Little Mathematics Library

∞

V. A. USPENSKY

GÖDEL'S INCOMPLETENESS THEOREM

Mir Publishers · Moscow

Rs 5/-

GÖDEL'S INCOMPLETENESS THEOREM

ПОПУЛЯРНЫЕ ЛЕКЦИИ ПО МАТЕМАТИКЕ

В. А. Успенский

ТЕОРЕМА ГЁДЕЛЯ О НЕПОЛНОТЕ

Издательство «Наука» Москва

LITTLE MATHEMATICS LIBRARY

V. A. Uspensky

GÖDEL'S INCOMPLETENESS THEOREM

ИЗДАТЕЛЬСТВО МИР

MIR PUBLISHERS
MOSCOW

Translated from the Russian
by Neal Koblitz

First published 1987

Revised from the 1982 Russian edition

На английском языке

Printed in the Union of Soviet Socialist Republics

Contents

Preface

There are some topics in mathematics which, though enjoying a certain fame, have traditionally been considered either too complicated or of insufficient importance to be included in the core mathematical curriculum. Customarily, such subjects are relegated to optional units of the syllabus, independent study projects, seminar papers, and math club talks. Among such topics are several which remain in this nebulous status only because of inertia. An example is Gödel's incompleteness theorem.

Although very many mathematicians (and nonmathematicians) have heard of Gödel's theorem, very few could explain exactly what the theorem says, let alone how it is proved. Yet the result is so important, and the reasons for the fundamental incompleteness (i.e., the impossibility of ever attaining a situation in which every true statement can be proved) are so simple, that Gödel's theorem can be taught to first-year college students. Moreover, the only prerequisites for understanding the proof are familiarity with the terminology of set theory (the words "set," "function," "domain of definition" and the like) and a certain facility at understanding mathematical arguments. Thus, the proof is even accessible to an ambitious high school student.

The method of proving Gödel's theorem in this book is different from Gödel's own method. Our method relies upon elementary concepts from the theory of algorithms. All of the necessary background information from this theory will be explained as needed, so that as a by-product of the proof the reader will become familiar with the basic facts of the theory of algorithms.

This book is based on an article I wrote in the *Uspekhi Matematicheskikh Nauk* journal, vol. 29, no. 1 (1974). Of course, the intended readership is quite different, so the essay had to be rewritten. In particular, I have omitted discussions of more specialized questions, and also all references to the original publications; the curious reader can find these in the article mentioned above. At the same time, I have expanded the section on the connection between the semantic and syntactic formulations of the incompleteness theorem, and have added appendixes on Tarski's theorem on the inexpressibility of truth and on the justification of the arithmeticity axiom.

The plan of the book is as follows. In Sec. 1 we state the incompleteness theorem and explain the precise meaning of each element in the statement of the theorem. In particular, the notion of a deductive system, which is central to the book, is introduced; Sec. 2 contains an informal exposition of some initial concepts from the theory of algorithms, which are then used to give our first criteria for completeness and incompleteness. In Sec. 3 we continue our study of the incompleteness criteria; and in Sec. 4 we cover the language of formal arithmetic, define exactly what it means for a statement in that language to be true, and give a precise statement of Gödel's incompleteness theorem for formal arithmetic. Section 5 contains a development of the ideas concerning algorithms which were briefly described in Sec. 2, culminating in three axioms for the theory of algorithms. In Sec. 5 we complete the proof of the incompleteness theorem for formal arithmetic.

The book concludes with seven appendixes, which are written in a somewhat more condensed style, but still without assuming any special knowledge. In the first appendix we examine the connection between the existence of true statements which cannot be proved and the existence of statements which cannot be either proved or disproved. In the second appendix we prove Tarski's theorem on the inexpressibility of truth, which is a strengthening of Gödel's theorem. The third appendix is concerned with justifying one of the axioms of the theory of algorithms in Sec. 5, namely, the arithmeticity axiom. For this purpose we introduce a particular class of algorithms, the "address programs," and we verify the arithmeticity of functions which are computable by this class of algorithms. In the fourth appendix the completeness and incompleteness criteria in Sec. 2 are applied to languages connected with what are called "associative calculi." The fifth appendix describes the original formulation of the incompleteness theorem which Gödel himself gave. The sixth appendix contains exercises for the preceding sections. Finally, the last appendix gives answers and hints for these exercises. The appendixes are mutually independent, and so can be read in any order, except that Appendix C in certain places assumes a familiarity with some concepts from Appendix B.

If after reading this book the reader would like to know more about mathematical logic and the theory of algorithms, he or she can turn to the books listed at the end of the book.

Vladimir Uspensky

1. Statement of the Problem

The incompleteness theorem, for which we will give a precise statement in this section and later a proof, says roughly the following: under certain conditions *in any language there exist true but unprovable statements.*

When we state the theorem this way, almost every word needs some explanation. Thus, we must start by explaining the meaning of these words.

1.1. Language. We shall not give the most general possible definition of a language, but rather shall limit ourselves to those language concepts which we shall later need. There are two such concepts: the "alphabet of a language" and the "set of true statements in a language."

1.1.1. Alphabet. By an *alphabet* we mean a finite list of elementary signs (i.e., things which cannot be split up into smaller units). These signs are called the *letters* of our alphabet. By a *word* in the alphabet we mean a finite sequence of letters. For example, the usual words in the English language (including proper names) are words in a 54-letter alphabet (26 small letters, 26 capital letters, the hyphen and apostrophe). As another example the natural numbers written in decimal form are words in a ten-letter alphabet whose letters are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. We shall use ordinary capital letters to denote alphabets. If L is an alphabet, then L^∞ denotes the set of all words in the alphabet L. We shall assume that any language has an alphabet such that all the expressions in the language (i.e., the names of various objects, statements concerning these objects, etc.) are words in this alphabet. For example, any sentence in English, or in fact any English language text, may be regarded as a word in the alphabet which is obtained by expanding the 54-letter alphabet to include punctuation marks, a space for use between words, a sign for paragraph indentation, and perhaps a few other useful signs. When we assume that the expressions in a language are words in some alphabet, we thereby rule out "multilayered" expressions, such as $\int_a^0 f(x)\,dx$. But this restriction is not very serious, because any such expression can be "stretched out" into a linear form using suitable notational conventions.

Any set M which is contained in L^∞ is called a *word set in the alphabet* L. If we say simply that M is a *word set*, we mean that it is a word set in some alphabet. Now the above assumption about a language can be rephrased as follows: in any language the set of expressions is a word set.

1.1.2. The Set of True Statements. We assume that we are given a subset T of the set L^∞ (where L is the alphabet of the language under consideration) which is called the set of "true statements" (or simply "truths"). In going right to the subset T we are omitting such intermediate steps as: firstly, specifying which words of all the possible ones in the alphabet L are correctly formed *expressions* in the language, i.e., have a definite meaning in our interpretation of the language (for example, $2 + 3$, $x + 3$, $x = y$, $x = 3$, $2 = 3$, $2 = 2$ are correctly formed expressions, while $+ = x$ is not); secondly, which of all the expressions are *formulas*, i.e., in our interpretation make statements which may depend on a parameter (for example, $x = 3$, $x = y$, $2 = 3$, $2 = 2$); thirdly, specifying which of all the possible formulas are *closed formulas*, i.e., statements which do not depend on parameters (for example, $2 = 3$, $2 = 2$); and finally, which of all the possible closed formulas are *true statements* (for example, $2 = 2$).

1.1.3. For our purposes it will be enough to consider that a language is completely defined as soon as we are told its alphabet L and the subset T of L^∞ We shall call any such $\langle \mathrm{L}, T\rangle$ a *fundamental pair*.

1.2. Unprovable. "Unprovable" means not provable, and "provable" means having a proof.

1.3. Proof. Although the term "proof" is perhaps the most important in mathematics*, it does not have an exact definition. The full notion of a proof, with all its ramifications,

* Bourbaki begins his *Foundations of Mathematics* with the words, "From the time of the Greeks, to say 'mathematics' has meant the same as to say 'proof.'"

belongs as much to the realm of psychology as to mathematics. After all, a proof is simply an argument which we find so convincing that we are ready to use it to convince others.

