COMPUTABILITY *and* LOGIC

FIFTH EDITION

GEORGE S. BOOLOS
JOHN P. BURGESS
RICHARD C. JEFFREY

CAMBRIDGE

CAMBRIDGE www.cambridge.org/9780521877527

This page intentionally left blank

Computability and Logic, Fifth Edition

Computability and Logic has become a classic because of its accessibility to students without a mathematical background and because it covers not simply the staple topics of an intermediate logic course, such as Gödel's incompleteness theorems, but also a large number of optional topics, from Turing's theory of computability to Ramsey's theorem. This fifth edition has been thoroughly revised by John P. Burgess. Including a selection of exercises, adjusted for this edition, at the end of each chapter, it offers a new and simpler treatment of the representability of recursive functions, a traditional stumbling block for students on the way to the Gödel incompleteness theorems. This new edition is also accompanied by a Web site as well as an instructor's manual.

"[This book] gives an excellent coverage of the fundamental theoretical results about logic involving computability, undecidability, axiomatization, definability, incompleteness, and so on."

– *American Math Monthly*

"The writing style is excellent: Although many explanations are formal, they are perfectly clear. Modern, elegant proofs help the reader understand the classic theorems and keep the book to a reasonable length."

– *Computing Reviews*

"A valuable asset to those who want to enhance their knowledge and strengthen their ideas in the areas of artificial intelligence, philosophy, theory of computing, discrete structures, and mathematical logic. It is also useful to teachers for improving their teaching style in these subjects."

– *Computer Engineering*

Computability and Logic

Fifth Edition

GEORGE S. BOOLOS

JOHN P. BURGESS
Princeton University

RICHARD C. JEFFREY

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK
Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521877527

First published in print format 2007

ISBN-13 978-0-511-36668-0 eBook (EBL)
ISBN-10 0-511-36668-X eBook (EBL)

ISBN-13 978-0-521-87752-7 hardback
ISBN-10 0-521-87752-0 hardback

ISBN-13 978-0-521-70146-4 paperback
ISBN-10 0-521-70146-5 paperback

For
SALLY
and
AIGLI
and
EDITH

Contents

BASIC METALOGIC

FURTHER TOPICS

Preface to the Fifth Edition

The original authors of this work, the late George Boolos and Richard Jeffrey, stated in the preface to the first edition that the work was intended for students of philosophy, mathematics, or other fields who desired a more advanced knowledge of logic than is supplied by an introductory course or textbook on the subject, and added the following:

> The aim has been to present the principal fundamental theoretical results *about* logic, and to cover certain other meta-logical results whose proofs are not easily obtainable elsewhere. We have tried to make the exposition as readable as was compatible with the presentation of complete proofs, to use the most elegant proofs we knew of, to employ standard notation, and to reduce *hair* (as it is technically known).

Such have remained the aims of all subsequent editions.

The "principal fundamental theoretical results *about* logic" are primarily the theorems of Gödel, the completeness theorem, and especially the incompleteness theorems, with their attendant lemmas and corollaries. The "other meta-logical results" included have been of two kinds. On the one hand, filling roughly the first third of the book, there is an extended exposition by Richard Jeffrey of the theory of Turing machines, a topic frequently alluded to in the literature of philosophy, computer science, and cognitive studies but often omitted in textbooks on the level of this one. On the other hand, there is a varied selection of theorems on (in-)definability, (un-)decidability, (in-)completeness, and related topics, to which George Boolos added a few more items with each successive edition, until by the third, the last to which he directly contributed, it came to fill about the last third of the book.

When I undertook a revised edition, my special aim was to increase the pedagogical usefulness of the book by adding a selection of problems at the end of each chapter and by making more chapters independent of one another, so as to increase the range of options available to the instructor or reader as to what to cover and what to defer. Pursuit of the latter aim involved substantial rewriting, especially in the middle third of the book. A number of the new problems and one new section on undecidability were taken from Boolos's *Nachlass*, while the rewriting of the précis of first-order logic – summarizing the material typically covered in a more leisurely way in an introductory text or course and introducing the more abstract modes of reasoning that distinguish intermediate- from introductory-level logic – was undertaken in consultation with Jeffrey. Otherwise, the changes have been my responsibility alone.

The book runs now in outline as follows. The basic course in intermediate logic culminating in the first incompleteness theorem is contained in Chapters 1, 2, 6, 7, 9, 10, 12, 15, 16, and 17, minus any sections of these chapters starred as optional. Necessary background on enumerable and nonenumerable sets is supplied in Chapters 1 and 2. All the material on computability (recursion theory) that is strictly needed for the incompletness theorems has now been collected in Chapters 6 and 7, which may, if desired, be postponed until after the needed background material in logic. That material is presented in Chapters 9, 10, and 12 (for readers who have not had an introductory course in logic including a proof of the completeness theorem, Chapters 13 and 14 will also be needed). The machinery needed for the proof of the incompleteness theorems is contained in Chapter 15 on the arithmetization of syntax (though the instructor or reader willing to rely on Church's thesis may omit all but the first section of this chapter) and in Chapter 16 on the representability of recursive functions. The first completeness theorem itself is proved in Chapter 17. (The second incompleteness theorem is discussed in Chapter 18.)

A semester course should allow time to take up several supplementary topics in addition to this core material. The topic given the fullest exposition is the theory of Turing machines and their relation to recursive functions, which is treated in Chapters 3 through 5 and 8 (with an application to logic in Chapter 11). This now includes an account of Turing's theorem on the existence of a universal Turing machine, one of the intellectual landmarks of the last century. If this material is to be included, Chapters 3 through 8 would best be taken in that order, either after Chapter 2 or after Chapter 12 (or 14).

Chapters 19 through 21 deal with topics in general logic, and any or all of them might be taken up as early as immediately after Chapter 12 (or 14). Chapter 19 is presupposed by Chapters 20 and 21, but the latter are independent of each other. Chapters 22 through 26, all independent of one another, deal with topics related to formal arithmetic, and any of them could most naturally be taken up after Chapter 17. Only Chapter 27 presupposes Chapter 18. Users of the previous edition of this work will find essentially all the material in it still here, though not always in the same place, apart from some material in the former version of Chapter 27 that has, since the last edition of this book, gone into *The Logic of Provablity*.

All these changes were made in the fourth edition. In the present fifth edition, the main change to the body of the text (apart from correction of errata) is a further revision and simplification of the treatment of the representability of recursive functions, traditionally one of the greatest difficulties for students. The version now to be found in section 16.2 represents the distillation of more than twenty years' teaching experience trying to find ever easier ways over this hump. Section 16.4 on Robinson arithmetic has also been rewritten. In response to a suggestion from Warren Goldfarb, an explicit discussion of the distinction between two different kinds of appeal to Church's thesis, avoidable and unavoidable, has been inserted at the end of section 7.2. The avoidable appeals are those that consist of omitting the verification that certain obviously effectively computable functions are recursive; the unavoidable appeals are those involved whenever a theorem about recursiveness is converted into a conclusion about effective computability in the intuitive sense.

On the one hand, it should go without saying that in a textbook on a classical subject, only a small number of the results presented will be original to the authors. On the other hand, a textbook is perhaps not the best place to go into the minutiæ of the history of a field. Apart from a section of remarks at the end of Chapter 18, we have indicated the history of the field for the student or reader mainly by the names attached to various theorems. See also the annotated bibliography at the end of the book.

