1

Preliminaries

1. Sets and *n*-tuples

We shall often be dealing with *sets* of objects of some definite kind. Thinking of a collection of entities as a *set* simply amounts to a decision to regard the whole collection as a single object. We shall use the word *class* as synonymous with *set*. In particular we write N for the set of *natural numbers* $0, 1, 2, 3, \ldots$. In this book the word *number* will always mean *natural number* except in contexts where the contrary is explicitly stated.

We write

$$a \in S$$

to mean that a belongs to S or, equivalently, is a member of the set S, and

$$a \notin S$$

to mean that a does not belong to S. It is useful to speak of the *empty set*, written $\varnothing$, which has no members. The equation $R = S$, where R and S are sets, means that R and S are *identical as sets*, that is, that they have exactly the same members. We write $R \subseteq S$ and speak of R as a *subset* of S to mean that every element of R is also an element of S. Thus, $R = S$ if and only if $R \subseteq S$ and $S \subseteq R$. Note also that for any set R, $\varnothing \subseteq R$ and $R \subseteq R$. We write $R \subset S$ to indicate that $R \subseteq S$ but $R \neq S$. In this case R

is called a *proper subset* of S. If R and S are sets, we write $R \cup S$ for the *union* of R and S, which is the collection of all objects which are members of either R or S or both. $R \cap S$, the *intersection* of R and S, is the set of all objects that belong to both R and S. $R - S$, the set of all objects that belong to R and do not belong to S, is the *difference* between R and S. S may contain objects not in R. Thus $R - S = R - (R \cap S)$. Often we will be working in contexts where all sets being considered are subsets of some fixed set D (sometimes called a *domain* or a *universe*). In such a case we write $\bar{S}$ for $D - S$, and call $\bar{S}$ the *complement* of S. Most frequently we shall be writing $\bar{S}$ for $N - S$. The De Morgan identities

$$\overline{R \cup S} = \bar{R} \cap \bar{S},$$

$$\overline{R \cap S} = \bar{R} \cup \bar{S}$$

are very useful; they are easy to check and any reader not already familiar with them should do so. We write

$$\{a_1, a_2, \ldots, a_n\}$$

for the set consisting of the n objects $a_1, a_2, \ldots, a_n$. Sets that can be written in this form as well as the empty set are called *finite*. Sets that are not finite, e.g., N, are called *infinite*. It should be carefully noted that a and $\{a\}$ are not the same thing. In particular, $a \in S$ is true if and only if $\{a\} \subseteq S$. Since two sets are equal if and only if they have the same members, it follows that, for example, $\{a, b, c\} = \{a, c, b\} = \{b, a, c\}$. That is, the order in which we may choose to write the members of a set is irrelevant. Where order is important, we speak instead of an n-tuple or a *list*. We write n-tuples using parentheses rather than curly braces:

$$(a_1, \ldots, a_n).$$

Naturally, the elements making up an n-tuple need not be distinct. Thus $(4, 1, 4, 2)$ is a 4-tuple. A 2-tuple is called an *ordered pair*, and a 3-tuple is called an *ordered triple*. Unlike the case for sets of one object, we *do not distinguish between the object a and the* 1*-tuple* (a). The crucial property of n-tuples is

$$(a_1, a_2, \ldots, a_n) = (b_1, b_2, \ldots, b_n)$$

if and only if

$$a_1 = b_1, \quad a_2 = b_2, \quad \ldots, \quad \text{and} \quad a_n = b_n.$$

If $S_1, S_2, \ldots, S_n$ are given sets, then we write $S_1 \times S_2 \times \cdots \times S_n$ for the set of all n-tuples $(a_1, a_2, \ldots, a_n)$ such that $a_1 \in S_1, a_2 \in S_2, \ldots, a_n \in S_n$.

$S_1 \times S_2 \times \cdots \times S_n$ is sometimes called the *Cartesian product* of $S_1, S_2, \ldots, S_n$. In case $S_1 = S_2 = \cdots = S_n = S$ we write S^n for the Cartesian product $S_1 \times S_2 \times \cdots \times S_n$.