1.3.1. When written down, a proof becomes a word in some alphabet P, just as English language texts are words in an alphabet L, as mentioned above. All proofs comprise a subset (a rather wide-ranging subset, to be sure) of P^∞. We shall not attempt to give a precise definition for this "naive" and "absolute" concept of proof, or, equivalently, for the corresponding subset of P^∞. Instead, we shall study a formal analog of this notion of proof, for which we shall still use the term "proof." This analog has two essential features which are different from the intuitive notion (though the intuitive idea of a proof reflects these features to some degree). In the first place, we shall allow different concepts of proof, i.e., different proof-subsets of P^∞, and, in fact, we shall also allow the alphabet P to vary. In the second place, for each such concept of proof we shall require that there be an effective method, or algorithm (a precise definition of this term will be given in Sec. 2), which verifies whether or not a given word in the alphabet P is a proof. We shall also assume that there is an algorithm which, given a proof, determines what statement it proves. (In many situations, the statement being proved is simply the last statement in the sequence of steps which make up the proof.)

1.3.2. Thus, our final definition is as follows:

1°. We have an alphabet L (the *language alphabet*) and an alphabet P (the *proof alphabet*).

2°. In the set P^∞ we are given a subset P, whose elements are called *proofs*. We further assume that we have an algorithm which, given an arbitrary word in the alphabet P, enables us to determine whether or not it belongs to P.

3°. We have a function δ (to *determine what is being proved)* whose domain of definition Δ satisfies $P \subseteq \Delta \subseteq P^\infty$ and whose range of values is in L^∞ We assume that we have an algorithm which computes this function (the precise meaning of the words "an algorithm computes a function" will be explained in Sec. 2). We shall say that an element p of P is a proof of the word $\delta\ (p)$ in the language alphabet L.

1.3.3. A triple $\langle P, P, \delta \rangle$ which satisfies conditions 1°-3° is called a *deductive system* over the alphabet L.

1.3.4. For the benefit of the reader who is familiar with the usual way of characterizing a "proof" in terms of "axioms" and "rules of deduction," we now explain how this method can be regarded as a special case of the definition in Subsec. 1.3.2. That is, a proof is usually defined to be a sequence of expressions in a language such that each term either is an axiom or else is obtained from the earlier terms using one of the rules of deduction. If we add a new letter $*$ to our language alphabet, we can write out such a proof as a word in the resulting alphabet: a sequence of expressions $\langle C_1, C_2, \ldots, C_n \rangle$ becomes the word $C_1*C_2* \ldots *C_n$. The function which determines what is being proved simply takes from such a word the part that follows the last $*$. The algorithms required by the definition in Subsec. 1.3.2 can easily be constructed once we specify any of the customary meanings of "axiom" and "rules of deduction."

1.4. Attempts at a Precise Formulation of the Incompleteness Theorem.

1.4.1. First Attempt. "Under certain conditions, given a fundamental pair $\langle \mathrm{L}, T \rangle$ and a deductive system $\langle \mathrm{P}, P, \delta \rangle$ over L, there always exists a word in T which does not have a proof." This statement is still too vague. In particular, we could obviously think up many deductive systems having very few provable words. For example, there are no provable words at all in the empty deductive system (where $P = \varnothing$).

1.4.2. Second Attempt. There is another more natural approach. Suppose we are given a language, in the precise meaning that we are given a fundamental pair $\langle \mathrm{L}, T \rangle$. We now look for a deductive system over L (intuitively, we look for techniques of proof) in which we can prove as many words in T as possible, ideally, all words in T. Gödel's theorem describes a situation in which such a deductive system (in which every word of T has a proof) does not exist. Thus, we would like to make the following statement: "Under certain conditions concerning the fundamental pair $\langle \mathrm{L}, T \rangle$ there does not exist a deductive system over L in which every word in T has a proof." However, this statement is clearly false, since one need only take the deductive system with $\mathrm{P} = \mathrm{L}$, $P = \mathrm{P}^\infty$ and $\delta(p) = p$ for all p in P^∞; then every word in L^∞ is trivially provable. Thus, we need a restriction on the deductive systems that we are allowed to use.

1.5. Consistency.

It is natural to require that only "true statements," i.e., words in T, can be proved. We say that a deductive system $\langle \mathrm{P}, P, \delta \rangle$ is *consistent relative to* (or *for*) the fundamental pair $\langle \mathrm{L}, T \rangle$ if we have $\delta(P) \subseteq T$. In what follows, we shall only be interested in consistent deductive systems. If we have a language, it is very tempting to try to find a consistent deductive system in which every true statement is provable. The version of Gödel's theorem which we shall study states precisely that, under certain conditions concerning the fundamental pair, it is impossible to find such a deductive system.

1.6. Completeness. We say that a deductive system $\langle \mathrm{P}, P, \delta \rangle$ is *complete relative to* (or *for*) the fundamental pair $\langle \mathrm{L}, T \rangle$ if we have $\delta(P) \supseteq T$. Our statement of the incompleteness theorem now takes the following form:

under certain conditions concerning the fundamental pair $\langle \mathrm{L}, T \rangle$, *there does not exist any deductive system over* L *which is both complete and consistent relative to* $\langle \mathrm{L}, T \rangle$.

For now we shall be satisfied with this formulation. In later sections we shall specify the conditions which the fundamental pair must satisfy.

2. Basic Concepts from the Theory of Algorithms and Their Application

Conditions for the nonexistence of a complete and consistent deductive system can easily be given in terms of the theory of algorithms.

For now, we shall only need the most general intuitive idea of what an *algorithm* is: a set of instructions which, given an *input* (also called the *initial data* or the *argument*) from some set of *possible inputs* (for the given algorithm), enables us to obtain an *output* if such an output exists or else obtain nothing at all if there is no output for our particular input. Notice that the set of possible inputs consists of all inputs to which the algorithm can be applied, not only those for which the algorithm gives an output. If there is an output for a particular input, then we say that the algorithm can be *applied* to this input and *processes* it to give the corresponding output.

For our purposes, in order to avoid unnecessary digressions, we shall suppose that the inputs and outputs of an algorithm are words. More precisely: every algorithm has an *input alphabet*, so that all possible inputs are words in this alphabet, and an *output alphabet*, so that all outputs are words in this output alphabet. This means, for example, that in order to work with algorithms which process pairs or sequences of words, we must first write such pairs or sequences as single words in a new alphabet. To be definite, whenever we have an alphabet L we shall agree to let a star stand for a new letter not in our alphabet L (thus, the star denotes different letters in different situations). We shall let L_* denote the alphabet obtained by adding the star symbol to our original alphabet L. In Subsec. 1.3.4 we already agreed to write a sequence of words $\langle C_1, \ldots, C_n \rangle$ in the alphabet L as the single word $C_1 * \ldots * C_n$ in the alphabet L_*. For example, a pair $\langle C_1, C_2 \rangle$ will be written as $C_1 * C_2$ in the alphabet L_*.

Furthermore, suppose that for fixed n we have a set of n alphabets $L_1, L_2, \ldots, L_n$. We then let $*$ denote a letter that is not in any of the alphabets L_i, and we write a sequence $\langle C_1, \ldots, C_n \rangle$ in which each C_i is a word in the corresponding alphabet L_i, as the single word $C_1 * \ldots * C_n$ in the alphabet $(L_1 \cup L_2 \cup \ldots \cup L_n)_*$. We shall let $L_1^\infty \times \ldots \times L_n^\infty$ denote the set of all such sequences of n words in the respective alphabets, or equivalently, the set of corresponding single words formed using the star symbol.

The set of all inputs that can be processed by a given algorithm is called the *domain of applicability* of the algorithm. Any algorithm defines a function, namely, the function which associates the corresponding output to every element in the domain of applicability. Thus, the domain of definition of this function is precisely the domain of applicability of the algorithm. We say that the algorithm *computes* the function that is defined in this way.

We shall let $A(x)$ denote the output obtained by applying the algorithm A to the input x, and for brevity we write $A(\langle x_1, x_2, \ldots, x_n \rangle)$ simply as $A(x_1, x_2, \ldots, x_n)$. Then the definition of the term "computes" can be rephrased as follows: the algorithm A computes the function f if we have $A(x) \simeq f(x)$ for all x. Here $\simeq$ is the "conditional equality" sign, which is defined as follows: $A \simeq B$ either if A and B are both undefined, or if A and B are both defined and are the same.

A function which can be computed by some algorithm is called a *computable* function. Thus, in part 3° of the definition of a proof (see Subsec. 1.3.2) we are saying that the function which yields the statement being proved must be a computable function.

Because of our assumptions about the meaning of an algorithm, for every computable function we must have two alphabets such that all possible arguments of the function are words in the first alphabet and all possible values of the function are words in the second alphabet.