There remains the pleasant task of expressing gratitude to those (beyond the dedicatees) to whom the authors have owed personal debts. By the third edition of this work the original authors already cited Paul Benacerraf, Burton Dreben, Hartry Field, Clark Glymour, Warren Goldfarb, Simon Kochen, Paul Kripke, David Lewis, Paul Mellema, Hilary Putnam, W. V. Quine, T. M. Scanlon, James Thomson, and Peter Tovey, with special thanks to Michael J. Pendlebury for drawing the "mop-up" diagram in what is now section 5.2.

In connection with the fourth edition, my thanks were due collectively to the students who served as a trial audience for intermediate drafts, and especially to my very able assistants in instruction, Mike Fara, Nick Smith, and Caspar Hare, with special thanks to the last-named for the "scoring function" example in section 4.2. In connection with the present fifth edition, Curtis Brown, Mark Budolfson, John Corcoran, Sinan Dogramaci, Hannes Eder, Warren Goldfarb, Hannes Hutzelmeyer, David Keyt, Brad Monton, Jacob Rosen, Jada Strabbing, Dustin Tucker, Joel Velasco, Evan Williams, and Richard Zach are to be thanked for errata to the fourth edition, as well as for other helpful suggestions.

Perhaps the most important change connected with this fifth edition is one not visible in the book itself: It now comes supported by an instructor's manual. The manual contains (besides any errata that may come to light) suggested hints to students for odd-numbered problems and solutions to all problems. Resources are available to students and instructors at www.cambridge.org/us/9780521877527.

January 2007 JOHN P. BURGESS

Computability Theory

1

Enumerability

Our ultimate goal will be to present some celebrated theorems about inherent limits on what can be computed and on what can be proved. Before such results can be established, we need to undertake an analysis of computability and an analysis of provability. Computations involve positive integers 1, 2, 3, ... in the first instance, while proofs consist of sequences of symbols from the usual alphabet A, B, C, ... or some other. It will turn out to be important for the analysis both of computability and of provability to understand the relationship between positive integers and sequences of symbols, and background on that relationship is provided in the present chapter. The main topic is a distinction between two different kinds of infinite sets, the enumerable and the nonenumerable. This material is just a part of a larger theory of the infinite developed in works on set theory: the part most relevant to computation and proof. In section 1.1 we introduce the concept of enumerability. In section 1.2 we illustrate it by examples of enumerable sets. In the next chapter we give examples of nonenumerable sets.

1.1 Enumerability

An *enumerable*, or *countable*, set is one whose members can be enumerated: arranged in a single list with a first entry, a second entry, and so on, so that every member of the set appears sooner or later on the list. Examples: the set P of positive integers is enumerated by the list

$$1, 2, 3, 4, \ldots$$

and the set N of natural numbers is enumerated by the list

$$0, 1, 2, 3, \ldots$$

while the set P^- of negative integers is enumerated by the list

$$-1, -2, -3, -4, \ldots.$$

Note that the entries in these lists are not numbers but numerals, or names of numbers. In general, in listing the members of a set you manipulate names, not the things named. For instance, in enumerating the members of the United States Senate, you don't have the senators form a queue; rather, you arrange their *names* in a list, perhaps alphabetically. (An arguable exception occurs in the case where the members of the set being enumerated are themselves linguistic expressions. In this case we can plausibly speak of arranging the members themselves in a list. But we might also speak of the entries in the list as *names of themselves* so as to be able to continue to insist that in enumerating a set, it is *names* of members of the set that are arranged in a list.)

By courtesy, we regard as enumerable the empty set, $\varnothing$, which has no members. (*The* empty set; there is only one. The terminology is a bit misleading: It suggests comparison of empty sets with empty containers. But sets are more aptly compared with contents, and it should be considered that all empty containers have the same, null content.)

A list that enumerates a set may be finite or unending. An infinite set that is enumerable is said to be *enumerably infinite* or *denumerable*. Let us get clear about what things count as infinite lists, and what things do not. The positive integers can be arranged in a single infinite list as indicated above, but the following is not acceptable as a list of the positive integers:

$$1, 3, 5, 7, \ldots, 2, 4, 6, \ldots$$

Here, all the odd positive integers are listed, and then all the even ones. This will not do. In an acceptable list, each item must appear sooner or later as the nth entry, for some *finite* n. But in the unacceptable arrangement above, none of the even positive integers are represented in this way. Rather, they appear (so to speak) as entries number $\infty + 1$, $\infty + 2$, and so on.

To make this point perfectly clear we might define an enumeration of a set not as a listing, but as an arrangement in which each member of the set is *associated with* one of the positive integers 1, 2, 3, ... Actually, a list *is* such an arrangement. The thing named by the first entry in the list is associated with the positive integer 1, the thing named by the second entry is associated with the positive integer 2, and in general, the thing named by the nth entry is associated with the positive integer n.

In mathematical parlance, an infinite list determines a *function* (call it f) that takes positive integers as *arguments* and takes members of the set as *values*. [Should we have written: 'call it "f",' rather than 'call it f'? The common practice in mathematical writing is to use special symbols, including even italicized letters of the ordinary alphabet when being used as special symbols, as names for themselves. In case the special symbol happens also to be a name for something else, for instance, a function (as in the present case), we have to rely on context to determine when the symbol is being used one way and when the other. In practice this presents no difficulties.] The value of the function f for the argument n is denoted $f(n)$. This value is simply the thing denoted by the nth entry in the list. Thus the list

$$2, 4, 6, 8, \ldots$$

which enumerates the set E of even positive integers determines the function f for which we have

$$f(1) = 2, \qquad f(2) = 4, \qquad f(3) = 6, \qquad f(4) = 8, \qquad f(5) = 10, \ldots.$$

And conversely, the function f determines the list, except for notation. (The same list would look like this, in Roman numerals: II, IV, VI, VIII, X, ..., for instance.) Thus, we might have defined the function f first, by saying that for any positive integer n, the value of f is $f(n) = 2n$; and then we could have described the list by saying that for each positive integer n, its nth entry is the decimal representation of the number $f(n)$, that is, of the number $2n$.

Then we may speak of sets as being enumerated by functions, as well as by lists. Instead of enumerating the odd positive integers by the list 1, 3, 5, 7, . . . , we may enumerate them by the function that assigns to each positive integer n the value $2n - 1$. And instead of enumerating the set P of all positive integers by the list 1, 2, 3, 4, . . . , we may enumerate P by the function that assigns to each positive integer n the value n itself. This is the *identity function*. If we call it id, we have $\mathrm{id}(n) = n$ for each positive integer n.