2. Functions

Functions play an important role in virtually every branch of pure and applied mathematics. We may define a function simply as a set f, all of whose members are ordered pairs and that has the special property

$$(a, b) \in f \text{ and } (a, c) \in f \quad \text{implies} \quad b = c.$$

However, intuitively it is more helpful to think of the pairs listed as the rows of a table. For f a function, one writes $f(a) = b$ to mean that $(a, b) \in f$; the definition of function ensures that for each a there can be at most one such b. The set of all a such that $(a, b) \in f$ for some b is called the *domain* of f. The set of all $f(a)$ for a in the domain of f is called the *range* of f.

As an example, let f be the set of ordered pairs (n, n^2) for $n \in N$. Then, for each $n \in N$, $f(n) = n^2$. The domain of f is N. The range of f is the set of perfect squares.

Functions f are often specified by *algorithms* that provide procedures for obtaining $f(a)$ from a. This method of specifying functions is particularly important in computer science. However, as we shall see in Chapter 4, it is quite possible to possess an algorithm that specifies a function without being able to tell which elements belong to its domain. This makes the notion of a so-called *partial function* play a central role in computability theory. A *partial function on a set S* is simply a function whose domain is a subset of S. An example of a partial function on N is given by $g(n) = \sqrt{n}$, where the domain of g is the set of perfect squares. If f is a partial function on S and $a \in S$, then we write $f(a)\downarrow$ and say that $f(a)$ is *defined* to indicate that a is in the domain of f; if a is not in the domain of f, we write $f(a)\uparrow$ and say that $f(a)$ is *undefined*. If a partial function on S has the domain S, then it is called *total*. Finally, we should mention that the empty set $\varnothing$ is itself a function. Considered as a partial function on some set S, *it is nowhere defined.*

For a partial function f on a Cartesian product $S_1 \times S_2 \times \cdots \times S_n$, we write $f(a_1, \ldots, a_n)$ rather than $f((a_1, \ldots, a_n))$. A partial function f on a set S^n is called an *n-ary partial function on S*, or a function of n variables on S. We use *unary* and *binary* for 1-ary and 2-ary, respectively. For n-ary partial functions, we often write $f(x_1, \ldots, x_n)$ instead of f as a way of showing explicitly that f is n-ary.

Sometimes it is useful to work with particular kinds of functions. A function f is *one–one* if, for all x, y in the domain of f, $f(x) = f(y)$ implies $x = y$. Stated differently, if $x \neq y$ then $f(x) \neq f(y)$. If the range of f is the set S, then we say that f is an *onto* function with respect to S, or simply that f is *onto* S. For example, $f(n) = n^2$ is one–one, and f is onto the set of perfect squares, but it is not onto N.

We will sometimes refer to the idea of *closure*. If S is a set and f is a partial function on S, then S is *closed under* f if the range of f is a subset of S. For example, N is closed under $f(n) = n^2$, but it is not closed under $h(n) = \sqrt{n}$ (where h is a total function on N).

3. Alphabets and Strings

An *alphabet* is simply some finite nonempty set A of objects called *symbols*. An n-tuple of symbols of A is called a *word* or a *string* on A. Instead of writing a word as $(a_1, a_2, \ldots, a_n)$ we write simply $a_1 a_2 \cdots a_n$. If $u = a_1 a_2 \cdots a_n$, then we say that n is the length of u and write $|u| = n$. We allow a unique null word, written 0, of length 0. (The reason for using the same symbol for the number zero and the null word will become clear in Chapter 5.) The set of all words on the alphabet A is written A^*. Any subset of A^* is called a *language on* A or a *language with alphabet* A. We do *not* distinguish between a symbol $a \in A$ and the word of length 1 consisting of that symbol. If $u, v \in A^*$, then we write $\widehat{uv}$ for the word obtained by placing the string v after the string u. For example, if $A = \{a, b, c\}$, $u = bab$, and $v = caa$, then

$$\widehat{uv} = babcaa \qquad \text{and} \qquad \widehat{vu} = caabab.$$

Where no confusion can result, we write uv instead of $\widehat{uv}$. It is obvious that, for all u,

$$u0 = 0u = u,$$

and that, for all u, v, w,

$$u(vw) = (uv)w.$$

Also, if either $uv = uw$ or $vu = wu$, then $v = w$.

