23.5.1 Given integer functions $f_0, f_1, f_2, \ldots$, define an integer function f such that $f(m)/f_n(m) \to \infty$ as $m \to \infty$, for each n. *Hint*: Arrange that $f(m) \geq n f_n(m)$ for all $m \geq n$.

23.5.2 Show that if $a_0 < a_1 < a_2 < \cdots$ is a bounded sequence of real numbers, then $a =$ least upper bound of $\{a_0, a_1, a_2, \ldots\}$ is a "diagonal number" of the sequence in the following sense. There are integers $k_0 < k_1 < k_2 < \cdots$ such that the decimal digits of a exceed those of a_n after the k_nth place.

The last exercise applies the diagonal construction to *any* set I, to show that I has more subsets than members.

23.5.3 Let I be any set, and let $\{S_i\}$ be a collection of subsets of I in one-to-one correspondence with the elements i of I. Show that the natural "diagonal" set S of this collection is a subset of I unequal to each S_i.

23.6 Computability

The notion of computability was first formalized by Turing (1936) and Post (1936), who arrived independently at a definition of computing machine, now called a *Turing machine*. A Turing machine M is given by two finite sets, $\{q_0, q_1, \ldots, q_m\}$ of *internal states* and $\{s_0, s_1, \ldots . s_n\}$ of *symbols*, and a *transition function* T that formalizes the behavior of M for pairs (q_i, s_j). M is visualized as having an infinite tape, divided into squares, each of which can carry one of the symbols s_j. (For most purposes, M is assumed to start on a tape with all but finitely many squares blank: s_0 is taken to denote the blank symbol.) Depending on its internal state q_i, M will make a *transition*: changing s_j to s_k, then moving one square right or left and going into a new state q_l. Thus the transition function is given by finitely many equations

$$T(q_i, s_j) = (m, s_k, q_l),$$

where $m = \pm 1$ indicates a move to right or left.

To use M to compute a function $f : \mathbb{N} \to \mathbb{N}$, some convention must be adopted for inputs (arguments of f) and outputs (values of f). The simplest is seen in Figure 23.1. M starts in state q_0 on the leftmost 1 of a block of n 1's, on an otherwise blank tape, and halts on the leftmost 1 of a block of $f(n)$ 1's, on an otherwise blank tape. M halts by virtue of entering a *halting state*, that is, a state q_h for which M has no transition from the pair $(q_h, 1)$. A *computable function* f is one that can be represented in this way by a Turing machine M.

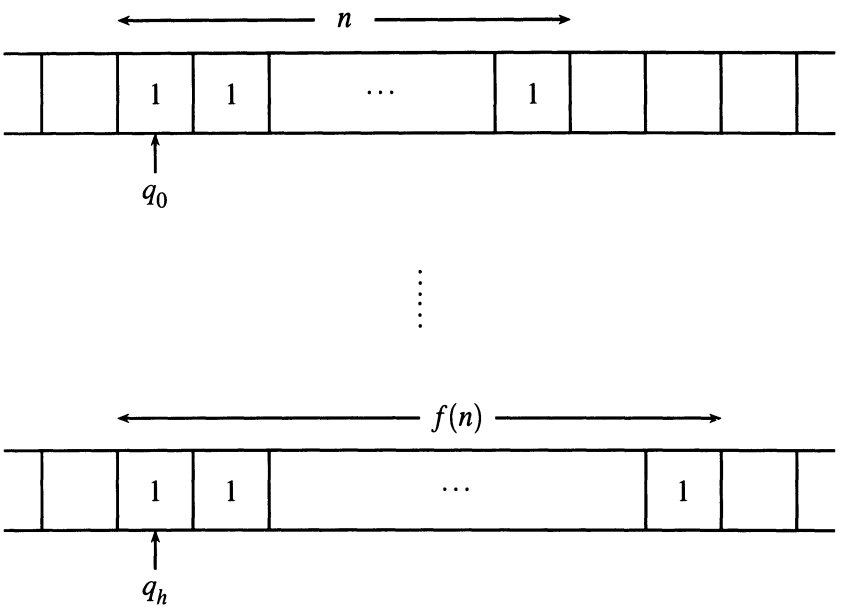

Figure 23.1: Computing a function by Turing machine

It follows that there are only countably many computable functions f : $\mathbb{N} \to \mathbb{N}$ since there are only countably many Turing machines. In fact, we can compute a list of all Turing machines by first listing the finitely many machines with one transition, then those with two transitions, and so forth. This may seem to contradict the discovery from the previous section that a list of all computable functions cannot be computed, but, as Turing (1936) realized, it does not. The catch is that not all machines define functions, and *it is impossible to pick out those that do.* Of course, it is possible to rule out any machine that halts in a situation unlike that in Figure 23.1; the difficulty is in knowing whether halting is going to occur. It is precisely this difficulty that prevents computation of the diagonal function.