If one function enumerates a nonempty set, so does some other; and so, in fact, do infinitely many others. Thus the set of positive integers is enumerated not only by the function id, but also by the function (call it g) determined by the following list:

$$2, 1, 4, 3, 6, 5, \ldots.$$

This list is obtained from the list 1, 2, 3, 4, 5, 6, . . . by interchanging entries in pairs: 1 with 2, 3 with 4, 5 with 6, and so on. This list is a strange but perfectly acceptable enumeration of the set P: every positive integer shows up in it, sooner or later. The corresponding function, g, can be defined as follows:

$$g(n) = \begin{cases} n + 1 & \text{if } n \text{ is odd} \\ n - 1 & \text{if } n \text{ is even.} \end{cases}$$

This definition is not as neat as the definitions $f(n) = 2n$ and $\mathrm{id}(n) = n$ of the functions f and id, but it does the job: It does indeed associate one and only one member of P with each positive integer n. And the function g so defined does indeed enumerate P: For each member m of P there is a positive integer n for which we have $g(n) = m$.

In enumerating a set by listing its members, it is perfectly all right if a member of the set shows up more than once on the list. The requirement is rather that each member show up *at least once*. It does not matter if the list is redundant: All we require is that it be complete. Indeed, a redundant list can always be thinned out to get an irredundant list, since one could go through and erase the entries that repeat earlier entries. It is also perfectly all right if a list has gaps in it, since one could go through and close up the gaps. The requirement is that every element of the set being enumerated be associated with some positive integer, not that every positive integer have an element of the set associated with it. Thus flawless enumerations of the positive integers are given by the following repetitive list:

$$1, 1, 2, 2, 3, 3, 4, 4, \ldots$$

and by the following gappy list:

$$1, -, 2, -, 3, -, 4, -, \ldots.$$

The function corresponding to this last list (call it h) assigns values corresponding to the first, third, fifth, . . . entries, but assigns no values corresponding to the gaps (second, fourth, sixth, . . . entries). Thus we have $h(1) = 1$, but $h(2)$ is nothing at all, for the function h is *undefined* for the argument 2; $h(3) = 2$, but $h(4)$ is undefined; $h(5) = 3$, but $h(6)$ is undefined. And so on: h is a *partial function* of positive integers; that is, it is defined only for positive integer arguments, but not for all such arguments. Explicitly, we might define the partial function h as follows:

$$h(n) = (n + 1)/2 \quad \text{if } n \text{ is odd.}$$

Or, to make it clear we haven't simply forgotten to say what values h assigns to even positive integers, we might put the definition as follows:

$$h(n) = \begin{cases} (n + 1)/2 & \text{if } n \text{ is odd} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Now the partial function h is a strange but perfectly acceptable enumeration of the set P of positive integers.

It would be perverse to choose h instead of the simple function id as an enumeration of P; but other sets are most naturally enumerated by partial functions. Thus, the set E of even integers is conveniently enumerated by the partial function (call it j) that agrees with id for even arguments, and is undefined for odd arguments:

$$j(n) = \begin{cases} n & \text{if } n \text{ is even} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The corresponding gappy list (in decimal notation) is

$$-, 2, -, 4, -, 6, -, 8, \ldots .$$

Of course the function f considered earlier, defined by $f(n) = 2n$ for all positive integers n, was an equally acceptable enumeration of E, corresponding to the gapless list 2, 4, 6, 8, and so on.

Any set S of positive integers is enumerated quite simply by a partial function s, which is defined as follows:

$$s(n) = \begin{cases} n & \text{if } n \text{ is in the set } S \\ \text{undefined} & \text{otherwise.} \end{cases}$$

It will be seen in the next chapter that although every set of positive integers is enumerable, there are sets of others sorts that are not enumerable. To say that a set A is enumerable is to say that there is a function all of whose arguments are positive integers and all of whose values are members of A, and that each member of A is a value of this function: For each member a of A there is at least one positive integer n to which the function assigns a as its value.

Notice that nothing in this definition requires A to be a set of positive integers or of numbers of any sort. Instead, A might be a set of people; or a set of linguistic expressions; or a set of sets, as when A is the set $\{P, E, \varnothing\}$. Here A is a set with three members, each of which is itself a set. One member of A is the infinite set P of all positive integers; another member of A is the infinite set E of all even positive integers; and the third is the empty set $\varnothing$. The set A is certainly enumerable, for example, by the following finite list: P, E, $\varnothing$. Each entry in this list names a member of A, and every member of A is named sooner or later on this list. This list determines a function (call it f), which can be defined by the three statements: $f(1) = P$, $f(2) = E$, $f(3) = \varnothing$. To be precise, f is a *partial function* of positive integers, being undefined for arguments greater than 3.

In conclusion, let us straighten out our terminology. A *function* is an assignment of *values* to *arguments*. The set of all those arguments to which the function assigns values is called the *domain* of the function. The set of all those values that the function assigns to its arguments is called the *range* of the function. In the case of functions whose arguments are positive integers, we distinguish between *total* functions and *partial* functions. A total function of positive integers is one whose domain is the whole set P of positive integers. A partial function of positive integers is one whose domain is something less than the whole set P. From now on, when we speak simply of a *function of positive integers*, we should be understood as leaving it open whether the function is total or partial. (This is a departure from the usual terminology, in which *function* of positive integers always means *total function*.) A set is *enumerable* if and only if it is the range of some function of positive integers. We said earlier we wanted to count the empty set $\varnothing$ as enumerable. We therefore have to count as a partial function the *empty function* e of positive integers that is undefined for all arguments. Its domain and its range are both $\varnothing$.

It will also be important to consider functions with two, three, or more positive integers as arguments, notably the addition function $\mathrm{sum}(m, n) = m + n$ and the multiplication function $\mathrm{prod}(m, n) = m \cdot n$. It is often convenient to think of a two-argument or two-place function on positive integers as a one-argument function on *ordered pairs* of positive integers, and similarly for many-argument functions. A few more notions pertaining to functions are defined in the first few problems at the end of this chapter. In general, *the problems at the end should be read as part of each chapter*, even if not all are going to be worked.

1.2 Enumerable Sets

We next illustrate the definition of the preceding section by some important examples. The following sets are enumerable.

1.1 Example (The set of integers). The simplest list is $0, 1, -1, 2, -2, 3, -3, \ldots$. Then if the corresponding function is called f, we have $f(1) = 0$, $f(2) = 1$, $f(3) = -1$, $f(4) = 2$, $f(5) = -2$, and so on.

1.2 Example (The set of ordered pairs of positive integers). The enumeration of pairs will be important enough in our later work that it may be well to indicate two different ways of accomplishing it. The first way is this. As a preliminary to enumerating them, we organize them into a rectangular array. We then traverse the array in *Cantor's zig-zag* manner indicated in Figure 1.1. This gives us the list

$$(1, 1), (1, 2), (2, 1), (1, 3), (2, 2), (3, 1), (1, 4), (2, 3), (3, 2), (4, 1), \ldots.$$

If we call the function involved here G, then we have $G(1) = (1, 1)$, $G(2) = (1, 2)$, $G(3) = (2, 1)$, and so on. The pattern is: First comes the pair the sum of whose entries is 2, then

$$\begin{array}{cccccc}
(1,1) - (1,2) & (1,3) & (1,4) & (1,5) & \cdots \\
(2,1) \quad (2,2) & (2,3) & (2,4) & (2,5) & \cdots \\
(3,1) \quad (3,2) & (3,3) & (3,4) & (3,5) & \cdots \\
(4,1) \quad (4,2) & (4,3) & (4,4) & (4,5) & \cdots \\
(5,1) \quad (5,2) & (5,3) & (5,4) & (5,5) & \cdots \\
\vdots \qquad \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

Figure 1-1. Enumerating pairs of positive integers.

come the pairs the sum of whose entries is 3, then come the pairs the sum of whose entries is 4, and so on. Within each block of pairs whose entries have the same sum, pairs appear in order of increasing first entry.

