Chapter 6

Limitations of the Discrete

My purpose here has been ... to show that a specific Gödel proposition — neither provable nor disprovable using the axioms and rules of the formal system under consideration — is clearly *seen*, using our insights into the meanings of the operations in question, to be a *true* proposition!

- Roger Penrose (ENM, p. 116)

The import of Goedel's conclusions is far-reaching, though it has not yet been fully fathomed. ... Goedel's conclusions also have a bearing on the question whether calculating machines can be constructed which would be substitutes for a living mathematical intelligence. ... There is no immediate prospect of replacing the human mind by robots. ... None of this is to be construed, however, as an invitation to despair, or as an excuse for mystery mongering.

- Ernest Nagel and James R. Newman (GP/WM)

In the preceding chapters we have discussed the 2500 year history of two related ideas, one epistemological, the other mathematical. The epistemological idea is that knowledge can be represented in the formulas of a calculus and that cognition is calculation — formal manipulation of those formulas. The mathematical idea is the arithmetization of geometry, which is motivated by the belief that the discrete is fundamentally more comprehensible than the continuous. The latter theme will be brought to its conclusion in this chapter, for we will consider several results that place fundamental limits on the arithmetization of geometry and on the axiomatization of mathematics. Although these results were established in the 1930s and were well-known to the scientific community by mid-century, philosophers, psychologists and AI researchers continued to use discrete, symbolic representations through most of the twentieth century. Therefore, the next two chapters will continue the historical presentation, and discuss the use of calculi in philosophy, cognitive science and artificial intelligence.

In this chapter we will consider important theorems proved by Gödel, Turing and other logicians, and I will try to explain the proofs of these theorems. Nonmathematical readers may wonder why they are being subjected to these proofs, but the quotations that open this chapter show the reason. Gödel's theorem rivals quantum mechanics in the number of unwarranted conclusions it has engendered, often by mathematically sophisticated commentators. No doubt I am also misinterpreting the significance of these results, but I hope at least that readers who understand the proofs will be in a better position to draw their own informed conclusions about their significance. Nevertheless, some technical issues have been separated out, and I suggest that the remainder be skimmed if the going gets too tough. Be cognizant though of the risk you run by taking this route.¹

Like a Rorshach test, quantum mechanics and Gödel's theorem invite the projection of our fears and hopes, and the popular fascination with these two ideas is perhaps a reflection of profound societal changes now in progress.

¹Cognoscenti will no doubt be outraged by my informality, but I have tried to steer a middle course, avoiding a myriad of uninteresting details, while allowing a majority of readers to grasp the essence of the proofs.

6.1 Undecidable Propositions

6.1.1 Gödel's Incompleteness Theorem

If an axiomatic system is consistent and complete, then for each proposition P, exactly one of the pair P and not-P is provable.² This is clearly the most desirable situation, since then the axioms say neither too much nor too little. One of the landmarks of twentieth century logic is Kurt Gödel's 1931 proof that no "reasonably powerful" axiomatic system can be both consistent and complete. So that you will understand the significance of this theorem I will sketch its proof. (If you are interested in the details, see the appendices to this section, beginning on p. 295.)

I've said that Gödel's result applies to "reasonably powerful axiomatic systems." What exactly does this mean? It will be most clear after we've completed the proof, for then you will be able to see what we've assumed. But I can give a rough definition now. We will make use of the usual laws of logic, including the law of the excluded middle. However, the proof is completely constructive, and appeals only to simple properties of the natural numbers. Thus it is acceptable even to intuitionists. Further we will assume that our axiomatic system is completely formal, so that the axioms are just strings of characters and the rules of inference are just string replacement rules (such as Markov algorithms). Since strings of characters can be encoded as integers (just think of the bit strings representing both), the resources of such an axiomatic system are adequate for talking about axiomatic systems (including itself). In the following, let \mathcal{A} be any reasonably powerful axiomatic system.

To prove the completeness of an axiomatic system we must show that every proposition is decidable; to prove its incompleteness we must show that at least one is undecidable. Gödel's great accom-

²Gödel (1906–1978) himself has provided a fairly readable, although somewhat oversimplified, overview of his proof in his original paper (Davis, Undec., pp. 5–9). A well-known popular account is Nagel & Newman (GP), which is abbreviated in Nagel & Newman (GP/WM). See also "Gödel's Theorem" in Edwards (EP, Vol. 3, pp. 348–357). A good general reference for this chapter is Kneebone (MLFM), although there are many other discussions of these topics.

plishment was to consider the possibility that axiomatizations of significant mathematics are incomplete, and thus to try to construct a counterexample.³

Our task is: given a consistent axiomatic system \mathcal{A} , construct a proposition Ω guaranteed to be undecidable in \mathcal{A} . One way to accomplish this is to make Ω a proposition of \mathcal{A} that asserts its own unprovability; then assuming the decidability of Ω will lead to a contradiction. For if Ω is provable then it's true, and hence unprovable (since Ω asserts its own unprovability). Conversely, if $\neg \Omega$ is provable (and hence true), then Ω must not be provable (since \mathcal{A} is consistent), which means Ω is true (since it asserts Ω 's unprovability). Again we have a contradiction. Thus we will have the incompleteness of \mathcal{A} if such an Ω can be constructed.

If Ω asserts its own unprovability then it is a proposition about formulas in \mathcal{A} and their derivability from the axioms of \mathcal{A} by its rules of inference. Hence Ω is a proposition about *strings* and their relationships. Thus, if Ω is to be expressible in \mathcal{A} then \mathcal{A} must be able to express propositions about strings. Any reasonably powerful axiomatic system can do so; in fact it's sufficient that \mathcal{A} be able to express propositions in elementary number theory (such as divisibility and prime numbers, see "Gödel Numbers," p. 295).

Let ω be the string representing Ω ; we've seen that the incompleteness of \mathcal{A} will be established if Ω asserts its own unprovability:

$$\Omega \equiv \neg \mathsf{Provable}(\omega)$$

Unfortunately we have no guarantee such an ω exists. Indeed, since this equation looks suspiciously like Russell's paradoxical set (p. 230), we are well advised to question its existence and to seek a constructive definition. Our doubts are confirmed by attempting an explicit definition of Ω by replacing ω by '¬**Provable**(ω)':

 $\Omega \equiv \neg \mathsf{Provable}(\omega)$

³Mathematicians had good reason to be optimistic about proving consistency and completeness; recall the consistency and completeness results discussed in Section 5.3. Von Neumann is reported to have reproached himself for not proving the incompleteness result because he had never seriously considered its possibility.

$$= \neg \mathsf{Provable}(`\neg \mathsf{Provable}(\omega)`) = \neg \mathsf{Provable}(`\neg \mathsf{Provable}(`\neg \mathsf{Provable}(\omega)`)`) :$$

Thus Ω looks like an infinite formula, which violates the finitary assumptions of formal systems. Therefore the construction of Ω must take a different tack.

Suppose we make a list of all the decidable propositions of some form. If we can then construct a proposition of this form that is guaranteed to not be in the list, then we will have constructed an undecidable proposition. This suggests that we use a diagonalization proof such as Cantor used to show that for any list of rational numbers there is a real number that does not appear in that list (Section 5.1.3).

