Diagonalization

In the preceding chapter we introduced the distinction between enumerable and non-enumerable sets, and gave many examples of enumerable sets. In this short chapter we give examples of non-enumerable 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 \( \emptyset \), 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:

\[(*) \quad \text{For each positive integer } n, \text{ } n \text{ is in } \Delta(L) \text{ if and only if } n \text{ is not in } S_n.\]

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 \( n \)th set in the list \( L \). Thus, if \( S_3 \) happens to be the set \( E \) of all 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 \( m \),

\[
\Delta(L) = S_m.
\]

[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

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

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, \ldots 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 \( S_1, S_2, \ldots \) as represented by functions \( s_1, s_2, \ldots \) 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 \( n \)th row represents the function \( s_n \) and thus represents the set \( S_n \). That is,
the $n$th row

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

is a sequence of zeros and ones in which the $p$th 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 $d$th row of Figure 2-1 would then agree with the diagonal sequence entry by entry:

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

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 $n$th row—we should have

$$s_n(1) = 1 - s_1(1), \quad s_n(2) = 1 - s_2(2), \ldots, \quad s_n(m) = 1 - s_m(m), \ldots$$

But the $m$th of these equations cannot hold. [Proof: $s_m(m)$ must be zero or one. If zero, the $m$th equation says that $0 = 1$. If one, the $m$th 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 $n$th 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.

```
0  1/16  1/8  1/4  1/2  1  
```

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

Proof: If \( \xi \) is a real number and \( 0 < \xi < 1 \), then \( \xi \) has a decimal expansion \( \ldots x_1x_2x_3 \ldots \) where each \( x_i \) is one of the cyphers 0–9. Some numbers have two decimal expansions, since for instance \( 0.3999 \ldots = 0.4000 \ldots \); so if there is a choice, choose the one with the 0s rather than the one with the 9s. Then associate to \( \xi \) the set of all positive integers \( n \) such that a 1 appears in the \( n \)th 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.](image-url)
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 \( n \)th set in the irredundant enumeration just described.

This set does not appear in that enumeration. For it cannot appear at the \( n \)th 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 \( n \)th 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.

### 3 Turing Computability

A function is effectively computable if there are definite, explicit rules by following which one could in principle compute its value for any given arguments. This notion will be further explained below, but even after further explanation it remains an intuitive notion.

In this chapter we pursue the analysis of computability by introducing a rigorously defined notion of a Turing-computable function. It will be obvious from the definition that Turing-computable functions are effectively computable. The hypothesis that, conversely, every effectively computable function is Turing computable is known as Turing's thesis. This thesis is not obvious, nor can it be rigorously proved (since the notion of effective computability is an intuitive and not a rigorously defined one), but an enormous amount of evidence has been accumulated for it. A small part of that evidence will be presented in this chapter, with more in chapters to come. We first introduce the notion of Turing machine, give examples, and then present the official definition of what it is for a function to be computable by a Turing machine, or Turing computable.

A superhuman being, like Zeus of the preceding chapter, could perhaps write out the whole table of values of a one-place function on positive integers, by writing each entry twice as fast as the one before; but for a human being, completing an infinite process of this kind is impossible in principle. Fortunately, for human purposes we generally do not need the whole table of values of a function \( f \), but only need the values one at a time, so to speak; given some argument \( n \), we need the value \( f(n) \). If it is possible to produce the value \( f(n) \) of the function \( f \) for argument \( n \) whenever such a value is needed, then that is almost as good as having the whole table of values written out in advance.

A function \( f \) from positive integers to positive integers is called effectively computable if a list of instructions can be given in that principle make it possible to determine the value \( f(n) \) for any argument \( n \). (This notion extends in an obvious way to two-place and many-place functions.) The instructions must be completely definite and explicit. They should tell you at each step what to do, not tell you to go ask someone else what to do, or to figure out for yourself what to do: the instructions should require no external sources of information, and should require no ingenuity to execute, so that one might hope to automate the process of applying the rules, and have it performed by some mechanical device.

There remains the fact that for all but a finite number of values of \( n \), it will be infeasible in practice for any human being, or any mechanical device, actually to carry