If u is a string, and $n \in N$, $n > 0$, we write

$$u^{[n]} = \underbrace{u\,u \cdots u}_{n}.$$

We also write $u^{[0]} = 0$. We use the square brackets to avoid confusion with numerical exponentiation.

If $u \in A^*$, we write u^R for u written backward; i.e., if $u = a_1 a_2 \cdots a_n$, for $a_1, \ldots, a_n \in A$, then $u^R = a_n \cdots a_2 a_1$. Clearly, $0^R = 0$ and $(uv)^R = v^R u^R$ for $u, v \in A^*$.

4. Predicates

By a *predicate* or a *Boolean-valued function* on a set S we mean a *total* function P on S such that for each $a \in S$, either

$$P(a) = \text{TRUE} \qquad \text{or} \qquad P(a) = \text{FALSE},$$

where TRUE and FALSE are a pair of distinct objects called *truth values*. We often say $P(a)$ *is true* for $P(a) = \text{TRUE}$, and $P(a)$ *is false* for $P(a) = \text{FALSE}$. For our purposes it is useful to identify the truth values with specific numbers, so we set

$$\text{TRUE} = 1 \qquad \text{and} \qquad \text{FALSE} = 0.$$

Thus, a predicate is a special kind of function with values in N. Predicates on a set S are usually specified by expressions which become statements, either true or false, when variables in the expression are replaced by symbols designating fixed elements of S. Thus the expression

$$x < 5$$

specifies a predicate on N, namely,

$$P(x) = \begin{cases} 1 & \text{if } x = 0, 1, 2, 3, 4 \\ 0 & \text{otherwise.} \end{cases}$$

Three basic operations on truth values are defined by the tables in Table 4.1. Thus if P and Q are predicates on a set S, there are also the predicates $\sim P$, $P \,\&\, Q$, $P \vee Q$. $\sim P$ is true just when P is false; $P \,\&\, Q$ is true when both P and Q are true, otherwise it is false; $P \vee Q$ is true when either P or Q or both are true, otherwise it is false. Given a predicate P

Table 4.1

p	$\sim p$
0	1
1	0

p	q	$p \,\&\, q$	$p \vee q$
1	1	1	1
0	1	0	1
1	0	0	1
0	0	0	0

on a set S, there is a corresponding subset R of S, namely, the set of all elements $a \in S$ for which $P(a) = 1$. We write

$$R = \{a \in S | P(a)\}.$$

Conversely, given a subset R of a given set S, the expression

$$x \in R$$

defines a predicate on S, namely, the predicate defined by

$$P(x) = \begin{cases} 1 & \text{if } x \in R \\ 0 & \text{if } x \notin R. \end{cases}$$

Of course, in this case,

$$R = \{x \in S | P(x)\}.$$

The predicate P is called the *characteristic function* of the set R. The close connection between sets and predicates is such that one can readily translate back and forth between discourse involving one of these notions and discourse involving the other. Thus we have

$$\{x \in S \mid P(x) \,\&\, Q(x)\} = \{x \in S \mid P(x)\} \cap \{x \in S \mid Q(x)\},$$
$$\{x \in S \mid P(x) \vee Q(x)\} = \{x \in S \mid P(x)\} \cup \{x \in S \mid Q(x)\},$$
$$\{x \in S \mid \sim P(x)\} = S - \{x \in S \mid P(x)\}.$$