Therefore we will consider propositions of the form P(s) where s is a string and P is a property of strings. We call such a property decidable if for each s the proposition P(s) is decidable, and we call it undecidable if there is at least one s for which P(s) is undecidable. Each property P is represented by a string p and each proposition P(s)is represented by a string, which we write subst(p,s), which refers to the result of substituting s into p (see "Class Expressions," p. 296 for details).

Now consider all the strings p_1, p_2, \ldots representing decidable properties of strings; let P_i be the corresponding properties. Since $P_i(p_j)$ is decidable for every i and j, we can make a table showing the truth or falsity of these propositions:

	p_1	p_2	p_3	p_4	p_5	p_6	• • •
P_1	Т	Т	F	F	Т	F	• • •
P_2	F	Т	Т	F	F	Т	• • •
P_3	F	F	F	F	F	F	• • •
P_4	F	Т	F	Т	F	Т	
÷	•	÷	÷	:	÷	÷	

Since we assume the list includes *all* decidable properties of strings, we can construct an undecidable property by making it the negation of the

diagonal (shown in boxes):

Thus we want $Q(p_i) \equiv \neg P_i(p_i)$.

Since Q is a property of strings we must define it in terms of the string p_i rather than the property P_i it represents; therefore we represent the proposition $P_i(p_i)$ by the string $subst(p_i, p_i)$. Further, since the P_i are decidable properties, $P_i(p_i)$ is true just when $Provable[subst(p_i, p_i)]$ is true (see p. 296). These observations permit an explicit definition of the property Q:

$$Q(p) \equiv \neg \mathsf{Provable}[\mathsf{subst}(p, p)] \tag{6.1}$$

Clearly Q cannot appear in the list of decidable properties since $Q(p_i)$ is the negation of $P_i(p_i)$ for every i.

Since Q is an undecidable property there must be at least one string s for which Q(s) is an undecidable proposition. Taking a clue from Russell's Paradox, we try q, the string representing Q. Then we have:

$$Q(q) \equiv \neg \mathsf{Provable}[\mathsf{subst}(q, q)] \tag{6.2}$$

This is exactly the string required, as we can see by letting $\Omega \equiv Q(q)$ and $\omega = \text{subst}(q, q)$, which is the string representing Ω . Then Eq. 6.2 becomes

$$\Omega \equiv \neg \mathsf{Provable}(\omega) \tag{6.3}$$

Thus $\Omega \equiv Q(q)$ is exactly the undecidable proposition we sought. See Figures 6.1 and 6.2.

In an inconsistent axiomatic system every formula is provable. On the other hand, as we've just shown, in a consistent system that's sufficiently powerful (i.e., powerful enough to talk about strings or numbers) there is always a proposition Ω that's undecidable. (And note that we can actually construct this proposition; refer to "Definiteness of Ω ," p. 297, to see it.) Thus, if such a system is consistent, it cannot be complete, and if it's complete it cannot be consistent. Alternately, no

288



Figure 6.1: Gödel's Theorem, First Part. The diagram depicts the first part of the proof of Gödel's Incompleteness Theorem, showing that Ω is not provable. The outer box represents the axiomatic system \mathcal{A} , the inner box represents a, the axiomatic system encoded in Gödel numbers or in some other way that allows it to be a subject within \mathcal{A} . Dotted arrows indicate deductions from hypotheses later determined to be false; thin undotted arrows indicate true deductions. Lines with dots on both ends connect contradictory situations. Thick arrows indicate the possibility or impossibility of derivations from 'Axioms', the axioms of \mathcal{A} , or from 'axioms', the encoded axioms of \mathcal{A} in a. We begin at 'hyp.' with the hypothesis that Ω is derivable from the axioms of \mathcal{A} . The dotted arrow shows that $\neg \mathsf{Provable}[\omega]$ is also provable. But the latter is provable if and only if ω is not derivable from the encoded axioms of a, as indicated by the crossed arrow in the inner box. Now derivations in a mirror those in \mathcal{A} , so we must conclude that Ω is not derivable in \mathcal{A} . This contradicts the hypothesis so we conclude that Ω is not provable in \mathcal{A} .



Figure 6.2: Gödel's Theorem, Second Part. The diagram depicts the second part of the proof of Gödel's Incompleteness Theorem, showing that $\neg\Omega$ is not provable in \mathcal{A} . Start with the hypothesis (marked 'hyp.') that $\neg\Omega$ is derivable from the axioms of \mathcal{A} , from which we conclude $\neg\Omega$. A dotted line indicates that the consistency of \mathcal{A} allows us to conclude that Ω is not derivable. Therefore, in the encoded system a we know ω is likewise underivable. Hence we know $\neg \text{Provable}(\omega)$, but this is exactly Ω , which contradicts $\neg\Omega$. Therefore we reject the hypothesis and conclude $\neg\Omega$ is not provable in \mathcal{A} .

reasonably powerful axiomatic system can be both consistent and complete. This is *Gödel's Incompleteness Theorem*. Clearly, this result was devastating to the formalist program.⁴

6.1.2 Corollaries to Gödel's Theorem

Gödel dealt an additional blow to the formalist program, for he showed that a (sufficiently powerful) consistent axiomatic system cannot prove its own consistency. This result is a simple corollary to the incompleteness proof. To see this, let C be any proposition asserting the consistency of \mathcal{A} ; since everything is provable in an inconsistent system (p. 244), C can be an assertion that some well-formed formula is unprovable. This will do:

$$C \equiv \neg \mathsf{Provable}(`0 = 1')$$

The proof of Gödel's Theorem can be easily formalized in \mathcal{A} , so we know that if \mathcal{A} is consistent, then Ω is not provable in it. Since C implies the consistency of \mathcal{A} we can prove the implication:

$$C \rightarrow \neg \mathsf{Provable}(\omega)$$

But $\Omega \equiv \neg \mathsf{Provable}(\omega)$, so we can likewise prove in \mathcal{A} the implication:

$$C \to \Omega$$

Thus, if C (the consistency of \mathcal{A}) were provable in \mathcal{A} , then Ω would also be provable in \mathcal{A} . But since we've seen that Ω is not provable in \mathcal{A} , the consistency of \mathcal{A} must be likewise unprovable.

In summary, Gödel showed that any reasonably powerful, consistent axiomatic system must have undecidable propositions, and that among these is the fact of its own consistency!

⁴The proof outlined above requires \mathcal{A} to satisfy a stronger property than simple consistency; it's called ω -consistency. I pass over this detail for three reasons: (1) reasonable axiomatic systems are ω -consistent; (2) the use of ω -consistency is buried in the proof of the correctness of **Provable** (p. 296), which I've omitted; and (3) Rosser (ESTGC) showed that Gödel's Theorem can be strengthened so as to require only consistency.

We turn to a surprising observation. We have seen that the formula Ω , which asserts the unprovability of Ω , is undecidable in the axiomatic system. Nevertheless, I claim that Ω is true, and prove it by the following *metamathematical* reasoning. We supposed that Ω is provable, and reached a contradiction. Therefore, applying the usual proof by contradiction, we must conclude that Ω is unprovable. That is, we have proved (metamathematically) that Ω is unprovable (in the axiomatic system). Since Ω asserts the unprovability of Ω in the axiomatic system [recall $\Omega \equiv \neg \text{Provable}(\omega)$], we have proved Ω metamathematically. We've decided the undecidable proposition! Of course there's no contradiction here. We proved that Ω was undecidable in the given axiomatic system. It was this very fact that allowed us to then decide Ω by metamathematical reasoning — outside the system.