As for the second way, we begin with the thought that while an ordinary hotel may have to turn away a prospective guest because all rooms are full, a hotel with an enumerable infinity of rooms would always have room for one more: The new guest could be placed in room 1, and every other guest asked to move over one room. But actually, a little more thought shows that with foresight the hotelier can be prepared to accommodate a busload with an enumerable infinity of new guests each day, without inconveniencing any old guests by making them change rooms. Those who arrive on the first day are placed in *every other* room, those who arrive on the second day are placed in every other room *among those remaining vacant*, and so on. To apply this thought to enumerating pairs, let us use up every other place in listing the pairs $(1, n)$, every other place then remaining in listing the pairs $(2, n)$, every other place then remaining in listing the pairs $(3, n)$, and so on. The result will look like this:

$$(1, 1), (2, 1), (1, 2), (3, 1), (1, 3), (2, 2), (1, 4), (4, 1), (1, 5), (2, 3), \ldots .$$

If we call the function involved here g, then $g(1) = (1, 1)$, $g(2) = (2, 1)$, $g(3) = (1, 2)$, and so on.

Given a function f enumerating the pairs of positive integers, such as G or g above, an a such that $f(a) = (m, n)$ may be called a *code number* for the pair (m, n). Applying the function f may be called *decoding*, while going the opposite way, from the pair to a code for it, may be called *encoding*. It is actually possible to derive mathematical formulas for the encoding functions J and j that go with the decoding functions G and g above. (Possible, but not necessary: What we have said so far more than suffices as a proof that the set of pairs is enumerable.)

Let us take first J. We want $J(m, n)$ to be the number p such that $G(p) = (m, n)$, which is to say the place p where the pair (m, n) comes in the enumeration corresponding to G. Before we arrive at the pair (m, n), we will have to pass the pair whose entries sum to 2, the two pairs whose entries sum to 3, the three pairs whose entries sum to 4, and so on, up through the $m + n - 2$ pairs whose entries sum to $m + n - 1$.

The pair (m, n) will appear in the mth place after all of these pairs. So the position of the pair (m, n) will be given by

$$[1 + 2 + \cdots + (m + n - 2)] + m.$$

At this point we recall the formula for the sum of the first k positive integers:

$$1 + 2 + \cdots + k = k(k + 1)/2.$$

(Never mind, for the moment, where this formula comes from. Its derivation will be recalled in a later chapter.) So the position of the pair (m, n) will be given by

$$(m + n - 2)(m + n - 1)/2 + m.$$

This simplifies to

$$J(m, n) = (m^2 + 2mn + n^2 - m - 3n + 2)/2.$$

For instance, the pair (3, 2) should come in the place

$$(3^2 + 2 \cdot 3 \cdot 2 + 2^2 - 3 - 3 \cdot 2 + 2)/2 = (9 + 12 + 4 - 3 - 6 + 2)/2 = 18/2 = 9$$

as indeed it can be seen (looking back at the enumeration as displayed above) that it does: $G(9) = (3, 2)$.

Turning now to j, we find matters a bit simpler. The pairs with first entry 1 will appear in the places whose numbers are odd, with $(1, n)$ in place $2n - 1$. The pairs with first entry 2 will appear in the places whose numbers are twice an odd number, with $(2, n)$ in place $2(2n - 1)$. The pairs with first entry 3 will appear in the places whose numbers are four times an odd number, with $(3, n)$ in place $4(2n - 1)$. In general, in terms of the powers of two ($2^0 = 1$, $2^1 = 2$, $2^2 = 4$, and so on), (m, n) will appear in place $j(m, n) = 2^{m-1}(2n - 1)$. Thus (3, 2) should come in the place $2^{3-1}(2 \cdot 2 - 1) = 2^2(4 - 1) = 4 \cdot 3 = 12$, as indeed it does: $g(12) = (3, 2)$.

The series of examples to follow shows how more and more complicated objects can be coded by positive integers. Readers may wish to try to find proofs of their own before reading ours; and for this reason we give the statements of all the examples first, and collect all the proofs afterwards. As we saw already with Example 1.2, several equally good codings may be possible.

1.3 Example. The set of positive rational numbers

1.4 Example. The set of rational numbers

1.5 Example. The set of ordered triples of positive integers

1.6 Example. The set of ordered k-tuples of positive integers, for any fixed k

1.7 Example. The set of finite sequences of positive integers less than 10

1.8 Example. The set of finite sequences of positive integers less than b, for any fixed b

1.9 Example. The set of finite sequences of positive integers

1.10 Example. The set of finite sets of positive integers

1.11 Example. Any subset of an enumerable set

1.12 Example. The union of any two enumerable sets

1.13 Example. The set of finite strings from a finite or enumerable alphabet of symbols

Proofs

Example 1.3. A positive rational number is a number that can be expressed as a *ratio* of positive integers, that is, in the form m/n where m and n are positive integers. Therefore we can get an enumeration of all positive rational numbers by starting with our enumeration of all pairs of positive integers and replacing the pair (m, n) by the rational number m/n. This gives us the list

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

or, simplified,

$$1, 1/2, 2, 1/3, 1, 3, 1/4, 2/3, 3/2, 4, 1/5, 1/2, 1, 2, 5/1, 1/6, \ldots.$$

Every positive rational number in fact appears infinitely often, since for instance $1/1 = 2/2 = 3/3 = \cdots$ and $1/2 = 2/4 = \cdots$ and $2/1 = 4/2 = \cdots$ and similarly for every other rational number. But that is all right: our definition of enumerability permits repetitions.

Example 1.4. We combine the ideas of Examples 1.1 and 1.3. You know from Example 1.3 how to arrange the positive rationals in a single infinite list. Write a zero in front of this list, and then write the positive rationals, backwards and with minus signs in front of them, in front of that. You now have

$$\ldots, -1/3, -2, -1/2, -1, 0, 1, 1/2, 2, 1/3, \ldots$$

Finally, use the method of Example 1.1 to turn this into a proper list:

$$0, 1, -1, 1/2, -1/2, 2, -2, 1/3, -1/3, \ldots$$

Example 1.5. In Example 1.2 we have given two ways of listing all pairs of positive integers. For definiteness, let us work here with the first of these:

$$(1, 1), (1, 2), (2, 1), (1, 3), (2, 2), (3, 1), \ldots.$$

Now go through this list, and in each pair replace the second entry or component n with the pair that appears in the nth place on this very list. In other words, replace each 1 that appears in the second place of a pair by (1, 1), each 2 by (1, 2), and so on. This gives the list

$$(1, (1, 1)), (1, (1, 2)), (2, (1, 1)), (1, (2, 1)), (2, (1, 2)), (3, (1, 1)), \ldots$$

and that gives a list of triples

$$(1, 1, 1), (1, 1, 2), (2, 1, 1), (1, 2, 1), (2, 1, 2), (3, 1, 1), \ldots.$$

In terms of functions, this enumeration may be described as follows. The original enumeration of pairs corresponds to a function associating to each positive integer n

a pair $G(n) = (K(n), L(n))$ of positive integers. The enumeration of triples we have just defined corresponds to assigning to each positive integer n instead the triple

$$(K(n), K(L(n)), L(L(n))).$$

We do not miss any triples (p, q, r) in this way, because there will always be an $m = J(q, r)$ such that $(K(m), L(m)) = (q, r)$, and then there will be an $n = J(p, m)$ such that $(K(n), L(n)) = (p, m)$, and the triple associated with this n will be precisely (p, q, r).

