches, one containing both 1: p and 0: p, the other with both 1: q and 0: q. We label these branches *dead* and conclude that 0: α is impossible, i.e. that α is a tautology.
- On the other hand, if *at least one* branch is without any crash-pair, then the label of the root is possible, provided we have really decomposed all decomposable formulae in that branch. We label these branches *alive*; we can read off any one of them a valuation that gives the root formula the value indicated in the label. In most texts, the rather bland terms *closed* and *open* are used in place of *dead* and *alive*.

For a second example, consider the formula $\alpha = ((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow r)$, and test to determine whether it is a tautology. We get the labelled tree of Fig. 8.2. It contains three branches. The leftmost and rightmost ones each contain a crash-pair (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

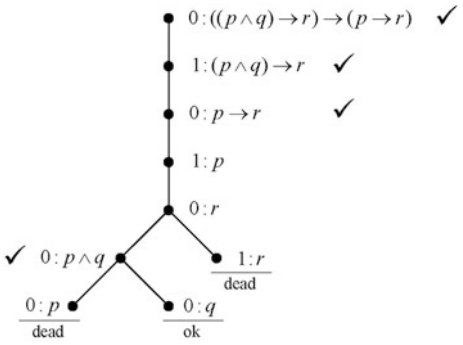

Fig. 8.2 Truth-tree for $((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow r)$

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 valuation $v(p) = 1$, $v(r) = v(q) = 0$. The valuation generated by this assignment agrees with the label for every formula on the branch; in particular, the root formula α gets the value 0 and so is not a tautology.

The construction of such trees always terminates: as we go down branches we work with shorter and shorter formulae until we reach elementary letters and can decompose no further. A rigorous proof of this would be by structural induction on formulae. Evidently, the method is algorithmic and can be programmed. It is certainly less boring than a full truth-table and can sometimes be quite interesting. At the same time, it is usually quicker to calculate—although it must be admitted that in worst-case examples, it can turn out to be just as exponential.

There are 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 Fig. 8.2, 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. The 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 reduces repetition of a given labelled node on different branches. If we were to decompose the second node of the tree in Fig. 8.2 before decomposing the third one, we would repeat 1: p, 0: r on at least two branches. But whichever order of decomposition is followed, the final verdict is the same.

Other tricks for avoiding unnecessary labour stem from the fact that we can sometimes stop constructing a tree before it is finished. This can happen in two distinct ways.

- If a branch contains a crash-pair before having been fully decomposed, we can already declare that branch dead without completing its decomposition, and pass on to other branches. This occasionally saves some work.
- If we have a branch all whose nodes have been decomposed, free of crash-pairs, then we can declare that this branch is *live*, so that the label of the root (and of everything on the branch) is possible, without bothering to complete any other branches. This often saves a lot of work. For an example, see end-of chapter Exercise 8.4.

To take full advantage of the latter kind of economy, one can develop *heuristics* to guide the choice, when one forks, which side of the fork to construct first and how far down to go before looking at the other side, in effect balancing *depth-first* and *breadth-first* strategies of construction. Such heuristics can make a considerable difference to efficiency, but their study takes us beyond the limits of this introduction.

Tables 8.12 and 8.13 recapitulate the definite and indefinite (resp. non-forking and forking) decompositions that one may carry out. Their justification is immediate from the truth-tables for the connectives concerned.

Table 8.12 Definite decompositions

1: $\neg\alpha$	0: $\neg\alpha$	1: $\alpha \wedge \beta$	0: $\alpha \vee \beta$	0: $\alpha \rightarrow \beta$
0: α	1: α	1: α 1: β	0: α 0: β	1: α 0: β

Table 8.13 Forking decompositions

1: $\alpha \vee \beta$		0: $\alpha \wedge \beta$		1: $\alpha \rightarrow \beta$		1: $\alpha \leftrightarrow \beta$		0: $\alpha \leftrightarrow \beta$	
1: α	1: β	0: α	0: β	0: α	1: β	1: α 1: β	0: α 0: β	1: α 0: β	0: α 1: β

Inspecting the tables, one sees that we have two rules for decomposing each connective, one for sign 1 and the other for sign 0. Note that the rules for $\neg$ are definite no matter what the sign (1 or 0); those for $\wedge$, $\vee$, $\rightarrow$ are definite or fork depending on the sign, and both of the rules for $\leftrightarrow$ fork irrespective of sign. Most common student errors: (1) misremembering the rules for $\rightarrow$, usually from not having remembered its truth-table well enough, (2) forgetting that for $\leftrightarrow$ we always need to make two entries on each side of the fork. The rule for decomposing 0: $\alpha \rightarrow \beta$ will play a very important role in Chap. 11 when we consider so-called 'relevance logics' and we already give it a special name, *counter-case*.

Exercise 8.5 (1)

(a) What would the decomposition rules for exclusive disjunction look like? Would they be definite, or fork? How many outputs? Any connection with the rules for other connectives?

(b) Use the method of truth-trees to test whether the following formulae are tautologies: (i) $(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$, (ii) $(p \rightarrow q) \vee (q \rightarrow p)$, (iii) $(p \rightarrow q) \rightarrow \neg(p \rightarrow \neg q)$, (iv) $(p \leftrightarrow q) \vee (p \leftrightarrow r) \vee (q \leftrightarrow r)$.

Solution

(a) Recall the truth-table for exclusive disjunction given in Exercise 1.4.2 (3) of Chap. 1. We write it as +, using the same sign as for the corresponding operation of symmetric difference of sets (see end-of-chapter Exercise 1(1)). The decomposition rules are as follows.

1: $\alpha + \beta$		0: $\alpha + \beta$	
1: α 0: β	0: α 1: β	1: α 1: β	0: α 0: β

Both rules fork, with two outputs on each branch. They are like those for $\leftrightarrow$ but inverted: the output for 1: $\alpha + \beta$ is the same as for 0: $\alpha \leftrightarrow \beta$ and dually. This is because $\alpha + \beta \dashv\models \neg(\alpha \leftrightarrow \beta)$.

(b) We give the verdicts only: (i), (ii), (iv) are tautologies but (iii) is not. *End of solution*.

Clearly, truth-trees may also be used to determine whether a formula is a contradiction, whether two formulae are tautologically equivalent, one formula tautologically implies another or, more generally, whether a finite set of formulae tautologically implies a formula.

- To test whether α is a contradiction, check whether its negation $\neg\alpha$ is a tautology.
- To test whether $\alpha \dashv\models \beta$, check whether the formula $\alpha \leftrightarrow \beta$ is a tautology.
- To test whether $\alpha \models \beta$, check whether the formula $\alpha \rightarrow \beta$ is a tautology.
- To test whether $\{\alpha_1, \ldots, \alpha_n\} \models \beta$, whether $(\alpha_1 \wedge \ldots \wedge \alpha_n) \rightarrow \beta$ is a tautology.

Exercise 8.5 (2)

(a) Use truth-trees to determine whether (i) $(p \vee q) \rightarrow r \dashv\models (p \rightarrow r) \wedge (q \rightarrow r)$, (ii) $(p \wedge q) \rightarrow r \dashv\models (p \rightarrow r) \vee (q \rightarrow r)$, (iii) $p \wedge \neg p \dashv\models q \leftrightarrow \neg q$.
(b) Use truth-trees to determine whether $\{\neg q \vee p, \neg r \rightarrow \neg p, s, s \rightarrow \neg r, t \rightarrow p\} \models \neg t \wedge \neg q$.

Solution outline

(a) All three tautological equivalences hold; in Fig. 8.3 we give the tree for (ii). To reduce clutter, we omit the vertical links in the definite (non-forking) decomposition steps.
(b) The tautological implication holds. The tree has seven branches, arising from forking nodes labelled 1: $\varphi \rightarrow \psi$, 1: $\varphi \vee \psi$ and 0: $\varphi \wedge \psi$, so manual presentation needs a full sheet of paper and small handwriting.

A word on presentation. The truth-trees in this chapter are said to be *signed*, since each node is labelled by a formula accompanied by the sign 1 or 0 for truth or falsehood. It is possible to do away with the signs, by making the formula φ do the work of 1: φ and its negation $\neg\varphi$ carry out the job of 0: φ.

To illustrate, in such a presentation of the tree in Fig. 8.2, the root node would be labelled by the formula $\neg[((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow r)]$; the three leaves, from left to right, would be labelled $\neg p$, $\neg q$, r, and so on for the intermediate nodes. The decomposition rules themselves are rewritten accordingly; for example, the rule allowing passage from 0: $\alpha \rightarrow \beta$ to 1: α and 0: β becomes one to pass from $\neg$

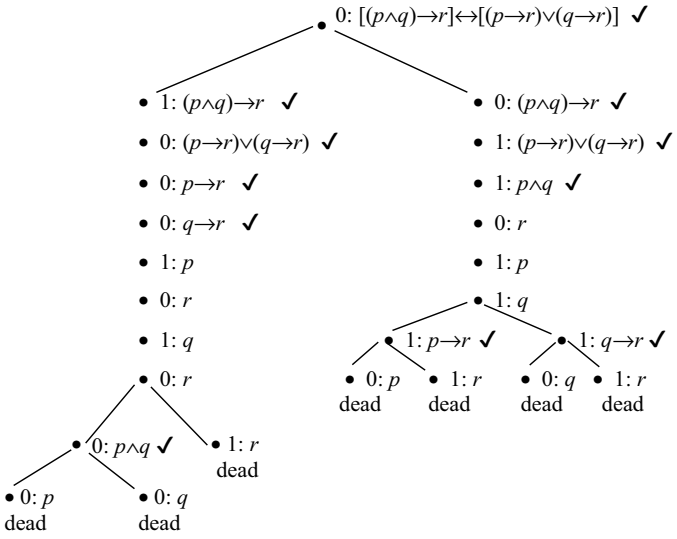

Fig. 8.3 Truth-tree for Exercise 8.5 (2)(a) (ii)

($\alpha \rightarrow \beta$) to α and $\neg\beta$. A crash-pair becomes a pair of nodes, on the same branch, one of which is labelled by a formula φ and the other labelled by $\neg\varphi$.

The difference between the two displays is essentially notational: in the unsigned format the object-language connective of negation does the work that the met-alinguistic attribution of falsehood does in the signed one. Textbook expositions seem to show a slight preference for signed trees.

8.6 End-of-Chapter Exercises

Exercise 8.6 (1) Truth-functional connectives

(a) Construct, from the appropriate truth-table, 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 by a less-than-full dnf?

(b) Show that the connective-set $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$ is not functionally complete. *Hint*: Think about the top row of the truth-table, in order to show that negation is not expressible.

(c) Show that the pair $\{\neg, \leftrightarrow\}$ is not functionally complete. This exercise is quite challenging; do not be disappointed if you get stuck. *Hint*: It will suffice to show that conjunction is not expressible. To do that, show that every formula φ built from at most two letters p, q using at most the connectives $\neg$, $\leftrightarrow$ is 'even', in the sense that $v(\varphi) = 1$ for an even number of the four assignments v: $\{p,q\} \rightarrow \{1,0\}$.

(d) If, when reading Chap. 4, you were not in a position to do Exercise 4.6.2 illustrating definitions and proofs by structural recursion/induction in propositional logic, you should now be in aposition to do so and check with the solution provided there.

Solution

(a) For f_2: $(\alpha \wedge \beta) \vee (\alpha \wedge \neg\beta)$; for f_{11}: $(\alpha \wedge \neg\beta) \vee (\neg\alpha \wedge \neg\beta)$; for f_{15}: $\neg\alpha \wedge \neg\beta$. Yes, f_2 and f_{11} respectively have α and $\neg\beta$ as less-than-full dnfs, while there is no way to simplify the above dnf for f_{15}. By the way, the notion of simplicity is, itself, far from simple; comparison can depend on the tasks that we wish to perform with the items in question, giving rise to non-equivalent definitions. In this example, however, there can hardly be any intuitive disagreement.

(b) Note that when $v(\alpha) = 1 = v(\beta)$ then $v(\alpha \wedge \beta) = 1 = v(\alpha \vee \beta) = v(\alpha \rightarrow \beta) = v(\alpha \leftrightarrow \beta)$. Using that fact, an easy structural induction shows that when $v(p) = 1$ for every elementary letter p (corresponding to the top row of a truth-table) then $v(\varphi) = 1$ for every formula φ that is constructed using at most those connectives. But negation and, indeed, any truth-functional connective that is contrarian in the sense defined in Exercise 8.3.3 (3), gets the value false in the top row. That is, $v(\neg p) = 0$ and, more generally $v(\varphi) = 0$, for any contrarian formula φ when v puts $v(p) = 1$ for every elementary letter p. So, neither negation nor any other contrarian connective can be expressed using only the connectives $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$.

(c) It suffices to show that there is no formula φ using at most the connectives $\neg$, $\leftrightarrow$ such that $p \wedge q \nvDash \varphi$. Now, we may assume *wlog* (see the logic box in Sect. 7.6.2 of Chap. 7) that any such φ contains at most the letters p, q since we can substitute p, say, for all the letters other than those two and we still have $p \wedge q = \sigma(p \wedge q) \nvDash \sigma(\varphi)$. Call a formula φ built from the letters p, q *even* iff $v(\varphi) = 1$ for an even number of the four assignments v: $\{p, q\} \rightarrow \{1,0\}$. Clearly, $p \wedge q$ is odd, so, it will suffice to show that every formula φ built from the letters p, q using at most $\neg$, $\leftrightarrow$ is even. This we do by induction. The basis, where φ is an elementary letter, is trivial. For the induction step, we need to consider each of $\neg$, $\leftrightarrow$. The case for $\neg$ is also trivial: if φ is even then $\neg\varphi$ is even since four minus an even number is even. The case for $\leftrightarrow$ is the tricky part of the proof.

(d) Suppose that α, β are both even; we need to show that $\alpha \leftrightarrow \beta$ is also even. Now $\alpha \leftrightarrow \beta \nvDash (\alpha \wedge \beta) \vee (\neg\alpha \wedge \neg\beta)$, so it will suffice to show that formula to be even. Suppose for *reductio* that it is odd. Since its disjuncts are mutually inconsistent, exactly one of $\alpha \wedge \beta$, $\neg\alpha \wedge \neg\beta$ is odd. Consider the case that $\alpha \wedge \beta$ is even while $\neg\alpha \wedge \neg\beta$ is odd; the other case is similar. Since $\alpha \nvDash (\alpha \wedge \beta) \vee (\alpha \wedge \neg\beta)$ and both α, $\alpha \wedge \beta$ are even, we know that $\alpha \wedge \neg\beta$ is even. But since $\neg\beta \nvDash (\alpha \wedge \neg\beta) \vee (\neg\alpha \wedge \neg\beta)$ and we know that $\alpha \wedge \neg\beta$ is even while $\neg\alpha \wedge \neg\beta$ is odd, we may conclude that $\neg\beta$ is odd, so β is also odd, contradicting the supposition that it is even.

Exercise 8.6 (2) Tautologies etc.

(a) In informal mathematics we often say briefly, when asserting a chain of equivalences, 'α iff β iff γ', just as in arithmetic we say $a = b = c$. What does such an informal statement mean: $\alpha \leftrightarrow (\beta \leftrightarrow \gamma)$, $(\alpha \leftrightarrow \beta) \leftrightarrow \gamma$, or $(\alpha \leftrightarrow \beta)$ $(\beta \leftrightarrow \gamma)$? 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) $\varnothing \vDash \alpha$, (iii) $\beta \vDash \alpha$ for every formula β.

(d) Show by structural induction that every formula constructed using the zero-ary connective $\bot$ and the usual $\neg$, $\wedge$, $\vee$ but without any elementary letters, is either a tautology or a contradiction.

(e) In Exercise 1.4.3 (1) (h) of Chap. 1, we saw that $A\backslash(B\backslash C) \subseteq (A\backslash B) \cup C$ for arbitrary sets A, B, C. What would the counterpart of this be in propositional logic, expressed using the usual classical connectives?

Solution

(a) The phrase 'α iff β iff γ' in informal mathematics is used to mean $(\alpha \leftrightarrow \beta)$ $(\beta \leftrightarrow \gamma)$. This is not tautologically equivalent to either of the other two, which are, however, tautologically equivalent to each other (associativity for $\leftrightarrow$). For this reason, a little comma after the β would add clarity to the English phrase, just as commas would clarify 'eats shoots and leaves' when referring to a cowboy in a western saloon rather than a panda in the forest.

(b) If α is a contradiction, then there is no valuation v with $v(\alpha) = 1$ so, *a fortiori* no matter what β is chosen, there is no valuation v with both $v(\alpha) = 1$ and $v(\beta) = 0$ so $\alpha \vDash \beta$. Conversely, if α is not a contradiction, then there is some valuation v with $v(\alpha) = 1$ so, choosing β as an elementary letter p not occurring in α and putting $v(p) = 1$, we have both $v(\alpha) = 1$ and $v(\beta) = 0$, so $\alpha \nvDash \beta$. Alternatively, choose β to be any contradiction.

(c) Outline: reason dually to the solution to (b).

(d) Basis: $\bot$ is a contradiction, by its truth-conditions (Sect. 8.4.3). Induction step: If α is a tautology or a contradiction then $\neg\alpha$ is a contradiction or a tautology. If each of α, β is a tautology or a contradiction then $\alpha \wedge \beta$ is a contradiction unless both components are tautologies, in which case $\alpha \wedge \beta$ is a tautology; dually for $\vee$.

(e) $\alpha \wedge \neg(\beta \wedge \neg\gamma) \vDash (\alpha \wedge \neg\beta) \vee \gamma$.

Exercise 8.6 (3) Normal forms

(a) Find a dnf for the formula $r \rightarrow \neg(q \vee p)$ using (i) the semantic method and (ii) the method of successive syntactic transformations.

(b) Use your work in (a) to find a cnf for the same formula by the same two methods.

(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) Find most modular versions of your answers to (c).
(e) True or false? For each, give a proof or counter-example. (i) The least letter-set of a formula is empty iff the formula is either a tautology or a contradiction. (ii) If the same letter is eliminable from each of α, β then it is eliminable from $\alpha \wedge \beta$. (iii) The least letter-set of a disjunction is the union of the least letter-sets of its disjuncts.

Solution

(a) (i) Make an eight-row truth-table for $r \rightarrow \neg(q \vee p)$. It comes out true in five rows; write down the disjunction of the five basic conjunctions that correspond to those rows, giving $(p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge \neg q \wedge \neg r)$. Remark: The more rows (i.e. valuations on the letters) make the formula true, the more basic conjunctions there are to disjoin, so the more tedious this method becomes when done manually. (ii) $r \rightarrow \neg(q \vee p) \dashv\vDash \neg r \vee \neg(q \vee p) \dashv\vDash \neg r \vee (\neg q \wedge \neg p)$, which is in (non-full) dnf.
(b) (i) There are just three rows that make $r \rightarrow \neg(q \vee p)$ false, so we form the corresponding three basic disjunctions and conjoin them, giving $(\neg p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee \neg r) \wedge (p \vee \neg q \vee \neg r)$ in full cnf. (ii) Distribute the last formula in the chain of equivalences of (a) (ii) to get $\neg r \vee (\neg q \wedge \neg p) \dashv\vDash (\neg r \vee \neg q) \wedge (\neg r \vee \neg p)$ whose RHS is in cnf.
(c) $A \dashv\vDash \{\bot\}$ with least letter-set $\varnothing$ and all three letters eliminable; B is already in least letter-set form since none of its letters are eliminable; $C \dashv\vDash \{s \wedge q, \neg p\}$ with least letter-set $\{p, q, s\}$ and r eliminable.
(d) Put $A' = \{\bot\}$ with splitting $\{\varnothing\}$ of the empty letter-set; $B' = \{r, (p \wedge q) \vee s\}$ with partition $\{\{r\}, \{p, q, s\}\}$; $C' = \{\neg p, q, s\}$ with partition $\{\{p\}, \{q\}, \{s\}\}$.
(e) (i) Yes; immediate from the definitions. (ii) Yes; if $\alpha \dashv\vDash \alpha'$ and $\beta \dashv\vDash \beta'$ where p occurs in neither of α', β', then $\alpha \wedge \beta \dashv\vDash \alpha' \wedge \beta'$ and p does not occur in the RHS. (iii) No; taking an elementary letter p, the least letter-set of each of p, $\neg p$ is $\{p\}$ but the least letter-set of $p \vee \neg p$ is $\varnothing$.

Exercise 8.6 (4) Truth-trees

Use truth-trees to determine whether each the following hold: (i) $p \rightarrow (q \wedge r) \dashv\vDash (p \rightarrow q) \wedge (p \rightarrow r)$, (ii) $p \rightarrow (q \vee r) \dashv\vDash (p \rightarrow q) \vee (p \rightarrow r)$, (iii) $p \rightarrow (q \rightarrow r) \vDash (p \rightarrow q) \rightarrow r$.

Solution outline

Implications (i), (ii) both hold, but (iii) does not. Figure 8.4 gives a truth-tree for (v) with three live branches on the left. If one has the flair or good luck to construct, say, the leftmost branch first, one can save work by omitting all nodes to the right below the one labelled 1: $q \rightarrow r$.

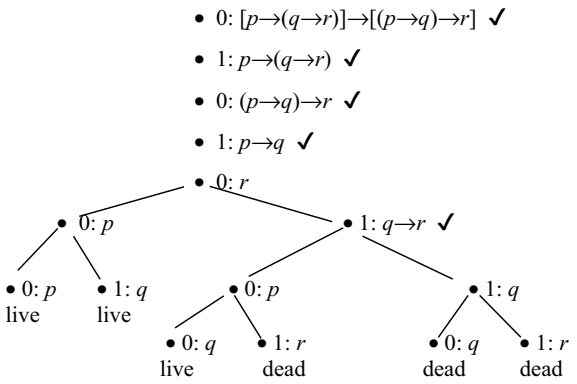

Fig. 8.4 Truth-tree for end-of-chapter Exercise 8.4 (iii)

8.7 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, as explained in the preface, it has some serious disadvantages so we have reversed the order. One of the few texts written for computer science students that also does that is James Hein *Discrete Structures, Logic and Computability*. Jones and Bartlett 2002 (second edition), chapter 6.

Computer science students may like to look at chapter 1 of Michael Huth & Mark Ryan *Logic in Computer Science*, Cambridge University Press 2000, and the rather more mathematical chapters 1–4 of Mordechai Ben-Ari *Mathematical Logic for Computer Science*, Springer 2012 (third edition). Philosophy students may prefer L.T.F. Gamut *Logic, Language, and Meaning. Volume I: Introduction to Logic*. University of Chicago Press 1991, chapters 1–2, while general readers can derive profit from Wilfrid Hodges *Logic*, Penguin 1977, sections 1–25.

For semantic decomposition trees, there is a gentle introduction in Colin Howson *Logic with Trees*, Routledge 1997, chapters 1–4 as well as a solid on-line resource written by Peter Smith at https://www.logicmatters.net/wp-content/uploads/2019/10/TruthTreesPL1.pdf.

Something About Everything: Quantificational Logic

9

Chapter Outline

Although fundamental to logic, truth-functional connectives have very limited expressive power. In this chapter we go further, explaining the basic ideas of *quantificational* logic (also known as *first-order* or *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*, showing how they can be used to express complex relationships. With no more than an intuitive understanding of them, some of the basic *logical equivalences* involving quantifiers can already be appreciated. For a deeper understanding, we then present the *semantics* of the language, which is still bivalent, yet goes beyond truth-tables. This semantics may be given in two versions, *substitutional* and *x-variant*, which however are equivalent under a suitable condition. 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* turning on the quantifiers, some with the identity relation.

9.1 The Language of Quantifiers

We have been using quantifiers informally throughout the book, and made a few remarks on them in a logic box of Chap. 2. Recall that there are two quantifiers $\forall$ and $\exists$, 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,
https://doi.org/10.1007/978-3-030-42218-9_9

9.1.1 Some Examples

Before getting systematic, we give some examples of statements in ordinary English and their representation using quantifiers, commenting on their salient features.

Each of these examples raises a host of questions, whose answers will emerge as we continue through the chapter. The numbers below correspond to the rows in the Table 9.1.

1. This symbolization uses the universal quantifier $\forall$, the variable x, two predicate letters, the truth-functional connective $\rightarrow$. Could we use a different variable, say y? Why are we using $\rightarrow$ here instead of $\wedge$? Can we express this using $\exists$ instead of $\forall$? What is its relation to $\forall x(Px \rightarrow Cx)$?
2. Why are we using $\wedge$ here instead of $\rightarrow$ as we did in the first statement? Can we express it with $\forall$ instead of $\exists$? Is it logically implied by the first statement? What is its relation to $\exists x(Cx \wedge \neg Px)$?
3. Can we express this using $\exists$, $\wedge$, $\neg$? What is its relation to $\forall x(Cx \rightarrow \neg Px)$?
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 logically 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 getting rid of some negations? Does the meaning change if we reverse the initial quantifiers?
10. Shouldn't the second universal quantifier take a different variable? Can we move the second quantifier 'out to the front'? What about the first quantifier?

Table 9.1 Examples of quantified statements

	English	Symbols
1	All composer are poets	$\forall x(Cx \rightarrow Px)$
2	Some composers are poets	$\exists x(Cx \wedge Px)$
3	No poets are composers	$\forall x(Px \rightarrow \neg Cx)$
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	$\forall x[(Mx \wedge Sx) \rightarrow \exists y(Wy \wedge Ay \wedge Byx)]$
8	No man is older than his father	$\neg \exists x(Oxf(x))$
9	The successors of distinct integers are distinct	$\forall x \forall y[\neg(x = y) \rightarrow \neg(s(x) = s(y)]$
10	The English for this should be familiar from Chap. 4!	$\{P0 \wedge \forall x(Px \rightarrow Ps(x))\} \rightarrow \forall x(Px)$

Exercise 9.1.1
On the basis of your informal experience with quantifiers, have a go at answering the bulleted questions.

Solution promise
If you feel confident about most of your answers, congratulations—you seem to have a head start! But keep a record of them to check later! If some of questions leave you puzzled or uncertain, don't worry; answers will become apparent as we go through the chapter and we will state them explicitly at the end.

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 there, along with the quantifiers. But to allow the latter to do their work, we also need some further ingredients: individual constants, variables, predicate letters, the identity relation sign and (optionally) function letters.

In each of the categories of basic terms, function letters and predicates, we assume that we have an infinite supply of signs. These can be referred to by using numerical subscripts e.g. $f_1, f_2, \ldots$ 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 $R^2_1, R^2_2, \ldots$. 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 set of (*well-formed*) *formulae*. 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)
Which of the following are terms, where f is 2-place and g is 1-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), (iv) Yes. (v) No: g is 1-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 sometimes do the same in what follows. *End of solution*.

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 $(t_1 = t_2)$ where = 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 use different styles of bracketing, e.g. square and curly. The special relation sign = for identity is customarily infixed, whereas all other predicates and function symbols are prefixed. For this reason, the parentheses and commas in terms and atomic formulae without the identity sign can in principle be omitted without ambiguity, which we will sometimes do to improve 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 $\forall$ is the universal quantifier', rather than 'The symbol '$\forall$' 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'?

Hatter Yes and no. Strictly speaking one should distinguish between use and mention, in this context between the use of a symbol like the quantifier in the object language and its mention when talking about it in our metalanguage. That can be done by various typographical devices such as quotation marks, italics, underlining or font change. 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; in most cases it is perfectly clear what is meant.

Alice What about the identity sign? Is that part of the object language or the metalanguage?

Hatter Well, both: the author is abusing notation by allowing the same sign, namely = , to play both roles. It serves as a component of formulae in the object language and as a sign in our metalanguage to represent the identity of two items of the object language. Purists may not be happy about this and, indeed, in earlier editions of this book the author took pains to distinguish the two signs. But classroom experience has convinced him that this was quite pedantic; students worked without confusion with the standard sign for identity in both contexts.

Another example of deliberate abuse of notation can be seen in our definition of the set of formulae. In the recursion step we used one particular variable, namely x. But what we really meant was that we can use any of the variables $x, y, z, \ldots$ in the way described, so that $\forall y(\alpha), \forall z(\alpha), \ldots$ are formulae whenever α is a formula. The reason for articulating with a sample variable is that it keeps the notation down. In our metalanguage, we already have *meta-variables* $\alpha, \beta, \ldots$ ranging over formulae; we don't want to add special signs (say Gothic letters) to serve as meta-variables ranging over the set of variables of the object language, and yet others to range over the set of constants etc., unless we really have to.

Exercise 9.1.2 (2)
Which of the following are formulae, where R is a 2-place predicate and P is a 1-place predicate, and f, g are as in the previous exercise? In the negative cases, give a brief reason. (a) $R(a,a)$, (b) $R(x,b) \wedge \neg P(g(y))$, (c) $R(x,P(b))$, (d) $P(P(x))$, (e) $\exists P(Px)$, (f) $\forall x(Px \rightarrow Rxy)$, (g) $\forall x \exists y(Rxy)$, (h) $\forall x \forall x(Px)$.

Solution

(a) (b) Yes.

(c) (d) No, because $P(b)$, $P(x)$ are *formulae*, not *terms*, 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.

(e) No: the quantifier has a predicate attached to it, rather than a variable. Such expressions are not formulae of first-order logic. They are admitted in what is called *second-order logic* which, however, goes beyond our remit.

(f) (g) Yes, provided we understand them as having some commas and/or brackets omitted without ambiguity.

(h) Yes, despite the fact that both quantifiers have the same variable attached to them. To be sure, such formulae are rather pathological, as we will see later, and some presentations do exclude them. But the standard presentations find it convenient to allow them in.

Exercise 9.1.2 (3)
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 'is a tiger', D for '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 someone, (g) Everyone who loves someone is loved

by that person, (h) There are exactly two prime numbers less than 5, (i) My mother's father is older than my father's mother.

Solutions

(a) It 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! Notice that the universal quantifier is not explicit in the English, but it evidently part of the meaning. Well, to be honest, in everyday street English, the statement would usually mean something less radical, along the lines of 'lions and tigers are almost always dangerous' but, in our treatment of first-order logic, we will be understanding the universal quantifier literally: 'all' means 'all', no exceptions.

(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*, writing simply $\forall x(Rx \rightarrow \neg Ex)$. Declaring a domain of discourse is often useful when symbolizing; but it is legitimate only when the statement says nothing about anything outside the domain.