Although this is an important point, too much can be made of it. For example, the metamathematical proof has been the basis for claims that informal mathematics is inherently more powerful than formal mathematics (Penrose, *ENM*). Therefore the metamathematical proof deserves some scrutiny.

To many people the term *metamathematical* suggests some kind of supramathematical intuition, but, as we've seen, it simply denotes the use of mathematical techniques to reason about mathematics (see Section 5.1.4). This is exactly what we did in Gödel's proof when we defined predicates such as Provable and IsaProof (p. 295). Thus Gödel's proof is metamathematical. Also, contrary to some claims (Penrose, ENM), there is nothing inherently unformalizable about the metamathematical proof (see "Formalizing the Metamathematical Proof," p. 297). It is different from Gödel's proof in that it talks about the truth of propositions, whereas Gödel's talks only about their provability. Nevertheless, it's a routine exercise (see below) to construct a formal system \mathcal{A}' capable of expressing propositions about the truth of the propositions of another system \mathcal{A} . Similarly an \mathcal{A}'' can be constructed that can express the semantics of \mathcal{A}' , and so on as necessary. It could be objected that this very argument shows the greater power of informal mathematics, since the informal metamathematical proof is valid for any axiomatic system \mathcal{A} , whereas the formal version requires constructing a new axiomatic system \mathcal{A}' for each \mathcal{A} . Indeed, informal mathematics can talk about the truth of *its own* propositions.

292

6.1. UNDECIDABLE PROPOSITIONS

But even this self-descriptive ability may be formalized, since we can construct an axiomatic system \mathcal{A}^* capable of expressing propositions about its own semantics. If we do so, however, we will make an interesting discovery: such an axiomatic system must be inconsistent since it is powerful enough to express a contradiction analogous to the Liar Paradox (p. 231): Define the predicate $Q(p) \equiv \neg P(p)$, where P is the interpretation of p, and consider the truth of Q(q), where q is the encoded representation of Q.⁵

Again it might seem that the greater power of informal mathematics has been established, since a formal system with its expressive power must be inconsistent, but this does not follow. Since the Liar Paradox can also be expressed in informal mathematics, it follows that *informal mathematics is inconsistent*, just like \mathcal{A}^* . Indeed, the original Liar Paradox (p. 231) is a creature of informal logic, which is also inconsistent.

The phenomenon to be explained is not the *power* of informal reasoning, since it's already so powerful that it permits the Liar Paradox. Rather, the mystery to be solved is the process by which the community of mathematicians avoids perpetually encountering contradictions. It seems there must be *nonlogical constraints* that keep reasoning in check; I will address this issue in more detail in Section **??** (see also MacLennan, DD).

In the 60 years since Gödel published his result there has been little consensus about its implications. However, we can make the following observations. First, the result is extremely robust; it does not depend on details of the formal system. Obvious escapes, such as going to multivalued logics (logics with truth values in addition to true and false), do not change the result. There *are* systems ("semiformal" systems) that are complete and sufficiently powerful to prove their own consistency, but they diverge radically from the finitary goals of formalism. For example, some have infinitely large rules of inference, while others

⁵This assumes the axiomatic system assigns a truth value to every proposition and so also to Q(q). It is of course possible to design a self-referential axiomatic system if it does not assign a truth value to propositions such as Q(q). It also assumes \mathcal{A}^* is powerful enough to talk about its own syntax (for which arithmetic is sufficient), and to talk about its own semantics (for which set theory is sufficient). See Beth (FM, pp. 335–345) for a detailed discussion.

permit infinitely long proofs (Edwards, EP, Vol. 3, p. 355).

Certainly, if we restrict our attention to formal systems in the conventional sense, which presumes that they are finite (Section 5.5), then Gödel's theorem applies. Any such system (unless it's extraordinarily weak) must have at least one undecidable proposition (unless the additional proposition made it inconsistent). And even if we add this proposition as an additional axiom, the resulting formal system must still have undecidable propositions. And yet all these propositions may be decided by metamathematical reasoning (which is just the garden variety mathematical reasoning applied to formal systems). Thus it seems that there is a sense in which a formal system can never capture the process of mathematics. This much is clear. Further implications are much less apparent. (See also Section 6.4.)

294

Gödel Numbers

Gödel wanted to reason about proofs, so he needed a representation for formulas and sequences of formulas. Now we would use strings of characters or linked lists, but Gödel didn't have these computer programming concepts, so he represented a sequence of numbers n_1, n_2, \ldots, n_k by the number

$$N = p_1^{n_1} \times p_2^{n_2} \times \cdots \times p_k^{n_k}$$

where p_1, \ldots, p_k are the first k prime numbers. By the prime factorization theorem, the *i*-th element of the sequence could be extracted by calculating the exponent of p_i in N. Sequences of characters were then represented by sequences of numbers, each number representing a character (now, we would probably use its ASCII code). Recall that Leibniz used the prime factorization theorem to represent finite sets (p. 116).

When we deal with an axiomatic system metamathematically, we treat formulas and proofs as string of characters (or, equivalently, natural numbers). Thus a relationship among formulas, such as being derivable by a given rule of inference, is just a relationship among strings (or natural numbers). Although it's tedious, it's not hard to define a predicate **Provable** so that **Provable**(p) means that the string p is derivable in the axiomatic system \mathcal{A} from its axioms and by its rules of inference. Just to give the idea, here is the beginning of the top-down definition of this predicate:

 $\begin{array}{lll} \mathsf{Provable}(e) &\equiv & \exists p \{ \mathsf{ProofOf}(p, e) \} \\ \mathsf{ProofOf}(p, e) &\equiv & \mathsf{IsaProof}(p) \land e = \mathsf{last}(p) \\ & \mathsf{IsaProof}(p) &\equiv & \mathsf{Axiom}(p) \lor \\ & & \exists q \, \exists s \{ p = \mathsf{postfix}(q, s) \land \mathsf{DerivableFrom}(s, q) \} \end{array}$

These definitions make use of simple operations on sequences of strings (such as **last**, which returns the last element of the sequence, and **postfix**, which adds an element to the end of the sequence), which also must be defined. Ultimately we get down to basic properties of strings (such as one being a substring of another), but these are easy to define in any reasonably powerful axiomatic system. If these definitions are carried out correctly, then we will be able to prove:

P is provable in \mathcal{A} if and only if $\mathsf{Provable}(p)$ is provable in \mathcal{A} , where *p* is the string representing proposition *P*.

Class Expressions

By formula we mean a syntactically legal string in the language of the formal system \mathcal{A} , and by class expression we mean a formula with one free variable (i.e., one variable not "bound" by a quantifier). This is an example of a class expression:

$$\exists m \{n = 2 \times m\},\$$

(In this case 'n' is a free variable and 'm' a bound variable.) Intuitively, this formula denotes the class of all even numbers.

It is simple to write a program that substitutes one string for another. Therefore we assume that we have a function **subst** such that **subst**(p, s) replaces the free variable of the class expression p by the string s. For example, if $p = \exists m \{n = 2 \times m\}$ ', then subst(p, `17') replaces 'n' by '17' yielding:

$$subst(p, '17') = `\exists m \{17 = 2 \times m\}'$$

As we've said, a class expression is intended to represent the class of numbers possessing the denoted property. Thus the formula returned by subst(p, s) can be interpreted as the proposition that s is a member of the class defined by p.