To indicate that two expressions containing variables define the same predicate we place the symbol $\Leftrightarrow$ between them. Thus,

$$x < 5 \Leftrightarrow x = 0 \vee x = 1 \vee x = 2 \vee x = 3 \vee x = 4.$$

The De Morgan identities from Section 1 can be expressed as follows in terms of predicates on a set S:

$$P(x) \,\&\, Q(x) \Leftrightarrow \sim(\sim P(x) \vee \sim Q(x)),$$
$$P(x) \vee Q(x) \Leftrightarrow \sim(\sim P(x) \,\&\, \sim Q(x)).$$

5. Quantifiers

In this section we will be concerned exclusively with predicates on N^m (or what is the same thing, m-ary predicates on N) for different values of m. Here and later we omit the phrase "on N" when the meaning is clear.

Thus, let $P(t, x_1, \ldots, x_n)$ be an $(n + 1)$-ary predicate. Consider the predicate $Q(y, x_1, \ldots, x_n)$ defined by

$$Q(y, x_1, \ldots, x_n) \Leftrightarrow P(0, x_1, \ldots, x_n) \vee P(1, x_1, \ldots, x_n) \vee \cdots \vee P(y, x_1, \ldots, x_n).$$

Thus the predicate $Q(y, x_1, \ldots, x_n)$ is true just in case there is a value of $t \leq y$ such that $P(t, x_1, \ldots, x_n)$ is true. We write this predicate Q as

$$(\exists t)_{\leq y} P(t, x_1, \ldots, x_n).$$

The expression "$(\exists t)_{\leq y}$" is called a *bounded existential quantifier*. Similarly, we write $(\forall t)_{\leq y} P(t, x_1, \ldots, x_n)$ for the predicate

$$P(0, x_1, \ldots, x_n) \,\&\, P(1, x_1, \ldots, x_n) \,\&\, \cdots \,\&\, P(y, x_1, \ldots, x_n).$$

This predicate is true just in case $P(t, x_1, \ldots, x_n)$ is true for *all* $t \leq y$. The expression "$(\forall t)_{\leq y}$" is called a *bounded universal quantifier*. We also write $(\exists t)_{< y} P(t, x_1, \ldots, x_n)$ for the predicate that is true just in case $P(t, x_1, \ldots, x_n)$ is true for at least one value of $t < y$ and $(\forall t)_{< y} P(t, x_1, \ldots, x_n)$ for the predicate that is true just in case $P(t, x_1, \ldots, x_n)$ is true for all values of $t < y$.

We write

$$Q(x_1, \ldots, x_n) \Leftrightarrow (\exists t) P(t, x_1, \ldots, x_n)$$

for the predicate which is true if there exists some $t \in N$ for which $P(t, x_1, \ldots, x_n)$ is true. Similarly, $(\forall t) P(t, x_1, \ldots, x_n)$ is true if $P(t, x_1, \ldots, x_n)$ is true for all $t \in N$.

The following generalized De Morgan identities are sometimes useful:

$$\sim (\exists t)_{\leq y} P(t, x_1, \ldots, x_n) \Leftrightarrow (\forall t)_{\leq y} \sim P(t, x_1, \ldots, x_n),$$

$$\sim (\exists t) P(t, x_1, \ldots, x_n) \Leftrightarrow (\forall t) \sim P(t, x_1, \ldots, x_n).$$

The reader may easily verify the following examples:

$$(\exists y)(x + y = 4) \Leftrightarrow x \leq 4,$$

$$(\exists y)(x + y = 4) \Leftrightarrow (\exists y)_{\leq 4}(x + y = 4),$$

$$(\forall y)(xy = 0) \Leftrightarrow x = 0,$$

$$(\exists y)_{\leq z}(x + y = 4) \Leftrightarrow (x + z \geq 4 \,\&\, x \leq 4).$$

6. Proof by Contradiction

In this book we will be calling many of the assertions we make *theorems* (or *corollaries* or *lemmas*) and providing *proofs* that they are correct. Why are proofs necessary? The following example should help in answering this question.