(c) $\forall x(Lxa \rightarrow Lxb)$. Here, we declare our domain of discourse to be the set of all people, write Lxy stands for 'x likes y' and use individual constants for Albert and Betty. To reduce clutter, we have omitted commas between the terms of a relational expression when they are simply variables or constants.

(d) (e) In many contexts, the English words 'every' and 'any' tend do much the same work but in others, for example negative ones, they can be used to say markedly different things. Taking the domain of discourse to be people (or some appropriate set of people that contains both Albert and Betty) we can write (d) as $\neg[\forall x(Lbx \rightarrow Lax)]$ while (e) is $[\forall x(Lbx \rightarrow \neg Lax)]$. The difference between them is manifested in the different scopes of the negation operator. These sentences can also be written using the existential quantifier, but we will come to that a little later.

(f) (g) In examples like these, you should first try to understand intuitively the difference of meaning. For (f), you need three quantifiers over the domain of people: $\forall x[\exists y(Lxy) \rightarrow \exists z(Lzx)]$ with bracketing as indicated. We can, however, make use of only two variables, writing $\forall x[\exists y(Lxy) \rightarrow \exists y(Lyx)]$—the reason for this will become clear as we proceed. For (g), on the other hand, two quantifiers suffice and, despite the idiomatic English, both are universal: $\forall x \forall y[Lxy \rightarrow Lyx]$.

(h) The problem here is how to say, with the limited means available, 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 that property and (2) everything with that property is identical to at least one of them. This goes into the language of first-order logic as $\exists x \exists y[Px \wedge Py \wedge (x < 5) \wedge (y < 5) \wedge \forall z\{(Pz \wedge (z < 5)) \rightarrow ((z = x) \vee (z = y))\}]$, where we take the set of positive integers as our domain of discourse and Px means that x is prime.

(i) The easiest way is with function letters $f(x)$, $m(x)$ for the father, mother of x, $O(x,y)$ for x is older than y, and a constant a for myself: $O(f(m(a),m(f(a))$.

No quantifiers or truth-functional connectives needed. It can be done with relations in place of the functions, but in a more complex way saying, in effect, that I have exactly one mother, who has exactly one father, and I have exactly one father, who has exactly one mother, and all of the fathers of my mothers are older than any of the mothers of my fathers. Quite long in words, and just as long when put in symbols.

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 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* all occurrences of the same variable x occurring in α, unless some other quantifier occurring inside α already binds them; it is convenient to say that it also binds the occurrence of the variable x that is attached to it. 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 also called *sentences*.

A full translation into symbols of any complete sentence of English should have all its variables bound, i.e. 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 symbolic representations there are closed formulae.

Exercise 9.1.3

(a) Identify the free and bound occurrences of variables in the formula $\forall z[Rxz \rightarrow \exists y(Rzy)]$.
(b) Identify the free and bound occurrences of variables in the formula $\forall x\exists y(Rxy) \vee \forall z(Py \rightarrow \forall x(Rzx \wedge Ryx))$.
(c) Consider the formula $\forall x\exists y\forall x\exists y(Rxy)$. Which occurrences of y do the two existential quantifiers bind?
(d) Which of the formulae mentioned in this exercise are closed?

Solution

(a) In $\forall z[Rxz \rightarrow \exists y(Rzy)]$, all occurrences of z and of y are bound, while the unique occurrence of x is free.
(b) In $\forall x\exists y(Rxy) \vee \forall z(Py \rightarrow \forall x(Rzx \wedge Ryx))$, all occurrences of x, z are bound while, for y, the occurrences to the left of $\vee$ are bound and those on the right are free. This example illustrates the fact that a single variable can have some

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: e.g. 5, Charlie Chaplin, London
	Variables	$x, y, z, \ldots$	Range over specific objects, combine with quantifiers to express generality
Function letters	1-Pace	$f, g, h, \ldots$	Form compound terms out of simpler terms, starting from the basic ones
	n-Place		
Predicates	1-Place	$P, Q, R, \ldots$	E.g. is prime, is funny, is polluted
	2-Place		E.g. is smaller than, resembles
	n-Place		E.g. lies between (3-place)
	Special relation sign	$=$	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

occurrences bound, others free, in the same formula. It also illustrates how a variable can be bound in a formula while free in one of its sub-formulae; for example, x is bound in $\forall x \exists y(Rxy)$, but is free in its sub-formula $\exists y(Rxy)$.

(c) In the formula $\forall x \exists y \forall x \exists y(Rxy)$, the innermost $\exists$ binds both the occurrence of y attached to it and the y in Rxy. The outer $\exists$ binds only its attached occurrence of y.

(d) Only $\forall x \exists y \forall x \exists y(Rxy)$ is closed; the others are open.

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 a little and continue to cultivate intuitions; with good intuitions, the formal definitions are easier to assimilate.

9.2.1 Quantifier Interchange

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'.

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 the following table, the formulae on the left are logically equivalent to those on the right.

Here α stands for any formula of quantificational logic. We will re-use 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 perhaps $\exists x\neg(\alpha)$,$\forall x\neg(\alpha)$?

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.

Actually, any one of the four equivalences in Table 9.3 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) $\neg\forall x(\alpha \rightarrow \beta) \dashv\vDash \exists x(\alpha \wedge \neg\beta)$, (ii) $\neg\exists x(\alpha \wedge \beta) \dashv\vDash \forall x(\alpha \rightarrow \neg\beta)$, (iii) $\forall x(\alpha \rightarrow \beta) \dashv\vDash \neg\exists x(\alpha \wedge \neg\beta)$, (iv) $\exists x(\alpha \wedge \beta) \dashv\vDash \neg\forall x(\alpha \rightarrow \neg\beta)$. Make free use of the principle of replacement in your work.

Solution

(a) To show $\neg\exists x(\alpha) \dashv\vDash \forall x(\neg\alpha)$: LHS $\dashv\vDash \neg\exists x(\neg\neg\alpha) \dashv\vDash \neg\neg\forall x(\neg\alpha) \dashv\vDash$ RHS by replacement using the first equivalence in the table, double negation. To show that $\forall x(\alpha) \dashv\vDash \neg\exists x(\neg\alpha)$: LHS $\dashv\vDash \neg\neg\forall x(\alpha) \dashv\vDash$ RHS by double negation then replacement using the first equivalence table.

(b) (i) $\neg\forall x(\alpha \rightarrow \beta) \dashv\vDash \exists x\neg(\alpha \rightarrow \beta) \dashv\vDash \exists x(\alpha \wedge \neg\beta)$. (ii) $\neg\exists x(\alpha \wedge \beta) \dashv\vDash \forall x\neg(\alpha \wedge \beta) \dashv\vDash \forall x(\alpha \rightarrow \neg\beta)$. (iii) $\forall x(\alpha \rightarrow \beta) \dashv\vDash \neg\exists x\neg(\alpha \rightarrow \beta) \dashv\vDash \neg\exists x(\alpha \wedge \neg\beta)$. (iv) $\exists x(\alpha \wedge \beta) \dashv\vDash \neg\forall x\neg(\alpha \wedge \beta) \dashv\vDash \neg\forall x(\alpha \rightarrow \neg\beta)$. *End of solution*.

Table 9.3 Quantifier interchange equivalences

$\neg\forall x(\alpha)$	$\exists x(\neg\alpha)$
$\neg\exists x(\alpha)$	$\forall x(\neg\alpha)$
$\forall x(\alpha)$	$\neg\exists x(\neg\alpha)$
$\exists x(\alpha)$	$\neg\forall x(\neg\alpha)$

Table 9.4 Distribution equivalences

$\forall x(\alpha \wedge \beta)$	$\forall x(\alpha) \wedge \forall x(\beta)$
$\exists x(\alpha \vee \beta)$	$\exists x(\alpha) \vee \exists x(\beta)$

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' or, more idiomatically though rather more loosely, 'some αs are not βs'.

Exercise 9.2.1 (2)
To what English statement-forms do the equivalences in Exercise 9.2.1 (1) (b) correspond?

Solution

(b)(ii) LHS: It is not the case that at least one α is a β. Using 'some': It is not the case that some αs are βs. RHS: No αs are βs.
(b)(iii) LHS: All αs are βs'. RHS: It is not the case that at least one α is not a β. Using 'some': It is not the case that some αs are not βs.
(b)(iv) LHS: At least one α is a β. Using 'some': Some αs are βs. RHS: It is not the case that no αs are βs'.

English has many other idioms for expressing these basic relationships, as do all other natural languages. They can also be expressed in the language of elementary set theory using the notations of Chap. 1. They were first subjected to systematic study by Aristotle but without the notion of a variable, nor that of a truth-functional connective, whose roles became clear only in the late nineteenth century.

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 existential quantification distributes over disjunction.

Why does the universal quantifier get on so well with conjunction? Essentially, because it is, itself, like a long conjunction. Suppose we specify a finite domain of discourse with just n elements, naming all of them by n individual constants $a_1, \ldots, a_n$. Then saying that every element of the domain has the property P amounts to saying, of each element in the domain, that it has that property, in other words, given the choice of domain and the names for all its elements, $\forall x(Px)$ says the same as $Pa_1 \wedge \ldots \wedge Pa_n$. The latter is called a *finite transform* of the former and we write $\forall x(Px) \approx_n Pa_1 \wedge \ldots \wedge Pa_n$.

Of course, if we change the size of the domain, then we change the length of the conjunction. To make the transform work we must have enough constants to name

all the elements of the domain (it doesn't matter, however, if some of them are named more than once). But, even with these provisos, the principles for $\forall$ reflect those for $\wedge$, while those for $\exists$ resemble familiar ones for $\vee$.

Exercise 9.2.2

(a) Write out $\forall x(Px \wedge Qx)$ as a conjunction in a domain of three elements named a, b, c. Then write out $\forall x(Px) \wedge \forall x(Qx)$ in the same way and explain in terms of familiar tautological equivalences why they are equivalent.
(b) What does $\exists x(Px)$ amount to in a domain of n elements?
(c) 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?

Solution

(a) $\forall x(Px \wedge Qx) \approx_3 (Pa \wedge Qa) \wedge (Pb \wedge Qb) \wedge (Pc \wedge Qc)$ while $\forall x(Px) \wedge \forall x(Qx) \approx_3 (Pa \wedge Pb \wedge Pc) \wedge (Qa \wedge Qb \wedge Qc)$. They are equivalent by several applications of association and commutation for conjunction.
(b) Call the domain elements $a_1, \ldots, a_n$. Then $\exists x(Px) \approx_3 Pa_1 \vee \ldots \vee Pa_n$.
(c) They correspond to four forms of de Morgan.

9.2.3 Vacuity and Re-lettering

The last intuitive equivalences that we mention are the vacuity and re-lettering principles. Two vacuity equivalences are expressed in Table 9.5, where all formulae in a given row are equivalent.

In other words, to quantify twice in a row on the same variable does no more than quantifying on it once. To say, 'there is an x such for every x we have Px', as in the top row right column, is just a redundant way of saying 'Px for every x'. Note that in these vacuity equivalences it is the 'inner' quantifier that 'takes precedence' and that *the variables must be the same*: $\forall x(Rxy)$ is *not* logically equivalent to $\forall y \forall x(Rxy)$, nor to $\exists y \forall x(Rxy)$.

The equivalences of Table 9.5 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 accords with this principle, take φ to be $\forall x(\alpha)$ in the top row, while in the bottom row put φ to be $\exists x(\alpha)$.

Table 9.5 Two vacuity equivalences

$\forall x(\alpha)$	$\forall x \forall x(\alpha)$	$\exists x \forall x(\alpha)$
$\exists x(\alpha)$	$\exists x \exists x(\alpha)$	$\forall x \exists x(\alpha)$

Table 9.6 Two more vacuity equivalences

Proviso: when there are no free occurrences of x in α	
$\forall x(\alpha \vee \beta)$	$\alpha \vee \forall x(\beta)$
$\exists x(\alpha \wedge \beta)$	$\alpha \wedge \exists x(\beta)$

Another two vacuity principles are given in Table 9.6 where, again, all formulae in a given row are equivalent. Note that, this time, the equivalences are asserted only in the circumstance that no occurrences of the variable x occur free in α.

Exercise 9.2.3 (1)

(a) Use vacuity and distribution equivalences as well as suitable tautological equivalences to show that (i) $\exists x(Px \vee \exists y \forall y(Qy)) \dashv\vDash \exists x \forall y(\neg Px \rightarrow Qy)$ and (ii) $\forall x(Px \rightarrow \exists y(Qy)) \dashv\vDash \forall x \exists y(Px \rightarrow Qy)$.

(b) In each of the above, the RHS has a special form. Can you guess what it is?

Solution

(a)(i) $\exists x(Px \vee \exists y \forall y(Qy)) \dashv\vDash \exists x(Px \vee \forall y(Qy)) \dashv\vDash \exists x \forall y(Px \vee Qy) \dashv\vDash \exists x \forall y(\neg Px \rightarrow Qy)$. The three equivalences are justified respectively by Tables 9.5 and 9.6 (noting that y does not occur free in Px), and a tautological equivalence.

(a)(ii) $\forall x(Px \rightarrow \exists y(Qy)) \dashv\vDash \forall x(\neg Px \vee \exists y(Qy)) \dashv\vDash \forall x(\exists y(\neg Px) \vee \exists y(Qy)) \dashv\vDash \forall x \exists y(\neg Px \vee Qy) \dashv\vDash \forall x \exists y(Px \rightarrow Qy)$. The four equivalences are justified respectively by: a tautological equivalence, the general vacuity principle enunciated after Table 9.5, a distribution principle from Table 9.4, another tautological equivalence.

(b) In each of the two equivalences, the RHS is a formula with all its quantifiers 'up the front', that is, it is of the form $\mathcal{Q}_1 x_1, \ldots, \mathcal{Q}_n x_n(\alpha)$ where each $\mathcal{Q}_i$ is a universal or existential quantifier and α is a formula with no quantifiers in it. *End of solution*.

Formulae with all quantifiers 'up the front', in the sense defined in the solution to Exercise 9.2.3 (1) (b), are said to be in *prenex* (or *maxiscope*) *normal form*. It can be shown that every formula of quantificational logic is logically equivalent to (at least) one of that kind, which can be found algorithmically by successive transformations using principles such as distribution, vacuous quantification (already described) and re-lettering, which we shall explain in a moment.

Prenex normal forms can be very useful for the mathematical analysis of theories expressed in the language of first-order logic, as well as for programming purposes; but they are not always easy to process in the head, without pencil and paper. Humans are better at handling formulae in *miniscope form*, that is, where each quantifier has the smallest possible scope, and this fact is reflected by the frequency with which ordinary speech tends to correspond to the latter rather than the former. There is a striking example of this in the first end-of-chapter exercise.

We will not be considering either of these kinds of normal form in this book, but readers wishing to look into prenex normal forms can easily find material in the references at the end of this chapter. Unfortunately, there has been little formal study of miniscope forms, as they are of more interest to cognitive scientists and linguists than to the mathematicians and computer scientists who have carried out most of the research in mathematical logic over the last century.

To explain re-lettering, we begin with some examples. Intuitively, it is clear that $\forall x(Px) \nvdash \forall y(Py)$ and $\forall x(Px \wedge Qy) \nvdash \forall z(Pz \wedge Qy)$—although $\forall x(Px \wedge Qy)$ is not equivalent to $\forall y(Py \wedge Qy)$ where the free occurrence of x in in the former has become a bound occurrence of y in the latter. Again, $\forall x\exists y(Rxy) \nvdash \forall x\exists z(Rxz) \nvdash \forall y\exists z(Ryz) \nvdash \forall y\exists x(Ryx)$—although $\forall x\exists y(Rxy)$ is certainly not equivalent to $\forall x\exists y(Ryx)$ where the variables attached to the quantifies have been left unchanged but those on the body Rxy are reversed to Ryx; nor is $\forall x\exists y(Rxy)$ equivalent to $\forall x\exists x(Rxx)$, where two distinct variables, bound by different quantifiers, become a single variable bound by just the inner quantifier.

The general principle of *re-lettering of bound variables* lying behind the equivalences thus needs careful enunciation. 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 $\varphi \nvdash \varphi'$, 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; so too is the restriction to replacing only bound occurrences of x in φ.

Exercise 9.2.3 (2)

(a) Use re-lettering to show in several steps that $\exists x\forall y(Rxy) \nvdash \exists y\forall x(Ryx)$.
(b) Can re-lettering be used to show that $\exists x\forall y(Rxy) \nvdash \exists y\forall x(Rxy)$?
(c) Is there an analogous principle of re-lettering of free variables?

Solution

(a) You need to bring in and then eliminate a fresh variable z. In detail, $\exists x\forall y(Rxy) \nvdash \exists z\forall y(Rzy) \nvdash \exists z\forall x(Rzx) \nvdash \exists y\forall x(Ryx)$. The second equivalence is legitimate because the variable x that it puts in place of y is fresh to the outcome of the first equivalence; similarly, the variable y that the third equivalence puts in place of z is fresh to the outcome of the second equivalence.
(b) No. Syntactically, the reason is that no matter how many replacements you make, the variable attached to a quantifier will continue to occupy the same place in the body of the formula, that is, the part between parentheses. Intuitively, if one thinks of meanings, the two sentences say different things. We will soon be able to make this more precise using a semantics for first-order logic.
(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 incorrect 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 already familiar from propositional logic: assignments, valuations, tautological implication and so on. It turns out that there are two ways of doing this; we begin by explaining the common ingredients.

9.3.1 The Shared Part of the Two Semantics

Just as in propositional logic, valuations are certain functions on the set of formulae into the set $\{0,1\}$, and they are defined recursively from assignments. Both notions are considerably more complex than their propositional counterparts, to cope with the more complex language. Our terminology and notation will follow as closely as possible the propositional case, to highlight the parallels and locate the substantive differences.

A *domain* (aka *universe*) *of discourse* is any non-empty set, in this context written D. An *assignment wrt domain D* is any function v that assigns a value to each constant sign, variable, function letter and predicate letter of the language in the following manner.

- For each constant sign a, $v(a)$ is an element of the domain, i.e. $v(a) \in D$,
- Likewise, for each variable x, $v(x)$ is an element of the domain, i.e. $v(x) \in D$,
- For each n-place function letter f, $v(f)$ is an n-argument function over the domain, i.e. $v(f)$: $D^n \rightarrow D$,
- For each n-place predicate P other than the identity symbol, $v(P)$ is an n-place relation over the domain, i.e. $v(P) \subseteq D^n$.
- For the identity symbol, we put $v(=)$ to be the identity relation over the domain.

The last clause of this definition means that we are giving the identity symbol special treatment. Whereas other predicate symbols of the language may be interpreted as *any* relations of the right arity (i.e. number of places) over the domain, the identity symbol is always read as the relation of *identity* over the domain. Such a semantics is 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 well-behaved in a certain respect (sometimes known as *congruence*) in its interaction with all function letters and predicates. However, in this chapter we will confine ourselves, as is customary, to standard assignments.

Given an assignment wrt domain D, we define valuations v^+ recursively, in two stages. The first stage defines recursively $v^+(t)$ for any term t of the language, following the definition of the terms themselves. Recall from Sect. 9.1.2 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.

$v^+(a) = v(a)$ for every constant a of the language
$v^+(x) = v(x)$ for every variable x 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 value assigned to the function letter f, which will be a function on D^n into D, and applying that function to the values, assigned or already defined, of the terms $t_1, \ldots, t_n$. The values of those terms are all elements of D, and so the value of the compound term $f(t_1, \ldots, t_n)$ is also an element of D. The notation may at first make the definition look complicated, but the underlying idea is natural and simple. As v^+ is a uniquely determined extension of v we will, as soon as the definitions are over, follow the same 'abuse of notation' as in propositional logic and write it simply as v, thereby eliminating an endless repetition of superscripts.

There are some contexts in which the notation could be simplified further. When considering only *one* valuation, we could reduce clutter by writing $v^+(t)$ as $\underline{t}$, that is, by simply underlining the term. But in contexts where we are considering more than one valuation (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, which would soon become cumbersome.

Exercise 9.3.1
Rewrite the recursion step of the definition of v^+ using the underlining notation.

Solution
$\underline{f(t_1,\ldots,t_n)} = \underline{f}(\underline{t_1},\ldots,\underline{t_n})$, where on the LHS the whole expression is underlined while on the RHS the signs f, $t_1,\ldots,t_n$ are underlined. This is certainly less cluttered than writing $v^+(f(t_1, \ldots, t_n)) = (v(f))(v^+(t_1), \ldots, v^+(t_n))$, but to understand it one has to look carefully for the underlining. *End of solution*.

The second stage in the recursive definition of a valuation v^+ given an assignment v defines $v^+(\alpha) \in \{0,1\}$ for any formula of the language. As is to be expected, the definition follows the recursive construction of formulae. In each clause, 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 $t_1 = t_2$: $v^+(t_1 = t_2) = 1$ iff $(v^+(t_1), v^+(t_2)) \in v(=)$. Since $v(=)$ is required to be the identity relation over D, this means that $v^+(t_1 = t_2) = 1$ 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 former is familiar:

- *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 lies at 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. Passions can run high over which is better to use; in the author's view, sometimes one is more convenient, sometimes the other—in any case, it is advisable to understand both of them. They are known as the *substitutional* and *x-variant* readings of the quantifiers.

9.3.2 Substitutional Reading

The substitutional reading of the quantifiers is not often used by mathematicians but is frequently employed by computer scientists and philosophers; we present it first because, in the author's experience, non-mathematical students find it easier to grasp. We emphasize, however, that it is no less rigorous than the *x*-variant reading that will follow.

As promised earlier, from this point onwards we simplify ('abuse') notation by dropping the superscripted plus sign from v^+ when speaking about valuations: v will stand for both the valuation and its underlying assignment. As the valuation is an extension of the assignment which, in turn is a restriction of the valuation to the domain of ingredients of the language, this should not lead to any confusion and context will also make it clear, in any instance, which is meant.

First, one needs to explain a particular kind of substitution. Given a formula α and a variable x, write $\alpha_{x:=t}$ 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 $\alpha_{x:=a} = \forall z[Raz \rightarrow \exists x(Rzx)]$, obtained by replacing the unique free occurrence of x in α by the constant a, leaving the two bound occurrences of x untouched.

Exercise 9.3.2 (1)

(a) Let α be the same formula $\forall z[Rxz \rightarrow \exists x(Rzx)]$. Write out $\alpha_{x:=b}$, $\alpha_{y:=b}$, $\alpha_{z:=b}$.
(b) Do the same for the formula $\forall x\exists y(Rxy) \vee \exists z[Py \rightarrow \forall x(Rzx \wedge Ryx)]$.

Solution

(a) $\alpha_{x:=b} = \forall z[Rbz \rightarrow \exists x(Rzx)]$ but $\alpha_{y:=b} = \forall z[Rxz \rightarrow \exists x(Rzx)] = \alpha = \alpha_{z=b}$, since there are no free occurrences of y in α, nor any free occurrences of z in α. In this case we say that the substitution is *vacuous*.
(b) Calling the formula in question β, we have $\beta_{x:=b} = \beta = \beta_{z:=b}$, since both substitutions are vacuous. But $\beta_{y=b} = \forall x\exists y(Rxy) \vee \exists z[Pb \rightarrow \forall x(Rzx \wedge Rbx)]$, replacing the two free occurrences of y in β by b. *End of solution*.

We are now ready to give the substitutional reading of the quantifiers under a valuation v with domain D.

We first make sure that the function v, restricted to the set of all constants of the language, is surjective, that 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 constant signs to the language to be able to extend the assignment function to ensure that it is onto D and consider that extension. Then we evaluate the quantifiers as follows.

- $v(\forall x(\alpha)) = 1$ iff $v(\alpha_{x:=a}) = 1$ for every constant symbol a of the language (thus extended).
- $v(\exists x(\alpha)) = 1$ iff $v(\alpha_{x:=a}) = 1$ for at least one constant symbol a of the language (thus extended).

The reason for requiring surjectivity 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'. There is no need to require injectivity.

Alice Box: Oddities

Alice One moment, there is something funny here! The surjectivity requirement has a strange consequence: the language itself is not fixed independently of its interpretation, since the supply of constants in the language may vary according to the 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 philosophers and 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)

(a) With a domain D consisting of two items 1 and 2, construct a valuation that makes the formula $\forall x(Px \vee Qx)$ true but makes $\forall x(Px) \vee \forall x(Qx)$ false. Show in detail your calculations of the respective truth-values.

(b) Do the same for $\forall x \exists y(Rxy)$ and $\exists y \forall x(Rxy)$.

Solution

(a) With $D = \{1,2\}$, put $v(P) = \{1\}$, $v(Q) = \{2\}$, and let a, b be constants in the language 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 $v(\forall x(Px)) = 0$; likewise $v(Qa) = 0$ so that $v(\forall x(Qx)) = 0$. Thus $v(\forall x(Px) \vee \forall x(Qx)) = 0$ as desired.

(b) With $D = \{1,2\}$ we specify constant signs in the language that are assigned 1, 2 as their respective values. We could use a, b as in part (a) of this exercise, but we can simplify mental coordination by using the numerals '1', '2' as constants with $v(\text{'1'}) = 1$ and $v(\text{'2'}) = 2$; we can also abuse notation by omitting the inverted commas. Put $v(R) = \{(1,1), (2,2)\}$. Suggestion: for visualization, draw an arrow diagram (aka digraph, Chap. 2, Sect. 2.2.2). We claim that $v(\forall x \exists y(Rxy)) = 1$ while $v(\exists y \forall x(Rxy)) = 0$.

For the former, it suffices to check that both $v(\exists y(R1y)) = 1$ and $v(\exists y(R2y)) = 1$, so in turn it suffices to show that either $v(R11) = 1$ or $v(R12) = 1$, and either $v(R21) = 1$ or $v(R22)) = 1$. But, since $v(R) = \{(1,1), (2,2)\}$, we have $v(R11) = 1 = v(R22)$ which, by truth-functional calculation, does the job. For the latter, it suffices to show that both $v(\forall x(Rx1)) = 0$ and $v(\forall x(Rx2)) = 0$. But we have the first since $v(R21) = 0$ and we have the second since $v(R12)) = 0$. *End of solution*.

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 valuation with domain D, each named by a constant of the language. Then, under the substitutional reading, for any universally quantified formula we have $v(\forall x(\alpha)) = 1$ iff $v((\alpha_{x:=a}) = 1$ 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 $v(\alpha_{x:=a_1} \wedge \ldots \wedge \alpha_{x:=a_n}) = 1$. Similarly for the existential quantifier: $v(\exists x(\alpha)) = 1$ iff $v(\alpha_{x:=a_1} \vee \ldots \vee \alpha_{x:=a_n}) = 1$.

Thus the truth-value of a formula under a valuation v with a finite domain D of n elements can also be calculated by a *translation into a quantifier-free formula* (that is, the only logical operators in it are truth-functional connectives). We recursively translate $\forall x(\alpha)$ and $(\exists x(\alpha)$ into $(\alpha_{x:=a_1} \wedge \ldots \wedge \alpha_{x:=a_n})$ and $(\alpha_{x:=a_1} \vee \ldots \vee \alpha_{x:=a_n})$ respectively, where $a_1, \ldots, a_n$ are constants chosen to name all elements of D.

Exercise 9.3.2 (3)
Do Exercise 9.3.2 (2) (a) again, but this time translating into quantifier-free formulae and assigning truth-values.

Solution
Take constants a, b. The translation of $\forall x(Px \vee Qx)$ is $(Pa \vee Qa) \wedge (Pb \vee Qb)$ while the translation of $\forall x(Px) \vee \forall x(Qx)$ is $(Pa \wedge Pb) \vee (Qa \wedge Qb)$. Let v be the

propositional assignment that e.g. puts $v(Pa) = 1 = v(Qb)$ and $v(Pb) = 0 = v(Qa)$. Then by truth-tables $v((Pa \vee Qa) \wedge (Pb \vee Qb)) = 1$ while $v((Pa \wedge Pb) \vee (Qa \wedge Qb)) = 0$ as desired.

Alice Box: Why the colon?

Alice One last question. Why the colon in the subscripts of $\alpha_{x:=a}$, $\alpha_{x:=b}$?

Hatter We are substituting constants for variables, not conversely. Equality is symmetric, so we need a sign that reflects the difference. We borrow an idea from computing and write : = rather than = .

Alice Isn't that a bit pedantic? Aren't we trying to simplify notation?

Hatter Perhaps it is. One could omit the colon so long as one understands what one is doing. Keeping it has the incidental advantage of helping flag the difference between substitution and another operation called replacement, to be defined in Sect. 9.4.4 when analysing identity.

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 valuation v looked at the values of certain formulae that are (except in limiting cases) *different* from α under the *same* valuation v, the *x*-variant reading will consider the values of the *same* formula α under certain valuations that are (except in limiting cases) *different* from v.

Let v be any valuation with domain D, and let x be any variable of the language. By an *x-variant* of v we mean any valuation 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 valuation 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 valuation v' of v.
$v(\exists x(\alpha)) = 1$ $v'(\alpha) = 1$ for at least one *x*-variant valuation v' of v.

Exercise 9.3.3 (1)
Re-solve Exercise 9.3.2 (2) using the *x*-variant reading of the quantifiers and compare the calculations. In other words:

(a) With domain $D = \{1,2\}$, construct a valuation v such that under the x-variant reading of the quantifiers that makes $\forall x(Px \vee Qx)$ true but $\forall x(Px) \vee \forall x(Qx)$ false. Show your calculations.
(b) Do the same for $\forall x \exists y(Rxy)$ and $\exists y \forall x(Rxy)$.

Solution

(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 $v(\forall x(Px \vee Qx)) = 1$ while $v(\forall x(Px) \vee \forall x(Qx)) = 0$.

To show that $v(\forall x(Px \vee Qx)) = 1$, it suffices to show that $v'(Px \vee Qx) = 1$ for every x-variant valuation v' of 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 $v_{x:=2}(x) = 2$. Now, we have $v_{x:=1}(Px) = 1$ while $v_{x:=2}(Qx) = 1$, so $v_{x:=1}(Px \vee Qx) = 1 = v_{x:=2}(Px \vee Qx)$ and thus $v(\forall x(Px \vee Qx)) = 1$. To show that $v(\forall x(Px) \vee \forall x(Qx)) = 0$, 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 $v(\forall x(Px) \vee \forall x(Qx)) = 0$ as desired.
(b) Put $v(R) = \{(1,1), (2,2)\}$. We claim that $v(\forall x \exists y(Rxy)) = 1$ while $v(\exists y \forall x(Rxy)) = 0$. To show that $v(\forall x \exists y(Rxy)) = 1$, it suffices to show that $v'(\exists y(Rxy)) = 1$ for every x-variant valuation v' of 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 $v_{x:=2}(x) = 2$ (while leaving the value of y unchanged). Consider first $v_{x:=1}(\exists y(Rxy))$. To check that this equals 1, it suffices to observe that $v_{x:=1,y:=1}(Rxy) = 1$, where $v_{x:=1,y:=1}$ is the y-variant of $v_{x:=1}$ that gives y the value 1 (while keeping the value of x at 1). The verification that $v_{x:=2}(\exists y(Rxy)) = 1$ is similar with 2 in place of 1.