Definiteness of Ω

Notice that we have *constructed* the undecidable proposition Ω . To see this, recall

$$Q(q) \equiv \neg \mathsf{Provable}[\mathsf{subst}(s, s)] \text{ and}$$

 $q \equiv `\neg \mathsf{Provable}[\mathsf{subst}(s, s)]'$

Then expand the definition of Ω :

$$\begin{split} \Omega &\equiv Q(q) \\ &\equiv \neg \mathsf{Provable[subst}(q,q)] \\ &\equiv \neg \mathsf{Provable[subst}(`\neg \mathsf{Provable[subst}(s,s)]', \\ `\neg \mathsf{Provable[subst}(s,s)]')] \\ &\equiv \neg \mathsf{Provable}(`\neg \mathsf{Provable[subst}(`\neg \mathsf{Provable[subst}(s,s)]', \\ `\neg \mathsf{Provable[subst}(s,s)]')]') \end{split}$$

You can now see that Ω is a perfectly definite proposition; it and the corresponding string ω are 65 characters long (not counting blanks).

Formalizing the Metamathematical Proof

To carry out a formal equivalent of the metamathematical proof would require many tedious constructions that would add little to understanding. Therefore my goal here will be to give just enough detail to make it plausible that the proof can be formalized. As before, we have the axiomatic system \mathcal{A} and the undecidable proposition Ω constructed according to the Gödel procedure. Since $\Omega = \neg \text{Provable}(\omega)$ means that ω , the embedded replica of Ω , is not provable in a, the embedded replica of \mathcal{A} , we see that Ω makes a true assertion. However, since the proof refers to the meaning of Ω , it's necessary to construct a model for \mathcal{A} . Therefore, the formal system \mathcal{A}' in which the metamathematical proof will be expressed must be sufficiently powerful to allow the construction of formal interpretations. To accomplish this we need to be able to talk about the formulas of \mathcal{A} , for which arithmetic is sufficient, as we've seen, and we need to be able to define functions mapping these formulas into various subsets of the domain of interpretation, which is a set. Therefore, the mathematical apparatus of set theory is sufficient for defining interpretations, and set theory can be formalized by means of the Zermelo-Fraenkel axioms (p. 310) — though no one knows if they are consistent. Since ZF is sufficient to define arithmetic, we can take \mathcal{A}' to be ZF without loss of generality.

To show in \mathcal{A}' that Ω is true, we must formally derive $\mathcal{I}\{\Omega\}$, the interpretation in \mathcal{A}' of Ω . First express Gödel's proof formally in \mathcal{A}' ; it should be clear that this can be done, because the proof uses only the most elementary proof techniques. Suppose the formal expression of the result is the following proposition of \mathcal{A}' :

$$\mathsf{Consistent}(\mathcal{A}) \to \neg \mathsf{ProvableIn}(\Omega, \mathcal{A}).$$

Now Gödel's proof hinges on the construction of the embedded system a so that '**Provable**(ω)' is derivable in \mathcal{A} just when Ω is derivable in \mathcal{A} . Expressed formally in \mathcal{A}' this is:

```
\mathsf{ProvableIn}(\Omega, \mathcal{A}) \leftrightarrow \mathsf{ProvableIn}(\mathsf{'Provable}(\omega)', \mathcal{A}).
```

The interpretation of the latter proposition is:

 $\mathsf{ProvableIn}(\mathsf{'Provable}(\omega)\mathsf{'},\mathcal{A}) \leftrightarrow \mathsf{ProvableIn}(\omega,a).$

Now notice that the interpretation in \mathcal{A}' of Ω is:

 $\mathcal{I}\{\Omega\} \leftrightarrow \mathcal{I}\{`\neg\mathsf{Provable}(\omega)`\} \leftrightarrow \neg\mathsf{Provableln}(\omega,a).$

Combining the implications we have:

 $Consistent(\mathcal{A}) \to \mathcal{I}\{\Omega\}.$

Therefore, we have a formal proof that if \mathcal{A} is consistent then its Gödel proposition Ω is true. (Of course, an inconsistent axiomatic system has no models, and so we cannot even talk of its propositions being true or false.)

298

6.2 The Undecidable and the Uncomputable

6.2.1 Introduction

In this section we investigate Alan Turing's (1912–1954) famous proof of the undecidability of the halting problem.⁶ This result and its generalization — Rice's theorem — demonstrate inherent limitations to digital computation, and reveal an essential unpredictability in formal systems.

If you have ever programmed a computer you know that if you make a mistake your program may "go into an infinite loop." That is, it will run forever (or as long as you let it run), without ever stopping and returning an answer. A common predicament, when running a new program, is not knowing whether it's in an infinite loop. It's run for a minute so far, which is longer than you thought it should run. But does that mean that it's in an infinite loop, or only that it's slower than expected? You let it run another five minutes, and it still hasn't halted. Now you're becoming very suspicious, but you're still not sure that it won't return its answer in the next second or so. The trouble is of course that you never know for sure whether it will halt until in fact it does halts. It would surely be useful to have a way of telling in advance whether the program will halt. Then we would know we're not waiting in vain. This is the *halting problem*.

Since a program may halt on some inputs but not on others, we would like to know whether a given program will halt when run on a given input. A procedure (i.e., a program) for deciding this is called a *decision procedure* for the halting problem. We can imagine that this would be a very complicated procedure, analyzing the text of the program, and tracing its behavior on the given input. Nevertheless

⁶The primary source for this section is Turing (OCN), which is reprinted in Davis (Undec., pp. 116–154). Turing's proof is discussed in most books on computability theory and theoretical computer science.

it would be valuable. There are of course many other questions we would like to ask about programs (when run on given inputs), such as whether they will ever try to divide by zero, whether they will return a particular output, and on and on. It would be useful to have decision procedures for all these problems. The remarkable thing that Turing proved is that there is no decision procedure for the halting problem, and a simple extension of his proof shows that there is no decision procedure for just about any property of interest. To understand this fundamental limitation of computers, it's important to see how it's proved. Therefore I'll present an informal overview of Turing's proof (using modern programming notations rather than Turing machines). The similarity to Gödel's proof will be apparent.

6.2.2 Undecidability of the Halting Problem

This will be a proof by contradiction, much like Gödel's proof. Therefore we suppose that we have a Boolean-valued procedure Halts(p, i)which returns **true** if program P halts on input i, and returns **false** otherwise. We assume that the program P is represented as a string of characters p in the obvious way.⁷ Technically, p is a string representing a procedure declaration. For simplicity we will also assume that the input i is a string of characters; it will become clear that this does not limit the generality of the proof. In the Pascal⁸ programming language the declaration of Halts would look like this:

procedure Halts (p, i: string): Boolean;

300

⁷I will use capital letters such as P to refer to programs (you can think of them as machine code loaded into the computer's memory). I will use small letters such as p to refer to the *source code* for the program: a string of characters in some programming language. Strictly speaking, a program P can be applied to some input, but program text p cannot. Also, a decision procedure can analyze program text p, but not the program P itself. For the most part these distinctions can be ignored, however.

