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 $A, B, C, \ldots$ *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

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

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 $\text{sum}(m, n) = m + n$ and the multiplication function $\text{prod}(m, n) = m \cdot n$. 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

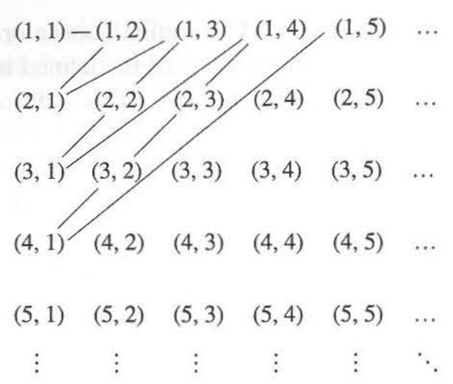

Figure 1-1. Enumerating pairs of positive integers.

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

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

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?