Example 1.6. The method by which we have just obtained an enumeration of triples from an enumeration of pairs will give us an enumeration of quadruples from an enumeration of triples. Go back to the original enumeration pairs, and replace each second entry n by the triple that appears in the nth place in the enumeration of triples, to get a quadruple. The first few quadruples on the list will be

$$(1, 1, 1, 1), (1, 1, 1, 2), (2, 1, 1, 1), (1, 2, 1, 1), (2, 1, 1, 2), \ldots.$$

Obviously we can go on from here to quintuples, sextuples, or k-tuples for any fixed k.

Example 1.7. A finite sequence whose entries are all positive integers less than 10, such as (1, 2, 3), can be read as an ordinary decimal or base-10 numeral 123. The number this numeral denotes, one hundred twenty-three, could then be taken as a code number for the given sequence. Actually, for later purposes it proves convenient to modify this procedure slightly and write the sequence *in reverse* before reading it as a numeral. Thus (1, 2, 3) would be coded by 321, and 123 would code (3, 2, 1). In general, a sequence

$$s = (a_0, a_1, a_2, \ldots, a_k)$$

would be coded by

$$a_0 + 10a_1 + 100a_2 + \cdots + 10^k a_k$$

which is the number that the decimal numeral $a_k \cdots a_2 a_1 a_0$ represents. Also, it will be convenient henceforth to call the initial entry of a finite sequence the 0th entry, the next entry the 1st, and so on. To decode and obtain the ith entry of the sequence coded by n, we take the quotient on dividing by 10^i, and then the remainder on dividing by 10. For instance, to find the 5th entry of the sequence coded by 123 456 789, we divide by 10^5 to obtain the quotient 1234, and then divide by 10 to obtain the remainder 4.

Example 1.8. We use a decimal, or base-10, system ultimately because human beings typically have 10 fingers, and counting began with counting on fingers. A similar base-b system is possible for any $b > 1$. For a *binary*, or base-2, system only the ciphers 0 and 1 would be used, with $a_k \ldots a_2 a_1 a_0$ representing

$$a_0 + 2a_1 + 4a_2 + \cdots + 2^k a_k.$$

So, for instance, 1001 would represent $1 + 2^3 = 1 + 8 = 9$. For a duodecimal, or base-12, system, two additional ciphers, perhaps * and # as on a telephone, would be needed for ten and eleven. Then, for instance, 1*# would represent $11 + 12 \cdot 10 + 144 \cdot 1 = 275$. If we applied the idea of the previous problem using base 12 instead of base 10, we could code finite sequences of positive integers less than 12, and not just finite sequences of positive integers less than 10. More generally, we can code a finite sequence

$$s = (a_0, a_1, a_2, \ldots, a_k)$$

of positive integers less than b by

$$a_0 + ba_1 + b^2a_2 + \cdots + b^k a_k.$$

To obtain the ith entry of the sequence coded by n, we take the quotient on dividing by b^i and then the remainder on dividing by b. For example, when working with base 12, to obtain the 5th entry of the sequence coded by 123 456 789, we divide 123 456 789 by 12^5 to get the quotient 496. Now divide by 12 to get remainder 4. In general, working with base b, the ith entry—counting the initial one as the 0th—of the sequence coded by (b, n) will be

$$\text{entry}(i, n) = \text{rem}(\text{quo}(n, b^i), b)$$

where quo(x, y) and rem(x, y) are the quotient and remainder on dividing x by y.

Example 1.9. Coding finite sequences will be important enough in our later work that it will be appropriate to consider several different ways of accomplishing this task. Example 1.6 showed that we can code sequences whose entries may be of any size but that are *of fixed length*. What we now want is an enumeration of *all* finite sequences—pairs, triples, quadruples, and so on—in a single list; and for good measure, let us include the 1-tuples or 1-term sequences (1), (2), (3), ... as well. A first method, based on Example 1.6, is as follows. Let $G_1(n)$ be the 1-term sequence (n). Let $G_2 = G$, the function enumerating all 2-tuples or pairs from Example 1.2. Let G_3 be the function enumerating all triples as in Example 1.5. Let $G_4, G_5, \ldots,$ be the enumerations of triples, quadruples, and so on, from Example 1.6. We can get a coding of all finite sequences by *pairs* of positive integers by letting any sequence s of length k be coded by the pair (k, a) where $G_k(a) = s$. Since pairs of positive integers can be coded by single numbers, we indirectly get a coding of sequences of numbers. Another way to describe what is going on here is as follows. We go back to our original listing of pairs, and replace the pair (k, a) by the ath item on the list of k-tuples. Thus (1, 1) would be replaced by the first item (1) on the list of 1-tuples (1), (2), (3), ... ; while (1, 2) would be replaced by the second item (2) on the same list; whereas (2, 1) would be replaced by the first item (1, 1) on the list of all 2-tuples or pairs; and so on. This gives us the list

$$(1), (2), (1, 1), (3), (1, 2), (1, 1, 1), (4), (2, 1), (1, 1, 2), (1, 1, 1, 1), \ldots.$$

(If we wish to include also the 0-tuple or empty sequence ( ), which we may take to be simply the empty set $\varnothing$, we can stick it in at the head of the list, in what we may think of as the 0th place.)

Example 1.8 showed that we can code sequences of any length whose entries are *less than some fixed bound*, but what we now want to do is show how to code sequences of any length whose entries may be *of any size*. A second method, based on Example 1.8, is to begin by coding sequences by pairs of positive integers. We take a sequence

$$s = (a_0, a_1, a_2, \ldots, a_k)$$

to be coded by any *pair* (b, n) such that all a_i are less than b, and n codes s in the sense that

$$n = a_0 + b \cdot a_1 + b^2 a_2 + \cdots + b^k a_k.$$

Thus (10, 275) would code (5, 7, 2), since $275 = 5 + 7 \cdot 10 + 2 \cdot 10^2$, while (12, 275) would code (11, 10, 1), since $275 = 11 + 10 \cdot 12 + 1 \cdot 12^2$. Each sequence would have many codes, since for instance (10, 234) and (12, 328) would equally code (4, 3, 2), because $4 + 3 \cdot 10 + 2 \cdot 10^2 = 234$ and $4 + 3 \cdot 12 + 2 \cdot 12^2 = 328$. As with the previous method, since pairs of positive integers can be coded by single numbers, we indirectly get a coding of sequences of numbers.

A third, and totally different, approach is possible, based on the fact that every integer greater than 1 can be written in one and only one way as a product of powers of larger and larger primes, a representation called its *prime decomposition*. This fact enables us to code a sequence $s = (i, j, k, m, n, \ldots)$ by the number $2^i 3^j 5^k 7^m 11^n \ldots$. Thus the code number for the sequence (3, 1, 2) is $2^3 3^1 5^2 = 8 \cdot 3 \cdot 25 = 600$.