Recall that a number is called a *prime* if it has *exactly two distinct divisors*, itself and 1. Thus 2, 17, and 41 are primes, but 0, 1, 4, and 15 are not. Consider the following assertion:

$$n^2 - n + 41 \text{ is prime for all } n \in N.$$

This assertion is in fact *false*. Namely, for $n = 41$ the expression becomes

$$41^2 - 41 + 41 = 41^2,$$

which is certainly not a prime. However, the assertion is true (readers with access to a computer can easily check this!) for all $n \leq 40$. This example shows that inferring a result about all members of an infinite set (such as N) from even a large finite number of instances can be very dangerous. A proof is intended to overcome this obstacle.

A proof begins with some initial statements and uses logical reasoning to infer additional statements. (In Chapters 12 and 13 we shall see how the notion of logical reasoning can be made precise; but in fact, our *use* of logical reasoning will be in an informal intuitive style.) When the initial statements with which a proof begins are already accepted as correct, then any of the additional statements inferred can also be accepted as correct. But proofs often cannot be carried out in this simple-minded pattern. In this and the next section we will discuss more complex proof patterns.

In a *proof by contradiction*, one begins by supposing that the assertion we wish to prove is false. Then we can feel free to use the negation of what we are trying to prove as one of the initial statements in constructing a proof. In a proof by contradiction we look for a pair of statements developed in the course of the proof which *contradict* one another. Since both cannot be true, we have to conclude that our original supposition was wrong and therefore that our desired conclusion is correct.

We give two examples here of proof by contradiction. There will be many in the course of the book. Our first example is quite famous. We recall that every number is either even (i.e., $= 2n$ for some $n \in N$) or odd (i.e., $= 2n + 1$ for some $n \in N$). Moreover, if m is even, $m = 2n$, then $m^2 = 4n^2 = 2 \cdot 2n^2$ is even, while if m is odd, $m = 2n + 1$, then $m^2 = 4n^2 + 4n + 1 = 2(2n^2 + 2n) + 1$ is odd. We wish to prove that the equation

$$2 = (m/n)^2 \tag{6.1}$$

has no solution for $m, n \in N$ (that is, that $\sqrt{2}$ is not a "rational" number). We suppose that our equation has a solution and proceed to derive a contradiction. Given our supposition that (6.1) has a solution, it must have a solution in which m and n are not both even numbers. This is true because if m and n are both even, we can repeatedly "cancel" 2 from numerator and denominator until at least one of them is odd. On the other hand, we shall prove that for every solution of (6.1) m and n must both be even. The contradiction will show that our supposition was false, i.e., that (6.1) has no solution.

It remains to show that in every solution of (6.1), m and n are both even. We can rewrite (6.1) as

$$m^2 = 2n^2,$$

which shows that m^2 is even. As we saw above this implies that m is even, say $m = 2k$. Thus, $m^2 = 4k^2 = 2n^2$, or $n^2 = 2k^2$. Thus, n^2 is even and hence n is even. ■

Note the symbol ■, which means "the proof is now complete."

Our second example involves strings as discussed in Section 3.

Theorem 6.1. Let $x \in \{a, b\}^*$ such that $xa = ax$. Then $x = a^{[n]}$ for some $n \in N$.

Proof. Suppose that $xa = ax$ but x contains the letter b. Then we can write $x = a^{[n]}bu$, where we have explicitly shown the first (i.e., leftmost) occurrence of b in x. Then

$$a^{[n]}bua = aa^{[n]}bu = a^{[n+1]}bu.$$

Thus,

$$bua = abu.$$

But this is impossible, since the same string cannot have its first symbol be both b and a. This contradiction proves the theorem. ■

Exercises

1. Prove that the equation $(p/q)^2 = 3$ has no solution for $p, q \in N$.
2. Prove that if $x \in \{a, b\}^*$ and $abx = xab$, then $x = (ab)^{[n]}$ for some $n \in N$.