We are especially interested in functions whose arguments and values are natural numbers (we shall always include zero in the natural numbers). Such functions are called *numerical functions*. In order to be able to speak of computable numerical functions, we must introduce algorithms which deal with numbers, and to do this we must first of all represent the natural numbers as words in some alphabet, called a *digital* alphabet. There are various ways of doing this, for example: (1) the binary system, in which numbers are written in the alphabet {0, 1}; (2) the decimal system, which uses the alphabet {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; (3) the system using a one-letter alphabet { | } with the number n written as the word || | (repeated n times); (4) the system in which n is written as the word (|| .   . |) (where | is repeated n times) in the three-letter alphabet { |, (,)}; and so on. We simply choose the most convenient system for our particular purposes. Each symbol for a number (in some fixed system) is called a *digit*. When we speak of algorithms and computable functions which process numbers, strictly speaking we shall mean algorithms and computable functions which process the digits used to write these numbers (in some chosen notational system).

Thus, the notion of a computable numerical function seems to depend upon our choice of notational system for writing numbers. However, it is easy to show that a numerical function which is computable in one notational system will be computable in any other, at least for a large class of notational systems. We shall say that two notational systems are equivalent if there exist an algorithm which processes an arbitrary number written in the first system and gives as output the same number written in the second system, and also an algorithm which processes a number written in the second system and gives as output the same number written in the first system. The examples of notational systems given above are obviously mutually equivalent.

We now show that a numerical function f which is computable in one notational system will also be computable in any equivalent notational system. Let C and D be algorithms which translate from the first notational system to the second and conversely, and let A be an algorithm which comenumerable but nondecidable subset of the set of natural numbers. The next lemma gives a condition for an enumerable set to be decidable.

Lemma 3. *A subset S of an enumerable set X is decidable relative to X if and only if both S and its complement $X \setminus S$ are enumerable.*

▶ If S is decidable, then so is $X \setminus S$, and so both S and $X \setminus S$ are enumerable, by Lemma 2. Conversely, suppose that both S and $X \setminus S$ are enumerable. If either one is empty, then S is decidable, by Lemma 1. Suppose that both S and $X \setminus S$ are nonempty, in which case they are enumerated by some computable functions f and g, respectively. Then, if we want to answer the question "Does x belong to S?" for an arbitrary x in X, we need only compute successively

$$f(0),\quad g(0),\quad f(1),\quad g(1),\quad f(2),\quad g(2),$$

until we encounter x. Note that x must eventually occur, since the above sequence exhausts all of X. If x occurs among the values of f, then x belongs to S; if x occurs among the values of g, then x does not belong to S. ■

Theorem 2. *The set of all proofs (for a given deductive system) is enumerable.*

▶ The set of all words in the proof alphabet is enumerable (see Example 2 above). Hence the theorem follows from Lemma 2. ■

Lemma 4 (the image of an enumerable set). *Suppose that R is an enumerable set and f is a computable function which is defined on all elements of R. Then $f(R)$ is an enumerable set.*

▶ If R is empty, then so is $f(R)$. If R is enumerated by the computable function ρ, then $f(R)$ is enumerated by the computable function $y = f(\rho(x))$. ■

Example 5. Let $\neg$ be a symbol in some alphabet L, and let $A \subseteq \mathrm{L}^{\infty}$. We let $\neg A$ denote the set of all words of the form $\neg a$, where $a \in A$. If we set $R = A$ and $f(a) = \neg a$ in Lemma 4, we see that enumerability of A implies enumerability of $\neg A$; and if we set $R = \neg A$ and $f(\neg a) = a$, we see that, conversely, enumerability of $\neg A$ implies enumerability of A.

Example 6. We claim that $\mathrm{L}^{\infty} \times \mathrm{L}^{\infty}$ is enumerable for any alphabet L. In fact, the sets $\mathbb{N}^2$ and L^{∞} are both enumerable (see Examples 1 and 2 above). Let L^∞ be enumerated by the computable sequence g. We define a computable function f on the set $\mathbb{N}^2$ by setting $f(a, b) = \langle g(a), g(b)\rangle$. Obviously, $f(\mathbb{N}^2) = L^\infty \times L^\infty$ and then our claim follows from Lemma 4.

As usual, we let $K_1 \times K_2 \times \quad \times K_n$ denote the direct product of the sets $K_1, K_2, . \quad ., K_n$, i.e., the set of all n-tuples $\langle k_1, k_2, \quad k_n\rangle$ such that $k_1 \in K_1, k_2 \in K_2, \quad . \quad . \quad ., k_n \in K_n$. In view of our notational convention at the beginning of the section, if $L_1, \quad L_n$ are alphabets and $K_1 \subseteq L_1^\infty, \quad ., K_n \subseteq L_n^\infty$, then the product $K_1 \times \quad \times K_n$ is a set of words in $L_1^\infty \times . \quad \times L_n^\infty$.

Corollary 1 of Lemma 4. *If $K_1, \quad ., K_n$ are enumerable sets, then so is the product $K_1 \times \quad . \times K_n$.*

▶ When $n = 2$, the proof follows the argument in Example 6 above. Then we proceed by induction, and apply Lemma 4 to the "obvious" computable function from $(K_1 \times \times K_n) \times K_{n+1}$ to $K_1 \times . \quad \times K_n \times K_{n+1}$. ■

The r-tuple $\langle C_{i_1}, \quad ., C_{i_r}\rangle$, where $i_1 \leqslant n, \quad ., i_r \leqslant n$, is called the *projection* of the n-tuple $\langle C_1, \ . ., C_n\rangle$ onto the $i_1, \quad ., i_r$ axes, and is denoted $\mathrm{pr}_{i_1, \quad ., i_r}\langle C_1, \quad ., C_n\rangle$. In particular, we have $\mathrm{pr}_1 \langle C_1, . . ., C_n\rangle = C_1$, $\mathrm{pr}_2 \langle C_1, \quad . \quad C_n\rangle = C_2$, etc. If $M \subseteq K_1 \times \quad \times K_n$, then we let $\mathrm{pr}_{i_1, \ldots, i_r} M$ denote the set of all possible projections $\mathrm{pr}_{i_1, \quad i_r} m$, where $m \in M$.

Corollary 2 of Lemma 4. *If $L_1, \ . ., L_n$ are alphabets, $i_1, \quad i_r$ are positive integers not exceeding n, and M is an enumerable subset of $L_1^\infty \times \quad \times L_n^\infty$, then $\mathrm{pr}_{i_1, \quad i_r} M$ is enumerable.*

▶ It suffices to use the computable function $x \mapsto \mathrm{pr}_{i_1, \quad i_r} x$. ■

Theorem 3. *The set of all provable words (for a given deductive system) is enumerable.*

▶ Let Q be the set of all provable words for the deductive system $\langle P, P, \delta\rangle$. Obviously $Q = \delta(P)$. But P is an enumerable set by Theorem 2. Hence Q is also enumerable, by Lemma 4. ■

It follows that if T is not an enumerable set, then it is impossible to find a complete and consistent deductive system for the pair $\langle L, T\rangle$, since the set Q of provable words for any consistent deductive system is a proper subset of T, and there will have to be an element in the complement $T \setminus Q$. Such an element is a true but unprovable statement!

Theorems 1 and 3 together give a condition on a fundamental pair which is necessary and sufficient for the existence of a complete and consistent deductive system for the pair. This condition is enumerability of the set of all truths. One would expect (and this is what turns out to be the case) that in a "rich" or "expressive" language the set of all truths is too complicated to be enumerable, and hence there are no complete and consistent deductive systems for such a language. However, the criterion in Theorems 1 and 3 is not very convenient to use, since it is often difficult to study the entire set T. In the next section we shall reformulate our criterion so as to make it more "applicable."

3. The Simplest Incompleteness Criteria

We now know that enumerability of the set T is equivalent to the existence of a complete and consistent deductive system for $\langle \mathrm{L}, T \rangle$.

However, we might be interested not in all truths in the language, but only in truths of a certain type or a certain class, much as a student studying for a math exam is not concerned with the truth of all mathematical statements, but only those which are likely to be encountered on the exam. For example, we might want to construct a deductive system in which one can derive all true statements of length at most 1000 and cannot derive any false statement of length at most 1000. In this case, for a statement of length greater than 1000 the question of whether or not it can be derived in the deductive system may have nothing to do with whether or not it is true. Moreover, in certain situations (such as the language of set theory), one cannot even define the set of all truths in their totality. This is why we restrict ourselves to considering consistency and completeness for subsets of the word set L^{∞} We now proceed to the formal definitions.