⁸Pascal is one of the most popular modern programming languages. Although I use its notation, it should be clear that the proof could be carried through using any programming notation, including Markov algorithms. Turing of course used Turing machines (Section 5.4.2). The principal reference for Pascal is Jensen & Wirth (PUMR).

begin

end {Halts};

Turing's proof, like Cantor's and Gödel's, is a diagonalization argument. In this case, since we are considering programs whose inputs are strings, the diagonal is where the program is applied to itself (more precisely, to the string representing itself). When a program is "self applied" in this way it will either halt or not. As is usual in diagonal proofs, we will construct a procedure that alters the diagonal. This procedure Q will halt if a given program does not halt when self applied, and will not halt, if the given program does halt under self-application. More precisely, Q(p) halts if and only if P(p) doesn't.

procedure Q (p: string); ... declaration of Halts ... begin if Halts (p, p) then 1: goto 1; end {Q};

This is how Q works. It takes the input string p (representing a program) and passes it to Halts as both the program and the input: Halts(p, p). We have assumed that Halts will tell us correctly whether P(p) halts. If P(p) does halt, then Q goes into an infinite loop (1: goto 1); otherwise it returns immediately (and therefore halts).

It should now be obvious how we will get our contradiction. Let q be the program text representing the declaration of Q:

q = 'procedure Q (p: string); . . . end {Q}; '

Consider the result of applying Q to this string, Q(q). As we saw, Q(p) halts if and only if P(p) doesn't halt. Therefore Q(q) halts if and only if Q(q) doesn't halt, which is a contradiction. More carefully, in

executing Q(q) we compute $\mathsf{Halts}(q, q)$. We have assumed that this tells us correctly whether Q(q) halts. But whatever Halts says is contravened by Q; if it says Q(q) halts, then Q loops forever; if it says it loops forever, then Q halts immediately. Thus our assumption, that Halts correctly decides the halting problem, must be wrong. We are forced to conclude that there is no decision procedure for the halting problem (see also "Picture of the Diagonal," p. 304).

6.2.3 General Undecidability

We have seen that the halting problem is undecidable. You might suspect that this is a peculiarity of this problem, and that other interesting problems might be decidable. Unfortunately this is not the case. There is a generalization of Turing's results, known as Rice's theorem (Rice, CRES), which says that all interesting problems are undecidable. It will be easier to say what is meant by "interesting" after I sketch the proof.

The proof follows the same outline as Turing's. Assume that we have a decision procedure DoesX(p, i), which tells us if a program P does something interesting X when applied to an input i. Then construct a diagonal procedure Q as before:

procedure Q (p: string); . . . declaration of DoesX . . . begin if DoesX (p, p) then don't do X else do X; end {Q};

In other words, if P(p) does X, then Q(p) doesn't do X; if P(p) doesn't do X then Q(p) does do it. The contradiction arises when we ask whether Q(q) does X, for Q is constructed so that Q(q) does X if and only if Q(q) doesn't do X. Therefore there can be no decision procedure for determining whether a program does X. But what is X?

It is virtually anything. The only real restriction is that it must be

in the power of the program to do it or not do it, otherwise we cannot construct Q. This includes just about any property of interest (e.g., dividing by zero, returning a particular number). Roughly, if it's not in the power of the language to do X, then there's not much point in a decision procedure that tells if a program does X, since in fact it never will.

If we look carefully at the proof of these undecidability results, then we can see some hidden assumptions in them. Bringing these assumptions to light will help us to understand the scope and limitations of these results. We have already noted that the proof assumes that it's possible to "do" or "not do" the thing in question. In general, most logical properties of the program are controllable, although some physical properties (such as the amount of space or time used by the program) may not be. Another assumption is that the procedure Q can be constructed. For example, we have assumed that we can perform a conditional test (if ... then ... else ...), although this is hardly a questionable assumption. More significantly, we have assumed that there is no limit on the size of a program. For example, if the largest program allowed were one million characters, and if it took 999999 characters to define DoesX, then we would not be able to construct Q; it would be too big. Of course, when we define programming languages, and study the logical properties of computers, we avoid putting arbitrary limits on the sizes of things. On the other hand, it's important to keep in mind that most of these results depend on the potential infinities (i.e. infinite producibility, p. 255) that abound in the theory of computation and formal language theory. All real computers are finite, as are the programs that run on them. Real computers are equivalent to finitestate machines, not Turing Machines. Therefore we must be careful in applying these undecidability results to real computers and programs. (See the appendices beginning on p. 304 for the halting problem for finite state machines, and for an example of a property to which Rice's theorem does not apply.)

We have been talking about programs, but they are just the final culmination of the idea of a calculus: finite arrangements of uninterpreted tokens manipulated mechanically according to finite, formal rules. Thus these undecidability results inform us of the inherent limitations of discrete formal systems (calculi). On one hand, formal systems are too weak: they are incapable of deciding many interesting questions, in particular, most any property of formal systems in general. On the other hand, they are too powerful. They are so unpredictable that most of their interesting properties are undecidable by any rigorous (mechanizable) process. Formal systems are too weak to determine their own power.

Picture of the Diagonal

Let p_1, p_2, \ldots be a list of all the procedure declarations and P_1, P_2, \ldots the corresponding procedures. Then we can make a table of the truth value returned for each pair (P_i, p_j) . The table might look like this:

	p_1	p_2	p_3	p_4	p_5	p_6	•••
P_1	Т	Т	F	F	Т	F	• • •
P_2	\mathbf{F}	Т	Т	\mathbf{F}	\mathbf{F}	Т	• • •
P_3	F	F	F	\mathbf{F}	\mathbf{F}	\mathbf{F}	
P_4	\mathbf{F}	Т	F	Т	F	Т	• • •
:		÷	•	•	÷	÷	

The diagonal elements are in boxes. We have constructed a procedure Q whose halting behavior differs from the diagonal:

Now we can see the contradiction directly. We have assumed that every program appears somewhere in the list P_1, P_2, \ldots But by construction Q differs in its halting behavior from every program in the list. Therefore Q cannot appear in the list. Since the existence of the procedure Halts is the only questionable thing required for the definition of Q, it must be Halts that doesn't exist.

Example of a Property Not Covered

For an example of a property X not under the control of the procedure Q, consider "halts in ten seconds". Intuitively is seems like this property ought to be decidable: just run the program for ten seconds, and at the end of that time return true if the program has halted and false if it hasn't. And in fact our proof of the general undecidability result does not contradict this, for it's not necessarily in our power to "do X". For example, suppose **DoesX** requires at least ten seconds to run (as it would for our hypothesized decision procedure). Then it's no longer in Q's power to halt within ten seconds, since more than ten seconds have already elapsed. Of course this does not mean that "halts in ten seconds" *is* decidable; it only means that our proof does not show that it's undecidable. However, our intuitive argument suggests it is decidable.

Decidability of Halting Problem for FSMs

All real computers are finite state machines (all the memory cells, registers, etc. together can be in only a finite — though very large — number of states). But we can decide the halting problem for an N-state finite state machine as follows: Run the machine for N + 1 cycles. If it has not halted by N +1 cycles, then it's in an infinite loop, since there are only N states, and whenever it returns to a previously visited state it must thereafter repeat the states that followed that state. (This assumes that the machine is deterministic, i.e., that its future action is determined by its present state.)