To show that $v(\exists y \forall x(Rxy)) = 0$, we need to show that $v'(\forall x(Rxy)) = 0$ for every y-variant valuation v' of v. But there are only two such y-variants, $v_{y:=1}$ and $v_{y:=2}$, defined by putting $v_{y:=1}(y) = 1$ and $v_{y:=2}(y) = 2$ (while leaving the value of x unchanged). Consider first $v_{y:=2}(\forall x(Rxy))$. To check that this equals 0, it suffices to observe that $v_{y:=2,x:=1}(Rxy) = 0$ where $v_{y:=2,x:=1}$ is the x-variant of $v_{y:=2}$ that gives x the value 1 (while keeping the value of y at 2). The verification that $v_{y:=1}(\forall x(Rxy)) = 0$ is similar with 2 and 1 interchanged. *End of solution.*

In Exercise 9.3.3 (1), we used a notation for x-variants, $v_{x:=1}$, $v_{x:=2}$ etc., that allows us to keep track of what is going on. It runs parallel to the notation for the substitutional reading of the preceding section, but with $x{:}=1$, $x{:}=2$ etc. subscripted to valuations v for the x-variant reading and to formulae α for the substitutional reading. At some deep level, the two procedures are thus 'doing the same thing'.

Evidently, detailed verifications like those in Exercises 9.3.2 (2) and 9.3.3 (1) are tedious when done by hand, even with very small domains, and become more than exponentially so as the size of the domain increases. Moreover, for infinite domains one cannot simply run through a finite number of substitutions or make a finite

number of x-variants; verification of the truth-value of a formula under a valuation into an infinite domain requires more abstract reasoning and is not in general algorithmic. However, once one has done a number of verifications, one can begin to 'see' the value of fairly simple formulae under a given valuation without writing out all the details. One can also program a computer to do the job, as far as specified finite domains are concerned.

Exercise 9.3.3 (2)
In the opening paragraph of this section the substitutional and x-variant readings were contrasted by saying that whereas the substitutional reading of $\forall x(\alpha)$ under v considers the values of certain formulae that are (except in limiting cases) *different* from α under the *same* valuation v, the x-variant reading considers the values of the *same* formula α under certain valuations that are (except in limiting cases) *different* from v. Why the qualifications 'except in limiting cases'?

Solution
For the substitutional reading: in the limiting case that there are no free occurrences of x in α, then $\alpha_{x:=a}$ is the formula α itself. For the x-variant reading: in the limiting case that $v(x) = a \in D$, then $v_{x:=a}$ is the valuation v itself. *End of solution.*

It should be clear from the exercises that, under both the substitutional and the x-variant readings, the truth-value of a formula α under a valuation v is independent of the value that v gives to the variables that do not occur free in α. For example, in the formula $\forall x(Px \vee Qy)$ the variable x does not occur free—all its occurrences are bound by the initial universal quantifier—so the truth-value of that formula under a valuation v is independent of the value $v(x)$ of the variable x—although it does depend on the value the truth-value $v(y)$. Hence, for sentences (aka closed formulae) α, since no variable occurs free in α, $v(\alpha)$ is always independent of $v(x)$ for any variable x whatsoever.

Alice Box: Truth of formulae that are not closed

Alice I have been browsing in several other books on logic written for students of mathematics. They all use the x-variant account, but in a rather different way.

Hatter How is that?

Alice Under the definitions in this chapter, a formula always gets a specific truth-value under a given valuation 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, then in general it is *neither true nor false in a given model*! What is going on?

Hatter That is another way of building the x-variant semantics. It is different in some of the terminology, but in the end it does the same job. The two accounts agree on closed formulae.

The Hatter's response is correct but far too laconic; Alice deserves rather more, although students who have not been exposed to the alternative way of proceeding may happily skip to the next section.

Under the presentation that Alice has been reading, formulae are evaluated in exactly the same way as above, but with a difference in terminology. Rather than speaking of truth and falsity and writing $v(\alpha) = 1$ and $v(\alpha) = 0$, the valuation is said to 'satisfy' or 'not satisfy' the formula, writing $v \vDash \alpha$ or $v \nvDash \alpha$. A 'model' is understood as a valuation stripped of the values assigned to variables, and a formula is called 'true' in a given model iff it is satisfied by *every valuation* extending that model to cover variables, that is, by every valuation that agrees with the model but in addition gives values to the variables of the language. A formula is said to be false in the model iff it is satisfied by *none* of those valuations.

Now, using that terminology, a formula containing free occurrences of variables will in general be neither true nor false in a given *model*: some assignments of values in the domain to its free variables may satisfy it, others not. The simplest example is the atomic formula Px. The model with domain $D = \{1,2\}$ and $v(P) = \{1\}$ does not make Px true, nor false in the sense we are now considering, since the valuation with $v(x) = 1$ satisfies Px while the valuation with $v(x) = 2$ does not. But when a formula is closed (that is, 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 the present text does not follow the alternative manner of presentation is that it creates unnecessary difficulties for students. Having become accustomed to the fundamental bivalence of propositional logic, they are then asked to deploy the terms 'true' and 'false' in a trivalent way in quantificational logic—and yet are often told, quite correctly, that this is still classical two-valued logic, not some many-valued one. A sure recipe for classroom confusion! The terminology followed in this book (along with quite a few others) retains bivalence for the terms 'true' and false', just as in propositional logic, rather than complicate matters by making those terms trivalent and shifting bivalence to fresh terms 'satisfy' and 'does not satisfy'

9.4 Semantic Analysis

We now apply the semantic tools developed in Sect. 9.3 to analyse the notion of logical implication in the presence of quantifiers and establish some fundamental rules. They are even more fundamental, though more technical and less familiar, than the equivalences listed in tables of Sect. 9.2.

9.4.1 Logical Implication

To define notions of logical relationship and logical status in quantificational logic, we simply take the definitions of the corresponding notions in truth-functional logic (Chap. 8, Sect. 8.3) and re-understand them in terms of the more elaborate notion of a valuation in the first-order context. To mark the difference of context, the convention is to reserve the term 'tautological' for the propositional context and speak instead of 'logical' this and that.

- A set A of formulae is said to *logically imply* a formula β iff there is no valuation 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, $A \models \beta$, and we also say that β is a *logical consequence* of A.
- α is *logically equivalent* to β, and we write $\alpha \dashv\models \beta$, iff both $\alpha \models \beta$ and $\beta \models \alpha$. Equivalently: $\alpha \dashv\models \beta$ iff $v(\alpha) = v(\beta)$ for every valuation v.
- α is a *logically true*, and we write $\models\alpha$, iff $v(\alpha) = 1$ for every valuation v; it is a *contradiction* iff $v(\alpha) = 0$ for every valuation v: it is *contingent* iff it is neither logically true nor a contradiction.
- More generally, a set A of formulae is *satisfiable* iff there is some valuation v such that $v(\alpha) = 1$ for every formula $\alpha \in A$. Otherwise, it is *unsatisfiable*.

Evidently, the bulleted concepts above are interrelated in the same way as their counterparts in propositional logic with the same verifications (Sect. 8.3). For example, $\models \alpha$ iff $\varnothing \models \alpha$.

In the exercises of Sect. 9.3 we checked some negative results; in particular, we saw that $\forall x(Px \vee Qx) \not\models \forall x(Px) \vee \forall x(Qx)$ and $\forall x\exists y(Rxy) \not\models \exists y\forall x(Rxy)$. In both instances, we found a valuation 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 $\alpha \not\models \beta$ but such that $v(\beta) = 1$ for every valuation v in any finite domain with $v(\alpha) = 1$. In other words, there are some non-implications that 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 second of the non-implications just mentioned 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 but 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, here written (…), remains unchanged. The figure is complete in the sense that when there is no path in it following arrows from φ to ψ, then in fact $\varphi \not\models \psi$.

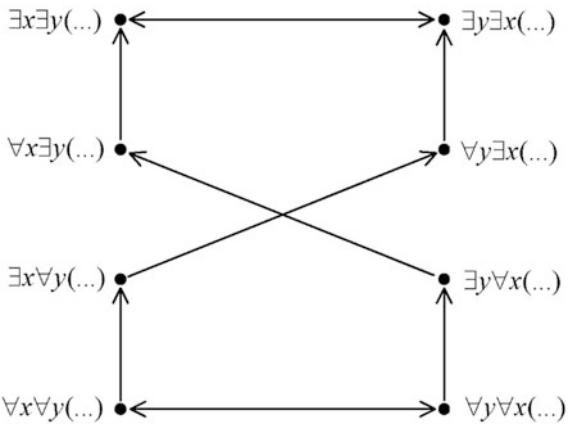

Fig. 9.1 Logical relations between alternating quantifiers

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 valuation in the smallest domain that you can get away with. Use either the x-variant or substitutional account of the quantifier in your verification as you prefer, but in either case follow the notation that was given in Sect. 9.3.

Solution
This is not quite the same as Exercise 9.3.2 (2) (b). There it was shown, in effect, that $\forall x \exists y(Rxy) \nvDash \exists y \forall x(Rxy)$ permuting quantifiers-with-attached-variables. Now we check that $\exists x \forall y(Rxy) \nvDash \exists y \forall x(Rxy)$, keeping the order of quantifiers fixed but permuting the variables attached.

The smallest domain in which this can be done has two elements. Put $D = \{1,2\}$ and $v(R) = \{(1,1), (1,2)\}$ (draw a digraph for visualization). We claim that $v(\exists x \forall y (Rxy)) = 1$ while $v(\exists y \forall x(Rxy)) = 0$, verifying with the substitutional reading of the quantifiers. To check that $v(\exists x \forall y(Rxy)) = 1$, it suffices to show that either $v(\forall y (R1y)) = 1$ or $v(\forall y(R2y)) = 1$; the former holds since both $v(R11)) = 1$ and $v(R12) = 1$ immediately from the valuation of R. To show $v(\exists y \forall x(Rxy)) = 0$ it suffices to show that both $v(\forall x(Rx1)) = 0$ and $v(\forall x(Rx2)) = 0$; the former holds since $v(R21)) = 0$ while the latter holds since $v(R22) = 0$. *End of solution*.

So far, we have been using the semantics to get negative results. We can also use it to get positive ones.

Exercise 9.4.1 (2)
Verify $\exists x \forall y(\alpha) \vDash \forall y \exists x(\alpha)$ using the x-variant reading of the quantifiers.

Solution
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 $v_{x=a}$. Since $v_{x:=a}(\forall y(\alpha)) = 1$, we have $v_{x:=a,y:=b}(\alpha) = 1$; likewise, since $v_{y:=b}(\exists x(\alpha)) = 0$, we get $v_{y:=b,x:=a}(\alpha) = 0$. 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 was implicitly presupposed in the formulation of the problem. For the former is obtained by re-specifying the value given to x and then re-specifying the value given to the different variable y, while the latter re-specifies in reverse order. So $1 = v_{x:=a,y:=b}(\alpha) = v_{x:=a,y:=b}(\alpha) = 0$, giving us a contradiction.

Alice Box: This looks like a fraud

Alice You make me laugh! In Exercise 9.4.1 (2) you are verifying an intuitively obvious principle and, to do it, you use the very same principle! OK, you do it in the metalanguage rather than the object language, but it still looks like a fraud.

Hatter A disappointment, yes; a pain in the neck, I agree. But not a fraud. To establish a principle expressed in the object language, we must reason and our reasoning is carried out in a metalanguage. Back in propositional logic we did exactly the same—in a sense it was even worse, because there too we reasoned with quantifiers although they were not even present in the object language.

Alice So, what's the advantage? The metalinguistic verification is sometimes more complicated and less transparent than the item in the object-language that it is verifying. Why not just think intuitively in the object language?

Hatter Indeed, heuristically, one does just that, but it can quickly become vague and unmanageable without assistance. One advantage of using the formal semantics is that verification often becomes a matter of calculation. We have no need to guess, intuit, or carry loads of information in our heads; it is all down on paper.

Alice When dealing with infinite domains, calculation alone may not suffice.

Hatter Quite so. But with the resources supplied by the formal semantics, the task of making an informed guess, as well as that of articulating a watertight verification for it, become more straightforward. Moreover, checking whether a proposed verification really works is still a matter of calculation.

Exercise 9.4.1 (3)
Verify $\forall x(\alpha) \vDash \exists x(\alpha)$ using (i) the x-variant and then (ii) the substitutional reading of the quantifiers.

Solution
The phenomenon discussed by Alice and the Hatter appears again in the solution to this exercise. Let v be any valuation with domain $D \neq \varnothing$, and suppose $v(\forall x(\alpha)) = 1$. We need to check that $v(\exists x(\alpha)) = 1$.

(i) Using the x-variant reading: Since $v(\forall x(\alpha)) = 1$, we have $v'(\alpha) = 1$ for every x-variant v' of v. Hence, in particular, $v(\alpha) = 1$ and so $v'(\alpha) = 1$ for at least one x-variant v' of v and thus $v(\exists x(\alpha)) = 1$ as desired.
(ii) Using the substitutional reading: Since $v(\forall x(\alpha)) = 1$, we have $v(\alpha_{x:=a}) = 1$ for every constant a of the language. $D \neq \varnothing$ and every element of D is given a constant as name (Sect. 9.3.2), we have $v(\alpha_{x:=a}) = 1$ for at least one constant a so, by the evaluation rule for the existential quantifier, $v(\exists x(\alpha)) = 1$ as desired. *End of solution*.

We note in passing that some philosophers have seen the implication $\forall x(\alpha) \vDash \exists x(\alpha)$ as a defect of first-order logic. They feel that, for complete generality, one should admit the empty domain and extend the relevant definitions in such a way that there is a valuation v into it with $v(\forall x(\alpha)) = 1$ and $v(\exists x(\alpha)) = 0$. While this idea sounds attractive, its rigorous execution turns out to be tricky. The first point to notice is that there are no functions taking a non-empty set, such as the set of individual constants or that of the individual variables of the language, into the empty set. Thus, unless we modify the definitions, there are no assignments, hence no valuations, into the empty domain. To ensure that there is a valuation v into $\varnothing$ with $v(\forall x(\alpha)) = 1$ and $v(\exists x(\alpha)) = 0$ as proposed above, we have to reconstruct the semantics, beginning with the definition of an assignment. But, despite valiant attempts in the literature, it is still not clear how that can be done without affecting the treatment of non-empty domains or creating endless complications. Since, in practice, we never knowingly work with the empty domain, few logicians take that path.

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 already a term. So when we substitute t for free occurrences of x in α, it may happen that t is, or 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 $\alpha_{x:=y} = \exists y(Ryy)$, where the y that is introduced is in the scope of the existential quantifier.

Such substitutions are inappropriate from the point of view of two of the principles that we are about to articulate. We say that a substitution passing from α to $\alpha_{x:=t}$ is *clean* iff no free occurrence of x in α falls within the scope of a quantifier that binds some variable occurring in t.

Exercise 9.4.2
Let $\alpha = \exists y\{Py \vee \forall z[Rzx \wedge \forall x(Sax)]\}$. Which of the following substitutions are clean? (i) $\alpha_{x:=y}$, (ii) $\alpha_{x:=a}$, (iii) $\alpha_{x:=x}$, (iv) $\alpha_{x:=z}$, (v) $\alpha_{y:=z}$, (vi) $\alpha_{w:=y}$.

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 so, vacuously, the substitution $\alpha_{y:=z}$ is clean. (vi) Vacuously clean again: there are no occurrences of w in α at all. *End of solution*.

Exercise 9.4.2 illustrates several general facts, whose verifications are immediate from the definitions.

- 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 quantifiers, beginning with $\forall$. The first one for $\forall$ is a logical consequence, also known as a *first-level* rule, whilst the second is a rule allowing us to get one logical consequence from another, known as a *second-level* rule. They called $\forall^-$ (universal instantiation, UI) and $\forall^+$ (universal generalization, UG).

- $\forall^-$: $\forall x(\alpha) \vDash \alpha_{x:=t}$, provided the substitution $\alpha_{x:=t}$ is clean.
- $\forall^+$: Whenever $A \vDash \alpha$ then $A \vDash \forall x(\alpha)$, provided the variable x has no free occurrences in any formula $\alpha \in A$.

As one would expect, there are dual principles for $\exists$. They are dubbed $\exists^+$ (existential generalization, EG) and $\exists^-$ (existential instantiation, EI).

- $\exists^+ : \alpha_{x:=t} \vDash \exists x(\alpha)$, provided the substitution $\alpha_{x:=t}$ is clean.
- $\exists^-$: Whenever A, $\beta \vDash \gamma$ then A, $\exists x(\beta) \vDash \gamma$, provided the variable x has no free occurrences in any formula $\alpha \in A \cup \{\gamma\}$.

In the second-level rule $\forall^+$, the entire consequence $A \vDash \forall x(\alpha)$ serves as output of the rule; the input is the consequence $A \vDash \alpha$. 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 $\alpha_{x:=t}$ to $\exists x(\alpha)$, the substitution goes in the reverse direction, from the body α of $\exists x(\alpha)$ to $\alpha_{x:=t}$. 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. It is perhaps for this reason, as well as the apparent syntactic complexity of $\exists^-$ compared to that of $\forall^+$ that renders the rules for the existential quantifier more difficult to assimilate compared to their universal counterparts.

Alice Box: Why the minus sign in the acronym $\exists^-$?

Alice I'm rather 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 one. Why the minus sign in $\exists^-$? Surely we are *adding* the existential quantifier $\exists x$ to α.

Hatter To see the rationale for the name, one has to remember that $\exists^-$ is a second-level rule and think backwards. Imagine that you are given A, $\exists x(\beta)$ as premises and want to infer a conclusion γ. The rule tells us that to do that, *it suffices* to treat β as a premise in place of $\exists x(\beta)$ and head for the same goal γ (so long as the proviso is satisfied).

Alice So it is more subtle than what one is doing in the $\exists^+$ rule, where we are simply inferring $\exists x(\alpha)$ from $\alpha_{x:=t}$ (so long as the proviso of that rule is satisfied)?

Hatter Indeed. We are not *inferring* β from $\exists x(\beta)$—in general, that would be quite invalid. We are *moving procedurally* from considering $\exists x(\beta)$ as a premise to considering β in the same role, in the process of building a deduction of γ from the original premises A, $\exists x(\beta)$.

Alice I think I understand, but you seem to be describing a mental process more than a mathematical fact. Can we go deeper?

Hatter Indeed we can, and will do so in Chap. 10.

The following exercise is very boring, but you need to do it conscientiously in order to make sure that you have really understood what these two rules are saying.

Exercise 9.4.3

(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 that $\forall x \exists y(Rxyz) \vDash \exists y(Rzyz)$. Which of the following may be obtained from it by just 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 a single application of the second-level rule $\exists^{-}$? (i) Px, $\exists z \forall y(Ryz) \vDash Px \wedge \exists z(Rxz)$, (ii) Px, $\exists z \forall y(Ryz) \vDash Px \wedge \exists z(Rxz)$, (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) $\exists x(Px), \forall y(Ryz) \vDash Px \wedge \exists z(Rxz)$.

Be careful. In (a), (b) we are not asking whether the logical consequence actually holds, but whether it holds as an instance of the rule concerned. Similarly, in (c), (d) it is not a matter of whether one logical consequence holds given another, but whether it does so as an instance of the rule mentioned.

Solution

(a) (i), (ii) yes; (iii) no; (iv) yes; (v), (vi) and (vii) no; (viii) and (ix) yes; (x) no.
(b) (i) yes, (ii) no, (iii) yes, (iv) no, (v) yes.
(c) (i) no; (ii) yes; (iii), (iv) no; (v) yes.
(d) (i) yes, (ii) no, (iii) no, (iv) yes, (v) no.

9.4.4 Identity

So far, we have ignored the identity relation sign. Recall from Sect. 9.3.1 that this symbol is always given a highly constrained valuation. At the most lax, it is interpreted as a congruence relation over elements of the domain; in the standard semantics that is 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 that it is an equivalence relation: reflexive, symmetric, transitive (see Chap. 2). To be precise, each of the following formulae is logically true, in the sense defined in Sect. 9.4.1: $\forall x(x = x), \forall x \forall y[(x = y) \rightarrow (y = x)], \forall x \forall y \forall z[\{(x = y) \wedge (y = z) \rightarrow (x = z)\}]$.

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, here written $\alpha_{t::=t'}$ with a double colon to distinguish it from substitution, 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 $\alpha_{t::=t'}$ is said to be *clean* iff the occurrences of t' thereby introduced are also free in $\alpha_{t::=t'}$. Using these concepts, we can articulate the following *principle of replacement*:

α, $t = t' \vDash \alpha_{t::=t'}$, 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 $\exists y(Rxy)$, $x = z \vDash \exists y(Rzy)$, since the introduced variable z is free in $\exists y(Rzy)$ and the replacement is clean. On the other hand, if we take t' to be y and propose the inference $\exists y(Rxy)$, $x = y \vDash \exists y(Ryy)$, the introduced occurrence of y y is not free in $\exists y(Ryy)$, so the replacement is not clean and the step is not authorized.

The proof of the replacement principle can be carried out by a rather tedious induction on the complexity of the formula α; we omit it.

Exercise 9.4.4 (1)
Show that the principle of replacement authorizes $x = x$, $x = y \vDash y = x$.

Solution
Let $(x = x)_{x::=y}$ be the result of replacing the first occurrence of the variable x in the formula $x = x$ by the variable y, giving the formula $y = x$; since there are no quantifiers the replacement is clean. Thus by the replacement principle, $x = x$, $x = y \vDash y = x$. Replacement can also be used in a trickier way to yield the transitivity of identity.

9.5 End-of-Chapter Exercises

Exercise 9.5 (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.

(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.

Solution

(a) (1) Yes, we can use any variable, writing for example $\forall y(Cy \rightarrow Py)$. We are using $\rightarrow$ rather than $\wedge$ because we are not saying that everything in the domain of discourse (people) is *both* a composer *and* a poet; we are claiming that for any person in the domain, *if* he/she is a composer *then* he/she is a poet. We can write $\forall x(Cx \rightarrow Px)$ as $\neg\exists x(Cx \wedge \neg Px)$. The formulae $\forall x(Cx \rightarrow Px)$ and $\forall x(Px \rightarrow Cx)$ are said to be converses of each other (for obvious reasons, given what you know from propositional logic) and are independent in the sense that neither logically implies the other.

(a) (2) We are saying that someone in the domain is *both* a composer *and* a poet, whereas $\exists x(Cx \rightarrow Px)$ would say that there is somebody in the domain such that if he/she is a composer then he/she is a poet. The latter is automatically true if there is somebody in the domain who is not a composer! We can write $\exists x(Cx \wedge Px)$ as $\neg\forall x(Cx \rightarrow \neg Px)$. It is not logically implied by the first statement: when nobody in the domain is a composer, $\forall x(Cx \rightarrow Px)$ is automatically true while $\exists x(Cx \wedge Px)$ is false. $\exists x(Cx \wedge Px)$ is independent of $\exists x(Cx \wedge \neg Px)$ in the sense that neither logically implies the other; they are also jointly satisfiable.

(a) (3) We can write $\forall x(Px \rightarrow \neg Cx)$ as $\neg\exists x(Px \wedge Cx)$. The two statements are logically equivalent.

(a) (4) To simplify the expression, we let the domain of discourse be the set of all persons. Yes, the meaning does change: $\forall x\exists y(Lyx)$ says that everyone *is loved by* someone.

(a) (5) No, $\exists y\forall x(Lxy)$ is not equivalent to $\forall x\exists y(Lxy)$, but the former does logically imply the latter.

(a) (6) Yes, but it would be clumsy. Write Fy for 'y is equal to five' and symbolize as $\exists x(Px \wedge \forall y(Fy \rightarrow (x < y))$.

(a) (7) Yes, as $\forall x\exists y[(Mx \wedge Sx) \rightarrow (Wy \wedge Ay \wedge Byx)]$.

(a) (8) We can express $\neg\exists x(Oxf(x))$ as $\forall x\neg(Oxf(x))$; while this still contains a negation, it is no longer the principal connective. Writing Fyx for 'y is a father of x' we can write $\neg\exists x(Oxf(x))$ as $\neg\exists x\exists y(Fyx \wedge Oxy)$.

(a) (9) We can contrapose $\forall x\forall y[\neg(x = y) \rightarrow \neg(s(x) = s(y)]$ to $\forall x\forall y[(s(x) = s(y) \rightarrow (x = y)]$ and reverse the initial quantifiers to $\forall y\forall x$ without changing logical force.

(a) (10) There is no logical need for the second quantifier to take a different variable in $\{P0 \wedge \forall x(Px \rightarrow Ps(x))\} \rightarrow \forall x(Px)$, since the scope of the first quantifier has already ended before the second begins, but some may find it easier to read when a different variable is used.

Yes, we can move the second universal quantifier up the front without complications, getting the equivalent formula $\forall x[\{P0 \wedge \forall x(Px \rightarrow Ps(x))\} \rightarrow Px]$. But moving the other quantifier 'up the front' takes more work and yields a formula that is strange and indigestible, although equivalent to the original. A chain of equivalences follows, in which we bring first one quantifier to the front and then the

other. If we do this in reverse order, we can also get another formula in prenex normal form with the same body but with the two initial quantifiers reversed, from $\exists x \forall y$ to $\forall y \exists x$. That is a rather exceptional feature of the example; in general, one cannot reverse the order of a universal with an existential quantifier.

$$\{P0 \wedge \forall x(Px \rightarrow Ps(x))\} \rightarrow \forall x(Px) \dashv\models \forall x\{P0 \wedge (Px \rightarrow Ps(x))\} \rightarrow \forall x(Px)$$
$$\dashv\models \neg\forall x\{P0 \wedge (Px \rightarrow Ps(x))\} \vee \forall x(Px) \dashv\models \exists x\neg\{P0 \wedge (Px \rightarrow Ps(x))\} \vee \forall x(Px)$$
$$\dashv\models \exists x[\neg\{P0 \wedge (Px \rightarrow Ps(x))\} \vee \forall x(Px)] \dashv\models \exists x[\{P0 \wedge (Px \rightarrow Ps(x))\} \rightarrow \forall x(Px)]$$
$$\dashv\models \exists x[\{P0 \wedge (Px \rightarrow Ps(x))\} \rightarrow \forall y(Py)] \dashv\models \exists x\forall y[\{P0 \wedge (Px \rightarrow Ps(x))\} \rightarrow Py].$$

Exercise 9.5 (2) Valuations

(a) Consider the relation between valuations of being x-variants, for a fixed variable x. Is it an equivalence relation? And if the variable x is not held fixed?
(b) Find a formula using the identity sign that is true under every valuation whose domain has less than three elements, but is false under every valuation 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 sign, with the same properties.

Solution

(a) For a fixed variable x, it is an equivalence relation, as is easily checked from the definition. But, if the variable is not held fixed, it is not in general an equivalence relation: a y-variant of an x-variant of v will in general differ from v in the value given to both those two variables.
(b) $\exists x \exists y \forall z[z = x \vee z = y]$.
(c) Consider any domain D of more than two elements. Select two individuals a_1, a_2 from it, and define a valuation that treats every element of D other than a_1 as indistinguishable from a_2. This is possible since for all function signs and predicates other than the identity sign, there are no restrictions on the functions resp. relations of D that they can be given as values, so long as their arity is respected. Thus, for example, if P is a one-place predicate, we can define $v(P)$ in such a way that for all $a \in D$ with $a \neq a_1$, $a \in v(P)$ iff $a_2 \in v(P)$. Then it is not difficult to verify that every formula without the identity sign receives the same truth-value under this valuation as it would in the corresponding valuation in the two-element domain $D' = \{a_1, a_2\}$.

Exercise 9.5 (3) Semantic analysis

(a) Justify the general principle of vacuous quantification (Sect. 9.2.3) semantically, first using the substitutional reading of the quantifiers, and then using the x-variant reading.
(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, specify a valuation 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 simple formulae α, β of quantificational logic without the identity predicate such that (i) $\forall x(\alpha)$ is a contradiction but α is not, (ii) $\exists x(\beta)$ is logically true but β is not.

(d) Recall from the text that a set A of formulae is said to be *consistent* iff there is some valuation 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.

Solution

(a) The general principle of vacuous quantification says: 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. We consider the universal quantifier; the treatment of the existential is similar. Suppose there are no free occurrences of x in φ. (i) Under the substitutional reading, $v(\forall x(\varphi)) = 1$ iff $v(\varphi_{x:=a}) = 1$ for every constant symbol a of the language (suitable expanded, if needed). But since there are no free occurrences of x in φ, the substitutions are all empty and each such formula $\varphi_{x:=a}$ is just φ. Thus $v(\forall x(\varphi)) = 1$ iff $v(\varphi) = 1$. (ii) Under the x-variant reading, $v(\forall x(\varphi)) = 1$ iff $v_{x:=a}(\varphi) = 1$ for every x-variant $v_{x:=a}$ of v. But it is not difficult to show by a structural induction that, since x does not occur free in φ, each $v_{x:=a}(\varphi) = v(\varphi)$. Thus, $v(\forall x(\varphi)) = 1$ iff $v(\varphi) = 1$.

(b) Solution outline: (i) Yes, but its converse is not. (ii), (iii) No, but its converse is. (iv) Yes, but its converse is not. We verify for (ii). As a witness to LHS $\nvDash$ RHS, Put $D = \{1\}$, $v(P) = \varnothing = v(Q)$. You may find it easier to process this mentally if you translate from $\rightarrow$ to $\vee$. To show RHS $\vDash$ LHS, suppose $v(\text{LHS}) = 0$. Using say the substitutional reading, we have $v(Pa \rightarrow Qa) = 0$ for every constant a so, for every constant a, both $v(Pa) = 1$ and $v(Qa) = 0$. Since $v(Qa) = 0$ for every constant a, we also have $v(\exists x(Qx)) = 0$. Choosing the constant a with $v(a) = 1$, we also have $v(Pa) = 1$ so $v(\exists x(Px)) = 1$. Putting these together, $v(\text{RHS}) = 0$ as desired.