Let $\langle \mathrm{L}, T \rangle$ be a fundamental pair, let $\langle \mathrm{P}, P, \delta \rangle$ be a deductive system over L, and let Q be the set of all provable words. Suppose that $V \subseteq \mathrm{L}^{\infty}$. We say that the deductive system $\langle \mathrm{P}, P, \delta \rangle$

(a) is *consistent relative to* V, if $V \cap Q \subseteq V \cap T$;

(b) is *complete relative to* V, if $V \cap T \subseteq V \cap Q$.

Theorem 4. *Suppose that V is an enumerable subset of* L^{∞}, *and the set of true statements in V is not enumerable. Then there* *is no deductive system which is both consistent and complete relative to V.*

▶ By assumption, $V \cap T$ is not enumerable. In order for a deductive system to be complete and consistent relative to V, we must have $V \cap T = V \cap Q$. But $V \cap Q$ must be enumerable, because of Theorem 3 and the following lemma. ■

Lemma 5. *The set-theoretic union or intersection of two enumerable sets is enumerable.*

▶ Suppose that R and S are enumerable sets. We first prove that $R \cup S$ is enumerable. This is trivial if one of the two sets is empty. If both sets are nonempty, then we have $R = \{\rho(0), \rho(1), \ldots\}$ and $S = \{\sigma(0), \sigma(1), \ldots\}$, where ρ and σ are computable sequences. Then we can enumerate $R \cup S$ by means of the computable sequence f defined as follows: $f(2n) = \rho(n)$, $f(2n + 1) = \sigma(n)$. We now prove that $R \cap S$ is enumerable. If $R \cap S$ is empty, then it is enumerable, by definition. Otherwise there exists an a such that $a \in R \cap S$. Again suppose that R and S are enumerated by the computable functions ρ and σ. Since the set $\mathbb{N}^2$ is enumerable (see Example 1 in Sec. 2), it is enumerated by some computable function g. Each value of $g(n)$ is a pair of natural numbers; let us denote the two numbers in this pair by $\xi(n)$ and $\eta(n)$. The functions ξ and η are obviously computable. We introduce a function h by setting

$$h(n) = \begin{cases} \rho(\xi(n)), & \text{if } \rho(\xi(n)) = \sigma(\eta(n)), \\ a, & \text{otherwise.} \end{cases}$$

The function h is computable, and it enumerates the set $R \cap S$. ■

Remark 1. The condition in Theorem 4 is actually necessary as well as sufficient for there not to exist a deductive system, which is complete and consistent relative to V (The necessity of this condition is even true without assuming that V is enumerable.) Namely, if $V \cap T$ is enumerable, then the complete and consistent deductive system for $\langle \mathrm{L}, V \cap T \rangle$ which exists by Theorem 1, will also be a complete and consistent deductive system for $\langle \mathrm{L}, T \rangle$ relative to V.

Clearly, a deductive system is consistent (or complete) for $\langle \mathrm{L}, T \rangle$ if and only if it is consistent (respectively, complete) relative to any subset of L^{∞} Hence, if we have a consistent deductive system for $\langle \mathrm{L}, T \rangle$ and want to show putes the function f in the first notational system. (More precisely, the algorithm A computes the function of digits in the first notational system which corresponds to f, which is a function of numbers.) Then the following algorithm B will compute f in the second notational system (more precisely, B will compute the function of digits in the second notational system which corresponds to f):

$$B(x) \simeq CAD(x).$$

That is, the set of instructions for the algorithm B can be described as follows: "Translate the input x (a number written in the second notational system) into the first notational system, they apply the algorithm A, and then translate the output (if A gives an output) into the second notational system." In a similar way, the notion of an enumerable set of numbers, which will be defined below, can be shown to be independent of our choice of notational system for writing numbers.

Because of this, once we have a digital system, we shall not be pedantic about distinguishing between numbers and the digits which represent them. For example, we shall use the letter $\mathbb{N}$ to denote both the set of natural numbers and the set of digital representations of natural numbers.

A set is said to be *enumerable* if either it is the empty set, or it is the set of elements in some computable sequence (i.e., the set of values of some computable function which is defined on the natural numbers). We say that the function (or sequence) *enumerates* our set. Obviously, every enumerable set is a word set.

Example 1. The set $\mathbb{N}^2$ consisting of all pairs of natural numbers is enumerable. One possible choice of enumerating function is the function

$$\varphi(n) = \langle a, b\rangle, \quad \text{where} \quad n = 2^a(2b+1) - 1.$$

Example 2. For any alphabet L, the set L^∞ of all words in the alphabet is enumerable. One possible way to construct an enumerating sequence is as follows. First order the elements of L in an arbitrary way. Then list the words in L in the following order: with words of different lengths, the shorter one comes first, and with words of the same length we use alphabetical (also called lexicographical) order (i.e., when comparing two words, we find the first place, moving from left to right, where the letters are different, and then take the two words in the order in which those two different letters occur in our ordering of the alphabet L). By listing the words in this order, we obtain the required enumerating sequence.

In this second example, one might ask how we know that the sequence of words is enumerable, i.e., how can we obtain an algorithm which, given k, produces the kth term a_k of the sequence? Here is one possible algorithm: write out the first $k+1$ elements of the sequence (i.e., a_0, a_1, . ., a_k), and then take the last word that was listed.

Example 3. The computable function f which enumerates L^∞ and which was constructed in Example 2 gives us a one-to-one correspondence from $\mathbb{N}$ to L^∞. Hence, we have a well-defined inverse function f^{-1}, which gives a one-to-one correspondence from L^∞ to $\mathbb{N}$. This f^{-1} is also computable, for example, by the following algorithm: to compute $f^{-1}(a)$, successively write out $f(0)$, $f(1)$, $f(2)$, until you reach an n for which $f(n) = a$; this n is then $f^{-1}(a)$.

Example 4. If we have any two alphabets L_1 and L_2, the composition of the computable function mapping $\mathbb{N}$ onto L_2^∞ in Example 2 with the computable function mapping L_1^∞ onto $\mathbb{N}$ in Example 3 gives us a computable function which is a one-to-one correspondence between L_1^∞ and L_2^∞.

A subset S of a set A is said to be *decidable* relative to A if there exists an algorithm which determines whether or not an element of A belongs to S. That is, the algorithm processes all the elements of S to a single output x (for example, x is the word "yes") and processes all the elements of the complement $A \setminus S$ to a second output word y (for example, "no"; of course, it makes no difference which words x and y we choose). Obviously, a subset S is decidable relative to A if and only if the set $A \setminus S$ is decidable relative to A. In part 2° of the definition of a proof (see Subsec. 1.3.2) we required that the set of all proofs be a decidable subset of the set of all words in the proof alphabet.

From the definition of decidability it follows that the domain of applicability of the algorithm in the definition must include all of A. It makes no difference whatsoever what happens if the algorithm is applied to words not in A. For example, if we want to construct an algorithm which distinguishes between the poetry of Pushkin and the poetry of Lermontov (two famous Russian poets), in other words, if we want to prove that the set of poems of Pushkin is decidable relative to the set consisting of all the poems of Pushkin and Lermontov, then we do not at all care what output is obtained (or if nothing is obtained) when we apply our algorithm to the poetry of the Soviet poet Mayakovsky or to the Instructions for installing a washing machine.

One might ask the natural question: what happens if we use a more restrictive definition of decidability, and require that the algorithm in the definition be applicable *only* to elements of the set A? With this narrower definition, a subset S is decidable relative to A if and only if the characteristic function of S relative to A (i.e., the function defined on A which takes the value 1 on S and 0 on $A \setminus S$) is computable. As we shall see in Sec. 5 (Corollary 1 of the protocol axiom), the domain of applicability of an algorithm is always an enumerable set. Hence, only enumerable sets could have decidable subsets in the sense of this new, narrower definition of decidability. But if our set A is enumerable, then both definitions of decidable subsets are equivalent. Suppose, for example, that f is a computable function which enumerates A, and $\boldsymbol{B}$ is an algorithm which decides the subset S relative to A in the sense of our first definition. Then the following algorithm will also decide S relative to A while at the same time having A as its domain of applicability: take an arbitrary a, successively write out $f(0)$, $f(1)$, $f(2)$,   , and, as soon as you obtain $f(n) = a$ apply the algorithm $\boldsymbol{B}$ to a.