Example 1.10. It is easy to get an enumeration of finite sets from an enumeration of finite sequences. Using the first method in Example 1.9, for instance, we get the following enumeration of sets:

$$\{1\}, \{2\}, \{1, 1\}, \{3\}, \{1, 2\}, \{1, 1, 1\}, \{4\}, \{2, 1\}, \{1, 1, 2\}, \{1, 1, 1, 1\}, \ldots.$$

The set $\{1, 1\}$ whose only elements are 1 and 1 is just the set $\{1\}$ whose only element is 1, and similarly in other cases, so this list can be simplified to look like this:

$$\{1\}, \{2\}, \{1\}, \{3\}, \{1, 2\}, \{1\}, \{4\}, \{1, 2\}, \{1, 2\}, \{1\}, \{5\}, \ldots.$$

The repetitions do not matter.

Example 1.11. Given any enumerable set A and a listing of the elements of A:

$$a_1, a_2, a_3, \ldots$$

we easily obtain a gappy listing of the elements of any subset B of A simply by erasing any entry in the list that does not belong to B, leaving a gap.

Example 1.12. Let A and B be enumerable sets, and consider listings of their elements:

$$a_1, a_2, a_3, \ldots \qquad b_1, b_2, b_3, \ldots.$$

Imitating the *shuffling* idea of Example 1.1, we obtain the following listing of the elements of the union $A \cup B$ (the set whose elements are all and only those items that are elements either of A or of B or of both):

$$a_1, b_1, a_2, b_2, a_3, b_3, \ldots.$$

If the intersection $A \cap B$ (the set whose elements of both A and B) is not empty, then there will be redundancies on this list: If $a_m = b_n$, then that element will appear both at place $2m - 1$ and at place $2n$, but this does not matter.

Example 1.13. Given an 'alphabet' of any finite number, or even an enumerable infinity, of symbols $S_1, S_2, S_3, \ldots$ we can take as a code number for any finite string

$$S_{a_0} S_{a_1} S_{a_2} \cdots S_{a_k}$$

the code number for the finite sequence of positive integers

$$(a_1, a_2, a_3, \ldots, a_k)$$

under any of the methods of coding considered in Example 1.9. (We are usually going to use the third method.) For instance, with the ordinary alphabet of 26 symbols letters $S_1 =$ 'A', $S_2 =$ 'B', and so on, the string or word 'CAB' would be coded by the code for (3, 1, 2), which (on the third method of Example 1.9) would be $2^3 \cdot 3 \cdot 5^2 = 600$.

Problems

1.1 A (total or partial) *function* f from a set A to a set B is an assignment for (some or all) elements a of A of an associated element $f(a)$ of B. If $f(a)$ is defined for *every* element a of A, then the function f is called *total*. If every element b of B is assigned to some element a of A, then the function f is said to be *onto*. If no element b of B is assigned to more than one element a of A, then the function f is said to be *one-to-one*. The *inverse* function f^{-1} from B to A is defined by letting $f^{-1}(b)$ be the one and only a such that $f(a) = b$, if any such a exists; $f^{-1}(b)$ is undefined if there is no a with $f(a) = b$ or more than one such a. Show that if f is a one-to-one function and f^{-1}its inverse function, then f^{-1} is total if and only if f is onto, and conversely, f^{-1} is onto if and only if f is total.

1.2 Let f be a function from a set A to a set B, and g a function from the set B to a set C. The *composite* function $h = gf$ from A to C is defined by $h(a) = g(f(a))$. Show that:

(a) If f and g are both total, then so is gf.

(b) If f and g are both onto, then so is gf.

(c) If f and g are both one-to-one, then so is gf.

1.3 A *correspondence* between sets A and B is a one-to-one total function from A onto B. Two sets A and B are said to be *equinumerous* if and only if there is a correspondence between A and B. Show that equinumerosity has the following properties:

(a) Any set A is equinumerous with itself.

(b) If A is equinumerous with B, then B is equinumerous with A.

(c) If A is equinumerous with B and B is equinumerous with C, then A is equinumerous with C.

1.4 A set A has n elements, where n is a positive integer, if it is equinumerous with the set of positive integers up to n, so that its elements can be listed as $a_1, a_2, \ldots, a_n$. A nonempty set A is finite if it has n elements for some positive integer n. Show that any enumerable set is either finite or equinumerous with the set of all positive integers. (In other words, given an *enumeration*, which is to say a function from the set of positive integers onto a set A, show that if A is not finite, then there is a *correspondence*, which is to say a *one-to-one*, *total* function, from the set of positive integers onto A.)

1.5 Show that the following sets are equinumerous:

(a) The set of rational numbers with denominator a power of two (when written in lowest terms), that is, the set of rational numbers $\pm m/n$ where $n = 1$ or 2 or 4 or 8 or some higher power of 2.

(b) The set of those sets of positive integers that are either finite or cofinite, where a set S of positive integers is *cofinite* if the set of all positive integers n that are *not* elements of S is finite.

1.6 Show that the set of all finite subsets of an enumerable set is enumerable.

1.7 Let $A = \{A_1, A_2, A_3, \ldots\}$ be an enumerable family of sets, and suppose that each A_i for $i =$ 1, 2, 3, and so on, is enumerable. Let $\cup A$ be the union of the family A, that is, the set whose elements are precisely the elements of the elements of A. Is $\cup A$ enumerable?

2

Diagonalization

In the preceding chapter we introduced the distinction between enumerable and nonenumerable sets, and gave many examples of enumerable sets. In this short chapter we give examples of nonenumerable sets. We first prove the existence of such sets, and then look a little more closely at the method, called diagonalization, *used in this proof.*

Not all sets are enumerable: some are too big. For example, consider the set of *all sets of positive integers*. This set (call it P^*) contains, as a member, each finite and each infinite set of positive integers: the empty set $\varnothing$, the set P of all positive integers, and every set between these two extremes. Then we have the following celebrated result.

2.1 Theorem (Cantor's Theorem). The set of all sets of positive integers is not enumerable.

Proof: We give a method that can be applied to *any* list L of sets of positive integers in order to discover a set $\Delta(L)$ of positive integers which is not named in the list. If you then try to repair the defect by adding $\Delta(L)$ to the list as a new first member, the same method, applied to the augmented list L^* will yield a different set $\Delta(L^*)$ that is likewise not on the augmented list.

The method is this. Confronted with any infinite list L

$$S_1, S_2, S_3 \ldots.$$

of sets of positive integers, we define a set $\Delta(L)$ as follows:

(*) For each positive integer n, n is in $\Delta(L)$ if and only if n is *not* in S_n.

It should be clear that this genuinely defines a set $\Delta(L)$; for, given any positive integer n, we can tell whether n is in $\Delta(L)$ if we can tell whether n is in the nth set in the list L. Thus, if S_3 happens to be the set E of even positive integers, the number 3 is not in S_3 and therefore it *is* in $\Delta(L)$. As the notation $\Delta(L)$ indicates, the composition of the set $\Delta(L)$ depends on the composition of the list L, so that different lists L may yield different sets $\Delta(L)$.