(c) (i) Put $\alpha = Px \wedge \neg\forall x(Px)$, (ii) $\beta = Px \vee \neg\exists x(Px)$.

(d) (i) Vacuously, every valuation makes all elements of $\varnothing$ true.

(d) (ii) $\{\alpha\}$ is consistent iff there is some valuation v with $v(\alpha) = 1$, which holds iff α is not a contradiction.

(d) (iii) Suppose that A is consistent. Then there is a valuation v with $v(\alpha) = 1$ for all $\alpha \in A$. But $v(p \wedge \neg p) = 0$, so $A \nvDash p \wedge \neg p$. Conversely, suppose that A is inconsistent. Then there is no valuation v with $v(\alpha) = 1$ for all $\alpha \in A$ so, *a*

fortiori, there is no valuation v with both $v(p \wedge \neg p) = 0$ and $v(\alpha) = 1$ for all $\alpha \in A$.

(d) (iv) Let A be a set of formulae and $B \subseteq A$. If A is consistent then there is a valuation v with $v(\alpha) = 1$ for all $\alpha \in A$, in which case $v(\alpha) = 1$ for all $\alpha \in B$, so B is consistent.

(d) (v) Let A be a finite set of formulae and let $\wedge A$ be the conjunction of all its elements. Now, A is consistent iff there is a valuation v with $v(\alpha) = 1$ for all $\alpha \in A$, which holds iff $v(\wedge A) = 1$. The reason why we needed the qualification 'finite' is that the language of first-order logic does not contain infinitely long conjunctions.

Exercise 9.5 (4) Fundamental rules

(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 the rule $\exists^-$ 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 $\alpha_1, \ldots, \alpha_{n-1}$.

Solution

(a) Recall that the rule $\forall^+$ says: whenever $A \models \alpha$ then $A \models \forall x(\alpha)$, provided the variable x has no free occurrences in any formula $\alpha \in A$. Example of violation if its proviso is not respected: put $\alpha = Px$, $A = \{\alpha\}$. Clearly $\{Px\} \models Px$ but $\{Px\} \not\models \forall x(Px)$ as is easily witnessed in the two-element domain $\{1,2\}$ with $v(x) = 1$, $v(P) = \{1\}$.

(b) Recall that the rule $\exists^-$ says: whenever $A, \beta \models \gamma$ then $A, \exists x(\beta) \models \gamma$, provided the variable x has no free occurrences in any formula $\alpha \in A \cup \{\gamma\}$.

Example of violation if its proviso is not respected: put $A = \varnothing$, $\alpha = \beta = Px$. Clearly $\{Px\} \models Px$ but $\{\exists x(Px)\} \not\models Px$ as is easily witnessed in the two-element domain $\{1,2\}$ with $v(x) = 1$, $v(P) = \{0\}$.

It is no accident that the solutions to (a) and (b) are so similar; the rules $\forall^+$, $\exists^-$ are themselves essentially the same when we take for granted the principles of quantifier interchange and suitable tautologies. The same can be said for $\forall^-$, $\exists^+$.

9.6 Selected Reading

Essentially the same books as for propositional logic with later chapters; for example the following.

For computer science students: James Hein *Discrete Structures, Logic and Computability*, Jones and Bartlett 2002 (second edition), chapter 7; Michael Huth and Mark Ryan *Logic in Computer Science*, Cambridge University Press 2000,

chapter 4; on a more advanced level Mordechai Ben-Ami *Mathematical Logic for Computer Science*, Springer 2001 (second edition), chapters 5–7. For students of philosophy and linguistics: L.T.F. Gamut *Logic, Language, and Meaning Volume I: Introduction to Logic*, University of Chicago Press 1991, chapters 3–4. For the general reader: Wilfrid Hodges *Logic*. Penguin 1977, sections 34–41.

Just Supposing: Proof and Consequence

10

Chapter Outline

In the last two chapters we learned some of the basics of propositional and quantificational logic and, in particular, the relations of logical implication that they provide. In this chapter, we look at how simple logical 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 the closely related proof by cases), proof by contradiction, as well as arguments to and from an arbitrary instance. Their analysis leads us to distinguish *first-level* from *second-level* and *split-level* rules, articulate their recursive structures and explain the informal procedure, in mathematical practice, 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.

D. Makinson, *Sets, Logic and Maths for Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-42218-9_10

10.1.1 My First Derivation Tree

Suppose that we are given three propositions as assumptions (also called, more traditionally, *premises*) and that they are of the form $(p \vee \neg q) \rightarrow (s \rightarrow r)$, $\neg s \rightarrow q$, and $\neg q$. We want to infer r. How can we do it?

This is, in effect, a matter of showing that $\{(p \vee \neg q) \rightarrow (s \rightarrow r), \neg s \rightarrow q, \neg q\} \models r$ so, in principle, we could use a truth-table with $2^4 = 16$ rows, or a semantic decomposition tree as in Chap. 8. But there is another method that parallels much more closely how we would tackle 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 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 ($\vee^+$), 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, which takes up a lot of page-space but helps bring out clearly the structure of the derivation, is as a *labelled derivation tree* as in Fig. 10.1.

Such a tree is conventionally written with leaves at the top, since it 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 (resp. 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
What happens to the tree in Fig. 10.1 if we merge the two nodes labelled $\neg q$?

Solution
It would no longer be a tree, only a graph. Nothing intrinsically wrong with that, but it turns out to be more convenient to represent derivations as trees rather than as graphs. *End of solution*.

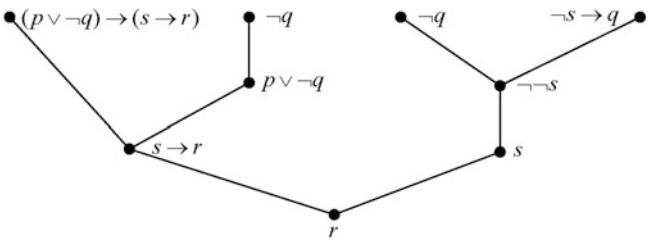

Fig. 10.1 Example of a labelled derivation tree

Table 10.1 Example of a derivation displayed 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

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 gives an annotated sequence of formulae corresponding to the tree in Fig. 10.1. The sequence itself is in the second column, numbered in the first column and with 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, and is the presentation that one finds most commonly in elementary logic texts. 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*.

Although tree presentations reveal structure more readily to the eye, linearization has practical advantages. It takes up less space on the handwritten page, and is easier to write down using current word-processing software. 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 the repeated formula $\neg q$ is a premise appearing only twice, each time as a leaf in the tree. But it is easy to imagine a long derivation in which a proposition is called upon, perhaps several times, near the conclusion. 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.

It should be intuitively clear from the example considered in Fig. 10.1 and Table 10.1, that we can algorithmically 'squeeze' any derivation tree into an annotated derivation sequence without repetitions; conversely, we can 'unfold' any linearly presented derivation into tree form. For the squeezing operation to be

well-defined, one needs a convention to choose the order in which nodes of the tree are listed that is consistent with each node being preceded by its parent(s). For the unfolding operation to be injective the linear presentation needs to have enough labelling to indicate which predecessor(s) each item was obtained from.

10.1.2 The Logic behind Chaining

That 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 (that is, corresponds to a logical implication in the sense of Chaps. 8 or 9) then their combination is valid? In other words, how can we be sure that the conclusion of the derivation (root of the tree) is implied by the premises (leaves) taken together, no matter how distant they are from each other? In the example, how can we be sure that the propositions (1) through (3) given as premises really do tautologically imply the proposition (8)?

The first answer that comes to mind is that we are just applying the transitivity of $\models$. 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 numbered taken together. We consider them in turn, using the numbers in the leftmost column of Table 10.1 as names of the corresponding formulae in the next column. 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 formula is a consequence of any set of formulae of which it is an element. Known as *strong reflexivity*, it says:

$$A \models \alpha \text{ whenever } \alpha \in A.$$

Formulae 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}. 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 formulae to A. In other words, increasing the stock of premises never leads us to drop a conclusion:

$$B \models \beta \text{ whenever both } A \models \beta \text{ and } A \subseteq B.$$

In our example $A = \{3\}$, $\beta = 4$ and $B = \{1,2,3\}$. What about 5? In the annotations, we noted that it is obtained from {1,4}, so monotony tells us that it implied by {1,2,3,4}. But 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 $\gamma = 5$.

$$A \models \gamma \text{ whenever both } A \models \beta \text{ for all propositions } \beta \in B \text{ and } A \cup B \models \gamma.$$

All three of these principles—strong reflexivity, monotony and cumulative transitivity—are needed to ensure that when each individual step in a derivation is justified by an inference relation $\models$ (in this example, that of tautological implication) then so too is the combination of those steps.

The surprising thing is that the three are not only necessary for the job; together they suffice to do it. Inference relations satisfying them are thus very important and are called *consequence relations*. In the next section we study them closely, showing that they do indeed suffice to guarantee the validity of chaining.

Exercise 10.1.2

(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) (i) Construct an elementary derivation for $\{\neg q \vee p, \neg r \rightarrow \neg p, s, s \rightarrow \neg r, t \rightarrow p\} \models \neg t \wedge \neg q$ as an annotated derivation sequence. (ii) What happens if you transform this sequence into a tree?

Solution

(a) To check that $\{1,2,3\} \models 6$ we first note that $\{2,3\} \models 6$ (by modus tollens, as recorded in the annotation) and then apply monotony. For $\{1,2,3\} \models 7$, we first note that $6 \models 7$ (by double negation elimination, as annotated), so $\{1,2,3,6\} \models 7$ by monotony. But we already know that $\{1,2,3\} \models 6$, so we may apply cumulative transitivity to get the desired $\{1,2,3\} \models 7$. For the last check, that $\{1,2,3\} \models 8$, note that $\{7,5\} \models 8$ (by modus ponens as indicated in the table) so by monotony $\{1,2,3,7,5\} \models 7$. But we already know that $\{1,2,3\} \models 5$ and $\{1,2,3\} \models 7$, so we may once more apply cumulative transitivity to get $\{1,2,3\} \models 7$ as desired. From these verifications, you may get the impression that when chaining we are implicitly using cumulative transitivity over, and over, again. That impression is correct.

(b) (i) See Table 10.2. (ii) In the derivation, we appeal to item 7 twice: first for a modus ponens and again for a disjunctive syllogism. Transforming this into a tree (rather than just into a graph) requires that we repeat the part of the tree that leads up to 7; that will give us 5 + 3 = 8 leaves and a total of 15 nodes (try it). The tree structure is clearer, but the repetition is tedious.

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 also conveniently be expressed as an operation.

Table 10.2 Answer to Exercise 10.1.2 (b)

Number	Formula	Obtained from	By rule
1	$\neg q \vee p$		Premise
2	$\neg r \rightarrow \neg p$		Premise
3	s		Premise
4	$s \rightarrow \neg r$		Premise
5	$t \rightarrow p$		Premise
6	$\neg r$	3, 4	Modus ponens
7	$\neg p$	6, 2	Modus ponens
8	$\neg t$	7, 5	Modus tollens
9	$\neg q$	7,1	Disjunctive syllogism
10	$\neg t \wedge \neg q$	8, 9	Conjunction

10.2.1 The Tarski Conditions

Let $\vdash$ 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. We use the symbol $\vdash$, similar to $\models$ for logical implication, because it will be treated as an abstraction on that relation. It is called a *consequence relation* iff it satisfies the three conditions that we isolated in the previous section.

Strong reflexivity: $A \vdash \alpha$ whenever $\alpha \in A$
Monotony: Whenever $A \vdash \beta$ and $A \subseteq B$ then $B \vdash \beta$
Cumulative transitivity: Whenever $A \vdash \beta$ for all $\beta \in B$ and $A \cup B \vdash \gamma$ then $A \vdash \gamma$

A few words on terminology 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. 'Monotony' is sometimes written 'monotonicity'. Cumulative transitivity is often called 'cut' (but be careful, there are a number of variants that are also given that name).

The three conditions are often called the *Tarski conditions* in honour of Alfred Tarski, who was perhaps the first to realize their joint importance. Algebraically-minded logicians often call consequence relations *logical closure relations*, as they are just a particular case of the closure relations/operations that play a very important role in abstract algebra. The notion of the closure of a set under a relation that we studied in Chap. 2 Sect. 2.7 is another particular case of the same general concept.

Given a relation $\vdash$ between sets of formulae and individual formulae, we reduce clutter by writing $\alpha_1,\ldots, \alpha_n \vdash \beta$ to abbreviate $\{\alpha_1,\ldots, \alpha_n\} \vdash \beta$ and in particular $\alpha \vdash \beta$ for $\{\alpha\} \vdash \beta$. To mark the contrast with strong reflexivity and cumulative transitivity, the conditions of reflexivity ($\alpha \vdash \alpha$) and transitivity (whenever $\alpha \vdash \beta$ and $\beta \vdash \gamma$ then $\alpha \vdash \gamma$) may sometimes be called *plain* reflexivity and transitivity.

Exercise 10.2.1 (1)

(a) Show that both tautological implication in propositional logic and logical implication in quantificational logic are indeed consequence relations.
(b) Show that (i) strong reflexivity immediately implies plain reflexivity, (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 strong or even plain reflexivity.

Solution

(a) We can use a single argument to cover both propositional and first-order logic, by considering valuations without bothering to specify which of the two languages we are considering. For strong reflexivity: Suppose $\alpha \in A$. Then any valuation that makes all formulae in A true, makes α true, so $A \models \alpha$. For monotony: Suppose $A \subseteq B$ and $A \models \beta$. Let v be any valuation that makes all formulae in B true. Then it makes all formulae in A true so, since $A \models \beta$, it makes β true. Cumulative transitivity is a tad less immediate. Suppose $A \models \beta$ for all $\beta \in B$ and $A \cup B \models \gamma$; we need to show $A \models \gamma$. 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 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, $v(\gamma) = 1$.
(b) (i) Assume strong reflexivity and consider any formula α. Put $A = \{\alpha\}$. Since $\alpha \in A$, strong reflexivity tells us that $A \vdash \alpha$, that is, $\alpha \vdash \alpha$. (ii) Assume plain reflexivity and monotony. Consider any formula α and set A with $\alpha \in A$. By plain reflexivity, $\alpha \vdash \alpha$, that is, $\{\alpha\} \vdash \alpha$ so, since $\{\alpha\} \subseteq A$, monotony tells us that $A \vdash \alpha$.
(c) The empty relation is the simplest. For an example that is less of a limiting case, put A to have the relation to α iff α is a tautology in A. *End of solution*.

While, as verified in Exercise 10.2.1 (1)(a), classical logical implication satisfies all three of the Tarski conditions, it should also be appreciated that neither monotony nor cumulative transitivity is appropriate, in unrestricted form, for relations of *uncertain inference* (or, as one says in another terminology, *non-deductive* as opposed to *deductive* inference). Monotony fails under both probabilistic and qualitative accounts of uncertain reasoning and, while cumulative transitivity is acceptable under some qualitative accounts, it too fails probabilistically. But that goes beyond our present concerns; the guide to further reading at the end of the chapter gives a pointer.

Exercise 10.2.1 (2)

(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 for uncertain reasoning.
(b) Give an intuitive explanation why plain transitivity is not appropriate for uncertain inference.

Solution

(a) Consider the following scenario. The testimony of a number of eyewitnesses, expressed in the premise-set A, gives reasonable grounds for believing that the suspect committed the crime γ. But traces of DNA left by the assassin on the murder weapon are quite different from that of the suspect; call this evidence β. Then A supports γ but $A \cup \{\beta\}$ may not support γ; in some cases, it may even support another conclusion that is incompatible with γ.
(b) 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 is respected by each of the separate steps. Thus, the strength of the chain is in general *even less than* that 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 $\triangleright$ be any relation between sets of formulae and individual formulae of a formal language. To guide intuition, think of $\triangleright$ 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 of interest. We do not assume that it satisfies any of the three Tarski conditions. 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 $\triangleright$ iff (1) $\alpha_n = \alpha$ and (2) for every $i \leq n$, either $\alpha_i \in A$ or $B \triangleright \alpha_i$ 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. For intuition, continue to think of these trees as with their leaves at the top and the root at the bottom, like that of Fig. 10.1. We say that a finite tree T whose nodes are labelled by formulae is an *elementary derivation* of α from A using the relation $\triangleright$ 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 $B \triangleright \beta$.

The *soundness theorem for chaining* tells us the following. Consider any relations $\triangleright$, $\vdash$ between sets of formulae and individual formulae of a formal language, such that $\triangleright \subseteq \vdash$ and $\vdash$ satisfies the three Tarski conditions. If there is an elementary derivation $\sigma = \alpha_1, \ldots, \alpha_n$ of α from A using $\triangleright$ then $A \vdash \alpha$.

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 Chap. 4 Sect. 4.5.2; in particular, it should be recalled that in this form of induction, we do not need a base (although one can put one in if one wishes), only an induction step.

Suppose that $\triangleright \subseteq \vdash$ and $\vdash$ satisfies the three Tarski conditions, and that there is an elementary derivation $\sigma = \alpha_1, \ldots, \alpha_n$ of α from A using $\triangleright$. To show that $A \vdash \alpha$, it suffices to show by induction on i that $A \vdash \alpha_i$ for all $i \leq n$. Let $k \leq n$ and suppose (induction hypothesis) that $A \vdash \alpha_i$ for all $i < k$; we need to show that $A \vdash \alpha_k$. Now, by the definition of an elementary derivation either $\alpha_k \in A$, or $B \triangleright \alpha_k$ for some $B \subseteq \{\alpha_1,\ldots,\alpha_{k-1}\}$. In the former case we have $A \vdash \alpha_k$ by strong reflexivity of $\vdash$. In the latter case, we have $B \vdash \alpha_k$ by the supposition that $\triangleright \subseteq \vdash$ so that (1) $A \cup B \vdash \alpha_k$ by monotony of $\vdash$. But by the induction hypothesis we have (2) $A \vdash \alpha_i$ for all $i \in B$. Applying cumulative transitivity to (2) and (1) gives us $A \vdash \alpha_k$ as desired.

Alice Box: What did we use in this proof?

Alice It's nice to see how that little proof appeals explicitly to each of the three Tarski conditions. But it does not seem to use the full power of monotony or cumulative transitivity.

Hatter How is that?

Alice In the applications of those two conditions we needed only the case that B is finite, even when A is infinite. 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, the proof goes through unchanged with versions of monotony and cumulative transitivity in which B is finite. For many inference relations $\vdash$, including classical consequence $\models$ in both propositional and first-order logic, the full and restricted versions are in fact equivalent. This is because those relations are *compact* in the sense that whenever $A \models \alpha$ then there is some finite subset $A' \subseteq A$ with $A' \models \alpha$. But for relations $\vdash$ that are not compact (and they do exist, for what is known as second-order classical logic as well some non-classical logics, all beyond our remit) such restricted versions of monotony and cumulative transitivity can be weaker than their full versions.

Exercise 10.2.2

(a) Explain why the converse of compactness holds trivially for every consequence relation.
(b) Show that if a relation $\vdash$ is both compact and monotonic, then whenever A is infinite and $A \vdash \beta$ there are infinitely many finite subsets $X \subseteq A$ with $X \vdash \beta$.

Solution

(a) The converse of compactness says that whenever there is some finite subset A' $\vdash A$ with $A' \vdash \alpha$, then $A \vdash \alpha$. This holds immediately by monotony.
(b) Suppose that $\vdash$ is both compact and monotonic, A is infinite and $A \vdash \beta$. By compactness, there is some finite subset $A' \vdash A$ with $A' \vdash \alpha$. Then there are infinitely many finite subsets A'' with $A' \subseteq A'' \subseteq A$. By monotony, we have A'' $\vdash \beta$ for all of them.

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 helpful, since its behaviour can usually be stated rather more succinctly in operational terms. Admittedly, the operational formulation does take a bit of time to get used to but, once you have it under your belt, you will find it very convenient.

Let A be any set of propositions expressed in a formal language L like that of propositional or first-order logic and let $\vdash$ be any inference relation for that language. We define C: $\mathcal{P}(L) \to \mathcal{P}(L)$ quite simply by putting $C(A)$, for any $A \subseteq L$, to be the result of gathering together all the propositions β such that $A \vdash \beta$. Briefly, $C(A) = \{\beta: A \vdash \beta\}$.

When $\vdash$ is a consequence relation, that is, satisfies the three Tarski conditions, then the operation C has the following three properties, also called *Tarski conditions*.

Inclusion: $A \subseteq C(A)$
Idempotence: $C(A) = C(C(A))$
Monotony: $C(A) \subseteq C(B)$ whenever $A \subseteq B$

These correspond to the properties for consequence as a relation and are easily derived from them (next exercise). Operations C: $\mathcal{P}(L) \to \mathcal{P}(L)$ satisfying these three conditions are known to logicians as *consequence operations* and are instances of the general mathematical notion of a *closure* operation; the letter C is used to recall those words. Be warned, however, that in topology closure operations are sometimes required to satisfy the equality $C(A) \cup C(B) = C(A \cup B)$ which does not hold for classical consequence (next exercise) and rarely holds for interesting non-classical consequence operations.

We have distinguished the notations for an arbitrary consequence relation $\vdash$ and the specific relation $\models$ of classical consequence. Similarly, while we write C for an arbitrary consequence relation, we will let Cn stand for the specific operation of classical consequence. By the definition, $Cn(A) = \{\beta: A \models \beta\}$.

Exercise 10.2.3

(a) Give a very simple example to show that (i) classical consequence does not satisfy the equality $Cn(A) \cup Cn(B) = Cn(A \cup B)$. Show that nevertheless, (ii) the LHS $\subseteq$ RHS half holds for every consequence relation (including, therefore, classical consequence).
(b) Show that the three Tarski conditions for logical consequence as an operation follow from those for consequence relations, via the definition $C(A) = \{\beta: A \vdash \beta\}$ given in the text.
(c) What would be the natural way of defining consequence as a relation back from consequence as an operation?
(d) 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 (c).
(e) (i) Express the equality $C(\{\alpha \vee \beta\}) = C\{\alpha\} \cap C\{\beta\}$ in terms of consequence relations and (ii) show that it holds for classical consequence $\models$.

Solution

(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 $Cn(A) \cup Cn(B) \subseteq Cn(A \cup B)$.
(b) For $A \subseteq C(A)$: Let $\alpha \in A$; we need to show that $\alpha \in C(A)$, in other words by the definition of C, that $A \vdash \alpha$. This is immediate by the Tarski condition of strong reflexivity for $\vdash$.

For $C(A) \subseteq C(A \cup B)$: Let $\alpha \in C(A)$, in other words, $A \vdash \alpha$. We need to show that $\alpha \in C(A \cup B)$, in other words that $A \cup B \vdash \alpha$. Since $A \subseteq A \cup B$, this is given immediately by the Tarski condition of monotony for $\vdash$.

For the identity $C(A) = C(C(A))$ we verify its two inclusions separately. Theinclusion $C(A) \subseteq C(C(A))$ is an instance of the inclusion $A \subseteq C(A)$ (substituting $C(A)$ for A), which we have already checked. For theconverse inclusion, suppose $\alpha \in C(C(A))$. We want to show that $\alpha \in C(A)$, i.e. that $A \vdash \alpha$. On the one hand, we have $C(A) = \{\beta: A \vdash \beta\}$ by the definition of C, so (i) $A \vdash \beta$ for all $\beta \in C(A)$. On the other hand, since $\alpha \in C(C(A))$ we also have by the definition of C that $C(A) \vdash \alpha$ so, by monotony for $\vdash$, (ii) $A \cup C(A) \vdash \alpha$. Applying cumulative transitivity for $\vdash$ to (i) and (ii) gives $A \vdash \alpha$ as desired. Remarks: Although it is short, this verification of $C(C(A)) \subseteq C(A)$ tends to give students headaches. Note that we need to appeal to both cumulative transitivity and monotony for $\vdash$, whereas for $A \subseteq C(A)$ we needed only strong reflexivity of $\vdash$ and, for $C(A) \subseteq C(A \cup B)$, just monotony of $\vdash$.

(c) Put $A \vdash \beta$ iff $\beta \in C(A)$.

(d) For strong reflexivity of $\vdash$, suppose $\alpha \in A$. Now, $A \subseteq C(A)$ by inclusion for C, so $\alpha \in C(A)$, so $A \vdash \alpha$ by the definition of $\vdash$.

For monotony of $\vdash$, suppose $A \vdash \alpha$. Then $\alpha \in C(A)$ by the definition of $\vdash$ so by monotony for C we have $\alpha \in C(A \cup B)$, that is $A \cup B \vdash \alpha$.

For cumulative transitivity of $\vdash$, suppose (i) $A \vdash \beta$ for all $\beta \in B$ and (ii) $A \cup B \vdash \gamma$; we need to show that $A \vdash \gamma$. By the definition of $\vdash$, (i) tells us that $B \subseteq C(A)$; by inclusion for C also $A \subseteq C(A)$, so $A \cup B \subseteq C(A)$, so $C(A) = (A \cup B) \cup C(A)$. But by (ii), $A \cup B \vdash \gamma$, that is, $\gamma \in C(A \cup B) \subseteq C((A \cup B) \cup C(A)) = C(C(A)) = C(A)$ using monotony for the inclusion and idempotence of C for the last equality, so finally $A \vdash \gamma$ as desired.

Students usually find the last verification rather tricky. It is important to keep clearly in mind what you are supposing and what you are trying to get. Note that we needed all three Tarski conditions on C to establish cumulative transitivity for $\vdash$.

(e) (i) $\alpha \vee \beta \vdash \gamma$ iff $\alpha \vdash \gamma$ and $\beta \vdash \gamma$. (ii) For classical consequence, this says $\alpha \vee \beta \models \gamma$ iff $\alpha \models \gamma$ and $\beta \models \gamma$ and we verify the contrapositive $\alpha \vee \beta \not\models \gamma$ iff $\alpha \not\models \gamma$ or $\beta \not\models \gamma$. Suppose LHS. Then for some valuation v, $v(\alpha \vee \beta) = 1$ while $v(\gamma) = 0$. Hence either $v(\alpha) = 1$ while $v(\gamma) = 0$, or $v(\beta) = 1$ while $v(\gamma) = 0$, so RHS as needed. For the converse, we run the same argument in reverse. Suppose RHS. Then for some valuation v, either $v(\alpha) = 1$ while $v(\gamma) = 0$, or $v(\beta) = 1$ while $v(\gamma) = 0$. Hence $v(\alpha \vee \beta) = 1$ while $v(\gamma) = 0$ and so RHS. If you don't like all this negativity, you can equally well argue positively, supposing LHS of the original to get its RHS and then conversely.

10.3 A Higher-Level Proof Strategy: Conditional Proof

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 finally proof to/from an arbitrary instance. Linguistically, 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 refer to as *first-level rules*.

We begin with the strategy known as *conditional proof*. We already said a few words about it in a logic box in Chap. 1, Sect. 1.2.2, but now we analyse the logical machinery behind it. We do so slowly and carefully, as conditional proof serves as an exemplar; after examining it, we can describe the other higher-level strategies rather more briskly.

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. 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 having used it to climb. This kind of reasoning is known as *conditional proof*, acronym CP.

In everyday mathematics it is relatively rare to prove a bare conditional $\beta \rightarrow \gamma$; usually it has, implicitly if not explicitly, one or more initial universal quantifiers. 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, taking the set of all positive integers as domain, may be written $\forall n(En \rightarrow En^2)$. 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, going on to show that n^2 is even. However, in our analysis we want to separate out the moves made for different logical connectives. Manipulations for the quantifiers will be discussed shortly; for now, we consider the situation in which conditional proof is the only method used beyond familiar chaining.

Consider the following example: 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) from which we wish to prove $(p \wedge s) \rightarrow (q \rightarrow u)$ (the $\beta \rightarrow \gamma$ of 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. But it is straightforward to do that with an elementary derivation using some of the tautological implications from Chap. 8.

Exercise 10.3.1
Write out the above proof in full, including the steps described as straightforward.

Solution
Given premises $(p \wedge q) \rightarrow r$, $(t \rightarrow \neg s)$, $(r \rightarrow (t \vee u))$, suppose $p \wedge s$ in order to get $q \rightarrow u$. For that, suppose q in order to get u. The rest is chaining elementary inferences, as follows. From $p \wedge s$ we have p (by simplification) and from that with q we have $p \wedge q$ so, by modus ponens with the first premise, r. Another modus ponens with the third premise gives $t \vee u$. But also, from $p \wedge s$ we have s (again by simplification), so (by double negation and modus tollens using the second premise) we get $\neg t$. Combining that with $t \vee u$ finally produces u (by disjunctive syllogism) as desired. *End of solution*.

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 since it is very often employed as, for example, in the argument used in the sample solution to Exercise 10.2.3 (e).

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. That's no problem for a computer but, manually, it is neither elegant nor convenient.
Alice	Well, by an elementary derivation, without suppositions?
Hatter	Try it. It will be tricky to find and, if you succeed, the elementary deviation will be much 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 $\vdash$. Quite simply, it is that $A \vdash \beta \rightarrow \gamma$ whenever $A \cup \{\beta\} \vdash \gamma$. Writing a slash for the central transition, it may be displayed as:

$$A \cup \{\beta\} \vdash \gamma \,/\, A \vdash \beta \rightarrow \gamma.$$

The conventions for understanding this parenthesis-free display are that connectives are most cohesive, then set-theoretic union, then the turnstile, finally the slash. Full bracketing would write $[(A \cup \{\beta\}) \vdash \gamma] \,/\, [A \vdash (\beta \rightarrow \gamma)]$ but that is rather distracting so we omit the parentheses, hopefully without any risk of confusion. Often one abbreviates $A \cup \{\beta\}$ as A, β to write the rule even more economically:

$$A, \beta \vdash \gamma \,/\, A \vdash \beta \rightarrow \gamma.$$

When visual intuition is more important than saving page-space, one often replaces the slash by an extended horizontal bar, writing:

$$\frac{A, \beta \vdash \gamma}{A \vdash \beta \rightarrow \gamma.}$$

The output $A \vdash \beta \rightarrow \gamma$ of the rule is called its *principal* (or *main*) *inference*, while $A, \beta \vdash \gamma$ is its *subordinate inference* (or *sub-proof*). The proposition β appears as the antecedent of the conclusion of the principal inference and functions as the *supposition* of the subordinate inference. Like the informal procedure that it underlies, the rule is known as *conditional proof* with acronym CP. It is often called $\rightarrow^{+}$ because an arrow is introduced as main connective of the conclusion of the principal inference.