Although real computers are finite-state machines, computer science theory uses the Turing machine model, since the number of states is so large. (If a computer has one megabyte of memory, then the number of states is $2^{8 \times 2^{20}} \approx 4 \times 10^{2525222}$, and this doesn't count internal registers or auxiliary memory, such as disks. In effect, it's presumed that this number is sufficiently large that it's a good approximation to infinity. Nevertheless, it's worth keeping in mind that real computers are not *in fact* equivalent to Turing machines. Infinite producibility is an idealizing assumption.

Relation of Turing's and Gödel's Theorems

It is not coincidental that Gödel's and Turing's proofs are so similar; they are really equivalent results. For example, you can see that Turing's theorem implies Gödel's as follows. It is straightforward, though rather tedious, to define in an axiomatic system \mathcal{A} a predicate Halts(p, i) that is true just when the procedure P (defined by string p) halts on input string i. (This assumes that \mathcal{A} is sufficiently rich to express the semantics of a programming language, but this requires nothing beyond elementary number theory.) The proposition Halts(q,q) (where q is the string representing procedure Q) must be undecidable in \mathcal{A} , since otherwise we could solve the halting problem as follows: Program a procedure to enumerate in order of increasing length all the proofs in \mathcal{A} . If $\mathsf{Halts}(q,q)$ is decidable, then we must eventually enumerate a proof of either $\mathsf{Halts}(q,q)$ or $\neg\mathsf{Halts}(q,q)$. Whichever we enumerate first gives the solution to the halting problem, but since this is unsolvable, the proposition Halts(q,q) must be undecidable. Thus \mathcal{A} is incomplete.

6.3 The Löwenheim-Skolem Theorem

6.3.1 Background

Gödel's Theorem says that we can never adequately axiomatize a modestly rich body of knowledge, since our axioms must be either inconsistent or incomplete. This is an important limitation on the power of formal systems. Before we leave this topic, however, we must discuss another result, which limits them in a different but equally significant way. This is the Löwenheim-Skolem Theorem (1915, 1920), which implies that no axiomatic system can uniquely characterize the real numbers, or even the integers.⁹ However, to state this theorem, we will need some terminology.

In Section 5.2 we said that a *model* is an interpretation that results in the axioms and theorems being true of the domain. Thus, in trying to axiomatize any body of knowledge it's important that the intended interpretation be a model of the axioms, and — if we want our axioms to uniquely characterize that body — that that interpretation be the only model. Axiomatic systems with only one model are sometimes called *categorical* axiom systems. (See also "Uniqueness of Models," on p. 309.)

There is of course no guarantee that an axiomatic system has a model. For example, as we would expect, inconsistent systems have no models, because their doing so would require contradictions to hold in fact. Technically, we say that an inconsistent system is not *satisfiable*. It is certainly not obvious that even consistent systems are in general satisfiable, but this is in fact the case, a result first proved by Gödel and sometimes known as his Completeness Theorem. To prove this result we construct ad hoc interpretations in which the symbols are essentially interpreted as themselves. Thus the domain of the interpretation is a set of formulas (symbol strings), and the function and relation symbols denote functions and relations on symbol strings. This is a technical trick that allows the theorem to be proved, but it also has important

⁹Leopold Löwenheim: 1878–1957; Thoralf Skolem: 1887–1963. Discussions of the Löwenheim-Skolem Theorem can be found in Kneebone (MLFM, pp. 70–78, 303–307) and the article "Systems, Formal, and Models of Formal Systems" in Edwards (EP, Vol. 8, pp. 61–74).

implications, to which we now turn.

6.3.2 The Theorem

We've argued that any consistent axiomatic system has a model in which the domain is a set of formulas. But formulas are finite strings of characters from a finite alphabet, and so the set of formulas must be at most denumerable (see below, p. 310). Thus a consistent axiomatic system has a denumerable model. This is essentially the Löwenheim-Skolem Theorem.

Now this is a remarkable result. To see this, consider a typical axiomatization of set theory, such as the Zermelo-Fraenkel axioms, so long as expressed in finite texts over a finite alphabet. These axioms are often taken as the foundations of all mathematics (see p. 310). In particular, the ZF (Zermelo-Fraenkel) axioms are sufficiently powerful to prove the existence of the real numbers, and to express Cantor's diagonalization proof of the nondenumerability of the reals. Thus the ZF axioms assert the existence of a nondenumerable domain of objects. But the Löwenheim-Skolem Theorem tells us that these axioms (if they are consistent) must have a denumerable model. Thus, even though we can prove in this system that there's no one-to-one correspondence between the integers and the real numbers, there is some denumerable domain of objects, which includes all the objects that are called real numbers in the system. This is Skolem's Paradox: In our system we prove the theorem that the individuals are not denumerable. By definition, the theorems make true statements about the domains of the system's models. But we know that this system has a model with a denumerable domain. How can this be?

The explanation of Skolem's Paradox seems to be this. The denumerable model of ZF set theory contains objects, functions and relations corresponding to the symbols of the axiom system. Suppose \Re represents the set of real numbers in this system. Since the model is denumerable, there will be a denumerable number of objects in the domain that make the proposition ' $x \in \Re$ ' true. Therefore, relative to the model, the reals are denumerable. On the other hand, we can prove in the system that there is no enumeration of all the x such that $x \in \Re$. But this means that there is no function in the domain capable of enu-

308

merating the objects that correspond to these x. Thus, although these objects are denumerable relative to the model, they are nondenumerable relative to the formal system. Although this explanation resolves the paradox, it leaves us without any absolute notion of denumerability. What had been well understood is now problematic.

It must be noted that Skolem's Paradox is not a flaw in some particular axiomatic system (such as Zermelo-Fraenkel set theory). It is a property of any consistent axiomatic system. From it we conclude that any attempt to axiomatize the real numbers (or any other nondenumerable domain) must fail, because the axioms permit as models essentially different (non-isomorphic), denumerable domains. We cannot have a categorical axiomatization of the real numbers (p. 307). In this sense the attempt to reduce the continuous to the discrete has failed. But the situation is worse than this, for a corollary of the Löwenheim-Skolem Theorem shows that a consistent axiomatic system must have models of all transfinite cardinalities (Edwards, EP, Vol. 8, p. 72). For example, an axiomatization of the *integers* must have models that are nondenumerable, and hence essentially different from the intended model. Thus we are denied even a categorical axiomatization of the integers!

Uniqueness of Models

It's easy to see that in a strict sense models cannot be unique. For example, the natural numbers 0, 1, 2, ... (with the usual interpretations) form a model for the Peano axioms, but so do the symbols $\Box, \Box \flat, \Box \flat \flat, ...$ (Interpret '0' as the symbol \Box and interpret 'succ(n)' to mean 'append the symbol \flat on the right end of the string n'.) Thus, the best that we can hope for is that all the models are *isomorphic* (i.e., there is a one-toone relationship between the objects of the two domains that preserves the functions and relations on the domains). In this case we have the correspondences

 $0 \sim \Box$, $1 \sim \Box \flat$, $2 \sim \Box \flat \flat$, ...

Also, corresponding to the successor function on the natural numbers we have the operation of appending \flat on the end of a string. Thus when we say that an axiomatic system is categorical, or that it has one model, we will intend by this that all its models are equivalent "up to isomorphism."

Why a Set of Formulas Must be Denumerable

To see why a set of formulas must be denumerable, suppose we have an alphabet of N characters; then every string of characters can be thought of as a base N number. Therefore to every formula there will correspond a natural number, and so there can be at most as many formulas as there are natural numbers.