To show that the set $\Delta(L)$ that this method yields is never in the given list L, we argue by *reductio ad absurdum*: we suppose that $\Delta(L)$ does appear somewhere in list L, say as entry number m, and deduce a contradiction, thus showing that the supposition must be false. Here we go. *Supposition*: For some positive integer m,

$$S_m = \Delta(L).$$

[Thus, if 127 is such an m, we are supposing that $\Delta(L)$ and S_{127} are the same set under different names: we are supposing that a positive integer belongs to $\Delta(L)$ if and only if it belongs to the 127th set in list L.] To deduce a contradiction from this assumption we apply definition (*) to the particular positive integer m: with $n = m$, (*) tells us that

$$m \text{ is in } \Delta(L) \text{ if and only if } m \text{ is } \textit{not} \text{ in } S_m.$$

Now a contradiction follows from our supposition: if S_m and $\Delta(L)$ are one and the same set we have

$$m \text{ is in } \Delta(L) \text{ if and only if } m \text{ is in } S_m.$$

Since this is a flat self-contradiction, our supposition must be false. For no positive integer m do we have $S_m = \Delta(L)$. In other words, the set $\Delta(L)$ is named nowhere in list L.

So the method works. Applied to any list of sets of positive integers it yields a set of positive integers which was not in the list. Then no list enumerates all sets of positive integers: the set P^* of all such sets is not enumerable. This completes the proof.

Note that results to which we might wish to refer back later are given reference numbers 1.1, 1.2, . . . consecutively through the chapter, to make them easy to locate. Different words, however, are used for different kinds of results. The most important general results are dignified with the title of ‘theorem’. Lesser results are called ‘lemmas’ if they are steps on the way to a theorem, ‘corollaries’ if they follow directly upon some theorem, and ‘propositions’ if they are free-standing. In contrast to all these, ‘examples’ are particular rather than general. The most celebrated of the theorems have more or less traditional names, given in parentheses. The fact that 2.1 has been labelled ‘Cantor’s theorem’ is an indication that it is a famous result. The reason is not—we hope the reader will agree!—that its proof is especially difficult, but that the *method* of the proof (*diagonalization*) was an important innovation. In fact, it is so important that it will be well to look at the proof again from a slightly different point of view, which allows the entries in the list L to be more readily visualized.

Accordingly, we think of the sets S_1, S_2, . . . as represented by functions s_1, s_2, . . . of positive integers that take the numbers 0 and 1 as values. The relationship between the set S_n and the corresponding function s_n is simply this: for each positive integer p we have

$$s_n(p) = \begin{cases} 1 & \text{if } p \text{ is in } S_n \\ 0 & \text{if } p \text{ is not in } S_n. \end{cases}$$

Then the list can be visualized as an infinite rectangular array of zeros and ones, in which the nth row represents the function s_n and thus represents the set S_n. That is,

$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & \\
\hline
s_1 & s_1(1) & s_1(2) & s_1(3) & s_1(4) & \\
s_2 & s_2(1) & s_2(2) & s_2(3) & s_2(4) & \\
s_3 & s_3(1) & s_3(2) & s_3(3) & s_3(4) & \\
s_4 & s_4(1) & s_4(2) & s_4(3) & s_4(4) & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Figure 2-1. A list as a rectangular array.

the nth row

$$s_n(1)s_n(2)s_n(3)s_n(4)\ldots$$

is a sequence of zeros and ones in which the pth entry, $s_n(p)$, is 1 or 0 according as the number p is or is not in the set S_n. This array is shown in Figure 2-1.

The entries in the diagonal of the array (upper left to lower right) form a sequence of zeros and ones:

$$s_1(1)\,s_2(2)\,s_3(3)\,s_4(4)\ldots$$

This sequence of zeros and ones (the *diagonal sequence*) determines a set of positive integers (the *diagonal set*). *The diagonal set may well be among those listed in L.* In other words, there may well be a positive integer d such that the set S_d is none other than our diagonal set. The sequence of zeros and ones in the dth row of Figure 2-1 would then agree with the diagonal sequence entry by entry:

$$s_d(1) = s_1(1), \qquad s_d(2) = s_2(2), \qquad s_d(3) = s_3(3), \ldots.$$

That is as may be: the diagonal set may or may not appear in the list L, depending on the detailed makeup of the list. What we want is a set we can rely upon *not* to appear in L, no matter how L is composed. Such a set lies near to hand: it is the *antidiagonal set*, which consists of the positive integers not in the diagonal set. The corresponding *antidiagonal sequence* is obtained by changing zeros to ones and ones to zeros in the diagonal sequence. We may think of this transformation as a matter of subtracting each member of the diagonal sequence from 1: we write the antidiagonal sequence as

$$1 - s_1(1), 1 - s_2(2), 1 - s_3(3), 1 - s_4(4), \ldots.$$

This sequence can be relied upon not to appear as a row in Figure 2-1, for if it did appear—say, as the mth row—we should have

$$s_m(1) = 1 - s_1(1), \qquad s_m(2) = 1 - s_2(2), \ldots, \qquad s_m(m) = 1 - s_m(m), \ldots.$$

But the mth of these equations cannot hold. [*Proof*: $s_m(m)$ must be zero or one. If zero, the mth equation says that $0 = 1$. If one, the mth equation says that $1 = 0$.] Then the antidiagonal sequence differs from every row of our array, and so the antidiagonal set differs from every set in our list L. This is no news, for the antidiagonal set is simply the set $\Delta(L)$. We have merely repeated with a diagram—Figure 2-1—our proof that $\Delta(L)$ appears nowhere in the list L.

Of course, it is rather strange to say that the members of an infinite set 'can be arranged' in a single list. By whom? Certainly not by any human being, for nobody has that much time or paper; and similar restrictions apply to machines. In fact, to call a set enumerable is simply to say that it is the range of some total or partial function of positive integers. Thus, the set E of even positive integers is enumerable because there are functions of positive integers that have E as their range. (We had two examples of such functions earlier.) Any such function can then be thought of as a program that a superhuman enumerator can follow in order to arrange the members of the set in a single list. More explicitly, the program (the set of instructions) is: 'Start counting from 1, and never stop. As you reach each number n, write a name of $f(n)$ in your list. [Where $f(n)$ is undefined, leave the nth position blank.]' But there is no need to refer to the list, or to a superhuman enumerator: anything we need to say about enumerability can be said in terms of the functions themselves; for example, to say that the set P^* is not enumerable is simply to deny the existence of any function of positive integers which has P^* as its range.

Vivid talk of lists and superhuman enumerators may still aid the imagination, but in such terms the theory of enumerability and diagonalization appears as a chapter in mathematical theology. To avoid treading on any living toes we might put the whole thing in a classical Greek setting: Cantor proved that there are sets which even Zeus cannot enumerate, no matter how fast he works, or how long (even, infinitely long).

If a set *is* enumerable, Zeus can enumerate it in one second by writing out an infinite list faster and faster. He spends $1/2$ second writing the first entry in the list; $1/4$ second writing the second entry; $1/8$ second writing the third; and in general, he writes each entry in half the time he spent on its predecessor. At no point *during* the one-second interval has he written out the whole list, but when one second has passed, the list is complete. On a time scale in which the marked divisions are sixteenths of a second, the process can be represented as in Figure 2-2.