Remark 1. Since any computable function, enumerable set, or decidable subset is given by some algorithm, we can use purely quantitative considerations to see that there must exist functions which are not computable, sets which are not enumerable, and subsets which are not decidable. (Here we always mean *word* sets, functions having *word* sets as their domain of definition, and so on.) Namely, any algorithm can, if we want, ultimately be written in the English language (perhaps with some mathematical symbols added), i.e., according to Subsec. 1.1.1, it can be written as a word in some rather large alphabet, and in any alphabet the set of all words is a countable set. Of course, while this argument proves the existence of nonalgorithmic objects, it is of no use in constructing individual examples of such sets and functions.

We are now ready to use these concepts from the theory of algorithms to study the question of whether a complete and consistent deductive system can exist.

Lemma 1. *For any word set X, the sets $\varnothing$ and X are decidable relative to X.*

▶[*] Let X be a word set in the alphabet L. It is enough to take the algorithm which gives the same output x for any input word in L^{∞} This algorithm decides the set $\varnothing$ and also the set X relative to X. ■[**]

Theorem 1. *If T is an enumerable set, then one can find a complete and consistent deductive system for the fundamental pair $\langle L, T\rangle$.*

▶ We have to give the triple $\langle P, P, \delta\rangle$. Recall that $\varnothing$ and P^{∞} are decidable relative to P^{∞}, by Lemma 1. If $T = \varnothing$, then we take the triple $\langle P, \varnothing, \delta\rangle$, where P and δ are arbitrary. If $T \neq \varnothing$, then $T = \{\tau(0), \tau(1), \tau(2), \quad .\}$, where τ is a computable function; we then identify n with the word $||\qquad|$ of length n and set $P = \{ | \}$, $P = P^{\infty}$, $\delta = \tau$. ■

Remark 2. This proof is not really as artificial as it might appear at first glance. In fact, if the set of truths in some language is enumerable, i.e. forms an enumerable sequence, then, in order to see that an expression belongs to this set (i.e., in order to prove that the expression is true) it suffices to give the number of the expression in the sequence (this number can thus be considered to constitute the proof).

The converse of Theorem 1 will be proved later (Theorem 3). First we shall need to prove some auxiliary lemmas.

Lemma 2 (enumerability of a decidable subset). *A decidable subset of an enumerable set is enumerable.*

▶ Suppose that $S \subseteq A$, and A is enumerated by the computable function f. If S is the empty set, then S is enumerable, by definition. If S is nonempty, then there exists an s such that $s \in S$. We set

$$g(n) = \begin{cases} f(n), & \text{if } f(n) \in S, \\ s, & \text{if } f(n) \in A \setminus S. \end{cases}$$

Clearly, g is a computable function which enumerates the set S. ■

From Lemma 2 it follows that any decidable subset of the set of natural numbers is enumerable. However, the converse is false: in Sec. 5 we shall construct an example of an

* The symbol ▶ marks the beginning of a proof.
** The symbol ■ marks the end of a proof.

that this deductive system is incomplete for ⟨L, T⟩, then we need only find a subset V of L^∞ relative to which our deductive system is incomplete. We now give a construction which enables us to find such a subset V in many important cases.

We shall say that *membership* in a set of natural numbers S is *expressible* by means of the fundamental pair ⟨L, T⟩ if there exists a computable function f (*expressing* membership in S) which is defined on the natural numbers, takes values in L^∞, and has the properties:

(1) if $n \in S$, then $f(n) \in T$,

(2) if $n \in \mathbb{N} \setminus S$. then $f(n) \in L^\infty \setminus T$.

The set V consisting of all values of such a function f is an enumerable set. Hence (by Theorem 4), if we know that the set $V \cap T$ of true statements in V is not enumerable, we can conclude that there does not exist a deductive system which is complete and consistent relative to V And, as we shall now see, $V \cap T$ is a nonenumerable set if the set S is not enumerable.

Lemma 6 (on the full preimage of an enumerable set). *Let f be a computable function whose domain of definition is an enumerable set.* Let B be an arbitrary enumerable set. Then the set $f^{-1}(B)$ is enumerable.*

▶ If $f^{-1}(B)$ is empty, then it is enumerable, by definition. Now suppose that $c \in f^{-1}(B)$, and the set B is enumerated by the computable function h. Suppose that the domain of definition of f is enumerated by the computable function g. In order to enumerate the set $f^{-1}(B)$ we proceed as follows.

We run through the set $\mathbb{N} \times \mathbb{N}$ and for each pair ⟨m, k⟩ we check whether or not f takes $g(m)$ (the "mth element listed in the domain of definition of f") to $h(k)$ (the "kth element listed in the set B"). If it does, then we include $g(m)$ in our list of the elements of $f^{-1}(B)$, and if it doesn't, then we simply list the element c.

More precisely, let ξ and η be defined as in the proof of Lemma 5. We set

$$\varphi(n) = \begin{cases} g(\xi(n)), & \text{if } f(g(\xi(n))) = h(\eta(n)), \\ c, & \text{otherwise.} \end{cases}$$

* Actually, the domain of definition of any computable function is enumerable; however, the proof of this fact requires some further study of algorithms, which we shall postpone until Sec. 5,

It is easy to see that φ is a computable function which enumerates the set $f^{-1}(B)$. ■

We now return to the line of thought we left to prove Lemma 6. Note that $S = f^{-1}(V \cap T)$. Thus, if S is not enumerable, then neither is $V \cap T$ (since, by Lemma 6, if $V \cap T$ were enumerable, then its full inverse image S would also be enumerable). In view of Theorem 4, we have thereby proved:

Theorem 5. *If there is a single nonenumerable set of natural numbers in which membership is expressible by means of the fundamental pair* ⟨L, T⟩, *then there cannot exist a complete and consistent deductive system for* ⟨L, T⟩. *Moreover, there cannot exist a deductive system which is both consistent and complete relative to the set of values of the function which expresses membership in our nonenumerable set.*

Remark 2. The sufficient condition in Theorem 5 is also a necessary condition for there not to exist a deductive system with the indicated properties. In fact, if there is no complete and consistent deductive system for ⟨L, T⟩, then T is nonenumerable (by Theorem 1). Meanwhile, L^{∞} is enumerable (see Example 2 in Sec. 2) and so is enumerated by some computable function f. Since $T = f(f^{-1}(T))$, it follows that the set $f^{-1}(T)$ is not enumerable (by Theorem 3). But the function f expresses membership in $f^{-1}(T)$ by means of the pair ⟨L, T⟩.

4. The Language of Arithmetic

In this section we apply the constructions of the earlier sections to the language of arithmetic. Intuitively speaking, the language of arithmetic is the language whose statements are given in terms of natural numbers and the addition and multiplication operations (using logical operations and the equals sign). In order to give a formal definition, we must construct a suitable fundamental pair. Of course, there are many possible ways to construct such a pair. For example, various different alphabets can be used. We shall choose a 14-letter alphabet A (our "arithmetic alphabet") consisting of the following symbols:

1°-2° the parentheses (and);
3° the symbol | for forming numbers;
4° the symbol x for forming variables;
5°-6° the addition sign $+$ and multiplication sign

7° the equals sign $=$;

8°-14° the logical symbols $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\exists$, $\forall$ (the intuitive meanings of these symbols are as follows: "it is false that," "and," "or," "if ..., then," "if and only if," "there exists such that," "for all").

In order to specify a suitable set of true statements, we must look at some questions involving syntax. That is, we must identify certain classes of words in A^∞ and study their structure.

We shall let α^n denote the word $\alpha...\alpha$ (repeated n times), where α is a letter. If $n = 0$, then the word α^n is empty (contains no letters). By a *number* we mean a word of the form $(|^n)$, where $n \geqslant 0$, and by a *variable* we mean a word of the form (x^n), where $n > 0$. In our intuitive interpretation of the language, the word $(|^n)$ is a way of writing the number n, and the word (x^n) is one of an infinite sequence of variables (we might need an arbitrarily large number of these variables to write a statement in arithmetic). We now give the following inductive definition of a *term*:

1° all numbers and all variables are terms;

2° if t and u are terms, then $(t + u)$ and $(t \cdot u)$ are terms.

Any variable which occurs in a term will be called a *parameter* of the term. A term which has no parameters is called a *constant*.

Example 1. The term $((|||)\ (||))$ is a constant. The terms $(( )\cdot(x))$ and $((|||) + (xx))$ are not constants: (x) is a parameter in the first of these terms and (xx) is a parameter in the second.

To any constant term we can associate a number, called its *value*, according to the following rules:

1° the value of $(|^n)$ is the number n.

2° the value of a constant term of the form $(t + u)$ is the sum of the values of the constant terms t and u, and the value of the constant term $(t \cdot u)$ is the product of the values of the constant terms t and u.