If it could be decided, for each machine M and each input, whether M eventually halts, then we could find the first machine to halt on input 1, the next after that to halt on input 2, the next after that to halt on input 3, and so on. By changing the corresponding outputs according to some rule (say, adding 1 if the output is a number, and taking the value 1 otherwise), we could compute a function different from each computable function.

This contradiction shows that the problem of deciding, given a machine and an input, whether halting eventually occurs, is *unsolvable*. This problem is called the *halting problem* and unsolvability means precisely that no Turing machine can solve it. That is, if the questions "Does M on input n eventually halt?" are formulated in some fixed finite alphabet, then there is no machine that, given these questions as inputs, will give their answers as outputs. The significance of this result is that, as far as we know, all possible rules or algorithms for the solution of problems can be realized by Turing machines. This is the "kind of miracle" referred to by Gödel (1946).

Now that computers are everywhere, it is taken for granted that "computability" has a precise, absolute meaning—synonymous with Turing machine computability. It is even a familiar fact that all computations can be done on a single, sufficiently powerful machine; this corresponds to the discovery of Turing (1936) of a *universal Turing machine*. However, these claims were surprising in the 1930s, particularly to Gödel, who had shown (1931) that the related notion of "provability" is *not* absolute. This will be discussed further in the next section. Briefly, the reason for the difference is that new computable functions cannot be created by diagonalization, whereas new theorems can.

The halting problem was of no obvious mathematical significance in 1936, but it seemed no more difficult than other unsolved algorithmic problems in mathematics. Thus for the first time it was reasonable to suspect that some ordinary mathematical problems were unsolvable. Moreover, if it could be shown that a solution of a particular problem P implied a solution of the halting problem, then the unsolvability of P would be rigorously established. This method was used to demonstrate the unsolvability of some problems in formal logic by Turing (1936) and Church (1936). Church (1938) also put forward a strong candidate for unsolvability in ordinary mathematics: the word problem for groups.

This is the problem of deciding, given a finite set of defining relations for a group G (Section 19.6) and a word w, whether $w = 1$ in G. There is more than a superficial analogy between the word problem and the halting problem. G corresponds to a machine M, words in G correspond to expression on M's tape, and $w = 1$ corresponds to halting. The defining relations of G roughly correspond to the transition function of M, but unfortunately there is no machine equivalent of the cancellation of inverses in G. This creates fierce technical difficulties, but they were overcome by

Novikov (1955). He succeeded in establishing the validity of the analogy and hence the unsolvability of the word problem. This led to unsolvability results for a host of significant mathematical problems, among them the homeomorphism problem mentioned in Section 22.7. [The reference given there, Stillwell (1993), also includes a proof of the unsolvability of the word problem.]

A profound reworking of Novikov's ideas, by Higman (1961), shows that computability is a mathematically natural concept in the context of groups. Higman showed that a finitely generated group H has a computable set of defining relations if and only if H is a subgroup of a finitely generated group F with a finite set of defining relations. Thus "computation" is the same as "generation" in a group that is "finitely defined" by generators and relations.

EXERCISES

Turing (1936) actually discovered the unsolvability of the halting problem by considering computable *real numbers* and applying the diagonal argument to them. The argument is similar to the one above using computable functions, but a little messier. Define a real number x to be *computable* if there is a Turing machine M that represents x in the following manner.

- Starting on a blank tape, M prints the decimal digits of x on successive squares of tape, eventually filling each square to the right of the square initially scanned (if necessary, printing all 0s beyond a certain point).
- The squares to the left may be used, and reused, for preliminary computation, but squares to the right, once written, may not be rewritten.

23.6.1 Show that there is no algorithm for recognizing the Turing machines that define real numbers in this way, since such an algorithm would give a way to compute a number different from every computable number.

23.6.2 Explain informally how each Turing machine M may be converted to a machine M' such that M defines a computable number if and only if M' does not halt.

23.6.3 Hence prove that no Turing machine can solve the halting problem.

23.7 Logic and Gödel's Theorem

Since the time of Leibniz, and perhaps earlier, attempts have been made to mechanize mathematical reasoning. Little success was achieved until the late nineteenth century, when the subject matter of mathematics was

clarified by defining all mathematical objects in terms of sets. The reduction of the many concepts of number, space, function, and the like, to the single concept of set brought with it a corresponding reduction in the number of axioms that seemed to be necessary for mathematics. At about the same time, investigation of the principles of logic by Boole (1847), and particularly Frege (1879), led to a system of rules by which all logical consequences of a given set of axioms could be inferred. These two lines of investigation together offered the possibility of a complete, rigorous, and, in principle, *mechanical* system for the derivation of all mathematics.