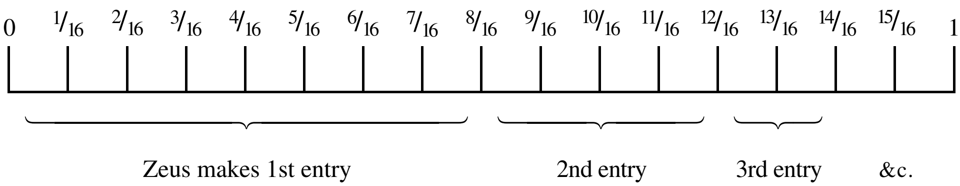

Figure 2-2. Completing an infinite process in finite time.

To speak of writing out an infinite list (for example, of all the positive integers, in decimal notation) is to speak of such an enumerator either working faster and faster as above, or taking all of infinite time to complete the list (making one entry per second, perhaps). Indeed, Zeus could write out an infinite sequence of infinite lists if he chose to, taking only one second to complete the job. He could simply allocate the first half second to the business of writing out the first infinite list ($1/4$ second for the first entry, $1/8$ second for the next, and so on); he could then write out the whole second list in the following quarter second ($1/8$ for the first entry, $1/16$ second for the next, and so on); and in general, he could write out each subsequent list in just half the time he spent on its predecessor, so that after one second had passed he would have written out every entry in every list, in order. But the result does not count as a single infinite list, in our sense of the term. In our sort of list, each entry must come some *finite* number of places after the first.

As we use the term 'list', Zeus has not produced a list by writing infinitely many infinite lists one after another. But he could perfectly well produce a genuine list which exhausts the entries in all the lists, by using some such device as we used in the preceeding chapter to enumerate the positive rational numbers. Nevertheless, Cantor's *diagonal argument* shows that neither this nor any more ingenious device is available, even to a god, for arranging all the sets of positive integers into a single infinite list. Such a list would be as much an impossibility as a round square: the impossibility of enumerating all the sets of positive integers is as absolute as the impossibility of drawing a round square, even for Zeus.

Once we have one example of a nonenumerable set, we get others.

2.2 Corollary. The set of real numbers is not enumerable.

Proof: If ξ is a real number and $0 < \xi < 1$, then ξ has a decimal expansion $.x_1x_2x_3\ldots$ where each x_i is one of the cyphers 0–9. Some numbers have two decimal expansions, since for instance $.2999\ldots = .3000\ldots$; so if there is a choice, choose the one with the 0s rather than the one with the 9s. Then associate to ξ the set of all positive integers n such that a 1 appears in the nth place in this expansion. Every set of positive integers is associated to some real number (the sum of 10^{-n} for all n in the set), and so an enumeration of the real numbers would immediately give rise to an enumeration of the sets of positive integers, which cannot exist, by the preceding theorem.

Problems

2.1 Show that the set of all subsets of an infinite enumerable set is nonenumerable.

2.2 Show that if for some or all of the finite strings from a given finite or enumerable alphabet we associate to the string a total or partial function from positive integers to positive integers, then there is some total function on positive integers taking only the values 1 and 2 that is not associated with any string.

2.3 In mathematics, the real numbers are often identified with the points on a line. Show that the set of real numbers, or equivalently, the set of points on the line, is equinumerous with the set of points on the semicircle indicated in Figure 2-3.

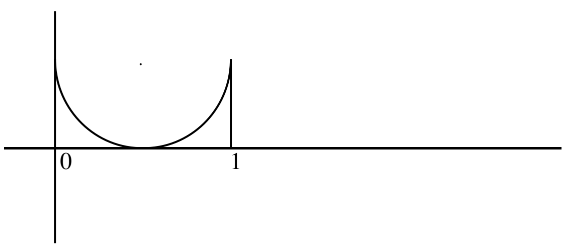

Figure 2-3. Interval, semicircle, and line.

2.4 Show that the set of real numbers ξ with $0 < \xi < 1$, or equivalently, the set of points on the interval shown in Figure 2-3, is equinumerous with the set of points on the semicircle.

2.5 Show that the set of real numbers ξ with $0 < \xi < 1$ is equinumerous with the set of *all* real numbers.

2.6 A real number x is called *algebraic* if it is a solution to some equation of the form

$$c_d x^d + c_{d-1} x^{d-1} + c_{d-2} x^{d-2} + \cdots + c_2 x^2 + c_1 x + c_0 = 0$$

where the c_i are rational numbers and $c_d \neq 0$. For instance, for any rational number r, the number r itself is algebraic, since it is the solution to $x - r = 0$; and the square root $\sqrt{r}$ of r is algebraic, since it is a solution to $x^2 - r = 0$.

(a) Use the fact from algebra that an equation like the one displayed has at most d solutions to show that every algebraic number can be described by a finite string of symbols from an ordinary keyboard.

(b) A real number that is not algebraic is called *transcendental*. Prove that transcendental numbers exist.

2.7 Each real number ξ with $0 < \xi < 1$ has a binary representation $0 \cdot x_1 x_2 x_3 \ldots$ where each x_i is a digit 0 or 1, and the successive places represent halves, quarters, eighths, and so on. Show that the set of real numbers, ξ with $0 < \xi < 1$ and ξ not a rational number with denominator a power of two, is equinumerous with the set of those sets of positive integers that are neither finite nor cofinite.

2.8 Show that if A is equinumerous with C and B is equinumerous with D, and the intersections $A \cap B$ and $C \cap D$ are empty, then the unions $A \cup B$ and $C \cup D$ are equinumerous.

2.9 Show that the set of real numbers ξ with $0 < \xi < 1$ (and hence by an earlier problem the set of *all* real numbers) is equinumerous with the set of all sets of positive integers.

2.10 Show that the following sets are equinumerous:

(a) the set of all pairs of sets of positive integers

(b) the set of all sets of pairs of positive integers

(c) the set of all sets of positive integers.

2.11 Show that the set of points on a line is equinumerous with the set of points on a plane.

2.12 Show that the set of points on a line is equinumerous with the set of points in space.

2.13 (*Richard's paradox*) What (if anything) is wrong with the following argument?

The set of all finite strings of symbols from the alphabet, including the space, capital letters, and punctuation marks, is enumerable; and for definiteness let us use the specific enumeration of finite strings based on prime decomposition. Some strings amount to definitions in English of sets of positive integers and others do not. Strike out the ones that do not, and we are left with an enumeration of all definitions in English of sets of positive integers, or, replacing each definition by the set it defines, an enumeration of all sets of positive integers that have definitions in English. Since some sets have more than one definition, there will be redundancies in this enumeration of sets. Strike them out to obtain an irredundant enumeration of all sets of positive integers that have definitions in English. Now consider the set of positive integers defined by the condition that a positive integer n is to belong to the set if and only if it does *not* belong to the nth set in the irredundant enumeration just described.

This set does *not* appear in that enumeration. For it cannot appear at the nth place for any n, since there is a positive integer, namely n itself, that belongs to this set if and only if it does *not* belong to the nth set in the enumeration. Since this set does not appear in our enumeration, it cannot have a definition in English. And yet it does have a definition in English, and in fact we have just given such a definition in the preceding paragraph.