Example 2. The constant term $((|||) + ((|)\cdot(||)))$ has value 5.

A word of the form $(t = u)$, where t and u are terms, will be called an *elementary formula*. We then give an inductive definition of a *formula*, as follows:

1° any elementary formula is a formula;

2° if α is a formula, then $\neg\alpha$ is a formula;

3° if α and β are formulas, then $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow$

$\to \beta)$ and $(\alpha \leftrightarrow \beta)$ are formulas;

4° if α is a formula and ξ is a variable, then $\exists\xi\alpha$ and $\forall\xi\alpha$ are formulas.

Example 3. The word

$$(\exists\ (x)\ \forall\ (xx)\ \neg\ ((x)=(xx)) \leftrightarrow \forall\ (xx)\ ((x)=(xx)))$$

is a formula.

For ease of reading, we shall abbreviate terms and formulas, writing n in place of $(\ |^n\ )$ and x_n in place of (x^n), and omitting outer parentheses. For example, the formula in Example 3 can be written in abbreviated form as follows:

$$\exists x_1\ \forall\ x_2\ \neg\ (x_1=x_2) \leftrightarrow \forall x_2\ (x_1=x_2).$$

The true statements in our language will be defined as a subset of the set of all formulas. But first we need to introduce the notions of a formula's parameters and the substitution of numbers in place of variables.

To every formula we associate a certain finite set of variables; these variables will be called the *parameters* of the formula. The set of parameters of a formula is defined inductively according to the following rules:

1° the set of parameters of an elementary formula $(t = u)$ consists of all of the parameters of the term t together with all of the parameters of the term u;

2° the formula $\neg\alpha$ has the same parameters as the formula α;

3° the set of parameters of the formula $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \to \beta)$ or $(\alpha \leftrightarrow \beta)$ consists of all of the parameters of the formula α together with all of the parameters of the formula β;

4° the set of parameters of the formula $\exists\xi\alpha$ or $\forall\xi\alpha$ consists of all of the parameters of the formula α except for ξ.

Example 4. The only parameter of the formula in Example 3 is x_1. In fact, the formula $(x_1=x_2)$ has parameters x_1 and x_2, as does the formula $\neg\ (x_1=x_2)$, the formula $\forall x_2\ \neg\ (x_1=x_2)$ has only one parameter x_1, and the formula $\exists x_1 \forall x_2\ \neg\ (x_1=x_2)$ has no parameters. Meanwhile, the formula $\forall x_2\ (x_1=x_2)$ has one parameter x_1.

More intuitively, the parameters are simply the variables which occur freely in the formula, i.e., which are not in the range of the quantifiers $\exists$ and $\forall$.

Formulas having no parameters are called *closed formulas.* The formula in Example 3 is not a closed formula. Closed formulas can be interpreted as statements about the set of natural numbers. A closed formula is said to be "true" or "false" in accordance with the rules described below (which agree with the intuitive meaning of the symbols in the formula). The closed formulas which are given the value "true" will form our set of "true statements of the language of arithmetic."

Example 5. The sentence "for every natural number except zero there exists a smaller natural number" can be translated into the following closed formula of arithmetic:

$$\forall x_1 (\neg (x_1 = 0) \to \exists x_2 \exists x_3 (\neg (x_3 = 0) \land ((x_2 + x_3) = x_1))).$$

Note that, because our language does not have the symbol $<$, we had to write "x_2 is less than x_1" in a roundabout way, namely:

$$\exists x_3 (\neg (x_3 = 0) \land ((x_2 + x_3) = x_1)).$$

Before describing how to determine the value of a closed formula, we need to introduce one final technical definition. We now define the *result of substituting the number n in place of the variable w in the formula* α. This result is a formula which is denoted $S_n^w \alpha$ and is defined inductively according to the following rules:

1° the result of substituting n in place of w in an elementary formula $(t = u)$ is simply the result of replacing all occurrences of the variable w by the number n;

2° $S_n^w \neg\alpha = \neg S_n^w \alpha$;

3° if λ is any of the symbols $\land$, $\lor$, $\to$, or $\leftrightarrow$, then

$$S_n^w (\alpha\lambda\beta) = (S_n^w \alpha\lambda S_n^w \beta);$$

4° if Q is one of the symbols $\forall$ or $\exists$ and if ξ is a variable, then the result of substituting n in place of w in the formula $Q\xi\alpha$ is the formula $Q\xi S_n^w \alpha$, provided that the variable w is different from the variable ξ; otherwise (if w and ξ are the same variable) the result of the substitution is simply the original formula $Q\xi\alpha$.

Example 6. If α is the formula in Example 3, then $S_5^{x_1}\alpha$ is $\exists x_1 \forall x_2 \neg (x_1 = x_2) \leftrightarrow \forall x_2 (5 = x_2)$, and $S_5^{x_2}\alpha$ is α. Notice that if we replaced *all* occurrences of x_1 in α by 5, then we would obtain the word $\exists 5 \forall x_2 \neg (5 = x_2) \to \forall x_2 (5 = x_2)$, which is not a formula. Thus, an important feature of our definition is that the occurrences of w which fall in the range of $\exists w$ or $\forall w$ are left unchanged when a number is substituted in place of w.

Lemma 7. *The set of parameters of the formula* $S_n^w\alpha$ *consists of all the parameters of* α *which are different from* w.

▶ This rather obvious fact can be proved by induction on the number of steps in the construction of α (or by induction on the length of the word α). ■

We are now ready to proceed to the determination of the *value* of a closed formula. As mentioned above, there are two possible values: "true" (T) and "false" (F). A closed formula having the value T will be called a "true statement," and a closed formula having the value F will be called a "false statement." We assign values to closed formulas using induction on the number of steps in the construction of the formula, as follows:

1° the closed formula $(t = u)$ is true if the values of the constant terms t and u are equal; otherwise it is false;

2° the formula $\rceil\alpha$ is true if α is a false statement; otherwise it is false;

3° the formula $(\alpha \wedge \beta)$ is true if both α and β are true statements, otherwise it is false;

4° the closed formula $(\alpha \vee \beta)$ is true if at least one of α or β is true, otherwise $(\alpha \vee \beta)$ is false;

5° the formula $(\alpha \rightarrow \beta)$ is false if α is a true statement and β is a false statement, otherwise $(\alpha \rightarrow \beta)$ is true;

6° the formula $(\alpha \leftrightarrow \beta)$ is true if α and β are closed formulas with the same truth value, otherwise $(\alpha \leftrightarrow \beta)$ is false;

7° the closed formula $\exists\xi\alpha$ is true if there exists a number n such that $S_n^\xi\alpha$ is a true statement; if no such number exists, then $\exists\xi\alpha$ is false;

8° the closed formula $\forall\xi\alpha$ is true if $S_n^\xi\alpha$ is true for all n; otherwise $\forall\xi\alpha$ is false.

In connection with 7° and 8° we note that $S_n^\xi\alpha$ is a closed formula, since the formula α has no parameters other than ξ (otherwise $\exists\xi\alpha$ and $\forall\xi\alpha$ would not be closed formulas).

Example 7. The formula in Example 5 is true. The formula in Example 3 is neither true nor false, since it is not a closed formula. However, the result of substitution of any number in place of x_1 in the formula in Example 3 is a true statement.

Thus, it is the true closed formulas which we have chosen to be the true statements of arithmetic. If we let T denote the set of true statements, we arrive at the fundamental pair

⟨A, T⟩ for the language of arithmetic. The question that interests us is whether there exists a complete and consistent deductive system for this pair. We shall use the criterion in the last section to show that no such deductive system exists.

To do this, we must show that there exists a nonenumerable set of natural numbers such that membership in this set is expressible by means of our fundamental pair ⟨A, T⟩. For this purpose we shall introduce a certain class of sets such that membership in any set in this class is expressible by means of ⟨A, T⟩. We shall then look for a nonenumerable set in this particular class. The class of sets we are speaking of is the class of so-called "arithmetic sets." It is defined as follows.

Let α be a formula having no parameters except perhaps the variable x_1. Then $S_n^{x_1}\alpha$ is a closed statement, true or false, for any n. We consider the set of such and only such numbers n for which $S_n^{x_1}\alpha$ is a true statement. We shall say that this set is *associated with* the formula α. Any set of numbers which is associated with some formula in A will be called a *Gödel-arithmetic* set, or, for brevity, simply an *arithmetic* set.