The most thorough attempt to realize this possibility was the massive *Principia Mathematica* of Whitehead and Russell (1910). *Principia* used axioms of set theory, together with a small collection of rules of inference, to derive a substantial part of ordinary mathematics in a completely formal language. The purpose of the formal language was to avoid the vagueness and ambiguity of natural language, so that proofs could be checked mechanically. Mechanical proof-checking was not then regarded as a goal in itself but rather as a guarantee of rigor. When Whitehead and Russell began writing their *Principia* in 1900, they believed that they were about to reach the nineteenth-century goal of a complete and absolutely rigorous mathematical system. They did not realize that the rigor of their system—the possibility of checking proofs mechanically—was in fact *incompatible* with completeness. Gödel (1931) showed that there are true sentences that can be expressed in the language of *Principia Mathematica* but that do not follow from its axioms. (Unless *Principia* is inconsistent, in which case all sentences follow from its axioms. The assumption of consistency is actually a weighty one, as we shall see by the end of this section.)

Gödel's theorem created a sensation when it first appeared. Not only did it shatter previous conceptions of mathematics and logic, but its proof was of a new and bewildering kind. Gödel exploited the mechanical nature of proof in *Principia* to define the relation "the nth sentence of *Principia* is provable" within the language of *Principia* itself. Using this, he was able to concoct a sentence that says, in effect, "This sentence is not provable." The Gödel sentence, if true, is therefore not provable. And if false, it is provable, and so *Principia* proves a false sentence. Either way, provability in *Principia* is not the same as truth.

Gödel's proof was very difficult for his contemporaries to understand. Combined with the novelty of treating symbols and sentences as mathematical objects was the near inconsistency of a sentence that expresses its

own unprovability (a sentence that says, "This sentence is not true" *is* inconsistent). Post (1944) presented Gödel's theorem in a less paradoxical way by deriving it from the classical diagonal argument. The key to Post's approach is the concept of a *recursively enumerable set.* A set W is called recursively enumerable if a list of its members can be computed, say by a Turing machine that prints them on its tape. (Of course if W is infinite, the computation lasts forever.) The paradigm of a recursively enumerable set is the set of theorems of a formal system, such as *Principia Mathematica.* For such a system one can compute a list of all sentences, a list of all finite sequences of sentences, and, by picking out those sequences that are proofs, a list of all theorems—since a theorem is simply the last line of a proof.

Post's idea was to look at the theorems about recursively enumerable sets proved in a given system Σ and to compute a "diagonal sentence" from them. Since recursively enumerable sets are associated with Turing machines, it is possible to enumerate the recursively enumerable subsets of $\mathbb{N}$ as $W_0, W_1, W_2, \ldots$ by letting W_n be the set of numbers output by the nth machine, under some reasonable convention. (Incidentally, there is no problem of picking out suitable machines, as there is for computable functions, since we do not mind if W_n is empty.) The diagonal set D equal to $\{n : n \notin W_n\}$, being unequal to each W_n, is of course not recursively enumerable, but the following set is:

$$\Pr(D) = \{n : \Sigma \text{ proves } \text{“}n \notin W_n\text{”}\}.$$

This "provable part" of D is recursively enumerable because we can list the theorems of Σ and pick out those of the form "$n \notin W_n$." $\Pr(D) \subseteq D$, assuming Σ proves only correct sentences, but $\Pr(D) \neq D$ since $\Pr(D)$ is recursively enumerable and D is not. This shows immediately that there is an n_0 in D that is not in $\Pr(D)$, that is, an $n_0 \notin W_{n_0}$ for which "$n_0 \notin W_{n_0}$" is not provable.

Better still, a specific n_0 with this property is the index of the recursively enumerable set $\Pr(D)$. If $W_{n_0} = \Pr(D)$, then $n_0 \in W_{n_0}$ is equivalent to $n_0 \in \Pr(D)$, which means that "$n_0 \notin W_{n_0}$" is provable. But then it is true that $n_0 \notin W_{n_0}$, assuming Σ proves only correct sentences, and we have a contradiction. Thus $n_0 \notin W_{n_0}$. This in turn is equivalent to $n_0 \notin \Pr(D)$, which means "$n_0 \notin W_{n_0}$" is not provable. (Notice, incidentally, that the last part of this argument reveals "$n_0 \notin W_{n_0}$" to be a sentence that expresses its own unprovability.)