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 other higher-level rules to 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* $A, \beta \vdash \gamma$ to the validity of another *inference* $A \vdash \beta \rightarrow \gamma$. 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 and are therefore called *first-level* rules.
- It is *correct* for the inference relation $\models$ of classical (tautological or first-order) logical implication. That context is the only one that we will be considering specifically in this chapter, but we will continue to formulate conditional proof in terms of an arbitrary consequence relation, using the sign $\vdash$, since it is also correct for many non-classical inference relations for logics.
- When carrying out a deduction, conditional proof makes life easier because it gives us more premises to play around with. While the principal inference $A \vdash \beta \rightarrow \gamma$ has n premises, the subordinate inference $A, \beta \vdash \gamma$ has $n + 1$ of them. 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 $\beta \rightarrow \gamma$ of the principal argument. In this way, it reduces the question of obtaining a conditional conclusion to that of getting a logically simpler one.

In all of the second-level rules that will be considering, indeed in all of those routinely employed in mathematics, the premises of the subordinate inference form a *superset* of the premises of the principal inference. This property permits great simplifications of presentation for derivations, as we will see shortly; we say that such rules are *incremental*. More specifically, for most of the rules the premises of the subordinate inference form a proper superset with just one additional element, functioning as the supposition of the sub-proof. In one of the second-level rules, for the universal quantifier, the premise sets of the principal and subordinate inferences are identical, with no supposition made.

Now, conditional proof may be written in another way, as what we will call a *split-level* rule. Whereas the second-level formulation allows us to pass from an inference $A,\ \beta \vdash \gamma$ to the corresponding inference $A \vdash \beta \rightarrow \gamma$, the *split-level* formulation authorizes passage from the propositions in A taken together with the inference $A,\ \beta \vdash \gamma$, to the proposition $\beta \rightarrow \gamma$. The split-level rule thus passes from various propositions plus an inference, to a proposition. It may be written with a slash:

$$A;\ A,\ \beta \vdash \gamma\ /\ \beta \rightarrow \gamma$$

or with a horizontal bar:

$$\frac{A;\ A,\ \beta \vdash \gamma}{\beta \rightarrow \gamma.}$$

This too is incremental, in that the premise-set $A \cup \{\beta\}$ of the input inference is a superset (by just one element) of the input set A of propositions. Second-level formulations are used in what are called *sequent calculi* while split-level ones are employed in *natural deduction*. Both were developed in the 1930s: sequent calculi by Gerhard Gentzen (building on earlier hints of Paul Hertz), natural deduction independently by Gentzen and Stanisław Jaśkowski.

Exercise 10.3.2
Show that conditional proof, in both its second-level and split-level forms, is correct for classical consequence.

Solution

(a) This can be done in essentially the same fashion for the two kinds of formulations. For brevity, write $v(A) = 1$, where A is a set of formulae, as shorthand for $v(\alpha) = 1$ for all $\alpha \in A$.

For the second-level version, we argue contrapositively: Suppose that $A \nvDash \beta \rightarrow \gamma$; we want to show that $A,\ \beta \nvDash \gamma$. By the supposition, there is a valuation v with $v(A) = 1$ and $v(\beta \rightarrow \gamma) = 0$. Then $v(\beta) = 1$, $v(\gamma) = 0$, so $v(A \cup \{\beta\}) = 1$ while $v(\gamma) = 0$, so $A,\ \beta \nvDash \gamma$ as desired.

For the split-level version: Suppose that $A,\ \beta \vDash \gamma$ and v is a valuation with $v(A) = 1$; we need to show that $v(\beta \rightarrow \gamma) = 1$. If $v(\beta) = 0$ then $v(\beta \rightarrow \gamma) = 1$ as desired. On the other hand, if $v(\beta) = 1$ then $v(A \cup \{\beta\}) = 1$ so $v(\gamma) = 1$ and thus again $v(\beta \rightarrow \gamma) = 1$ as needed. *End of solution*.

Derivations using only second-level rules are extremely tedious to construct by hand because of the immense amount of repetition that they generate. On the other hand, they can be very helpful for studying the behaviour of a logical system, because they lend themselves to the systematic application of complex inductive arguments on well-orderings of derivations. Such investigations make up what is

known as *proof theory*, a subject that forms a world of its own with a vast literature; there is a pointer in the guide at the end of the chapter.

Derivations using split-level rules lend less suited for abstract studies about logic, but they have a great advantage for conducting the everyday business of inference. When they are incremental in the sense defined above, derivations employing them can be 'flattened' to mimic first-level derivations, reducing repetition radically and paralleling the informal style that mathematicians have refined over the centuries to communicate their proofs with minimum fuss. We describe the flattening operation in the next section.

10.3.3 Flattening Split-Level Proofs

It is not possible to combine first-level rules directly with second-level ones. The inputs and and output of the former are propositions while the inputs and output of the latter are inferences, so an input or output of one cannot be, respectively, an output or input of the other. But first-level rules do combine easily with split-level ones. Since the output of a split-level rules is a proposition, as are some of is inputs, the former can readily serve as input and the latter as output of first-level rules.

A proof using both split-level and first-level rules will thus be a finite tree or sequence with a mix of nodes. Some nodes will be labelled by propositions while others will be labelled by inferences known as sub-proofs. When all higher-level rules employed are incremental, it is possible to rewrite such a tree or sequence as one where all nodes are labelled by propositions-with-annotations, thus mimicking a first-level derivation but with a bit more book-keeping. This is the operation we call *flattening*.

Flattening is most easily described in terms of sequences. In a sequence serving as a derivation using both first-level and split-level rules, we replace each point in the sequence that is labeled by a sub-proof, by a sequence of propositions beginning with the premises (if any) of the sub-proof that are *additional* to those already assumed, labelling them as *suppositions*, and continue with the propositions of the sub-proof until its conclusion is reached, at which point another label indicates that the supposition is 'de-supposed' or, as is usually said, *discharged*.

In the case of conditional proof in split-level format, A; A, $\beta \vdash \gamma$ / $\beta \rightarrow \gamma$, this means that we add to the premises A of the main proof the incrementing premise β of the sub-proof, calling it a supposition and making free use of it as well as the original premises in A until we reach the conclusion γ of the sub-proof, where we annotate that the supposition is no longer active (discharged) and, by the rule of conditional proof, claim that $\beta \rightarrow \gamma$ may be concluded from the original premises A alone. In brief, given the premises in A, and wanting to get $\beta \rightarrow \gamma$, the original split-level proof carries out a sub-proof of γ from $A \cup \{\beta\}$. The flattened version thus carries out the following procedure:

- Given A, wanting to get $\beta \rightarrow \gamma$:
- Suppose β

- Argue to γ,
- Discharge β,
- Conclude $\beta \rightarrow \gamma$ from A alone.

The advantage of such flattening is practical. By integrating the sub-proof into the main proof it cuts down on repetition. In the unflattened derivation we needed to repeat all the inputs in A as premises of the subordinate inference $A, \beta \vdash \gamma$; in the flattened version they appear only once. That may seem a very meagre gain but in practice it makes things much more manageable, at least for humans as contrasted with computers—especially when there are multiple embedded sub-proofs, each with many premises.

On the other hand, flattening can have the conceptual side-effect of moving the recursion contained in the very notion of split-level proof out of the spotlight to the back of the stage. By mimicking an elementary derivation, it can lead the unwary student into thinking that it is one. To avoid misunderstandings of that kind, a flattened presentation should retain enough annotative signals such as the label 'suppose' and indication of points of discharge, to let the reader know where the recursions are, permitting in principle recovery of the split-level or second-level derivations in full.

All of this applies, *mutatis mutandis*, to the other higher-level proof rules that will be explained in following sections. In each case, the higher-level rule is incremental, so that flattening is possible.

Recapitulating the chapter so far, we have noted two ways of reducing repetition in a derivation. One, described in Sect. 10.1.1, was called *squeezing*; the other, described in this section, is *flattening*. They should not be confused. In a nutshell, one can say that squeezing takes trees to sequences, while flattening puts split-level reasoning into a format resembling first-level reasoning. More specifically:

- Squeezing may be applied to any derivation in tree form, transforming it into a sequence accompanied by some labels to record its pattern of justification. This eliminates the obligation that trees impose on us to repeat propositions that are appealed to more than once, thereby also repeating the subtrees above them. This operation is always possible.
- On the other hand, flattening transforms arguments that combine first-level with split-level rules (independently of whether they are in tree or sequence form) into structures that mimic elementary derivations in that they contain only propositions, but which have the implicit application of higher-level rules flagged by further labels such as 'suppose' and indicating points of discharge. This allows us to forbear repeating in the sub-proofs all among their premises that are already premises of the main proof. It is possible when all split-level rules employed are incremental.

10.4 Other Higher-Level Proof Strategies

Now that we have a clear picture of what is going on behind the scenes in a conditional proof, we can review more easily other higher-level proof strategies, namely disjunctive proof and the closely related method of proof by cases, proof by contradiction, also known as *reductio ad absurdum*, and proofs to and from arbitrary instances.

10.4.1 Disjunctive Proof and Proof by Cases

We said a little about disjunctive proof in a logic box of Chap. 1, Sect. 1.4.2; now 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, so 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 original premises logically imply one of $\neg\beta_1$, $\neg\beta_2$, which will not often be the case. However, we can tackle the problem in another way, by a 'divide and rule' strategy. We first *suppose* β_1 and try to get the same conclusion, γ. If we succeed, we discharge that supposition without yet concluding anything, and *suppose* β_2 aiming again at γ. If we succeed again, we discharge the second supposition and conclude γ from whatever our original premises were. This is *disjunctive proof*, acronym DP.

Again, it is not usual in mathematical arguments for disjunctive proof to occur in isolation. More commonly, the information available will be of the kind $\forall x(\beta_1 \vee \beta_2)$ or $\forall x[\alpha \rightarrow (\beta_1 \vee \beta_2)]$, and we have to peel off the outer quantifier and, in the second kind, start a conditional proof 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 ready 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)

(a) Does it matter which disjunct we handle first?
(b) Do we need to require that the two disjuncts are exclusive, that is, that they cannot both be true?
(c) Construct informal verbal arguments explicitly using disjunctive proof to show that $(A \cup B) \times C \subseteq (A \times C) \cup (B \times C)$ and conversely.

Solution

(a) No, it does not matter which disjunct is dealt with first, because the two subordinate arguments are independent of each other. In practice, it is good style, because better communication, to cover the easier sub-proof first.

(b) No, we don't need to require that the two disjuncts are exclusive. When they are both true, the situation is covered twice, once by each of the sub-proofs. That said, it is a little inelegant to cover the same ground twice, and it is sometimes possible to choose one's disjuncts so that they are exclusive without introducing other complications.

(c) To show LHS $\subseteq$ RHS, suppose $(x,y) \in$ LHS. Then $x \in A \cup B$ and $y \in C$. By the former, either $x \in A$ or $x \in B$. Suppose first that $x \in A$. Then $(x,y) \in A \times C \subseteq$ RHS. Suppose alternatively that $x \in B$. Then $(x,y) \in B \times C \subseteq$ RHS. Thus in both cases $(x,y) \in$ RHS, as desired. For the converse, suppose $(x,y) \in$ RHS. Then either $(x,y) \in A \times C$ or $(x,y) \in B \times C$. Suppose first that $(x,y) \in A \times C$. Then $x \in A$ and $y \in C$ so $x \in A \cup B$ and $y \in C$ so $(x,y) \in$ LHS. Suppose alternatively that $(x,y) \in B \times C$. Then $x \in B$ and $y \in C$ so $x \in A \cup B$ and $y \in C$ so again $(x,y) \in$ LHS. *End of solution.*

What is the logical rule underlying disjunctive proof? Expressing it incrementally, it is:

$$A, \beta_1 \vee \beta_2, \beta_1 \vdash \gamma;\ A, \beta_1 \vee \beta_2, \beta_2 \vdash \gamma\ /\ A, \beta_1 \vee \beta_2 \vdash \gamma.$$

In other words, when the two inferences expressed by turnstiles to the left of the slash are correct, then so is the inference expressed by the turnstile to the right of the slash. Of course, for classical consequence (or anything reasonably like it) we have $\beta_1 \vdash \beta_1 \vee \beta_2$ and $\beta_2 \vdash \beta_1 \vee \beta_2$ so, for such consequence relations, the rule may be expressed more succinctly without $\beta_1 \vee \beta_2$ appearing as a premise in the subordinate inferences. That gives us the following more succinct and familiar version:

$$A, \beta_1 \vdash \gamma;\ A, \beta_2 \vdash \gamma\ /\ A, \beta_1 \vee \beta_2 \vdash \gamma.$$

This is the formulation of disjunctive proof that we will usually refer to, with acronym DP and often known as $\vee^-$ because it eliminates a disjunction.

Alice Box: Why is this described as elimination?

Alice Why do you say that DP eliminate disjunction? Surely it *introduces* the disjunction into the principal inference!

Hatter Indeed it does; in this respect, it is like the rule $\exists^-$ discussed in Chap. 9, Sect. 9.4.3. But notice that each of DP and $\exists^-$ introduces its connective into the *premises* of the principal inference, not into the conclusion of the principal inference. To appreciate the rationale

for the names $\exists^-$, $\vee^-$, you should understand that the minus signs do not refer to what happens in the transition from subordinate inference to principal inference, but rather what takes place in the passage from the premises of the principal inference to the conclusion of that same inference. Each of the rules $\vee^-$, $\exists^-$ eliminates its connective in that passage.

Alice That's weird!

Hatter What would you expect from a mad hatter? But seriously, it brings out a deep pattern that will be defined precisely at the end of the chapter

Exercise 10.4.1 (2)
Verify that the rule of disjunctive proof, in both its succinct and incremental versions, is correct for classical consequence.

Solution
Consider first the succinct version. Suppose A, $\beta_1 \vee \beta_2 \nvDash \gamma$. Then there is a valuation v such that $v(\beta_1 \vee \beta_2) = 1$, $v(\gamma) = 0$ and $v(A) = 1$ (recall from Exercise 10.3.2 that this is shorthand for $v(\alpha) = 1$ for all $\alpha \in A$). Then either $v(\beta_1) = 1$ or $v(\beta_2) = 1$. In the former case we have $v(A) = 1$, $v(\beta_1) = 1$, $v(\gamma) = 0$ so A, $\beta_1 \nvDash \gamma$. In the latter case we have $v(A) = 1$, $v(\beta_2) = 1$, $v(\gamma) = 0$ so A, $\beta_2 \nvDash \gamma$ and the verification is complete. The same verification works with minimal editing for the incremental formulation. *End of solution*.

Of course, in Exercise 10.4.1 (2) we are using disjunctive proof informally in the meta-language to verify the correctness of its formal articulation in the object-language; but this is a pervasive phenomenon that we have already seen several times in Chaps. 8 and 9 and there is no point in trying to disguise it.

What would a split-level formulation of disjunctive proof look like? Consider the succinct formulation, A, $\beta_1 \vdash \gamma$; A, $\beta_2 \vdash \gamma$ / A, $\beta_1 \vee \beta_2 \vdash \gamma$. We would move A, $\beta_1 \vee \beta_2$ across from its position as premise of the principal inference, to become a third input to the rule, giving us:

$$A,\ \beta_1 \vee \beta_2;\ A, \beta_1 \vdash \gamma; A, \beta_2 \vdash \gamma \,/\, \gamma.$$

In words: proposition γ is true whenever $\beta_1 \vee \beta_2$ as well as all propositions in A are true and the inferences A, $\beta_1 \vdash \gamma$ and A,$\beta_2 \vdash \gamma$ are both valid.

Just as for conditional proof, this split-level-formulation can be *flattened* to look like this:

Given A and $\beta_1 \vee \beta_2$, wanting to get γ:
Suppose β_1
Argue to γ
Discharge β_1

Suppose β_2
Argue to γ again
Discharge β_2
Conclude γ as desired.

For disjunctive proof, flattening reduces repetition even more than it does for conditional proof. A single application of the unflattened version of split-level DP would write all the propositions in A three times, once for each input; a flattened version writes them only once, in the top line of the above schematic presentation.

Exercise 10.4.1 (3)
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 semi-formal suppositional argument using both conditional and disjunctive proof strategies.

Solution
Assume as premises $p \rightarrow ((r \vee s) \rightarrow t)$, $q \rightarrow (\neg(s \vee u) \vee t)$, s; we want to get $(p \vee q) \rightarrow t$. Suppose (for a conditional proof) $p \vee q$; our goal now is t. Suppose first (for disjunctive proof) p. Then by modus ponens on the first premise we have $(r \vee s) \rightarrow t$. We also have s as third premise so an application of $\vee$ + provides $r \vee s$ so by modus ponens we have t as desired. Discharge the first supposition for disjunctive proof and make the second supposition q. Modus ponens on the second premise yields $\neg(s \vee u) \vee t$. But since we have s as third premise, we have $s \vee u$; double negating it $\neg\neg(s \vee u)$ permits us to carry out disjunctive syllogism yielding t again. Discharging the second supposition for disjunctive prof leaves us with t on the supposition $p \vee q$. Discharging that supposition by conditional proof gives us the original goal $(p \vee q) \rightarrow t$. *End of solution*.

If you find it helpful, you can set out the solution to Exercise 10.4.1 (3) as a numbered sequence with each item in the sequence annotated to indicate whether it is a premise, a supposition, obtained from what by which elementary rule, or obtained by discharging a supposition using a higher-level rule. That is the way in which textbooks focusing on natural deduction do it.

As remarked in Chap. 1, in mathematical practice, disjunctive proof often appears in a variant (though classically equivalent) form known as *proof by cases*. We have a premise-set A 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 them into two. We think up a suitable proposition β—which need not be mentioned explicitly in A or in γ—and consider separately two cases, one for β and the other for $\neg\beta$. Each of them may open the way to a neat proof of γ. The two proofs may quite similar but they can also turn out to be 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 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 it is natural to break the problem down into the cases that n is even and that it is

odd. In the chapter on trees, Sect. 7.6.2, we used a quite sophisticated proof by cases when showing that every cycle in an unrooted tree contains at least one repeated edge: we broke the situation into *three* cases (this, that, neither of the two) rather than just two, and this was done *inside* a proof by contradiction. You will find it profitable to revisit that proof in the light of what you now know about its underlying logic.

The underlying second-level rule for proof by cases is:

$$A, \beta \vdash \gamma;\ A, \neg\beta \vdash \gamma \,/\, A \vdash \gamma.$$

Expressed in split-level form, this becomes:

$$A;\ A, \beta \vdash \gamma;\ A, \neg\beta \vdash \gamma \,/\, \gamma.$$

Flattening this split-level rule we get the procedure:

Given A, wanting to get γ
Suppose β
Argue to γ
Discharge β
Suppose $\neg\beta$
Argue to γ again
Discharge $\neg\beta$
Conclude γ as desired.

Exercise 10.4.1 (4)
Verify that the rule of proof by cases is correct for classical consequence.

Solution
Again, the verification is essentially the same for second-level and split-level formulations. It is slightly more elegant for the former, so we do it that way. Suppose $A \not\models \gamma$. Then there is a valuation v such that $v(A) = 1$, $v(\gamma) = 0$. If $v(\beta) = 1$ then $v(A \cup \{\beta\}) = 1$, $v(\gamma) = 0$ so $A, \beta \not\models \gamma$, while if $v(\beta) = 0$ then $v(A \cup \{\neg\beta\}) = 1$, $v(\gamma) = 0$ so $A, \neg\beta \not\models \gamma$. *End of solution.*

In our two arithmetical examples of proof by cases, it was rather obvious how best to choose the proposition β that defines the two cases; in such instances, the difference between proof by cases and disjunctive proof can be little more than a matter of verbal editing. But there are other examples, such as that about cycles in unrooted trees, in which it is not immediately obvious what would be a good choice of a proposition β to distinguish cases. Indeed, it can happen that finding one can be the key to solving the problem. Experience helps one do that, just as it helps us cut a roast turkey at the joints.

Exercise 10.4.1 (5)
Use proof by cases to give a simple proof that there are positive irrational numbers x, y such that x^y is rational.

Solution
We cook up two pairs (x,y) and (x',y') and show that at least one of them does the job, although we do not show which of them does.

Put $y = \sqrt{2}$, which is irrational. Now $\sqrt{2}^{\sqrt{2}}$ is either rational or irrational. In the former case put $x = \sqrt{2}$ and we are done. In the latter case, put $x = \sqrt{2}^{\sqrt{2}}$ which, by the supposition of the case, is irrational. Then $x^y = (\sqrt{2^{\sqrt{2}}})^{\sqrt{2}} = (\sqrt{2})^{\sqrt{2}\cdot\sqrt{2}} = (\sqrt{2})^2 = 2$ which is rational and again we are done. *End of solution*.

The solution to Exercise 10.4.1 (5), known as Jarden's proof, is widely cited as an example of an elementary proof that is *non-constructive*. It considers cases β, $\neg\beta$ (where β is a sentence, that is, without free variables) and proves the desired result separately for each case without knowing which of them really holds. Such arguments generally convey less information than constructive ones and some non-classical mathematicians go so far as to cast doubt on their validity. On the other hand, constructive proofs can be much longer and more intricate. In the case of the exercise, it is possible to render the proof constructive by establishing which of the two cases holds; in fact, it is the second case that holds, as can be shown by a long and difficult proof that we do not attempt to describe.

Sometimes, in a proof, one finds oneself iterating the splitting to get sub-cases, sub-sub-cases, and so on, ending up with dozens of them in several layers. Such arguments can be rather inelegant and are disdained by mathematicians when less fragmented ones can be found. However, computers are not so averse to them. A notorious example arose in the celebrated solution of the four-colour problem of 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 next method of argument has been famous since Greek antiquity. It is known as *proof by contradiction* or, using its Latin name, *reductio ad absurdum*, briefly *reductio*, and bearing the acronym RAA. It was sketched in an Alice box in Chap. 2 ; here we go further on a more analytic level.

Imagine that we are given a premise-set A and we want to get a conclusion β. We may even have tried various other methods and got stuck, so we try the following: we suppose the proposition $\neg\beta$ opposite to what we want to show, and seek to establish, on this basis, a contradiction. If we manage to do so, the proof of β from A is complete.

What is meant by 'a contradiction' here? We mean an *explicit* contradiction, that is, any statement γ and its negation $\neg\gamma$ which, in Chap. 8 Sect. 8.5 in the context of truth-trees, we called briefly *crash-pairs*. As remarked by the Hatter in the Alice box, it does not matter what contradiction it is, that is, how γ is chosen. Indeed, when constructing the proof, we often do not have a clear idea which contradiction we will end up with. Moreover, a little editing such as replacing a modus ponens by

a modus tollens can modify the contradiction that is obtained; but we don't care in the least.

Proof by contradiction is a universal rule in the sense that neither the premises nor the conclusion of the principal inference need be in any specific form. This is in contrast with conditional proof, which is designed specifically for getting a conditional conclusion, as well as disjunctive proof, which proceeds from a disjunctive premise. Sometimes, using *reductio* can make an argument a great deal easier due to the availability of the supposition as one more item available to work with, but on other occasions it will not make much difference. Some mathematicians prefer to use *reductio* only when they really need 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*. In such applications, it gets a contradiction from the supposition that nothing has that feature—but without thereby specifying an item possessing it. As we have already seen in Exercise 10.4.1 (4), the same can happen with proof by cases, but it is much more common in proofs by contradiction.

The most famous example of proof by contradiction, dating back to Greek antiquity, is a standard proof that $\sqrt{2}$ is irrational. Everybody learned it at school, but we recall it here to highlight the logical structure. The argument makes use of the fact that an integer n is even iff n^2 is even. Suppose that $\sqrt{2}$ is rational. Then by the definition of rationality, $\sqrt{2} = a/b$ where a, b are integers. We may assume *wlog* (without loss of generality, see Chap. 7, Sect. 7.6.2) that a, b share no factors other than 1. Since $\sqrt{2} = a/b$ we have $2 = (a/b)^2 = a^2/b^2$ so $a^2 = 2b^2$ so a^2 is even; hence a is even, that is, $a = 2c$ for some integer c. Substituting $2c$ for a in the equality $a^2 = 2b^2$ gives us $(2c)^2 = 2b^2$, hence $2b^2 = 4c^2$ so $b^2 = 2c^2$. Thus, b^2 is even, so b is even. Hence 2 is a common factor of a, b, giving us the desired contradiction.

We used proof by contradiction many times in previous chapters. In particular, Chap. 7 on trees was full of them—there were five in a row in Sect. 7.2.1 and one *reductio* inside another in Sect. 7.6.2. Reviewing those proofs will help appreciate the power and convenience of the method in practice.

Exercise 10.4.2 (1)
Given the premises $s \to q$, $(r \vee q) \to \neg p$, use proof by contradiction and elementary inferences to get $\neg(p \wedge s)$.

Solution
Suppose $\neg\neg(p \wedge s)$; we want to get an explicit contradiction. Eliminating the double negation, we have $p \wedge s$. From that we have each of s, p; using s with the first premise gives us q by modus ponens, from which we have $r \vee q$ and so by another modus ponens $\neg p$, giving us the desired contradiction. *End of solution*.

When we are aiming for a negative conclusion $\neg\beta$ as in Exercise 10.4.2 (1), we can evidently take a short cut and suppose β directly rather than suppose $\neg\neg\beta$ and then get rid of the double negation.

Writing $\perp$ to stand for any explicit contradiction, we can state the second-level rule underlying the procedure: $A \vdash \beta$ whenever $A, \neg\beta \vdash \perp$. With a slash for the central transition, this is written:

$$A, \neg\beta \vdash \perp \,/\, A \vdash \beta.$$

In *split-level* form, it becomes:

$$A;\, A, \neg\beta \vdash \perp \,/\, \beta.$$

Finally, a flattened rendering of the split-level version:

Given A, wanting to get β:
Suppose $\neg\beta$
Argue to an explicit contradiction
Discharge $\neg\beta$,
Conclude β as desired.

Exercise 10.4.2 (2)
Show that proof by contradiction is correct for classical consequence.

Solution
We consider the second-level formulation $A, \neg\beta \models \perp / A \models \beta$ and argue contrapositively. Suppose that $A \not\models \beta$; we want to show that $A, \neg\beta \not\models \perp$. By the supposition, there is a valuation v with $v(A) = 1$ and $v(\beta) = 0$, so $v(A \cup \{\neg\beta\}) = 1$. But since $\perp$ is shorthand for an explicit contradiction, we have $v(\perp) = 0$ so that $A, \neg\beta \not\models \perp$ as desired.

Alice Box: Indirect inference

Alice In other discussions, I have seen the term *indirect inference*. Is that the same as proof by contradiction? Or does it cover, more broadly, all higher-level strategies?

Hatter The term 'indirect inference' is more of a cognitive description than a logical one. It is used to mean any way of writing out an argument in which the order of development appears to 'swim against the stream of implication'. So understood, it covers more than proof by contradiction, but less than all higher-level proof procedures.

Alice For example?

Hatter Conditional proof, disjunctive proof and proof by cases are usually considered direct. Proof by contradiction, on the other hand, is always seen as indirect as it supposes the very opposite of what we want to prove.

Alice Any others?

Hatter The variant of conditional proof that we called 'contraposed conditional proof' in Sect. 8.3.1, is also often described as indirect. Presented as a second-level rule, it has the form $A \cup \{\neg\gamma\} \vdash \neg\beta$ / $A \vdash \beta \rightarrow \gamma$. It gives the impression of swimming against the direction of implication, and so is often called 'indirect conditional proof'.

Alice Any formal basis for that impression?

Hatter The supposition $\neg\gamma$ (resp. conclusion $\neg\beta$) of the subordinate inference is the negation of the consequent γ (resp. antecedent β) of the conclusion of the principal inference.

There is a sense in which, if we are given any one of the three higher-level rules that we have been discussing, the others become redundant. For example, we can make conditional proof do the work of disjunctive proof as follows. Given premises A, $\beta_1 \vee \beta_2$, first suppose β_1 to get γ, apply CP to get $\beta_1 \rightarrow \gamma$ and discharge β_1; then suppose β_2 to get γ, apply CP again to obtain $\beta_2 \rightarrow \gamma$ and discharge β_2. Finally, use the tautological implication $\beta_1 \rightarrow \gamma$, $\beta_2 \rightarrow \gamma$, $\beta_1 \vee \beta_2 \models \gamma$, which one could add to the list in Table 8.5 of Chap. 8, to conclude γ from A, $\beta_1 \vee \beta_2$ as desired. But this is a rather roundabout way to proceed so, in mathematical practice, disjunctive proof is considered as a strategy in its own right.

We end this section with some remarks for the philosophically inclined; others may skip to the next section. There are a few mathematicians and rather more philosophers of mathematics who are wary of the use of truth-values in the discipline. For them, the notion of truth has no meaning in mathematics beyond 'intuitively provable'; moreover falsehood means no more than '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* dating back to the early twentieth century, some classical principles go out the window. The most notorious of these is the tautology of excluded middle $\alpha \vee \neg\alpha$. But other principles are also affected; in particular, a number of classical equivalences involving negation are retained in one direction only. For example, using $\vdash$ for intuitionistically acceptable logical implication, classical double negation $\neg\neg\alpha \dashv\vdash \alpha$ is reduced to $\alpha \vdash \neg\neg\alpha$; contraposition $\alpha \rightarrow \beta \dashv\vdash \neg\beta \rightarrow \neg\alpha$ drops to $\alpha \rightarrow \beta \vdash \neg\beta \rightarrow \neg\alpha$; 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 of principles where negation does not figure. For example, the classical tautology $(\alpha \rightarrow \beta) \vee (\beta \rightarrow \alpha)$ is not intuitionistically acceptable, nor is the following classical tautology that uses implication alone: $((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \alpha$.

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 argument by cases in any form that presumes excluded middle is out; proof by contradiction is not accepted in the form that we have been using, but only in a form with interchanged negations: $A, \beta \vdash \perp$ / $A \vdash \neg\beta$. However, all that is beyond our remit.