Notice that this argument depends on both the alphabet and the sizes of the formulas being finite. As we've seen (Sec. 5.5), these have been characteristics of formal systems since Pythagoras' time.

Zermelo-Fraenkel Axioms

The Zermelo-Fraenkel axioms refers to the most commonly accepted axiomatization of set theory. Zermelo (1871-1953) proposed seven axioms in 1908; these were revised and two additional axioms were added in 1922 by Fraenkel (1891-1965) and Skolem (1887-1963). These Zermelo-Fraenkel-Skolem axioms are the most commonly used, and a version of them is shown below. Another common axiomatization was developed from 1925–1954 by von Neumann (1903-1957) and Bernays (1888-?). See Beth (FM, pp. 381-398) for a discussion.

- Axiom of Extensionality $\forall S \forall T [\forall x (x \in S \leftrightarrow x \in T) \rightarrow S = T]$. That is, if two sets have the same members, then they are the same set.
- Axiom of Empty Set $\exists S \forall x [\neg(x \in S)]$. That is, there is a set with no members.
- Axiom of Coupling $\forall x \forall y \exists S \forall z [z \in S \leftrightarrow (z = x \lor z = y)]$. That is, for any x and y, there is a set whose only members are x and y.

6.3. THE LÖWENHEIM-SKOLEM THEOREM

- Axiom of Power Sets $\forall S \exists P \forall T (T \in P \leftrightarrow T \subseteq S)$. That is, for any set S there is a set (the power set of S) whose members are the subsets of S.
- Axiom of Union $\forall S \exists U \forall x [x \in U \leftrightarrow \exists T (x \in T \land T \in S)]$. If S is a set of sets, then there is a set whose members are just the members of the members of S.
- Axiom of Infinity $\exists S[\emptyset \in S \land \forall x (x \in S \to x \cup \{x\} \in S)]$. This guarantees the existence of at least one infinite set, corresponding to the natural numbers, that's constructed from nested empty sets:

$$\{ \{\}, \{\{\}\}, \{\{\{\}\}\}, \dots \}.$$

- Axiom of Regularity $\forall S \exists T[S = \emptyset \lor (T \in S \land \forall x[x \in S \rightarrow \neg(x \in T)])]$. This axiom prohibits $S \in S$ and "unfounded" sets with infinite descending chains of members, $S_1 \ni S_2 \ni S_3 \ni \cdots$.
- Schema for Axioms of Replacement $\forall x \forall y \forall y' [(F(x, y) \land (x, y')) \rightarrow y = y'] \rightarrow \exists S \forall x [x \in S \leftrightarrow \exists w (w \in T \land F(w, x))]$, where F(x, y) does not contain y', T or w. This is an axiom schema, that is, a pattern for generating an axiom for each suitable, expressible relation F. This says, roughly, that there is a set corresponding to any "reasonable" property expressible in the system. More precisely, it says that for each expressible function F and each set T there is a set S = F[T] that is the image of T under F.
- Axiom of Choice $\forall y \forall z[(y \in S \land z \in S \land y \neq z) \rightarrow \exists v \forall w (v \in y \land [w \notin y \lor w \notin z])] \rightarrow \exists u \forall y [y \in S \rightarrow \exists v \forall t (t = v \leftrightarrow [t \in u \land t \in y])]$. This says, roughly, that for any indexed set of nonempty sets, there is a function of the indices that "chooses" members of the sets. That is, if S_a is nonempty for each $a \in I$, then there is a function F on I such that $F(a) \in S_a$.

6.4 Epistemological Implications

The conclusion is inescapable that even for such a fixed, well defined body of mathematical propositions, *mathematical thinking is, and must remain, essentially creative.* To the writer's mind, this conclusion must inevitable result in at least a partial reversal of the entire axiomatic trend of the later nineteenth and early twentieth centuries, with a return to meaning and truth as being of the essence of mathematics.

— Emil Post

6.4.1 Limitations of the Discrete

It is important to realize that the results described in this chapter apply to any body of formalizable (i.e. verbalizable) knowledge, not merely to theories that are currently expressed as formal systems.¹⁰ Thus these limitations apply to the very ideal to which scientific knowing has aspired, for we expect that the basic truths of a scientific theory should be expressible in a finite number of words, and we also expect that at each step in reasoning about this theory we need to consider only a finite number of words (and thus that the inferential processes are finitary). The result is that a reasonable scientific theory is formalizable, and therefore the Gödel and Löwenheim-Skolem results apply to it. In this section we consider briefly the implications of these results for scientific knowing.

Gödel's Incompleteness Theorem shows us that a formalizable body of knowledge must be incomplete. In other words, there must be some questions about the subject matter that the theory does not permit answering. Conversely, a reasonably rich, consistent, complete body of knowledge cannot be expressed in a finite number of words. Thus the complete understanding of any subject matter must take a very

¹⁰More accurately, they apply only if the required formal system is "reasonably powerful" (p. 285) and consistent. Since it seems likely that any scientific theory must include multiplication and division, the hypothesis of "reasonable power" will be suppressed in the following discussion.

different form from what has traditionally been expected of scientific knowledge. In this sense we can never say all that there is to know about a subject. (Compare Socrates, Section 2.4.3.)

Even if we cannot say everything about a subject, it would seem that we ought at least to be able to uniquely characterize what it is we are talking about, but the Löwenheim-Skolem Theorem says that this is not so. For example, we cannot characterize the real continuum in a finite number of words, since any attempt must also apply to sets of objects that are essentially different from the reals (i.e., that are not isomorphic to the reals).

Although it appeared that the arithmetization of geometry — the reduction of the continuous to the discrete — had been successfully accomplished (Section 4.3), we now see that it hasn't. Finite words cannot exhaust the continuum. Eudoxus and Euclid eschewed the arithmetization of geometry and founded each science on its own axioms (p. 61), but we now see that even this cannot succeed.

The limitations of the discrete go beyond its inability to encompass the continuous. As noted (p. 309), a corollary of the Löwenheim-Skolem Theorem shows that even the *integers* cannot be uniquely characterized in a finite number of words. Also, Gödel's Incompleteness Theorem applies to the integers. Like the Pythagoreans 2500 years ago, we have discovered in number theory an element that is irreducibly *irrational*, ultimately *illogical*. (Recall the full meanings of *rational* and *logos*, Section 2.2.2.) Thus the most important limitation of the discrete is not the lack of a theory of the continuous. Rather it is the weakness of any body of knowledge that is in principle formalizable, for such knowledge must be both incomplete and incapable of characterizing its subject matter, except in the most trivial cases.¹¹

6.4.2 Transcending the Discrete

There seem to be several possible ways of escaping from the limitations of the discrete. One comes from rejecting an assumption that we owe

¹¹In general, none of these limitations apply to calculi with a finite number of possible formulas, or to finite domains of interpretation. Nor do they apply to certain simple infinite systems, as we saw in Section 5.3. However we face these limitations in any system powerful enough to be mathematically interesting.

to Plato and Aristotle, the assumption that true knowledge can be expressed in a *closed deductive system*, that is, in a system in which all the truths derive from a finite number of explicitly stated axioms. What is the alternative? Since a formal system is defined by its axioms and rules of inference, we may allow either or both of these to be *openended*. I will briefly discuss the possibilities.