Arithmetic sets have several obvious properties:

Property 1. The complement of an arithmetic set is an arithmetic set. Namely, if M is associated with the formula α, then $(\mathbb{N}\setminus M)$ is associated with the formula $\neg\alpha$.

Property 2. The union or intersection of two arithmetic sets is an arithmetic set. Namely, if M_1 and M_2 are associated with α_1 and α_2, respectively, then $M_1 \cap M_2$ is associated with $(\alpha_1 \wedge \alpha_2)$, and $M_1 \cup M_2$ is associated with $(\alpha_1 \vee \alpha_2)$.

Property 3. Membership in any arithmetic set is expressible by means of ⟨A, T⟩. Namely, suppose that the arithmetic set M is associated with the formula α. We define the function f as follows: the value of f at n is the word $S_n^{x_1}\alpha$. Then f is a computable function which expresses membership in M.

The key step in our proof of Gödel's theorem is the following claim:

(∗) *there exists a nonenumerable arithmetic set.*

We shall postpone the proof of this claim until the next section. Once the claim has been proved, we can then conclude, because of Property 3 and Theorem 5, that:

there does not exist a complete and consistent deductive sys-

tem for the fundamental pair $\langle A, T\rangle$ *of the language of arithmetic.*

This result is Gödel's incompleteness theorem for formal arithmetic, it says that, given any carefully defined notion of proof, there exists either a provable but false statement in the language of arithmetic or else a true but unprovable statement in the language of arithmetic.

Remark 1. Suppose that M is a nonenumerable arithmetic set. According to the second part of Theorem 5, if f is any computable function expressing membership in M and V is its set of values, then there does not exist a deductive system which is both complete and consistent relative to V. Thus, if we have a consistent deductive system, we must be able to find true but unprovable statements by looking no farther than the sequence $f(0), f(1), f(2), \ldots$. We just saw that we can for f take the function $n \mapsto S_n^x\alpha$, where M is associated with α. If we choose f in this way, it is natural to interpret the word $f(n)$ as the statement that "$n \in M$." Hence, speaking informally, we see that a true but unprovable statement can be found (for any consistent deductive system!) among the statements of the form "$n \in M$." In the next section we shall see that the set M can be chosen in such a way that its complement $E = \mathbb{N} \setminus M$ is enumerable. Thus, there exists an enumerable set E such that for any consistent deductive system there is a true statement of the form "n does not belong to E" which is unprovable. (Note that, by Theorem 1, it would be impossible to have "n belongs to E" in place of "n does not belong to E" here.)

Remark 2. Several of the definitions in this section use induction on the number of steps in the construction of the terms and formulas. Here a possible difficulty arises. Suppose, for example, that a word X had the form $(\alpha \wedge \beta)$ and simultaneously had the form $(\alpha' \rightarrow \beta')$, where α, β, α', and β' are formulas. In this case the requirements of the sections of the inductive definition relating to formulas of the form $(\alpha \wedge \beta)$ and to formulas of the form $(\alpha' \rightarrow \beta')$ might contradict one another.

For this reason, when we give inductive definitions we must be sure that the terms and formulas can be analyzed in a unique way, i.e., that the different cases in the definition of a term and in the definition of a formula are mutually exclusive. When in our definitions a term or formula is obtained as a result of combining two terms or formulas, the terms or formulas which are being combined must be uniquely determined. This is the purpose for which the parentheses are used in formulas. If we want a formal proof that terms and formulas can be analyzed in a unique way, the following fact will be useful:

the number of left parentheses in a term or formula is equal to the number of right parentheses; and if the word X comes at the beginning of a term or formula and is not the whole term or formula, then the number of left parentheses in X is greater than the number of right parentheses.

It is amusing to note that the role parentheses play in preventing ambiguities in our formal language is analogous to the role of punctuation in the natural language of everyday speech. For example, where one places the comma in the sentence "Execute we cannot show mercy!" has a crucial effect on the meaning; the decision about where to put the comma amounts to a choice between "Execute $\wedge$ we cannot show mercy!" and "Execute we cannot $\wedge$ show mercy!" By the way, it is possible to find sentences in natural language with ambiguities that cannot be cleared up by punctuation.

5. Three Axioms for the Theory of Algorithms

5.0. Our goal now is to prove the claim (*) in the last section. However, the rather diffuse ideas about algorithms with which we have been satisfied in the earlier sections are not sufficient to prove this claim.

The traditional method of continuing our argument would be to refine the idea of an algorithm, i.e., replace the somewhat indefinite concept of algorithm which we have been using, which has the advantage of being completely general, by a more precise and restricted notion, i.e., by a "special type of algorithm." By the way, it should be mentioned that this narrower notion of an algorithm would have a claim to being equivalent to our original definition, in the sense that the class of computable functions which arises from one definition is the same as the class of computable functions arising from another (and, therefore, the class of enumerable sets is the same for each definition). This claim that the classes of computable functions (or enumerable sets) are the same does not have the status of a theorem that can be proved, it is rather a conjecture that can be verified in practice. We can then construct a precise mathematical theory of functions which are computable by the "special type of algorithm" (here the proof of facts analogous to those in Problems 9 and 10 for Appendix C turns out to be technically the most difficult part of this theory). The unprovable belief that the class of functions computable by the "special type of algorithm" coincides with the class of all computable functions is important only for the purpose of justifying the meaningfulness of the theory. For more details on one such traditional approach, see Appendix C.

However, here we shall choose another method. Without committing ourselves to a special type of algorithm, we shall instead impose some restrictions on our original conception of an algorithm. These restrictions will be stated in the form of three axioms: the protocol axiom, the program axiom, and the arithmeticity axiom.

5.1. The First Axiom. We consider the process of applying an arbitrary algorithm A to the input x to obtain the output y. We shall assume that all of the intermediate computations, the entire computing process leading from x to y (where the word "computation" is meant in the broadest possible sense, by no means including just numerical calculation) can be written down in a record in such a way that this "protocol" contains exhaustive information about the successive stages of the computing process.

Example 1. When checking a computer program it is often necessary to print out not only the final result but also all of the intermediate results. The resulting "computer protocol" is a word in the output alphabet of the computer, perhaps with the addition of a sign for a blank, a sign for a new line, etc.

Example 2. Suppose we want to check whether children learning how to add a column of figures have correctly understood the addition algorithm. We might require that in their written work, in addition to the final result, they also write down all of their steps in some agreed upon notational system. One might use a notational system for the computations in which, for example, the protocol for adding 68 and 9967 would be

		1	11	111	1111	1111	
68, 9967	68	68	68	68	68	68	10035
	9967	9967	9967	9967	9967	9967	
		5	35	035	0035	10035	

Each term in the protocol is either a decimal number (in our example 10035), or else a pair of numbers (in our example 68, 9967), or else a four-storey structure such as

$$\begin{array}{r} 11 \\ 68 \\ 9967 \\ 35 \end{array}$$

(the "basement" and "attic" may be empty). It is not hard to make the protocol into a word in some alphabet. Namely, one need only introduce some additional symbols so that, for example, the above four-storey structure can be written first as a table

*	*	1	1	*
*	*	*	6	8
*	9	9	6	7
*	*	*	3	5

and then as a word: (**11*/***68/*9967/***35). The entire protocol is written as follows:

(68 + 9967) (*****/***68/*9967/*****)(***1*/***68/
*9967/****5)(**11*/***68/*9967/***35)(*111*/***68/
*9967/**035)(1111*/***68/*9967/*0035)(1111*/***68/
*9967/10035)(10035).

In this notational system the protocol for adding any two numbers is a word in the 15-letter alphabet {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, (,), /, + *}.

These examples suggest the following general considerations. We shall suppose that:

(1) for each algorithm A there is an alphabet Π_0 (the protocol alphabet), and all protocols describing the operation of A for the various inputs in the algorithm's domain of applicability together form a subset P_0 of the set Π_0^∞;

(2) there exist computable functions α and ω such that, for each protocol p_0 in P_0, the values α (p_0) and ω (p_0) are, respectively, the input x and the output y for which the protocol p_0 was written (i.e., p_0 is a protocol for the processing of x into y);

(3) P_0 is decidable relative to Π_0^∞.

We restate this more briefly in the form of an axiom, which we shall call the *protocol axiom*:

for each algorithm A there exist an alphabet Π_0, a decidable subset P_0 of the set Π_0^∞, a computable function α, and a computable function ω, such that:

$A(x) = y$ if and only if there exists p_0 in P_0 for which $\alpha(p_0) = x$ and $\omega(p_0) = y$.