7. Mathematical Induction

Mathematical induction furnishes an important technique for proving statements of the form $(\forall n)P(n)$, where P is a predicate on N. One

proceeds by proving a pair of auxiliary statements, namely,

$$P(0)$$

and

$$(\forall n)(\textit{If } P(n) \textit{ then } P(n+1)). \tag{7.1}$$

Once we have succeeded in proving these auxiliary statements we can regard $(\forall n)P(n)$ as also proved. The justification for this is as follows.

From the second auxiliary statement we can infer each of the infinite set of statements:

$$\textit{If } P(0) \textit{ then } P(1),$$
$$\textit{If } P(1) \textit{ then } P(2),$$
$$\textit{If } P(2) \textit{ then } P(3), \ldots .$$

Since we have proved $P(0)$, we can infer $P(1)$. Having now proven $P(1)$ we can get $P(2)$, etc. Thus, we see that $P(n)$ is true for all n and hence $(\forall n)P(n)$ is true.

Why is this helpful? Because sometimes it is much easier to prove (7.1) than to prove $(\forall n)P(n)$ in some other way. In proving this second auxiliary proposition one typically considers some fixed but arbitrary value k of n and shows that if we assume $P(k)$ we can prove $P(k+1)$. $P(k)$ is then called the *induction hypothesis*. This methodology enables us to use $P(k)$ as one of the initial statements in the proof we are constructing.

There are some paradoxical things about proofs by mathematical induction. One is that considered superficially, it seems like an example of circular reasoning. One seems to be assuming $P(k)$ for an arbitrary k, which is exactly what one is supposed to be engaged in proving. Of course, one is not really assuming $(\forall n)P(n)$. One is assuming $P(k)$ for some *particular* k in order to show that $P(k+1)$ follows.

It is also paradoxical that in using induction (we shall often omit the word *mathematical*), it is sometimes easier to prove statements by first making them "stronger." We can put this schematically as follows. We wish to prove $(\forall n)P(n)$. Instead we decide to prove the *stronger* assertion $(\forall n)(P(n) \,\&\, Q(n))$ (which of course implies the original statement). Proving the stronger statement by induction requires that we prove

$$P(0) \,\&\, Q(0)$$

and

$$(\forall n)[\textit{If } P(n) \,\&\, Q(n) \textit{ then } P(n+1) \,\&\, Q(n+1)].$$

In proving this second auxiliary statement, we may take $P(k) \,\&\, Q(k)$ as our induction hypothesis. Thus, although strengthening the statement to

be proved gives us more to prove, it also gives us a stronger induction hypothesis and, therefore, more to work with. The technique of deliberately strengthening what is to be proven for the purpose of making proofs by induction easier is called *induction loading.*

It is time for an example of a proof by induction. The following is useful in doing one of the exercises in Chapter 6.

Theorem 7.1. For all $n \in N$ we have $\sum_{i=0}^{n}(2i + 1) = (n + 1)^2$.

Proof. For $n = 0$, our theorem states simply that $1 = 1^2$, which is true.

Suppose the result known for $n = k$. That is, our induction hypothesis is

$$\sum_{i=0}^{k} (2i + 1) = (k + 1)^2.$$

Then

$$\begin{aligned}\sum_{i=0}^{k+1} (2i + 1) &= \sum_{i=0}^{k} (2i + 1) + 2(k + 1) + 1 \\ &= (k + 1)^2 + 2(k + 1) + 1 \\ &= (k + 2)^2.\end{aligned}$$

But this is the desired result for $n = k + 1$. ■

Another form of mathematical induction that is often very useful is called *course-of-values induction* or sometimes *complete induction.* In the case of course-of-values induction we prove the single auxiliary statement

$$(\forall n)[\mathit{If}\ (\forall m)_{m<n} P(m)\ \mathit{then}\ P(n)], \tag{7.2}$$

and then conclude that $(\forall n)P(n)$ is true. A potentially confusing aspect of course-of-values induction is the apparent lack of an initial statement $P(0)$. But in fact there is no such lack. The case $n = 0$ of (7.2) is

$$\mathit{If}\ (\forall m)_{m<0} P(m)\ \mathit{then}\ P(0).$$

But the "induction hypothesis" $(\forall m)_{m<0} P(m)$ is entirely vacuous because there is no $m \in N$ such that $m < 0$. So in proving (7.2) for $n = 0$ we really are just proving $P(0)$. In practice it is sometimes possible to give a single proof of (7.2) that works for all n including $n = 0$. But often the case $n = 0$ has to be handled separately.