If the set of axioms is to be open-ended, then we must provide a *nondeductive* process for extending it. (The process must be nondeductive, since otherwise the new "axioms" are just theorems in a conventional axiomatic system.) Examples are such *ampliative* inferential processes as *induction* and *abduction*; these may lead to the invention or revision of axioms on the basis of observation (see "Definitions," p. 315.).

Ampliative inferential processes cannot be considered merely temporary measures necessary only until a science is completed. Rather, the *practice* of observation, invention and revision must be considered an integral part of the body of knowledge. The nature of such "practical knowledge" or skill is a central topic of Part III.

Traditionally, it has been assumed that there is a fixed set of finitary inferential processes — the laws of logic. Therefore, one possible means of transcending the discrete is to allow the set of inferential processes to be open-ended, to recognize situations in which the inferential resources of a formal system can be extended. Notice, however, that there cannot be formal rules for this extension, otherwise they could be made a part of a formal system, and the usual limitations would apply. This means that we cannot expect precise specifications of when an extension is allowed or of the results of the extension.

How then does inferential extension take place? It seems that there must be some process for proposing possible extensions and for judging if they should be made (else we will have no confidence in the resulting formal systems). There are several possibilities. One is that we may discover new, intuitively valid inferential principles that can be formalized as deductive rules. However, in the 2300 years since Aristotle first formalized logic, this has rarely happened, so it seems unlikely that this process will be a continuing source of new inference rules. (Remember: our goal is not to fill in the gaps in an otherwise complete theory (which

Definitions

- Ampliative: "In [ampliative inference] the facts summed up in the conclusion are not among those stated in the premisses.
 ... These are the only inferences which increase our real knowledge, however useful the others may be." (Peirce, CP, 2.669-693)
- **Abduction:** "Abduction consists in studying facts and devising a theory to explain them." (Peirce, *CP*, 5.145)
- **Induction:** "Induction is the experimental testing of a theory." (Peirce, *CP*, 5.145)

On Peirce's notions of ampliative (or synthetic) inference, abduction (or hypothesis) and induction, see Buchler (PWP, Ch. 11), Goudge (ToP, Ch. 6) and Rescher (PPS, Ch. 3).

is impossible), but to find processes that are continually productive of new rules of inference.)

A more likely possibility is that new rules will be identified and justified by their consequences. For example, new deductive rules may be accepted because they allow the derivation of classes of theorems for which we have empirical or other informal reasons to want to be provable. This is, necessarily, an informal, ampliative process. As in the case of open-ended axioms, we see that a body of knowledge with open-ended inferential processes must include a set of ampliative practices that cannot be expressed as formal rules.

Is there any sense in which an open body of knowledge could be considered complete? For this to be the case we would have to believe that the ampliative processes are adequate to answer any question askable in the system. Are there such processes? Many people believe that the empirical methods of the sciences are complete in this way, but the claim needs justification.

More generally we can ask, "What would be the nature of amplia-

Open-ended Mathematics

The need for nondeductive processes for extending the basic truths of a body of knowledge may not seem surprising for those sciences traditionally considered empirical (physics, biology, psychology, etc.). However the Gödel and Löwenheim-Skolem results show that ampliative inference is also necessary in mathematics. Although this is not widely acknowledged, the history of mathematics exhibits many nondeductive processes; see Section ??.

tive processes that *could* be complete?" In effect we want processes that are guaranteed to gives answers but that satisfy certain criteria of objectivity (e.g., public accessibility, replicability, criticizability). These are essentially social criteria, and their roles in a theory of knowledge are discussed later (Sections ??, ?? and ??).

There is a more radical way by which we may transcend the discrete. As noted previously (p. 293), there are "semiformal" systems that are both consistent and complete, but they diverge radically from the finitary assumptions that underlie the traditional view of scientific knowledge. We can entertain systems that are *infinitary* in either their axioms or rules of inference (or both).

Since the time of Pythagoras it has been assumed that the basic truths of a science must be (de)finite (i.e., finite and definite). One way to escape the limitations of the discrete is to reject this assumption by permitting axioms that are *essentially infinite*, either in number or structure.¹² For example certain propositions about all the points of a continuum cannot be expressed in a finite number of discrete symbols, although they could be finitely expressed in a *continuous language* (for example a language whose "formulas" are images drawn from a contin-

 $^{^{12}}$ By "essentially infinite" I mean that they cannot be generated by some regular (i.e. finitely specifiable) way.



Figure 6.3: Simple Example of a Continuous Rule. Just as an discrete axiom $P \lor P \Rightarrow P$ ' shows how to map the formulas of one discrete space into those of another (for example, this one takes $A \lor A$ ' into A', $(p \land q) \lor (p \land q)$ ' into $(p \land q)'$), so the graph above shows how each point in one continuum (represented by the x axis) can be mapped into a point in another continuum (the y axis). Note that the graph (the curved line) itself is the finite, written representation of the correspondence, just as the string $P \lor P \Rightarrow P'$ is in the discrete case. Note also that there are infinitely many graphs arbitrarily similar to the graph shown above.

uum). Further, if the axioms are themselves drawn from a continuum, then there will be axioms that are arbitrarily "close" to one another; in this sense we may call them *indefinite* axioms (Fig. 6.3).

There are two senses in which inference may be infinitary: in the rules of inference or in proofs. Traditionally, semiformal systems permit rules of inference with an infinite number of premises, such as the "rule of infinite induction" (Edwards, *EP*, Vol. 3, p. 355). For example, infinitary rules of inference could embody continua in their antecedent; it would be natural to have continuous inferential rules of this kind to go with the continuous axioms described above. Such rules could be finitely specifiable in a continuous language (as would continuous axioms), but the decision as to whether a rule is applicable might require arbitrarily precise discriminations.

Another source of infinitary inference is to allow infinite proofs. These could of course be proofs with an infinite number of discrete steps of the usual kind, but finite proof length can be preserved in a system with *continuous* proofs, in which the theorems evolve from the axioms by a continuous process. (Think of classical mechanics to picture this possibility.)

What are the implications of infinitary systems for knowledge representation? Here it will be helpful to distinguish the two senses of the (de)finite: the definite (or discrete) and the finite (or bounded), for while we've found the limitations of the discrete, there are still advantages to hewing to the bounded. The reason is that if we are interested in the representation of knowledge in people and computers, then we must limit ourselves to representations that are physically realizable, which means that they must be bounded (require finite matter, energy and time).

Hence, it seems that we may escape the limitations of calculi while saving their physical realizability by representing knowledge in continuous, bounded structures — what topologists call *continua*.¹³ When analyzed independently of its physical embodiment, such a structure may be called a *continuous formal system* or a *formal continuum*, the principal topic of Part IV.¹⁴

¹³This term is defined in slightly different ways by different authors; I will take a continuum to be a connected compact metric space with more than one point (Iyanaga & Kawada, EDM, §81C).

¹⁴MacLennan (LNAI) argues for the necessity of "continuous logics" and presents two examples. Both are based on continuous "propositions" and continuous rules of inference, but one has the traditional discrete derivations while the other uses continuous derivation. Some steps toward continuous formal systems can be found in MacLennan (CCKR, CSS, GAC, WLIOW, IS). Already a number of theoretical results show the ability of continuous computational models to transcend the limits of Turing computability (Pour-El & Richards, 1979, 1981, 1982; Stannett, 1990).