10.4.3 Proof Using Arbitrary Instances

In Chap. 9, Sect. 9.4.3, we articulated the second-level rules $\forall^+$ and $\exists^-$ for the quantifiers in classical logic, contrasting them with the first-level logical implications $\forall^-$ and $\exists^+$. Their formulations for classical consequence $\models$ were:

$\forall^+$: Whenever $A \models \alpha$ then $A \models \forall x(\alpha)$, provided the variable x has no free occurrences in any formula α in A
$\exists^-$: Whenever $A, \beta \models \gamma$ then $A, \exists x(\beta) \models \gamma$, provided the variable x has no free occurrences in any formula $\alpha \in A \cup \{\gamma\}$.

As also noted, $\forall^+$ is also commonly called 'universal generalization' (UG) and $\exists^-$ referred to as 'existential instantiation' (EI), although those names do little to bring out the second-order nature of the rules or reflect the logic of what is going on, being more apposite as descriptions of the flattened presentations of the split-level versions of these rules.

There is one respect in which $\forall^+$ differs from both its dual $\exists^-$ and all the second-level rules for propositional connectives that we have considered. *It makes no supposition*! The premises of the subordinate inference are exactly those of the principal inference. So, we cannot simply identify higher-level inference with the making and discharging of suppositions. Suppositions are a frequent and salient feature of such inferences, but not an invariable one. Nevertheless, like the other higher-level rules that we have considered, $\forall^+$ is incremental since the principal and subordinate inferences have the same premises.

On the other hand, the rule $\exists^-$, as we have written it, is not incremental: the premise $\exists x(\alpha)$ of the principal inference is not a premise of the subordinate one. Nevertheless, since $\alpha \models \exists x(\alpha)$, we may make it incremental by reformulating it a little redundantly, just as we did with disjunctive proof:

$\exists^-$: Whenever $A, \beta, \exists x(\beta) \models \gamma$ then $A, \exists x(\beta) \models \gamma$, provided the variable x has no free occurrences in any formula $\alpha \in A \cup \{\gamma\}$.

However, in practice, one usually works with the non-incremental formulation since it is simpler.

For arbitrary consequence relations $\vdash$, using the schematic slash notation and writing 'x not free in A' to abbreviate 'x is not free in any formula in A', the rules $\forall^+$ and $\exists^-$ say:

$\forall^+$: $A \vdash \alpha \,/\, A \vdash \forall x(\alpha)$ (x not free in A)
$\exists^-$: $A, \beta \vdash \gamma \,/\, A, \exists x(\beta) \vdash \gamma$ (x not free in A or in γ).

Written as split-level rules, distinguishing semi-colons for separate rule-inputs from commas for separate premises in an inferential input:

$\forall^+$: $A;\ A \vdash \alpha \,/\, \forall x(\alpha)$ (x not free in A)
$\exists^-$: $A;\ \exists x(\beta);\ A, \beta \vdash \gamma \,/\, \gamma$ (x not free in A or in γ).

Flattening $\forall^+$:

Given A, wanting to get $\forall x(\alpha)$:
Argue to α
Check that x is not free in A
Conclude $\forall x(\alpha)$ as desired.

Flattening $\exists^-$:

Given A, $\exists x(\beta)$, wanting to get γ:
Suppose β
Argue to γ
Check that x is not free in A or in γ
Conclude γ as desired.

Conventional mathematical English does not go in for full formal presentation of its propositions and so cannot routinely identify free and bound occurrences of variables in them. In that context, the flattened versions of $\forall^+$, $\exists^-$ are conveyed using terms like 'arbitrary' and 'choose'. It is in terms such as these that we have been carrying out our own reasoning in this book:

$\forall^+$: To show that every element of the domain has a certain property, it suffices to *consider an arbitrary* x and show that it has that property.
$\exists^-$: When we know that at least one element of the domain has a certain property, we may without loss of generality *choose an arbitrary* x and suppose that it has the property.

Alice Box: Arbitrary items?

Alice	I have a philosophical worry about this talk of 'arbitrary x', say an arbitrary positive integer or triangle. 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 quite right.
Alice	So how can we legitimately use the word?
Hatter	Well, it's just a manner of speaking; let's move on …

Alice was right to be puzzled. Her point was made already by the eighteenth-century philosopher Berkeley in a forceful critique of the mathematics of his time, so perhaps we should flesh out the Hatter's response a little. 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 that 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 attributes to x other than those that it asserts, or are implied, for all elements of the domain. The arbitrariness lies not in the item, but in the way in which we handle it.

Such a way of understanding talk of arbitrary items dispels the air of paradox around them, but it is still quite vague. In the end, the only way to make the notion perfectly precise is by using the precise language and concepts of first-order logic, requiring that a certain variable does not occur free in certain places. In this way, a conceptual puzzle is resolved by applying a banal syntactic criterion in the language of first-order logic.

10.4.4 Summary Discussion of Higher-Level Proof

The construction of logical derivations is a recursive affair. That is so even for chaining even though it uses only elementary rules, since chaining explores the closure of a set of premises under those rules. As observed in Sect. 10.2, the construction of elementary derivations by chaining makes implicit use of the three Tarski conditions for logical consequence and, while strong reflexivity is a first-level rule, the other two are second-level.

Exercise 10.4.4
Write the Tarski conditions of monotony and cumulative transitivity for inference relations as second-level rules in the schematic slash manner.

Solution
Monotony is most concisely expressed as $A \vdash \beta \ / \ A \cup X \vdash \beta$ although one could of course write: $A \vdash \beta \ / \ B \vdash \beta$ whenever $A \subseteq B$. Cumulative transitivity may be written: $A \vdash \beta$ for all $\beta \in B$; $A \cup B \vdash \gamma \ / \ A \vdash \gamma$. *End of solution*.

Recursion enters into derivations in another way when we use any of the second-level rules of conditional proof (in direct or contraposed versions), disjunctive proof (or its variant, proof by cases), *reductio ad absurdum*, and for the quantifiers, for then we are considering the closure of the set of all elementary derivations under those rules.

To interact with elementary rules, second-level ones may be expressed in split-level form and, to reduce repetition, derivations using split-level rules can be flattened to become annotated sequences that mimic elementary derivations.

While in theory we could restrict ourselves to first-level rules and the Tarski conditions, in practice all serious proof requires unremitting use of higher-level inference. On the other hand, it does not seem to call upon any rules of third level or beyond. For quick reference, we recapitulate the main second-level rules in Table 10.3.

Table 10.3 The most important second-level inference rules

	Conditional proof (CP, $\rightarrow^{+}$)	Disjunctive proof (DP, $\vee^{-}$)	Proof by contradiction (RAA, *reductio*)	Universal generalization (UG, $\forall^{+}$)	Existential instantiation (EI, $\exists^{-}$)
Subordinate inference(s)	$A, \beta \vdash \gamma$	$A, \beta_1 \vdash \gamma$; $A, \beta_2 \vdash \gamma$	$A, \neg\beta \vdash \gamma \wedge \neg\gamma$	$A \vdash \alpha$	$A, \beta \vdash \gamma$
Principal inference	$A \vdash \beta \rightarrow \gamma$	$A, \beta_1 \vee \beta_2 \vdash \gamma$	$A \vdash \beta$	$A \vdash \forall x(\alpha)$	$A, \exists x(\beta) \vdash \gamma$
Provisos	None	None	None	x not free in A	x not free in γ or in A

We end by fulfilling a promise made to Alice, to dig deeper into why disjunctive proof (DP) and existential instantiation (EI) are classified as elimination rules. When need some precise definitions of different types of second-level rules $A_1 \vdash \alpha_1$; …; $A_n \vdash \alpha_n$ / $B \vdash \beta$. In the first place, they may be partitioned into three categories: *structural*, *intelim* and *undistinguished*:

- A second-level inference rule $A_1 \vdash \alpha_1$; …; $A_n \vdash \alpha_n$ / $B \vdash \beta$ is said to be *structural* iff it contains no connectives at all.
- It is an *intelim* rule iff it contains only one connective and that connective occurs just once, with the unique occurrence being in the principal inference $B \vdash \beta$.
- It is *undistinguished* if it is neither structural nor intelim.

The intelim rules are partitioned into *introduction* and *elimination* rules:

- When the unique occurrence of the connective occurs in β (the conclusion of the output of the rule) then it is an introduction rule.
- When the unique occurrence of the connective occurs in B (the premise-set of the output of the rule) then it is an elimination rule.

It should be noted that in the literature, particularly of a philosophical kind, the notion of an intelim rule (and thus of its sub-categories, introduction and elimination) is sometimes used quite loosely with one or more of the above conditions relaxed. Sometimes two connectives are allowed, or the unique connective may occur twice in the principal inference, or it may also occur in the subordinate inference.

Exercise 10.4.5 Classify the second-level rules discussed in this chapter, namely monotony, cumulative transitivity, conditional proof, disjunctive proof, proof by cases, *reductio*, UG, EI as structural, introduction, elimination or undistinguished.

Solution

Monotony and cumulative transitivity are structural. CP is an introduction rule, as is UG (hence their icons $\rightarrow^{+}$, $\forall^{+}$ with a plus sign). On the definitions provided, DP and EI are elimination rules (signaled by icons $\vee^{-}$, $\exists^{-}$). This answers Alice's earlier questions about the nomenclature.

In contrast to DP, its cousin proof by cases is undistinguished, since negation appears in the subordinate inference. This difference is of little importance for classical deductive practice, but for some non-classical logics it is significant. For example, DP is accepted in the system of intuitionistic logic mentioned in Sect. 10.4.2 while proof by cases in the form considered here is not.

Reductio is the most interesting of our rules to classify. Inspecting its formulation, $A, \neg\beta \vdash \gamma \wedge \neg\gamma \,/\, A \vdash \beta$, we see that there is a negation (in fact, two) in the subordinate inference so, like proof by cases, it is undistinguished.

To be sure, a variant version of *reductio*, $A, \beta \vdash \gamma \wedge \neg\gamma \,/\, A \vdash \neg\beta$ gets rid of one the negations, but the other remains. Moreover, if we introduce into the language the zero-ary connective $\perp$ known as the falsum (see Chap. 8 Sect. 8.4.3) we can rewrite the rule as $A, \beta \vdash \perp \,/\, A \vdash \neg\beta$ thus eliminating both negations from the subordinate inference. But this manipulation introduces another connective, namely $\perp$, into the subordinate inference so we still do not have an intelim rule in the strict sense of the term.

Exercise 10.4.6

Recall from Exercise 8.3.3 (3) that a truth-functional connective $*(p_1, \ldots, p_k)$ ($k \geq 0$) is called *contrarian* iff $v(*(p_1, \ldots, p_k)) = 0$ when all $v(p_i) = 1$ ($i \leq k$). (a) Identify five familiar contrarian truth-functional connectives with $k \leq 2$. (b) Show that no contrarian truth-functional connective of any arity has a classically correct introduction rule.

Solution

(a) The truth-functional connectives $\neg$, $\oplus$ (exclusive disjunction), *nand* (not-both), $\downarrow$ (neither-nor), and the zero-ary falsum $\perp$ are all familiar contrarian ones. For a complete list, inspect Table 8.4 in Chap. 8, recalling that zero-place and one-place connectives reappear as two-place ones with one or both of their letters redundant.

(b) Consider any introduction rule $A_1 \models \alpha_1; \ldots; A_n \models \alpha_n \,/\, B \models \beta$ for classical consequence $\models$ and a k-place truth-functional connective $*(p_1, \ldots, p_k)$. Such a rule contains * as its unique connective and * occurs in it just once, with the unique occurrence being in β. Now substitute a classical tautology, e.g. $p \vee \neg p$, for all sentence letters in this scheme, writing σ for the substitution function. Since classical consequence is closed under substitution, we have $\sigma(A_1) \models \sigma(\alpha_1), \ldots, \sigma(A_n) \models \sigma(\alpha_n)$. Since the formulae in B contain no connectives, they are all sentence letters so that all formulae in $\sigma(B)$ are tautologies. Since the connective * occurs exactly once in β, we have $\beta = *(p_1, \ldots, p_k)$ for some sentence letters $p_1, \ldots, p_k$. Hence $\sigma(\beta) = \sigma(*(p_1, \ldots, p_k)) = *(\sigma(p_1), \ldots, \sigma(p_k))$. Now let v be any valuation. Then each $v(\sigma(p_i)) = 1$ so, since * is

contrarian, $v(\sigma(\beta)) = 0$ while $v(\sigma(B)) = 1$. Putting all this together, we have a substitution instance of the rule where all n subordinate inferences are classically correct while the principle inference is not, so the scheme fails for classical logic.

10.5 End-of-Chapter Exercises

Exercise 10.5 (1) Consequence relations

(a) Show that cumulative transitivity taken together with monotony implies (plain) transitivity, i.e. the principle that whenever $\alpha \vdash \beta$ and $\beta \vdash \gamma$ then $\alpha \vdash \gamma$.
(b) Let $\vdash$ be the relation defined by putting $A \vdash \beta$ iff either (i) A is a singleton $\{\alpha\}$ and $\alpha = \beta$ or (ii) A has more than one element. Show that $\vdash$ satisfies the three Tarski conditions.

Solution

(a) Suppose $\alpha \vdash \beta$, $\beta \vdash \gamma$. By monotony on the latter, $\{\alpha,\beta\} \vdash \gamma$, so $\alpha \vdash \gamma$ by cumulative transitivity.
(b) For strong reflexivity, suppose $\alpha \in A$. If $A = \{\alpha\}$, then $A \vdash \alpha$ by clause (i), while if A has more than one element, then again $A \vdash \alpha$ by clause (ii). For monotony, suppose $A \vdash \alpha$ and $A \subseteq B$. By the supposition, either $A = \{\alpha\}$ or A has more than one element. In the former case, $\alpha \in B$ so $B \vdash \alpha$ by strong reflexivity as already checked while in the latter case B also has more than one element so $B \vdash \alpha$ by clause (ii).

For cumulative transitivity, suppose $A \vdash \beta$ for all $\beta \in B$ and $A \cup B \vdash \gamma$; we need to check that $A \vdash \gamma$. We consider three cases. Case 1: $A = \varnothing$. Since $A \vdash \beta$ for all $\beta \in B$ we must have $B = \varnothing$ so $A \cup B = \varnothing$ contradicting $A \cup B \vdash \gamma$. Case 2: A is a singleton $\{\alpha\}$. Then since $A \vdash \beta$ for all $\beta \in B$ we must have $B \subseteq A$ so $A \cup B = A = \{\alpha\}$ so, since $A \cup B \vdash \gamma$, we have $\gamma = \alpha$ so $A \vdash \gamma$. Case 3: A has more than one element. Then immediately $A \vdash \gamma$ by clause (ii) and we are done.

Exercise 10.5 (2) Consequence operations

(a) In Exercise 10.2.3 (a) we noted that although the inclusion $C(A) \cup C(B) \subseteq C(A \cup B)$ holds for any consequence operation C, its converse $C(A \cup B) \subseteq C(A) \cup C(B)$ fails for $C = Cn$. Show that nevertheless $C(A \cup B) = C(C(A) \cup C(B))$ holds for all consequence operations C.
(b) (i) Show that for any consequence operation C, if $A \subseteq C(B)$ and $B \subseteq C(A)$ then $C(A) = C(B)$; (ii) express the principle in terms of consequence relations
(c) Show that $C(A) \cap C(B) = C(C(A) \cap C(B))$ for any consequence operation.

Solution

(a) In one direction: Since $C(A) \cup C(B) \subseteq C(A \cup B)$ as shown in Exercise 10.2.3 (a), monotony and idempotence give us $C(C(A) \cup C(B)) \subseteq CC(A \cup B) \subseteq C(A \cup B)$. For the converse, we have $A \subseteq C(A)$, $A \subseteq C(B)$ by strong reflexivity so, by basic set theory, $A \cup B \subseteq C(A) \cup C(B)$ and hence by monotony $C(A \cup B) \subseteq C(C(A) \cup C(B))$ as desired.

(b) (i) Suppose $A \subseteq C(B)$ and $B \subseteq C(A)$. Applying monotony and idempotence to the former gives $C(A) \subseteq CC(B) = C(B)$; applying the same to the latter gives $C(B) \subseteq CC(A) = C(A)$; put these together and we are done. (ii) Whenever both $A \vdash \beta$ for all $\beta \in B$ and $B \vdash \alpha$ for all $\alpha \in A$, then $A \vdash \gamma$ iff $B \vdash \gamma$ for all γ.

(c) LHS $\subseteq$ RHS is immediate by strong reflexivity; the converse inclusion is a bit trickier. Clearly $C(A) \cap C(B) \subseteq C(A)$ so, by monotony, $C(C(A) \cap C(B)) \subseteq CC(A) = C(A)$. Similarly, $C(A) \cap C(B) \subseteq C(B)$ so $C(C(A) \cap C(B)) \subseteq CC(B) = C(B)$. Putting these together gives us $C\{C(A) \cap C(B)\} \subseteq C(A) \cap C(B)$ as desired.

Exercise 10.5 (3) Higher-level rules

(a) Show how the rule of proof by cases for a consequence relation $\vdash$ may be obtained from disjunctive proof for $\vdash$ whenever $\vdash$ satisfies the condition that $A \vdash \beta \vee \neg\beta$ for all A, β.

(b) Show, conversely, how the rule of disjunctive proof for a consequence relation $\vdash$ may be obtained from proof by cases for $\vdash$ whenever $\vdash$ satisfies disjunctive syllogism, i.e. the condition that $\{\beta_1 \vee \beta_2, \neg\beta_1\} \vdash \beta_2$ for all formulae β_1, β_2.

(c) Write out the converses of conditional proof, *reductio*, $\forall^+$, $\exists^-$ and disjunctive proof and check whether they are classically correct.

Solution

(a) Assume that a consequence relation $\vdash$ satisfies disjunctive proof and suppose that both $A, \beta \vdash \gamma$ and $A, \neg\beta \vdash \gamma$; we need to show that $A \vdash \gamma$. Applying DP to the suppositions we have $A \cup \{\beta \vee \neg\beta\} \vdash \gamma$. But it is assumed that $A \vdash \beta \vee \neg\beta$, so $A \vdash \gamma$ by cumulative transitivity.

(b) This direction is a bit less immediate. Assume that a consequence relation $\vdash$ satisfies proof by cases and suppose that both $A \cup \{\beta_1\} \vdash \gamma$ and $A \cup \{\beta_2\} \vdash \gamma$; we need to show that $A \cup \{\beta_1 \vee \beta_2\} \vdash \gamma$. Proof by cases tells us that it will suffice to get both $A \cup \{\beta_1 \vee \beta_2\} \cup \{\beta_1\} \vdash \gamma$ and $A \cup \{\beta_1 \vee \beta_2\} \cup \{\neg\beta_1\} \vdash \gamma$. The former is immediate by monotony from the first supposition. For the second, monotony applied to the second supposition gives us $A \cup \{\beta_1 \vee \beta_2\} \cup \{\neg\beta_1\} \cup \{\beta_2\} \vdash \gamma$, and monotony applied to the assumption $\{\beta_1 \vee \beta_2, \neg\beta_1\} \vdash \beta_2$ provides $A \cup \{\beta_1 \vee \beta_2\} \cup \{\neg\beta_1\} \vdash \beta_2$. Cumulative transitivity thus yields $A \cup \{\beta_1 \vee \beta_2\} \cup \{\neg\beta_1\} \vdash \gamma$ and the verification is complete.

(c) All five converses are classically correct. For this reason, the rules are often said to be *invertible* (in contrast, neither of the two second-level Tarski conditions of monotony and cumulative transitivity is invertible). We state the five

converses for classical consequence $\models$ and verify their correctness. In the last three verifications, when we say "essentially because so-and-so" we mean that it is easily checked using both so-and-so and the Tarski conditions.

CP: Its converse is the rule $A \models \beta \rightarrow \gamma \,/\, A, \beta \models \gamma$. Verification: suppose $A, \beta \not\models \gamma$. Then there is a valuation v with $v(A) = v(\beta) = 1$ and $v(\gamma) = 0$ so $v(\beta \rightarrow \gamma) = 0$ hence $A \not\models \beta \rightarrow \gamma$.

Reductio: Its converse is $A \models \beta \;/\; A, \neg\beta \models \gamma \wedge \neg\gamma$. Verification: suppose $A, \neg\beta \not\models \gamma \wedge \neg\gamma$. Then there is a valuation v with $v(A) = v(\neg\beta) = 1$ so $v(\beta) = 0$ so $A \not\models \beta$.

$\forall^+$: Its converse is $A \models \forall x(\alpha) \,/\, A \models \alpha$. It is classically correct, irrespective of whether x occurs free in A, essentially because $\forall x(\alpha) \models \alpha$ is an instance of $\forall^-$ using the identity substitution (which automatically satisfies the $\forall^-$ proviso).

$\exists^-$: Its converse is $A, \exists x(\alpha) \models \beta \;/\; A, \alpha \models \beta$. It is classically correct, irrespective of whether x occurs free in A or in β, essentially because $\alpha \models \exists x(\alpha)$ is an instance of $\exists^+$ using the identity substitution (which automatically satisfies the $\exists^+$ proviso).

DP: Its converse is the rule-pair $A, \beta_1 \vee \beta_2 \models \gamma \,/\, A, \beta_1 \models \gamma$ and $A, \beta_1 \vee \beta_2 \models \gamma \,/\, A, \beta_2 \models \gamma$. Both are classically correct essentially because $\beta_1 \models \beta_1 \vee \beta_2$ and $\beta_2 \models \beta_1 \vee \beta_2$.

10.6 Selected Reading

The concept of a consequence operation/relation is explained in a Wikipedia entry at http://en.wikipedia.org/wiki/Consequence_operators; this presentation highlights algebraic and topological aspects, with lots of useful links. A succinct account can also be found in chapter 1 of Ryszard Wójcicki *Theory of Logical Calculi: Basic Theory of Consequence Operations,* Reidel 1988. For an examination of the extent to which the Tarski conditions succeed or fail in various logics of uncertain inference, both qualitative and probabilistic, see David Makinson *Bridges from Classical to Nonmonotonic Logic*, College Publications 2007.

Informal proof-strategies in mathematics are illustrated very clearly in the textbooks E.D. Bloch *Proofs and Fundamentals: A First Course in Abstract Mathematics*, Springer 2011 (second edition), chapter 2 and Daniel Velleman *How to Prove It: A Structured Approach*, Cambridge University Press 2006 (second edition), chapter 3. Section 10.2.6 of the Bloch text also contains useful advice on writing proofs in coherent and elegant English; in that connection, it is well worth reading the celebrated article of Paul Halmos 'How to write mathematics' in *L'Enseignement Mathématique* 16: 1970, now available on several websites including https://www.math.uh.edu/tomforde/Books/Halmos-How-To-Write.pdf.

For a first glimpse of the world of proof theory, see Jan von Plato 'The development of proof theory', *Stanford Encyclopedia of Philosophy* http://plato.stanford.edu/entries/proof-theory-development; this perspective is developed at length in Sara Negri & Jan von Plato *Structural Proof Theory*, Cambridge University Press 2008. A recent concise text is Hiroakira Ono *Proof Theory and Algebra in Logic*,

Springer 2019. For more on intelim rules in classical logic see David Makinson 'Intelim rules for classical connectives' via the author's webpage https://sites.google.com/site/davidcmakinson/.

This chapter has not sought to drill readers in building extended 'natural deductions' in formal notation; the reasons are explained in the Preface. Those wishing to cultivate the skill can find a classic presentation in Patrick Suppes *Introduction to Logic*, republished in 2003 in the series Dover Books in Mathematics. More recent presentations include Nicholas Smith *Logic: The Laws of Truth*, Princeton University Press 2012 and the third edition of Peter Smith *An Introduction to Formal Logic*, Cambridge University Press, expected 2020. These are only three of the great many textbook presentations with a wide variety of notations and variant procedures. Instructors may find interest in an overview of the diversity in Jeffry Pelletier & Allen Hazen 'Natural deduction', chapter 7 of Dov Gabbay & John Woods *Handbook of the History of Logic, Vol. 11: Central Concepts*, North Holland 2012.

Sticking to the Point: Relevance in Logic

11

Chapter Outline

When presenting classical propositional logic in Chap. 8, we observed that the connective of material implication, defined by its truth-table, makes no claim of any kind of 'relevance' between antecedent and consequent; nor does the relation of tautological implication require that premises and conclusion share any content. This leads to some rather surprising results, notably the so-called principles of 'explosion'. In this chapter, we explore a way of excluding such runaway principles while retaining a logic that is smoothly behaved and easily applied. The basic idea is to adapt the method of truth-trees, from Chap. 8 Sect. 8.5, by imposing syntactic constraints of 'actual use' of antecedents and consequents in the production of crash-pairs. Readers are encouraged to review Sect. 8.5 to come up to speed for the present chapter.

11.1 Some Curious Classical Principles

As usual, it is best to begin with some examples. Among tautologies of the form $\alpha \rightarrow \beta$, there are some where α appears to 'have nothing to do with' β. The simplest are the formulae $(p \wedge \neg p) \rightarrow q$, $p \rightarrow (q \vee \neg q)$, $(p \wedge \neg p) \rightarrow (q \vee \neg q)$ whose antecedents and consequents share no sentence letters, so that propositions instantiating them may be utterly unrelated to each other in content. The first of the three is widely known as *explosion* and, by extension, we will refer to the three respectively as *right*, *left* and *symmetric explosion*. Like all arrow formulae, their status as tautologies is echoed in corresponding classical logical consequences, namely $p \wedge \neg p \vDash q$, $p \vDash q \vee \neg q$, $p \wedge \neg p \vDash q \vee \neg q$, also called principles of explosion. More generally, when either α is a contradiction or β is a tautology, then $\alpha \vDash \beta$ holds and $\alpha \rightarrow \beta$ is a tautology, even when antecedent and consequent share no letters.

D. Makinson, *Sets, Logic and Maths for Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-42218-9_11

Among the tautologies, there are also arrow formulae whose antecedents and consequents do share letters but where, intuitively, relevance still seems to fail. For example, the formulae $\neg q \rightarrow (q \rightarrow p)$, $p \rightarrow (q \rightarrow p)$, $(\neg q \vee p) \rightarrow (q \rightarrow p)$ all share at least one letter between antecedent and consequent but, intuitively, they are dubious as candidates for logical truth when the arrow is read as expressing, in the object language, some kind of relevant implication. The same doubts arise for the corresponding tautological consequences $\neg q \vDash q \rightarrow p$, $p \vDash q \rightarrow p$, $\neg q \vee p \vDash q \rightarrow p$. For instance, in the second of the three, the mere truth of p hardly seems to guarantee that an arbitrary q relevantly implies p.

Finally, there are tautologies in which the principal connective is not arrow so that the letter-sharing requirement does not arise, that are difficult to swallow under a reading of their internal arrows as claiming relevance; a well-known example is the 'comparability' tautology $(p \rightarrow q) \vee (q \rightarrow p)$.

It is thus natural to wonder whether one can construct a logic that avoids these and any other principles that appear counter-intuitive when the arrow is understood as expressing some kind of relevance between antecedent and consequent, while remaining as close to classical propositional logic as is compatible with such a goal. That is the project of *relevance logic*, also known in Australian dialect as *relevant logic*. The trouble is that the counter-intuitive principles are interdependent with others that, at first sight, are quite innocuous. The point can be illustrated starkly for right explosion by what is known as the *Lewis derivation* or *Lewis dilemma*, named after the twentieth century logician C.I. Lewis, although it is known to go back at least as far as Alexander Neckam at the end of the twelfth century. It is most transparently formulated as a short 'natural deduction' from premise $p \wedge \neg p$ to conclusion q (Neckam chose 'Socrates is a man' for p and 'Socrates is a stone' for q). The names and acronyms for rules should be familiar from Chap. 10, which readers are encouraged to consult again whenever needed.

1. $p \wedge \neg p$	Supposition
2. p	From 1 by $\wedge^-$
3. $\neg p$	From 1 by $\wedge^-$
4. $p \vee q$	From 2 by $\vee^+$
5. q	From 3,4 by Disjunctive Syllogism (DS)

Clearly, if we are to reject the inference from $p \wedge \neg p$ to q, something here must go! We must abandon or at least restrict at least one of the first-level rules $\wedge^-$ (conjunction elimination, simplification), $\vee^+$ (disjunction introduction, addition), DS (disjunctive syllogism) or the second-level rules of cumulative transitivity and monotony which, as we saw in Chap. 10, are implicit in all derivations. All these paring options have received attention in the literature on relevance logic, each with its own peculiarities and, one might say, eccentricities.

Most people seeing the Lewis dilemma for the first time tend to regard disjunction introduction, applied here to pass from p to $p \vee q$, as the principle to be blamed; the arbitrarily chosen disjunct q looks like a Trojan horse for smuggling irrelevant material into its conclusion. On the other hand, it may be suggested that

the problem is not so much the *introduction of irrelevancies* as the subsequent *elimination of whatever material is still relevant to the starting point*, for which DS is responsible. On balance, one might say that blaming $\vee^+$ for the Lewis dilemma is intuitively more convincing but turns out to lead to less interesting logical systems than does attributing it to DS.

Whichever path is chosen, inferential practice takes a serious hit. You apply $\vee^+$ when, seeing that your watch is in the top drawer you conclude that it is in the chest of drawers; also, in mathematics, when you treat a set as either finite or countable having shown that it is countable. Disjunctive syllogism is also applied in many situations. In Stoic antiquity it was famous as 'the dog', for it was maintained that even a canine can reason in accord with it. When following a scent down a road and coming to a fork, the animal may sniff some distance down one branch finding nothing then, presuming that the quarry must have gone down one of the two, race down the other path without even bothering to sniff. Such behaviour bears all the signs of performing an inference from two premises $p \vee q$, $\neg p$ to the conclusion q, even though the agent is unable to articulate what it is doing. You behave similarly when, knowing that your watch is in the chest of drawers, and finding that it is not in the upper one, you conclude that it must be in one of the others. Readers who play sudoku will recognize that disjunctive syllogism is one of the most pervasive kinds of inference used in the game. Finally, having shown that a certain set is either infinite or empty, and checking that it is not empty, a mathematician will without hesitation conclude that it is infinite.

The approach followed in this chapter blocks DS while allowing $\vee^+$ to work unhindered. Readers may continue in this chapter with some scepticism but it is hoped that they will find the material interesting, illuminating, and even fun. The author has no metaphysical agenda: his purpose is not to reject classical logic as incorrect, as those working in relevance logic have sometimes done, but to explain one way of monitoring its application to block certain surprising inferences. It is logic at play rather than logic with a reformatory mission.