It seems that Post was aware of this approach to Gödel's theorem in the 1920s, before Gödel's own proof appeared. However, Post's more general view of incompleteness as a property of arbitrary recursively enumerable systems held him up until he was satisfied that computability was a mathematically definable concept. In December 1925 Post formulated a plan for proving *Principia Mathematica* incomplete but, as he later wrote, "The plan, however, included prior calisthenics at other mathematical and logical work, and did not count on the appearance of a Gödel!" [Post (1941), p. 418].

Gödel's theorem comes from reflection on the nature of proofs in ordinary mathematics. An even more devastating theorem, known as Gödel's second theorem, comes from reflection on the proof of Gödel's theorem itself. The latter proof, unusual though it is, can in fact be expressed in ordinary mathematical language.

We described Post's proof of Gödel's theorem in an informal language of Turing machines, but, with some effort, it can be expressed in a small language for number theory called *Peano Arithmetic* (PA). PA is a language of addition and multiplication on $\mathbb{N}$, with basic logic and mathematical induction as the proof machinery. Turing machines can be discussed in PA by interpreting sequences of symbols on the tape as numerals, so that the changes they undergo in the course of a computation become operations on numbers. Under this interpretation, "$n_0 \notin W_{n_0}$" and "Σ does not prove '$n_0 \notin W_0$'" become sentences of PA.

At this point it is important to remember the hypothesis about Σ used in the proof of Gödel's theorem; Σ proves only correct sentences. This assumption cannot be dropped (since one incorrect theorem usually allows *all* sentences to be derived), but it can be weakened to the consistency assumption that Σ does not prove the sentence "$0 = 1$." Since the latter assumption says that a certain element (the number of the sentence "$0 = 1$") does not belong to a certain recursively enumerable set (the set of theorems of Σ), it can be expressed as a sentence of PA, Con(Σ). In particular, PA can express its own consistency by the sentence Con(PA). After these modifications, Gödel's theorem for $\Sigma = \text{PA}$ becomes the following sentence of PA:

$$\text{Con(PA)} \Rightarrow \text{PA does not prove "}n_0 \notin W_{n_0}\text{."}$$

An equivalent sentence is simply

$$\text{Con(PA)} \Rightarrow n_0 \notin W_{n_0},$$

since, as we have seen, the sentence "$n_0 \notin W_{n_0}$" is equivalent to its own unprovability.

Now Gödel noticed that his proof could be carried out in PA. [He he did not publish the details, but the rather laborious verification was carried out by Hilbert and Bernays (1936)]. Consequently, if Con(PA) can be proved in PA, then so can "$n_0 \notin W_{n_0}$," by basic logic. But if PA is consistent, "$n_0 \notin W_{n_0}$" *cannot* be proved in it, by Gödel's theorem, hence neither can Con(PA). [Gödel of course had a different unprovable sentence, but it was similarly implied by Con(PA), and equivalent to its own unprovability.)]

Thus the assertion Con(PA) that the axioms of PA are consistent is in some way stronger than the axioms themselves. Similarly, if Σ is any system that includes PA (such as *Principia Mathematica* and other systems of set theory), then Con(Σ) cannot be proved in Σ, if Σ is consistent. This is Gödel's second theorem.

EXERCISE

It is instructive to spell out why the sentence "$n_0 \notin W_{n_0}$" expresses its own unprovability, if this is not already obvious.

23.7.1 Fill in the gap so as to establish a chain of equivalences:

$$n_0 \notin W_{n_0} \Leftrightarrow \cdots \Leftrightarrow \Sigma \text{ does not prove } "n_0 \notin W_{n_0}".$$

23.8 Provability and Truth

The previous section stressed that Gödel's theorem is a statement of alternatives: a formal system Σ either fails to prove a true sentence or else proves a false one. Gödel's second theorem identifies a sentence, Con(Σ), which is either true and unprovable or false and provable, but the proof does not say which alternative actually holds for a particular Σ, such as PA or *Principia Mathematica*. How could it, without violating Gödel's theorem itself? Unless Σ actually is *inconsistent*, there can be no formal proof that Con(Σ) is true!

Nevertheless, Gödel's theorem tells us that we have nothing to lose by adding Con(Σ) to the system Σ. If Σ is inconsistent, then it is already worthless, and we are no worse off for having added Con(Σ). And if Σ is consistent, we actually gain, because Con(Σ) is a new mathematical truth not provable from Σ alone. In this case, Gödel's theorem gives a way to transcend any given formal system. Knowing that Con(Σ) is beyond the scope of Σ (if Σ is consistent) is of practical value to mathematicians, as it