To see why course-of-values induction works, consider that, in the light of what we have said about the $n = 0$ case, (7.2) leads to the following

infinite set of statements:

$$P(0),$$
$$\textit{If } P(0) \textit{ then } P(1),$$
$$\textit{If } P(0) \,\&\, P(1) \textit{ then } P(2),$$
$$\textit{If } P(0) \,\&\, P(1) \,\&\, P(2) \textit{ then } P(3),$$
$$\vdots$$

Here is an example of a theorem proved by course-of-values induction.

Theorem 7.2. There is no string $x \in \{a, b\}^*$ such that $ax = xb$.

Proof. Consider the following predicate: *If* $x \in \{a, b\}^*$ and $|x| = n$, *then* $ax \neq xb$. We will show that this is true for all $n \in N$. So we assume it true for all $m < k$ for some given k and show that it follows for k. This proof will be by contradiction. Thus, suppose that $|x| = k$ and $ax = xb$. The equation implies that a is the first and b the last symbol in x. So, we can write $x = aub$. Then

$$aaub = aubb,$$

i.e.,

$$au = ub.$$

But $|u| < |x|$. Hence by the induction hypothesis $au \neq ub$. This contradiction proves the theorem. ■

Proofs by course-of-values induction can always be rewritten so as to involve reference to the principle that if some predicate is true for some element of N, then there must be a least element of N for which it is true. Here is the proof of Theorem 7.2 given in this style.

Proof. Suppose there is a string $x \in \{a, b\}^*$ such that $ax = xb$. Then there must be a string satisfying this equation of minimum length. Let x be such a string. Then $ax = xb$, but, if $|u| < |x|$, then $au \neq ub$. However, $ax = xb$ implies that $x = aub$, so that $au = ub$ and $|u| < |x|$. This contradiction proves the theorem. ■

Exercises

1. Prove by mathematical induction that $\sum_{i=1}^{n} i = n(n + 1)/2$.
2. Here is a "proof" by mathematical induction that if $x, y \in N$, then $x = y$. What is wrong?

Let

$$\max(x, y) = \begin{cases} x & \text{if } x \geq y \\ y & \text{otherwise} \end{cases}$$

for $x, y \in N$. Consider the predicate

$$(\forall x)(\forall y)[\textit{If } \max(x, y) = n, \textit{ then } x = y].$$

For $n = 0$, this is clearly true. Assume the result for $n = k$, and let $\max(x, y) = k + 1$. Let $x_1 = x - 1$, $y_1 = y - 1$. Then $\max(x_1, y_1) = k$. By the induction hypothesis, $x_1 = y_1$ and therefore $x = x_1 + 1 = y_1 + 1 = y$.

3. Here is another incorrect proof that purports to use mathematical induction to prove that all flowers have the same color! What is wrong?

   Consider the following predicate: If S is a set of flowers containing exactly n elements, then all the flowers in S have the same color. The predicate is clearly true if $n = 1$. We suppose it true for $n = k$ and prove the result for $n = k + 1$. Thus, let S be a set of $k + 1$ flowers. If we remove one flower from S we get a set of k flowers. Therefore, by the induction hypothesis they all have the same color. Now return the flower removed from S and remove another. Again by our induction hypothesis the remaining flowers all have the same color. But now both of the flowers removed have been shown to have the same color as the rest. Thus, all the flowers in S have the same color.

4. Show that there are no strings $x, y \in \{a, b\}^*$ such that $xay = ybx$.

5. Give a "one-line" proof of Theorem 7.2 that does not use mathematical induction.