11.2 A Bit of History

Historically, formal work on relevance-sensitive logics first got under way in the middle of the last century. Initially, it was dominated by the construction and study of what are called *Fregean* (or *Hilbertian*) axiom systems. They are defined recursively as the closure of a set of formulae (called the *axioms* of the system, usually presented as all instances of certain axiom schemes, optimally few in number and intuitively well-motivated) under certain rules (called *derivation rules*; formulae in the closure are called the *theorems* of the system.

For classical logic, one of the most popular Fregean axiomatizations works with a language using only the connectives $\neg$, $\rightarrow$, which we know to be functionally complete (Sect. 8.3.3 of Chap. 8). A marvel of compact elegance, it takes as axioms all formulae that are instances of any of the three schemes $\alpha \rightarrow (\beta \rightarrow \alpha)$, $(\alpha \rightarrow (\beta \rightarrow$

$\gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$, $(\neg\alpha \rightarrow \neg\beta) \rightarrow ((\neg\alpha \rightarrow \beta) \rightarrow \alpha)$ and it has a single derivation rule, which in this context is called *detachment*, telling us that whenever α, $\alpha \rightarrow \beta$ are theorems then β is a theorem. One of the basic results in the mathematical theory of classical logic, dating from the early twentieth century, is the *soundness-and-completeness theorem*: there are axiom systems (such as the one just described) whose set of theorems coincides with the set of tautologies (in the language of the system).

Nothing succeeds like success and so, for relevance logic, the idea was to construct a similar Fregean axiom system for a language containing a connective $\rightarrow$, now read as 'relevant implication', alongside the truth-functional connectives $\neg$, $\wedge$, $\vee$ (from which, as we saw in Chap. 8, one may easily define material implication). This project was marked by some success and, among those active in the field, a certain level of consensus grew around the axiom system R, which accepts all the principles that are involved in the Lewis derivation except for disjunctive syllogism.

For reference, we note that R has two derivation rules. One is detachment with respect to relevant implication, α, $\alpha \rightarrow \beta$ / β, echoing detachment with respect to material implication in the axiomatization of classical logic. The other is a rule which, in this context, is known as adjunction: α, β / $\alpha \wedge \beta$. It would be redundant if added to the usual axiomatizations of classical propositional logic, but it is not redundant in R. There are about a dozen axiom schemes, the exact number depending on how we count certain paired schemes. The following choice of axioms is quite standard for the system, as are their names, but we list them in an order that passes, very roughly, from those that are more transparent to others that are less so. The terminology 'introduction' and 'elimination' for some of these axioms reflects that for corresponding rules explained in Chap. 10.

$\alpha \rightarrow \alpha$	Identity
$\neg\neg\alpha \rightarrow \alpha$	Double Negation Elimination
$(\alpha \rightarrow \neg\beta) \rightarrow (\beta \rightarrow \neg\alpha)$	One form of Contraposition
$(\alpha \wedge \beta) \rightarrow \alpha$ and $(\alpha \wedge \beta) \rightarrow \beta$	$\wedge$-Elimination
$\alpha \rightarrow (\alpha \vee \beta)$ and $\beta \rightarrow (\alpha \vee \beta)$	$\vee$-Introduction
$((\alpha \rightarrow \beta) \wedge (\alpha \rightarrow \gamma)) \rightarrow (\alpha \rightarrow (\beta \wedge \gamma))$	$\wedge$-Introduction
$((\alpha \rightarrow \gamma) \wedge (\beta \rightarrow \gamma)) \rightarrow ((\alpha \vee \beta) \rightarrow \gamma)$	$\vee$-Elimination
$(\alpha \wedge (\beta \vee \gamma)) \rightarrow ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	Distribution
$(\alpha \rightarrow \beta) \rightarrow ((\beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \gamma))$	Suffixing
$(\alpha \rightarrow (\alpha \rightarrow \beta)) \rightarrow (\alpha \rightarrow \beta)$	Contraction
$\alpha \rightarrow ((\alpha \rightarrow \beta) \rightarrow \beta)$	Assertion

All these axioms become classical tautologies when $\rightarrow$ is read as material implication. Since each of the two derivation rules preserves the property of being a tautology it follows that, under that reading, all theorems of the system are tautologies (but not conversely).

The last of the axioms of R deserves special mention. It illustrates the difference between requiring merely the *relevance* of antecedent to consequent in an arrow formula, and requiring (alone or in addition) that there is of some kind of *necessity* in the connection between the two. They are not the same. On the one hand, the explosion formulae $(p \wedge \neg p) \rightarrow q$ and $p \rightarrow (q \vee \neg q)$ with the arrow read as material implication are logically necessary since they are tautologies but there is no relevance between antecedent and consequent. On the other hand, the 'assertion' scheme $\alpha \rightarrow ((\alpha \rightarrow \beta) \rightarrow \beta)$, last in the above list, is plausible when relevance alone is in question (and turns out to be acceptable on the analysis developed in this chapter), but is not so when the arrow is read as some kind of necessary implication. Another example of the same phenomenon is the formula $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow (\beta \rightarrow (\alpha \rightarrow \gamma))$, known as *permutation* or *exchange*, which is derivable in R. The two are intimately related; indeed given permutation, assertion becomes equivalent to $(\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \beta)$, which is just an instance of $\alpha \rightarrow \alpha$.

There is a broad consensus on how to build logics with a connective that can be read as expressing necessity; they are called *modal logics* (references can be found in the guide to further reading). One might wish to inject both relevance and necessity into the arrow connective. In the relevance logic literature, the systems called E (for entailment) and NR (for necessity with relevance) seek to do just that, rejecting both permutation and assertion. However, in the author's view, it is better to obtain a clear view of each of the two non-classical ingredients—modality and relevance—considered separately before attempting to combine them. Accordingly, in this chapter, we consider only the requirement of relevance. Roughly speaking, the goal is to examine a 'relevance-sensitive counterpart of material implication'.

The list of axioms of R is quite long, its elements disparate and there is no immediately obvious rationale for how, exactly, the line is drawn between what deserves to go in and what should be kept out. One naturally hankers for a semantic approach analogous to the valuation procedure for classical propositional logic, with respect to which the soundness and completeness of this axiom system (or another close to it) may be established.

Such an approach has in fact been devised; it is known as the *Routley-Meyer semantics* for relevance logic and a soundness-and-completeness theorem has been proven linking it with the above axiom system. For readers familiar with the Kripke semantics for modal logic (not discussed in this text) it may be helpful to say that Routley-Meyer abstracts on Kripke by replacing his two-place relations by three-place ones, between 'states' at which formulae may be counted as true or false under a given valuation. To calculate the truth-value of a formula $\alpha \rightarrow \beta$ at a given state, one needs to know the values of its antecedent and consequent, not only at that state but also at others specifiable in terms of the three-place relation. In addition, the truth-value of a formula $\neg\alpha$ at a given state is determined by that of α in another state that is selected by an auxiliary one-place function commonly known as a 'star function'. The approach is very flexible and can be adapted to yield a wide variety of logics. In particular, with suitable constraints on the three-place relation and the star function, along with a suitable articulation of the above idea for

evaluating the truth-value of an arrow formula at a given state, one can ensure that the semantics validates all and only the theorems of R.

However, the Routley-Meyer semantics has two major shortcomings, at least in so far as its application to relevance logic is concerned. One is that the constraints required on the three-place relations turn out to be arcane and intuitively uninterpretable—unlike the transparent constraints of reflexivity, transitivity and/or symmetry that are placed on two-place relations in the Kripke semantics for basic systems of modal logic. The other is that negation is evaluated in a non-classical way, despite the fact that the only connective that was initially in question was the arrow and even though the semantics ends up validating all classical tautologies in the connectives $\neg$, $\wedge$, $\vee$ alone.

The valuation of $\neg\alpha$, as true at a given state s iff α is false in state $*(s)$, has the effect of allowing both α, $\neg\alpha$ to be true at the same state. While this may be seen as no more than a technical device to make Routley-Meyer models output a desired set of formulae it has also had the unfortunate effect of encouraging flights of fancy and (to put it bluntly) mystification in the shape of a philosophy of *dialetheism* that hails the existence of true contradictions in logic, mathematics, thought and nature. In this chapter we have no need for such hypotheses.

A quite different way of articulating relevance logic is to take a classical system of *natural deduction* (see Chap. 10 especially Sect. 10.3.3) and restrict the rule of conditional proof (CP) to enforce a requirement of relevance. Recall that classical CP in, say, its second-level presentation, has the form $A, \beta \models \gamma / A \models \beta \rightarrow \gamma$. For relevance-sensitive natural deduction, it is required that the subordinate inference A, $\beta \models \gamma$ be carried out in a way that, in a certain syntactically defined sense, *actually uses* the supposition β.

The idea is intuitively natural and, again, may be elaborated in a manner that generates the same output as the axiom system R. The shortcomings are in the details of its articulation. While negation is not disturbed, as it is in the Routley-Meyer semantics, it turns out that inelegant and apparently ad hoc restrictions need to be placed on classical natural deduction rules for conjunction and disjunction (specifically on $\wedge^{+}$, $\vee^{-}$) in order to block 'cheating' manoeuvres that would otherwise get around the restriction on conditional proof and bring us back to classical logic for the arrow.

We do not go further into the quite labyrinthine details of these three approaches —Fregean axiomatization, Routley-Meyer style semantics and constrained natural deduction (there are pointers to good expositions at the end of the chapter). Instead, we adapt the method of semantic decomposition trees (truth-trees), as set out in Sect. 8.5 of Chap. 8. By suitably monitoring the classical decomposition rule for 0: $\alpha \rightarrow \beta$ and modifying that for 1: $\alpha \rightarrow \beta$, while leaving unchanged the rules for all the other connectives, one can generate an interesting logic for relevance-sensitive implication that turns out to be a little stronger than R.

11.3 Analyses of some Truth-Trees

In this section, we examine the classical truth-trees of some of the 'undesirable' tautologies mentioned in Sect. 11.1 and analyse what is going on in them. It is convenient to begin with right explosion and then look at disjunctive syllogism and comparability. Their trees will, for the duration of this section only, be constructed using the decomposition rules for classical logic that are summarized in Tables 8.12 and 8.13 of Chap. 8. The rule decomposing 0: $\alpha \rightarrow \beta$ to 1: α, 0: β on the same branch is called *counter-case*: its output nodes, labelled 1: α, 0: β, are called *critical nodes* and are *partners* of each other. They will play a central role in our analysis.

Although counter-case is the very same rule that was used for classical logic in Table 8.12 of Chap. 8, its status is rather different in the present context. Whereas for truth-functional arrow, critical nodes are *consequences* of their negated arrow formula, that is not the case for relevant arrow where the falsehood of $\alpha \rightarrow \beta$ can have two sources: the truth of α accompanied by the falsehood of β, or the lack of relevance between α and β. In the present context, passage from 0: $\alpha \rightarrow \beta$ to 1: α, 0: β serves as a *procedural* rather than an inferential step, with the critical nodes serving as *wlog suppositions* (see the logic box in Sect. 7.6.2 of Chap. 7, and further examples later in the same chapter). In other words, to obtain a crash pair from 0: $\alpha \rightarrow \beta$, it suffices to do so from the pair 1: α, 0: β.

Figure 11.1 gives the tree for right explosion. To reduce clutter, links are omitted in this and following figures when there is no branching. The annotations 'dead' and 'alive' on each branch are left for the reader to fill in if desired, likewise for the ticks signalling that decompositions have actually been carried out. For rapid visual inspection, however, critical nodes are written as $\odot$, with • for non-critical nodes; they will play a vital role in what follows.

The crash-pair is {1: p, 0: p}, Consider the two critical nodes obtained by applying the counter-case rule to the root 0: $(p \wedge \neg p) \rightarrow q$. One of them, 1: $p \wedge \neg p$, is used in getting the crash-pair but its partner 0: q is not. Using the same basic idea as is applied to conditional proof in relevant natural deduction (briefly described in Sect. 11.2), we will require that *if one critical node is used in obtaining the crash-pair then so is its partner*. We call this requirement *parity* and will define it rigorously in the following section. It sounds straightforward, but we need to be careful, since not all examples are as simple as right explosion! A truth-tree may

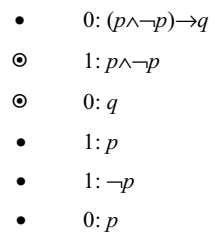

Fig. 11.1 Truth-tree for right explosion

have multiple branches and on any given branch there may be multiple critical pairs, also multiple crash-pairs—we will see examples of all of these as we go along. So, a rigorous definition of parity will need to juggle adroitly with branches, critical pairs and crash-pairs.

Alice Box: Nodes and labels

Alice Before we go any further, I have a question. Figure 11.1 represents nodes anonymously as points, written ⊙ or • according as they are or aren't critical nodes. But shouldn't they be given names to distinguish them from each other?

Hatter In Fig. 11.1, that is done by the labels, made up of a formula and a truth-value.

Alice Is that sufficient? After all, it might happen that we may have two distinct nodes bearing the same label at different places in a truth-tree so, if we refer to the node by the label alone, we will not have identified it fully.

Hatter Quite so. Putting your point briefly in the language of set theory: in some trees the labelling function is not injective. When that happens, as it will in a few of our examples, we will indeed need to give the nodes in question explicit names. But when it doesn't happen, as in Fig. 11.1, we can safely use the labels alone to refer to the nodes.

Alice And when it does happen, do we name just the nodes involved in the failure of injectivity, or all the nodes of the tree?

Hatter Both conventions are feasible but, in the interests of uniformity and visual harmony, I would suggest that when you need to name one node, you name them all

Let's now look at disjunctive syllogism, whose truth-tree in Fig. 11.2 has two branches. Given the fact that we are 'blaming' it for the Lewis derivation of right explosion, its truth- tree will be of particular interest. As there is more than one crash-pair in the tree, we use arrow annotations to signal them visually. If you prefer to signal crash-pair nodes by writing a sign such as $\perp$ next to them, feel free to do so but, to be clear about what clashes with what, you will need a bit more detail. In Fig. 11.2 you would need to write $\perp_1$ at nodes 5 and 7 of the left branch and $\perp_2$ at nodes 3 and 7 of the right branch.

Both branches of the tree crash, with crash-pair $\{0\colon p, 1\colon p\}$ on the left and $\{0\colon q, 1\colon q\}$ on the right. But the left branch does not crash with parity, since the critical node $1\colon (p \vee q) \wedge \neg p$ is used in getting the unique crash-pair on that branch while its partner $0\colon q$ is not. The lesson is that an adequate definition of parity will need to consider *all branches*. Lest it be thought that this lesson is peculiar to disjunctive syllogism, the same consideration arises for many other formulae that are close to right explosion, for example $((p \vee (q \wedge \neg q)) \rightarrow p$.

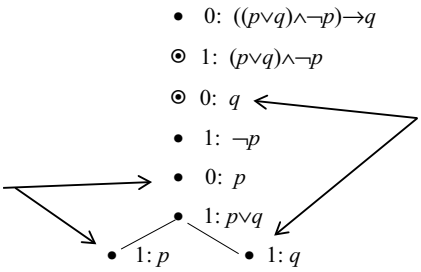

Fig. 11.2 Truth-tree for disjunctive syllogism

Alice Box: Disjunctive syllogism versus modus ponens

Alice I think that you have a difficulty here! In classical logic, disjunctive syllogism is essentially the same as modus ponens: the truth tables for $\vee$, $\neg$ and $\rightarrow$ make $((p \vee q) \wedge \neg p) \rightarrow q$ semantically indistinguishable from $((\neg p \rightarrow q) \wedge \neg p) \rightarrow q$. So, if the parity constraint knocks out the former, surely it will knock out the latter which, presumably, we do not want to do!

Hatter Indeed it would, if positive arrow nodes $1{:}\ \alpha \rightarrow \beta$ were decomposed by a forking rule as is customary for the classical arrow. Principles like contraposition would also be lost, as we will see in an exercise. But that is not the only way to decompose $1{:}\ \alpha \rightarrow \beta$. In the next section you will see how the forking rule for such nodes may be replaced by a non-forking one that leaves modus ponens, contraposition and their friends unscathed.

Alice I hope you keep your promise …

We consider two more examples of the decomposition of negative arrows $0{:}\ \alpha \rightarrow \beta$, focusing on the role of the parity condition. The tautology $(p \rightarrow q) \vee (q \ p)$, sometimes known as 'comparability', appears to be undesirable for relevant arrow. Its truth-tree in Fig. 11.3 has only one branch, but two crash-pairs and two pairs of critical nodes. The critical pairs are linked by crescents to highlight which node is a partner of which.

A peculiarity of this tree is that while each critical node is used in getting some crash-pair (namely, the crash-pair of which it is an element), there is no crash-pair such that, whenever a critical node n is used in getting it, then so is n's partner. Consider, for example, the first of the two crash-pairs, $\{1{:}\ p, 0{:}\ p\}$. Its two elements are themselves critical nodes, but they are not partners as they emanate from distinct applications of counter-case, as flagged by the crescents. Now, a node in a truth-tree is always the last item in the path of nodes used to reach it, so it is natural to understand the notion of dependency in such a way that each node depends on itself

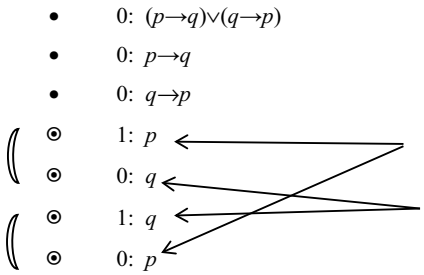

Fig. 11.3 Truth-tree for comparability

as well as on the preceding nodes in its path. With this understanding, the crash-pair $\{1{:}\ p, 0{:}\ p\}$ depends on the critical node $1{:}\ p$, but does not depend on its partner $0{:}\ q$. Similar considerations apply to the other crash-pair $\{0{:}\ q, 1{:}\ q\}$.

This example has two morals. One is that *when there is more than one crash-pair or more than one pair of critical nodes on a branch, we need to be careful about the order of quantifiers over them in our definition of parity*. The other is that *we need to be rigorous in our definition of dependence in a decomposition tree*.

Essentially the same considerations arise in the truth-tree for the tautology $p \to (p \to p)$, known as *mingle*. Its antecedent and consequent share a letter, but it is not generally accepted by relevance logicians (and is not derivable in the axiomatic system R). Its tree is given in Fig. 11.4. Two of the nodes have the same label $1{:}\ p$ so, following the Hatter's recommendation, we give them and all the other nodes explicit names.

Note that while n_1, n_3 are both obtained by counter-case they issue from different nodes, namely r, n_2 respectively. There are two overlapping crash-pairs, namely $\{n_1, n_4\}$, $\{n_3, n_4\}$, as indicated by the pointers. The first crash-pair $\{n_1, n_4\}$ depends on critical node n_4 but not on its partner n_3, their partnership highlighted by a crescent. The second crash pair $\{n_3, n_4\}$ depends on critical node n_2 but not on its partner n_1.

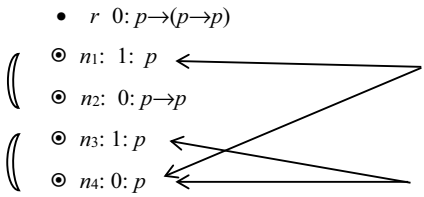

Fig. 11.4 Truth-tree for mingle

11.4 Direct Acceptability

We now articulate rigorous definitions of the ideas that were explored informally in Sect. 11.3, namely decomposition rules for positive and negative arrows as well as notions of crash-pair, critical pair, dependence, parity, and acceptability of a tree and formula.

Table 11.1 gives the rules for arrow. Just as in the classical context, negative arrows are still decomposed by counter-case without restriction on its actual application; the restriction will be a *global* one, on what can be done with the outputs in the entire tree, and will be defined in the parity condition shortly. On the other hand, positive arrows are no longer handled by implicative forking but by *modus ponens*. Eyebrows may rise at a description of modus ponens as a 'decomposition' rule, but the output is indeed a sub-formula of one of the two inputs.

For ease of reference, Table 11.2 recalls the decomposition rules for the connectives $\wedge$, $\vee$, $\neg$. They are unchanged from the classical context as described in Chap. 8 Sect. 8.5.

We recall from Chap. 8 that a *crash-pair* is a pair (1: ζ, 0: ζ) of nodes on the same branch, labelled by the same formula with opposite signs. On the other hand, a *critical pair* is a pair of nodes, labelled 1: φ, 0: ψ, that are introduced on the same branch by an application of the counter-case rule to a node labelled 0: $\varphi \rightarrow \psi$. The two *critical nodes* so introduced are said to be *partners* of each other. Note that critical nodes can sometimes themselves elements of crash-pairs; for example, when we decompose 0: $p \rightarrow p$ the two critical nodes form a crash-pair.

Dependence is defined recursively, going backwards. We say that a node x *depends* on a node n in a decomposition tree iff n is in the least set $D(x)$ of nodes such that $x \in D(x)$ and whenever $D(x)$ contains a non-root node then it also contains the node (or both nodes, for modus ponens) from which it was obtained by an application of a decomposition rule. Note that, as a limiting case, every node depends on itself and that the relation is transitive. The set $D(x)$ is referred to as *the trace* of x.

A *set of nodes* is said to depend on n iff at least one of its elements does so. In other words, writing X for any set of nodes and $D(X)$ for its trace, $D(X) = \cup\{D(x): x \in X\}$. In particular, for crash-pairs $X = \{1: \zeta, 0: \zeta\}$—which are the only sets of nodes whose dependency will interest us in what follows—we have $D(X) = D(\{1: \zeta, 0: \zeta\}) = D(1: \zeta) \cup D(0: \zeta)$.

Table 11.1 Decomposition rules for relevance-sensitive implication

Modus ponens	Counter-case
Given nodes m, m' on a branch, labelled by	Given a node m on a branch, labelled by
1: α, 1: $\alpha \rightarrow \beta$	0: $\alpha \rightarrow \beta$
we can add to that branch, without forking	
one further node n labelled by	two nodes n_1, n_2 labelled respectively by
1: β	1: α, 0: β

Table 11.2 Classical decomposition rules for $\wedge$, $\vee$, $\neg$

Given a node m on a branch, labelled by					
1: $\alpha \wedge \beta$	0: $\alpha \wedge \beta$	1: $\alpha \vee \beta$	0: $\alpha \vee \beta$	1: $\neg\alpha$	0: $\neg\alpha$
we can add to that branch					
two	two	two	two	one node n	
further nodes n_1, n_2, with the branch					
not forking	forking	forking	not forking		
where n_1, n_2 are labelled respectively by the formulae				labelled by	
1:α, 1:β	0: α, 0: β	1: α, 1: β	0: α, 0: β	0: α	1: α

A truth-tree is said to be *directly acceptable* iff for every branch B of the tree, there is a crash-pair $Z = \{(1: \zeta), (0: \zeta)\}$ on B such that for every critical node n, if Z depends on n then it also depends on the partner of n. When this condition is satisfied for a specific branch B and crash-pair Z on B, we also say briefly that B and Z *satisfy parity*.

Some readers may find this definition easier to digest if the words 'for every critical node n' are expanded to 'for every critical node n on B'; the two formulations are equivalent since whenever a node is on a branch B then so too is every node on which it depends. On the other hand, the order of the quantifiers in the definition is vital: $\forall_B \exists_Z \forall_n$ rather than in some other sequence. As often for such conditions, it can be convenient to replace the existential quantifier $\exists_Z$ by a choice function on B. So formulated, the definition requires that there is a choice function $B \mapsto Z_B$ taking each branch of the tree to a designated crash-pair lying on it, such that for every branch B and critical node n, either both n and its partner are in the trace of the designated crash-pair Z_B or neither of them are.

Finally, a *formula* φ is said to be directly acceptable iff it has some directly acceptable tree, that is, iff there is a directly acceptable truth-tree with root labelled 0: φ. A formula *scheme* is deemed directly acceptable iff all its instances are.

Inspecting Figs. 11.1 through 11.4 and the small number of trees obtainable from them by varying the order in which decomposition rules are applied, it is easy to check that none of the formulae $(p \wedge \neg p) \rightarrow q$ (right explosion), $((p \vee q) \wedge \neg p) \rightarrow q$ (disjunctive syllogism), $(p \rightarrow q) \vee (q \rightarrow p)$ (comparability), $p \rightarrow (p \rightarrow p)$ (mingle) are directly acceptable, in accord with our informal discussion in Sect. 11.3.

Alice Box: Why the existential quantier?

Alice Why the existential quantifier over crash-pairs on a branch? Why not require that for every branch B and every crash-pair Z on B, etc.?

Hatter For a branch to be dead (to use the terminology of Chap. 8) only one crash-pair is needed. Others may happen to turn up on the branch without being involved in that crash. If they fail to satisfy parity, it should not affect the situation.

Alice Can you give me an example?

Hatter Sure. Consider the contraction formula $(p \to (p \to q)) \to (p \to q)$, one of the axioms of the system R. It is easily checked to be directly acceptable, and so too is its instance $(p \to (p \to p)) \to (p \to p)$, where p is substituted for q. The tree for the latter is just a substitution instance of the tree for the former but, because of the identification of letters, it has a second crash-pair that happens to fail parity. That does not affect the damage done to the branch by the other crash-pair, which does respect parity.

The more applications of counter-case are carried out in a tree, the more care is needed in checking the traces of crash-pairs to see whether parity is satisfied. To identify the trace $D(x)$ for a given node x, do not just eyeball the tree as a whole or try to work downwards from the root. Work backwards from x by systematically applying the recursive definition of dependence. When there are at most two branches, each with its designated crash-pair, you can keep track of your work by placing ticks alongside the nodes, say on the left of them for the left branch and on the right for the right branch. You may find it helpful to use other annotations and pointers to record dependencies in more complex examples. We have not done that in this chapter as it quickly generates clutter that makes the figure annoying to read for anyone other than the person who constructed it.

Exercise 11.4.1
Check that (a) contraposition $(p \to q) \to (\neg q \to \neg p)$ is directly acceptable, but (b) it would not be so if we used implicative forking as decomposition rule for positive arrow formulae. (c) Compare the two truth-trees to explain 'geometrically' the difference of outcome. This fulfills a promise made to Alice.

Solution

(a) See Fig. 11.5. There is a unique branch with a unique crash-pair. There are two pairs of critical nodes, but parity is satisfied; indeed, every critical need is used in getting the crash-pair.

(b) See Fig. 11.6. Implicative forking applied to the second node creates two branches; we have delayed the branching as late as possible, but advancing it makes no difference to the analysis. The crash-pair in the left branch is $\{1{:}\ p, 0{:}\ p\}$ depending on critical node $0{:}\ \neg p$ but not on its partner $1{:}\ \neg q$. Similarly, the crash-pair in the right branch is $\{0{:}\ q, 1{:}\ q\}$, which depends on critical node $1{:}\ \neg q$ but not on its partner $0{:}\ \neg p$. Double trouble!

(c) The first seven nodes are the same in the two trees, but modus ponens in Fig. 11.5 allows us to omit the left branch of Fig. 11.6 and retain only the right one. Parity in that branch, which failed in Fig. 11.6, is now satisfied since $1{:}\ q$ is now obtained by modus ponens using $1{:}\ p$ as minor premise, which in turn depends on the critical node $0{:}\ \neg p$. *End of solution*.

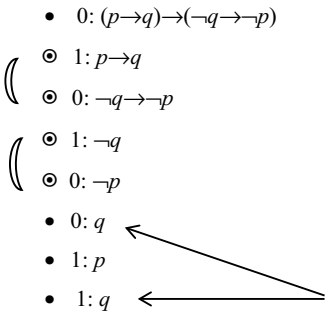

Fig. 11.5 Truth-tree for contraposition

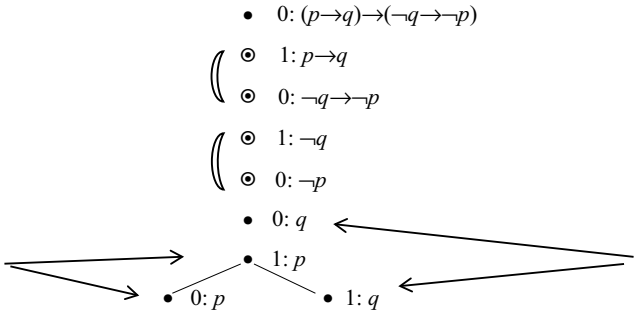

Fig. 11.6 Contraposition fails under implicative forking

Exercise 11.4.2

(a) Show that all classical tautologies in the connectives $\neg$, $\wedge$, $\vee$ are directly acceptable.
(b) Which of the axiom schemes of the system R, listed in Sect. 11.2, are directly acceptable?
(c) Which of the 'undesirable' tautologies $(\neg p \vee q) \rightarrow (p \rightarrow q)$, $(p \rightarrow (q \rightarrow p))$ are directly acceptable?
(d) What about the tempting formulae $(p \rightarrow q) \rightarrow (p \rightarrow (p \wedge q))$ and its dual $(p \rightarrow q) \rightarrow ((p \vee q) \rightarrow q)$?
(e) And the formulae $((p \rightarrow (q \rightarrow r)) \rightarrow (q \rightarrow (p \rightarrow r))$, $(p \rightarrow (q \vee (p \rightarrow q))) \rightarrow (p \rightarrow q)$?

Solution outline

(a) The truth-tree for such a formula uses only the classical decomposition rules of Table 11.2 and parity is vacuously satisfied since there are no critical nodes.
(b) All axioms of R turn out directly acceptable. The construction of their trees and inspections for parity are straightforward. When writing out the tree for the axiom of contraction, one can instantiate it to obtain a tree for $(p \to (p \to p)) \to (p \to p)$ to check out the Hatter's remark in the last Alice Box.
(c) Neither of the two formulae is directly acceptable. The first has two branches, both of which fail parity. The second has a single branch, which fails parity. Variations in the order of applying the decomposition rules do not remedy matters.
(d) Despite their initial attractiveness, these formulae are not directly acceptable. In both cases, every branch of the decomposition tree contains a crash-pair, but parity fails in one branch. For example, for the first formula, parity fails in the branch passing from 0: $p \wedge q$ to 0: p. Again, variations in order of rule-application do not repair the situation.
(e) Both formulae are directly acceptable. *End of solution*.

The first formula in Exercise 11.4.2 (e), $((p \to (q \to r)) \to (q \to (p \to r))$ is known as *permutation* and is very useful in manipulating formulae into equivalent forms. It is a theorem of R but is not generally accepted in systems like E that attempt to combine requirements of relevance and necessity in the arrow. The second formula, $(p \to (q \vee (p \to q))) \to (p \to q)$, has been called *skewed cases*. It and its one-letter substitution instance $(p \to (p \vee (p \to p))) \to (p \to p)$ are the simplest known examples of formulae that are directly acceptable but not theorems of R.

It is possible to show that direct acceptability satisfies the *letter-sharing condition*, in other words, φ shares at least one sentence letter with ψ whenever $\varphi \to \psi$ is directly acceptable. However, it also has a major limitation: it is not as broad as one would like it to be. There are formulae that appear perfectly reasonable when one reads the arrow as relevance-sensitive implication, but which are not directly acceptable. Perhaps the simplest example is $(\neg\neg p \to q) \to (p \to q)$, contrasting with its directly acceptable converse $(p \to q) \to (\neg\neg p \to q)$. Its truth-tree is given in Fig. 11.7.

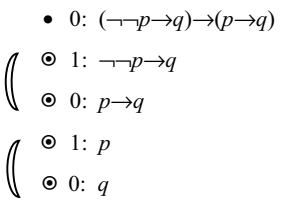

Fig. 11.7 Truth-tree for $(\neg\neg p \to q) \to (p \to q)$

In this tree, one would like to apply modus ponens to 1: $\neg\neg p \rightarrow q$ and its antecedent 1: $\neg\neg p$, but we do not have that antecedent at hand. We cannot obtain it by decomposing 1: p, since decomposition always eliminates a connective, never introduces one. We cannot apply implicative forking to 1: $\neg\neg p \rightarrow q$ since that rule has been replaced by modus ponens—and even if we retain and apply it, parity fails in both branches. Nor can we apply modus tollens to 1: $\neg\neg p \rightarrow q$ accompanied by 0: q, since modus tollens is not one of our direct decomposition rules. Direct decomposition can go no further.

We note in passing that the tree of Fig. 11.7 would still be blocked if we were not requiring parity. It thus illustrates the fact that if we were to replace implicative forking by modus ponens in the truth-trees for classical propositional logic, the method would become incomplete: not all tautologies would be validated.

Exercise 11.4.3

(a) Show that while the tautology $(p \rightarrow q) \rightarrow ((p \wedge p) \rightarrow q)$ is directly acceptable, its converse is not.
(b) Show that the tautology $\neg((p \vee \neg p) \rightarrow (q \wedge \neg q))$ is not directly acceptable.

Solution outline

(a) The truth-tree for the converse, $((p \wedge p) \rightarrow q) \rightarrow (p \rightarrow q)$, snags because we cannot go from 1: p to 1: $p \wedge p$ although we can decompose in the reverse direction.
(b) We can show, quite generally, that no negated conditional $\neg(\varphi \rightarrow \psi)$ has a directly acceptable tree. The root of such a tree is labelled 0: $\neg(\varphi \rightarrow \psi)$ and the only decomposition rule we can apply to it is one for negation, getting 1: $\varphi \rightarrow \psi$. The only rule that could possibly be applied to 1: $\varphi \rightarrow \psi$ is modus ponens, but that needs 1: φ as minor premise, which is not available in the tree thus far constructed. *End of solution*.

The negative results of Exercise 11.4.3 and Fig. 11.7 point to an intuitive difficulty with the definition of direct acceptability, since the formulae concerned seem quite agreeable when read 'relevantly'. The same results also reveal a formal weakness: that the set of directly acceptable formulae is not closed under detachment wrt the arrow. For each of the three formulae, call it ψ, we can find a formula φ such that both φ and $\varphi \rightarrow \psi$ are directly acceptable although we know that ψ is not. For example, we can check that both of φ: = $(p \rightarrow \neg\neg p)$ and $\varphi \rightarrow \psi$: = $(p \rightarrow \neg\neg p) \rightarrow ((\neg\neg p \rightarrow q) \rightarrow (p \rightarrow q))$ are directly acceptable while, as seen in Fig. 11.7, ψ: = $(\neg\neg p \rightarrow q) \rightarrow (p \rightarrow q)$ is not.

11.5 Acceptability

The question thus arises: Is there a principled way of extending our decomposition procedure to validate formulae such as those in Fig. 11.7 and Exercise 11.4.2 and, more generally, ensure closure under detachment—without losing the letter-sharing property enjoyed by direct acceptability?

We can get some mileage by adding modus tollens as a decomposition rule to accompany modus ponens, but that is not enough. We need a more radical remedy, which we articulate as a recursion with the directly acceptable trees as basis, to be closed under an appropriate operation.

Specifically, we define the notion of an *acceptable tree* as follows. *Basis*: Every directly acceptable tree is acceptable. *Recursion step*: If the formula $\varphi \rightarrow \psi$ is acceptable then, at any stage in the construction of a tree, we allow passage from a node labelled 1: φ to a node labelled 1: ψ (forwards clause) and we also allow passage from 0: ψ to 0: φ (backwards clause). A *formula* is deemed acceptable iff it has at least one acceptable decomposition tree.

Evidently, applications of the recursive step do not always decompose in a strict sense since in, the forwards clause, ψ may be more complex than φ and conversely in the backwards clause. Nevertheless we will call this a 'decomposition' step in an indirect sense of that term. Note that in the recursive rule the 'justifying' formula $\varphi \rightarrow \psi$ need not label any node of the tree. Observe also that both forward and backward clauses apply the recursive rule to a *single node*, not two, unlike the direct decomposition rule of modus ponens, which passes from *two nodes* as input.

We digress with a comment for those accustomed to working with unsigned truth-trees in classical logic (cf. the remarks at end of Chap. 8). If we use unsigned trees in the present context (as is done in the extended treatment referenced at the end of the chapter) then we can dispense with a backwards clause in the recursion step; only a forwards clause is needed. This could be seen as being a minor advantage for the unsigned option; on the other hand, the signed one seems to be easier for most students to assimilate. The choice between the two modes of presentation remains largely a matter of taste, as in classical logic.

Exercise 11.5.1
We have already noticed that although the tautologies (i) $(\neg\neg p \rightarrow q) \rightarrow (p \rightarrow q)$, (ii) $((p \wedge p) \rightarrow q) \rightarrow (p \rightarrow q)$, (iii) $\neg((p \vee \neg p) \rightarrow (q \wedge \neg q))$ are attractive under an understanding of arrow as relevant material implication, they are not directly acceptable. Show that they are acceptable.

Solution outline
In the truth-trees for these formulae we apply the recursive rule, in each instance appealing to a suitable directly acceptable formula as 'justification'. For (i) appeal to $p \rightarrow \neg\neg p$, for (ii) to $p \rightarrow (p \wedge p)$. For (iii), which we call *polarity*, appeal to $((p \vee \neg p) \rightarrow (q \wedge \neg q)) \rightarrow (\neg(p \vee \neg p) \vee (q \wedge \neg q))$. This looks rather long, but is merely an instance of the directly acceptable scheme $(\alpha \rightarrow \beta) \rightarrow (\neg\alpha \vee \beta)$. Figure 11.8 gives details for polarity.

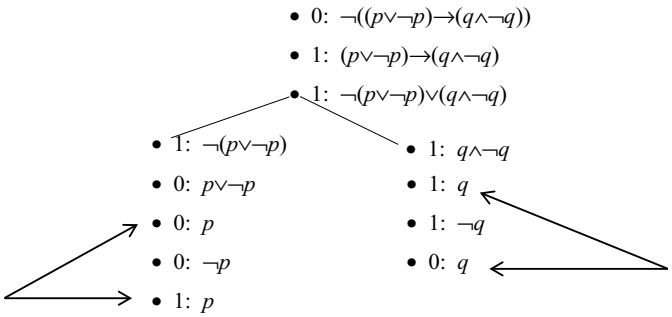

Fig. 11.8 Acceptable truth-tree for polarity

Exercise 11.5.2
Which among the three schemes for the popular Fregean axiomatization of classical logic mentioned in Sect. 11.2, are acceptable?

Solution outline
The second axiom scheme $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$ and the third one $(\neg\alpha \rightarrow \neg\beta) \rightarrow ((\neg\alpha \rightarrow \beta) \rightarrow \alpha)$ are acceptable; indeed, they are directly so. On the other hand, the first one, $\alpha \rightarrow (\beta \rightarrow \alpha)$, is not acceptable since its instance $p \rightarrow (q \rightarrow p)$ (which we call *mangle*) and even the more specific instance $p \rightarrow (p \rightarrow p)$ (*mingle*) are not acceptable. A decomposition tree for mingle was given in Fig. 11.4 of this chapter and the informal analysis there carries through unchanged under our formal definitions.

Alice Box: Negative outcomes

Alice Just a minute! I think that there is a gap in the verification that mingle is not acceptable. I see that the decomposition tree for it in Fig. 11.4 fails parity, but that tree does not even apply the recursive rule!

Hatter There are lots of ways in which we could apply the recursive rule to nodes in that tree, but I see none that gets a crash-pair with parity…

Alice Agreed. But that does not mean that no successful application is *possible*! Clearly, there are more than just 'lots' of ways of applying the recursive rule—there are infinitely many, giving rise to infinitely many candidate truth-trees for any given formula. How do we know in advance that none of them will satisfy parity, thus providing a clever or devious verification of acceptability?

Hatter Er, um …

Alice has an important point here. The failure of a specific tree to satisfy parity does not mean that there is no tree satisfying it. For *direct* acceptability, where the recursive rule is not available, we can get around the difficulty, because there is only a limited number of ways in which one can vary the order of application of the decomposition rules; the variant trees can thus be checked out one by one. But for acceptability, where the recursive rule is available and there is no limit on the number of trees that one could create, one needs a *proof that none of them are acceptable*. At the time of writing, techniques for such proof have not yet been developed; thus we have as yet *no rigorous method for establishing negative results for acceptability*.

In practice, after working with a variety of examples, one gets a feel for them and, for certain among them (such as mingle or explosion), one acquires intuitive confidence that there is no way of repairing the failure of parity that arose in the most natural tree. Nevertheless, we should acknowledge explicitly that in the absence of full proof, all negative assessments that arise from inspecting a single tree, or just a few of them, should be treated as presumptive, even if we sometimes neglect to include that qualification explicitly in what follows. In contrast, positive assessments of acceptability are fully verifiable with a single tree.

A closely related fact is that while the set of directly acceptable formulae is *computable* (decidable) in the sense that we can, in principle, write a computer program that will check out whether an arbitrarily given formula is directly acceptable, it is not clear whether the same holds for acceptability. Another way of saying the same thing, which will make sense to readers familiar with some basic notions from the theory of computability, is that while the set of acceptable formulae is *semi-computable* (recursively enumerable), it is not currently known whether its complement is also semi-computable. There is some reason to suspect that it is not so, for it is known that the set of theorems of the closely related axiomatic system R is not computable.

Exercise 11.5.3
Which of the tautologies $(\neg p \vee q) \rightarrow (p \rightarrow q)$, $(p \rightarrow q) \rightarrow (p \rightarrow (p \wedge q))$, $(p \rightarrow q) \rightarrow ((p \vee q) \rightarrow q)$, which we have already seen not to be directly acceptable, are nevertheless acceptable?

Solution outline
None of them appear to be acceptable; in each instance, attempts to use the recursive rule get nowhere. *End of solution*.

Notwithstanding the situation for negative assessments, acceptability has features that make it attractive. We list some of them, with verifications.

1. All directly acceptable formulae are acceptable.
2. The set of acceptable formulae is closed under substitution, adjunction (as conjunction is usually termed in this context) and detachment with respect to the arrow.
3. Every theorem of the axiomatic system R is acceptable.

4. The set of acceptable formulae coincides with the closure of the set of directly acceptable formulae under the decomposition rules of adjunction and detachment.

Feature (1) is immediate by the definition. By Exercise 11.4.2 (a), this also shows that all classical tautologies in the connectives $\neg$, $\wedge$, $\vee$ are acceptable. For (2), we have three closure conditions to establish. Verifying substitution, suppose that φ is acceptable and σ is a substitution function; we want to show that $\sigma(\varphi)$ is acceptable. Take an acceptable tree for φ and make the same substitution σ throughout the tree. Substitution instances of crash-pairs are still crash-pairs since $\sigma(\neg\varphi) = \neg\sigma(\varphi)$, and it is not difficult to check by induction on the construction of the tree that decomposition steps and parity are preserved.

For adjunction, suppose that φ, ψ are acceptable; we want to show that $\varphi \wedge \psi$ is acceptable. Construct a tree with root 0: $\varphi \wedge \psi$, decompose to 0: φ, 0: ψ on separate branches, copy-paste acceptable trees for φ, ψ. Clearly, parity is preserved.

For detachment, suppose φ and $\varphi \rightarrow \psi$ are both acceptable; we want to construct an acceptable truth-tree for ψ. Put 0: ψ as root, apply the 'backwards' part of the recursive rule using the acceptability of $\varphi \rightarrow \psi$ as justification, to get 0: φ, then paste in an acceptable tree for φ. Parity is preserved.

Since the axioms of system R are directly acceptable, as noted in Exercise 11.4.2 (b), point (3) is immediate from points (1), (2).

For (4), the inclusion RHS $\subseteq$ LHS is already given by properties (1) and (2). The verification of the converse inclusion is rather more complex and will not be given here (it can be found in the extended treatment referenced at the end of the chapter).

The operation of detachment mentioned in the points above should not be confused with that of modus ponens, nor with the forwards part of the recursive rule as these are defined in the chapter. We emphasized the distinction between the latter two when defining acceptability at the beginning of this section; this is a good moment to compare all three explicitly.

- *Modus ponens*, as here understood, is a rule that is applied *within* trees. It says that we may pass from a nodes labelled 1: $\varphi \rightarrow \psi$, 1: φ on a branch to a node labelled 1: ψ, irrespective of whether the formula $\varphi \rightarrow \psi$ is acceptable. The rule is legitimate for acceptable as well as directly acceptable trees.
- *The forwards clause of the recursive rule*, as defined in this section, is also applied *within* trees. It tells us that we may pass from a node labelled 1: φ on a branch to a node labelled 1: ψ on the same branch whenever the formula $\varphi \rightarrow \psi$ is acceptable, irrespective of whether any node on the branch is labelled 1: $\varphi \rightarrow \psi$. It is legitimate for acceptable trees, but not for directly acceptable ones.
- *Detachment*, in the present context, is not a rule for constructing trees. It is a closure condition that holds for the set of all formulae that have some acceptable tree. It does not hold for the base set of directly acceptable formulae.

Exercise 11.5.4

(a) Show that whenever $\neg\psi$ and $\varphi \rightarrow \psi$ are both acceptable, so too is $\neg\varphi$.
(b) Which of the following formulae are acceptable: (i) $((p \rightarrow q) \wedge (p \rightarrow \neg q)) \rightarrow (p \rightarrow (q \wedge \neg q))$, (ii) $(p \rightarrow q) \rightarrow ((p \rightarrow \neg q) \rightarrow (p \rightarrow (q \wedge \neg q)))$, (iii) $((p \rightarrow q) \wedge (p \rightarrow \neg q)) \rightarrow \neg p$, (iv) $(p \rightarrow q) \rightarrow ((p \rightarrow \neg q) \rightarrow \neg p)$?

Solution

(a) Suppose $\neg\psi$ and $\varphi \rightarrow \psi$ are both acceptable. Now the contraposition formula $(\varphi \rightarrow \psi) \rightarrow (\neg\psi \rightarrow \neg\varphi)$ is easily checked to be directly acceptable and so acceptable. Since acceptability is closed under detachment, $\neg\psi \rightarrow \neg\varphi$ is acceptable and so, by detachment again, $\neg\varphi$ is acceptable.
(b) (i) Acceptable, in fact directly so. It is, indeed, an instance of $\wedge$ -introduction, one of the axiom schemes of R all of which are directly acceptable as we saw in Exercise 11.4.2 (b). (ii) Not directly acceptable and presumably not acceptable. Comparing (i) with (ii) we see the important difference, in relevance logic, between the classically equivalent schemes $(\varphi \wedge \psi) \rightarrow \chi$ and $\varphi \rightarrow (\psi \rightarrow \chi)$. (iii) Acceptable, in fact directly so. (iv) Acceptable, in fact directly so. *End of solution*.

After having seen the failure of (ii) in Exercise 11.5.4 (b), one might be rather surprised at the positive status of (iv). Intuitively, one can understand the latter outcome by noting that, modulo a contraposition of $p \rightarrow \neg q$, it says the same as $(p \rightarrow q) \rightarrow ((q \rightarrow \neg p) \rightarrow \neg p)$, and that the latter follows naturally from $(p \rightarrow q) \rightarrow ((q \rightarrow \neg p) \rightarrow (p \rightarrow \neg p))$, which is an instance of the transitivity scheme (one of the axioms of R) together with the directly acceptable $(p \rightarrow \neg p) \rightarrow \neg p$.

Exercise 11.5.5

While modus ponens is one of the decomposition rules, *modus tollens* is not. In other words, we do not have a decomposition rule that authorizes passage from nodes labelled 1: $\varphi \rightarrow \psi$ and 0: ψ on a common branch to 0: φ on that same branch. Show that it is, however, *admissible for acceptability* in the sense that in acceptable trees, its application merely shortens a longer decomposition without it.

Solution

Let nodes labelled 1: $\varphi \rightarrow \psi$ and 0: ψ lie on a branch of a decomposition tree. We want to show that a node labelled 0: φ may legitimately be added to that branch. Now, an application of the forwards clause of the recursive rule, justified by the fact that $(\varphi \rightarrow \psi) \rightarrow (\neg\psi \rightarrow \neg\varphi)$ is directly acceptable, allows us to pass from 1: $\varphi \rightarrow \psi$ to 1: $\neg\psi \rightarrow \neg\varphi$. Moreover, the backwards clause of the recursive rule authorizes us to pass from 0: ψ to 0: $\neg\neg\psi$, justified by the fact that $\neg\neg\psi \rightarrow \psi$ is directly acceptable. From 0: $\neg\neg\psi$ we decompose to 1: $\neg\psi$, perform modus ponens on that in association with 1: $\neg\psi \rightarrow \neg\varphi$ to get 1: $\neg\varphi$, and finally decompose to 0: φ. *End of solution*.

Modus tollens can be a very useful short-cut in the construction of acceptable trees, as in the next exercise. Note, however, that the proof of admissibility in Exercise 11.5.5 appealed to both the forwards and backwards clauses of the recursive rule; the short-cut is not available for direct acceptability.

Exercise 11.5.6
Show that every formula is acceptably equivalent to some arrow formula. In other words, for every formula α (in the language of $\wedge$, $\vee$, $\neg$, $\rightarrow$) there is a formula β (in the same language) with principal connective arrow, such that both $\alpha \rightarrow \beta$ and $\beta \rightarrow \alpha$ are acceptable.

Solution outline
Put $\beta = (\alpha \rightarrow \alpha) \rightarrow \alpha$. Then it is easy to check that $\alpha \rightarrow \beta$ is directly acceptable. Using the fact that modus tollens is admissible as a decomposition rule for acceptability (Exercise 11.5.5), we can construct an acceptable tree for $\beta \rightarrow \alpha$ with a single branch of just six nodes. *End of solution*.

Exercise 11.5.7
Defining $\leftrightarrow$ in the usual way as the conjunction of arrows in both directions, show that $(\neg(p \rightarrow q) \rightarrow r) \leftrightarrow (p \rightarrow (\neg q \rightarrow r))$ is acceptable.

Solution outline
Since acceptability is closed under adjunction, it suffices to show that each of the two directions is acceptable. For $(\neg(p \rightarrow q) \rightarrow r) \rightarrow (p \rightarrow (\neg q \rightarrow r))$, see Fig. 11.9 which uses the admissible decomposition step of modus tollens. For the converse formula you are on your own.

We end the chapter by stating *a major open problem*. We know that direct acceptability satisfies the letter-sharing condition (Sect. 11.4). But does acceptability continue to satisfy it? In other words, is it the case that φ, ψ share at least one

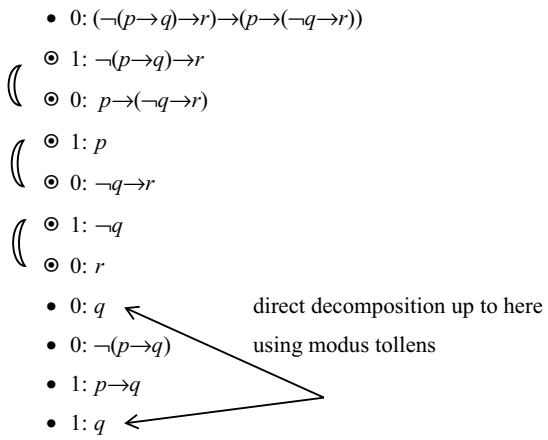

Fig. 11.9 Acceptable truth-tree for $(\neg(p \rightarrow q) \rightarrow r) \rightarrow (p \rightarrow (\neg q \rightarrow r))$

sentence letter whenever $\varphi \rightarrow \psi$ is acceptable? At the time of writing, this question is still open. If the answer turns out to be negative, the author apologizes for the time you have put into this chapter. Relevance logic is still in an unsettled state!

11.6 End-of Chapter Exercises

Exercise 11.6 (1) Packing and unpacking arrows

(a) (i) Show that the formula $p \rightarrow (q \rightarrow r)) \rightarrow ((p \wedge q) \rightarrow r)$ (*packing*, also known as *import*) is directly acceptable while (ii) its converse $((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$ (*unpacking*, *export*) is not directly acceptable. (b) Is unpacking nevertheless acceptable?

Solution

(a) (i) For packing, see the tree in Fig. 11.10, which clearly satisfies parity. (ii) For unpacking, the direct truth-tree in Fig. 11.11 grinds to a halt without reaching a crash-pair. Direct decomposition can go no further, so the formula is not directly acceptable. Note, in particular, that we are not allowed to conjoin 1: p and 1: q to get the antecedent of the second node and thus a modus ponens to 1: r. Do not confuse such conjunction of node labels within a tree with the property of adjunction, which tells us that the set of all acceptable formulae is closed under conjunction.

(b) No, it appears to be inacceptable. Although we can apply the recursive rule to the second node in Fig. 11.10 to get 1: $\neg(p \wedge q) \vee r$, then fork and continue to crash-pairs on each branch, parity fails. No other way of applying the recursive rule seems to get us round this failure, so the formula is presumably unacceptable.

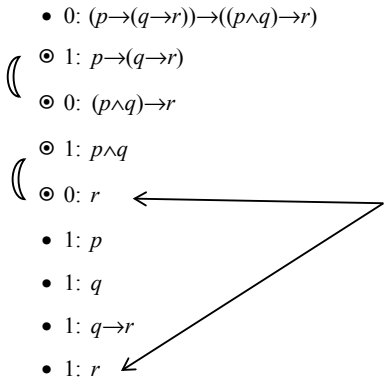

Fig. 11.10 Directly acceptable truth-tree for packing

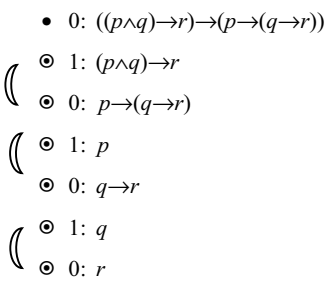

Fig. 11.11 Failed direct truth-tree for unpacking

Exercise 11.6 (2) Some enthymemes
We have seen in Exercise 11.4.2 (d) that the tempting formulae $(p \rightarrow q) \rightarrow (p \rightarrow (p \wedge q))$ and $(p \rightarrow q) \rightarrow ((p \vee q) \rightarrow q)$ are not directly acceptable; natural attempts to show acceptability by applying the recursive rule also fail. Show that we may, however, regard these formulae as *enthymemes*, in the sense that we can add (trivially directly acceptable) antecedents φ, ψ to obtain directly acceptable formulae $((p \rightarrow q) \wedge \varphi) \rightarrow (p \rightarrow (p \wedge q))$ and $((p \rightarrow q) \wedge \psi) \rightarrow ((p \vee q) \rightarrow q)$.

Solution outline
For the first formula, put $\varphi = p \rightarrow p$, obtaining $((p \rightarrow q) \wedge (p \rightarrow p)) \rightarrow (p \rightarrow (p \wedge q))$. This is an instance of the scheme $\wedge^{+}$, which we know from Exercise 11.4.2 (b) to be directly acceptable. For the second formula, put $\psi = q \rightarrow q$, forming $((p \rightarrow q) \wedge (q \rightarrow q)) \rightarrow ((p \vee q) \rightarrow q)$. This is an instance of the directly acceptable scheme $\vee^{-}$, which the same exercise tells us is directly acceptable.

Exercise 11.6 (3) Some tricky formulae
Check whether the following formulae are acceptable: (a) $(p \rightarrow \neg p) \rightarrow \neg(\neg p \rightarrow p)$; (b) $(((p \wedge q) \rightarrow r) \wedge (p \rightarrow q)) \rightarrow (p \rightarrow r)$; (c) $((p \rightarrow q) \rightarrow p) \rightarrow p$; (d) $((p \rightarrow p) \rightarrow p) \rightarrow p$, (e) $((p \leftrightarrow (p \rightarrow q)) \rightarrow q$. In (e), the biconditional abbreviates the conjunction of two conditionals.

Solution outline

(a) Acceptable, although not directly so. The key step in the decomposition is an application of the forwards clause of the recursive rule to a node labelled 1: $\neg p \rightarrow p$ to get 1: $p \vee p$. Inspection shows that parity is satisfied in the resulting tree.

(b) Acceptable, although not directly so. The key step in the decomposition is an application of the forwards clause of the recursive rule to a node labelled 1: $(p \wedge q) \rightarrow r$ to get 1: $\neg r \rightarrow \neg(p \wedge q)$. Careful inspection of the resulting tree shows that parity is satisfied. Note, on the other hand, that parity fails in the corresponding tree for the unpacked version $(p \rightarrow q) \rightarrow ((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow r))$ which, as far as one can tell, is not acceptable.

(c) Neither directly acceptable nor, as far as one can tell, acceptable since the most natural ways of using the recursive decomposition rule snag. In more detail, beginning its tree with root 0: $((p \to q) \to p) \to p$, counter-case gives 1: $(p \to q) \to p$ and 0: p but here, already, direct decomposition is stuck; no further rule can be applied. When the recursive rule is available we can continue with modus tollens (see Exercise 11.5.5) to get 0: $p \to q$, then counter-case to 1: p and 0: q, thus providing a crash-pair {0: p, 1: p}. However, the parity condition is not satisfied in this tree: of the two critical nodes 1: p and 0: q, only the former is in the trace of the crash-pair. Historical note: This formula is known as 'Peirce's law', after the late nineteenth century polymath Charles Sanders Peirce. It is a tautology, as is easily seen by translating the sub-formula $p \to q$ into $\neg p \vee q$.

(d) Acceptable, notwithstanding the fact that it is also an instance of the unacceptable (c). Indeed, it is also an instance of the acceptable formula $((p \to p) \to q) \to q$. More generally: whenever φ is acceptable then $(\varphi \to q) \to q$ is also acceptable. To verify this, construct a tree with root 0: $(\varphi \to q) \to q$, apply counter-case to get 1: $\varphi \to q$, 0: q, modus tollens (Exercise 11.5.5) to get 0: φ and finally paste in any acceptable tree for φ (there is at least one, since by supposition φ is acceptable). Clearly the resulting tree satisfies parity.

(e) Acceptable, although not directly so. The key step when constructing its tree is to use the directly acceptable contraction formula $((p \to (p \to q)) \to (p \to q)$ (Exercise 11.4.2 (b)) as justification for an application of the recursive rule.

11.7 Selected Reading

The approach in this chapter is set out in greater depth and detail in "Relevance-sensitive truth-trees", available via the author's webpage and also directly at https://drive.google.com/file/d/1Yw8J2QsJHhpaMa-E7mTS1RXe9rrYtsGe/view. This extends an initial presentation (with variant title) in the *Australasian Journal of Logic* 14:3 2017 http://ojs.victoria.ac.nz/ajl/article/view/4009/3617.

For some healthy skepticism about the project of relevance logic, see chapter 5 of John Burgess *Philosophical Logic*, Princeton University Press 2012. That is probably the best place to go before reading further, in order to keep a sense of balance.

Introductory entries on 'Relevant Logic' in *Wikipedia* and 'Logic, relevance' in the *Stanford Encyclopedia of Philosophy* review a variety of different approaches to the area, emphasizing the currently fashionable Routley-Meyer semantics. More detail and pointers to further literature can be found in J.M. Dunn & G. Restall's chapter on relevance logic in D. Gabbay & F. Guenthner eds. *Handbook of Philosophical Logic,* 2nd edn, Vol. 6, Amsterdam: Kluwer 2002 as well as the chapter by E. Mares & R. Meyer on relevant logics in L. Goble ed. *The Blackwell*

Guide to Philosophical Logic, Blackwell 2001. The route via natural deduction is clearly presented for students by S. Sterrett as Part 3 of *Three Views of Logic: Mathematics, Philosophy, and Computer Science* by D. Loveland, R. Hodel and S. Sterrett, Princeton University Press 2014.

First-degree relevance logic (that is, the fragment consisting of formulae $\alpha \rightarrow \beta$ where the only connectives in α, β are $\neg$, $\wedge$, $\vee$) is discussed in chapter 2 of the author's book *Topics in Modern Logic*, Methuen 1973, reprinted by Routledge 2020 as volume 14 of its collection of Library Editions in Logic.

For a discussion of (ultimately unsuccessful) attempts to use relevance logic in the axiomatization of set theory, see chapter 4 of Luca Incurvati *Conceptions of Set and the Foundations of Mathematics*, Cambridge University Press 2020. An instance of the formula in our end-of-chapter Exercise 11.6 (3) (e) plays a central role in the analysis.

Those wishing to study modal logic can get quite far with various entries in the *Stanford Encyclopedia of Philosophy*. Textbooks include G. Hughes & M. Cresswell's *A New Introduction to Modal Logic* London: Routledge 1996 directed mainly to students of philosophy and, for graduate students of mathematics and computing, P. Blackburn, M. de Rijke & Y. Venema *Modal Logic,* Cambridge University Press 2001.

Index

D. Makinson, *Sets, Logic and Maths for Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-42218-9

T