This axiom has the following

Corollary 1. *The domain of applicability and the set of outputs of an algorithm are enumerable sets.*

▶ The first of these sets is $\alpha(P_0)$, and the second is $\omega(P_0)$. Both of these sets are enumerable in view of Lemmas 2 and 4 and Example 2 in Sec. 2. ■

Corollary 2. *The domain of definition and the set of values of any computable function are enumerable sets.*

▶ This follows immediately from Corollary 1. ■

Corollary 3. *The graph of any computable function (i.e., the set of all pairs $\langle x, y\rangle$ for which $f(x) = y$) is an enumerable set.*

▶ We apply the protocol axiom to the algorithm which computes f and find the corresponding set P_0 and functions α and ω. We then construct a computable function ψ by setting $\psi(p) = \langle\alpha(p), \omega(p)\rangle$. Finally, we note that the graph of f is precisely the set $\psi(P_0)$, so it remains to apply Lemma 4. ■

Remark 1. We could have obtained Corollary 2 as a consequence of Corollary 3. Namely, we could apply Corollary 2 of Lemma 4 and note that the domain of definition and the set of values of a function are, respectively, $\mathrm{pr}_1 M$ and $\mathrm{pr}_2 M$, where M is the graph of the function.

Remark 2. Enumerability of the graph of a function is not only a necessary condition (as we established in Corollary 3) but also a sufficient condition for the function to be computable. In fact, if the graph is the empty set, then the function is nowhere defined and so is computable. If the graph of the function f is nonempty and is enumerated by the computable function ψ, then one can use the following algorithm to compute f: to compute the value $f(a)$, go through the pairs $\psi(0)$, $\psi(1)$, $\psi(2)$, ... until you obtain a pair whose first term is a; then $f(a)$ is the second term in this pair.

5.2. The Second Axiom. Functions whose arguments lie in X and whose values lie in Y are customarily called functions *from X to Y*. Similarly, an algorithm whose possible inputs lie in X and whose outputs lie in Y will be called an algorithm *from* X *to* Y Here we may take $X = K^\infty$ and $Y = L^\infty$ where K and L are alphabets. An algorithm from K^∞ to L^∞ is a set of instructions, i.e., a text in English or some other language (perhaps an artificial language created especially for writing algorithms). Although in concrete situations there is usually no problem in deciding whether or not a given text is an algorithm, nevertheless the notion of a set of instructions is too vague to enable us unambiguously to distinguish between a text which is a set of instructions and a text which is not a set of instructions. We do not have a single sufficiently precise way of understanding what a set of instructions means. The instructions could be written in any of many languages, and even within a single language the problem of interpreting the meaning of a text is rather complicated.

Nevertheless, we shall assume (this will be the program axiom) that it is possible to identify with full certainty a set consisting of all sets of instructions according to a single uniform interpretation of what that means. This class of sets of instructions will be representative in a sense that will be made more precise below. The sets of instructions in this representative class will be called *programs.*

We shall say that two algorithms are *equivalent* if they have the same domain of applicability and if they give the same output when they process any input in the domain of applicability. A set of algorithms from K^∞ to L^∞ will be said to be *representative* (for the alphabets K and L) if any algorithm from K^∞ to L^∞ is equivalent to some algorithm in our set. When we said before that we want to be able to identify this set with "full certainty," we meant that we want it to be a decidable subset of the set of all words in some alphabet. When we say that we should have a "single uniform interpretation" of what a program is, we mean that there should be an algorithm U which is applicable to pairs $\langle$program p, input $a\rangle$ and which gives as output the result of applying the program p to the input a. (Here a denotes an input for p; $\langle p, a\rangle$ is an input for U.)

Remark 3. It is not hard to show that any of the traditional approaches using a "special type of algorithm" can be reduced to the above scheme. Any such refinement of the notion of algorithm essentially amounts to a particular choice of a set P_1 of programs and an algorithm U which explains how to apply the program to initial data; it is then claimed (as an unprovable stipulation) that the set P_1 is representative.

Thus, we shall assume that:

(1) for any two alphabets K and L there is an alphabet Π_1 (the program alphabet) and a set P_1 of algorithms which are called programs and are written in the alphabet Π_1 (i.e., $P_1 \subseteq \Pi_1^\infty$);

(2) there exists an algorithm U from $\Pi_1^\infty \times K^\infty$ to L^∞ (the algorithm for applying a program) such that $U(p, a)$ is the result of applying p to a;

(3) the set P_1 is representative;

(4) the set P_1 is decidable relative to Π_1^∞.

Here we are by no means assuming that the alphabet Π_1, the set P_1 and the algorithm U can be chosen only in one way. Any triple $\langle \Pi_1, P_1, U \rangle$, where Π_1 is an alphabet, P_1 is the set of all programs written in this alphabet, and U is an algorithm describing how a program processes input, will be called a *programming method from* K^∞ *to* L^∞ Thus, for fixed K and L there may be various possible programming methods.

Remark 4. Our assumptions (1)-(4) do not fully define what a "programming method" means. That concept in its entirety will remain something for us to understand on an intuitive level. The above assumptions merely give some properties of this concept (and not all the properties, as a deeper analysis will show), properties which we are assuming are satisfied by some triple.

We now proceed to state the second axiom. But first we need some notation. Suppose that G is an arbitrary algorithm from $\Pi_1^\infty \times K^\infty$ to L_1^∞ If $p \in \Pi_1^\infty$, then we let G_p denote the following algorithm from K^∞ to L^∞: for any a in K^∞, take the output from applying G_p to a to be the result of applying G to the pair $\langle p, a \rangle$; in other words, $G_p(a) \simeq G(p, a)$. Using this notation, we can restate our assumptions (1)-(4) as the following *program axiom*:

for any two alphabets K *and* L *there exist an alphabet* Π_1, *a decidable subset* P_1 *of the set* Π_1^∞, *and an algorithm* U *from* $\Pi_1^\infty \times K^\infty$ to L^∞, *with the following properties: for every algorithm* A *from* K^∞ *to* L^∞ *there is a* p *in* P_1 *such that the algorithms* A *and* U_p *are equivalent.*

This axiom also has some important corollaries. But first we give a few definitions.

Suppose that I, X and Y are sets, and F is a function from $I \times X$ to Y. If i is an element of I, then we let F_i denote the function from X to Y which is defined on x for which the pair $\langle i, x\rangle$ is in the domain of definition of F and which takes the value $F(i, x)$ at such an x. Using the conditional equality sign, we can abbreviate this definition as follows: $F_i(x) \simeq F(i, x)$.

Now suppose that Φ is some class of functions from X to Y We shall say that a function F from $I \times X$ to Y is *universal* for the class Φ if the following two conditions hold:

1° the function F_i belongs to the class Φ for every $i \in I$;

2° each function in Φ is F_i for some i; in other words, for every $\varphi \in \Phi$ there exists $i \in I$ such that $\varphi(x) \simeq F(i, x)$ for all $x \in X$.

Corollary 1 of the program axiom. *Suppose that* K *and* L *are two alphabets, and* Φ *is the family of all computable functions from* K^∞ *to* L^∞. *Then there exists a computable function from* $\mathbb{N} \times K^\infty$ *to* L^∞ *which is universal for the class* Φ.

▶ Condition 1° automatically holds for any computable function F (since if F is computable, then so are all of the F_i). So we need only construct a computable function F from $\mathbb{N} \times K^\infty$ to L^∞ which satisfies condition 2°. We consider the alphabet Π_1, the decidable subset P_1 of the set Π_1^∞, and the algorithm U from $\Pi_1^\infty \times K^\infty$ to L^∞, the existence of which is ensured by the program axiom. Since P_1 is a decidable subset of an enumerable set, it is enumerable, by Lemma 2; let f be a function which enumerates P_1. Then we claim that the function F defined by the relation

$$F(i, x) \simeq U(f(i), x)$$

has the desired property.

To see this, let φ be any computable function from K^∞ to L^∞, and let A be an algorithm which computes φ, i.e., $A(x) \simeq \varphi(x)$ for all $x \in K^\infty$. By the program axiom, there exists a p in P_1 such that the following conditional equality holds for all $x \in K^\infty$:

$$U(p, x) \simeq A(x).$$

Since $p \in P_1$, we have $p=f(i)$ for some i; then for this i we have the chain of conditional equalities:

$$F(i, x) \simeq U(f(i), x) \simeq U(p, x) \simeq A(x) \simeq \varphi(x),$$

which shows that our function F satisfies condition 2° in the definition of a universal function. ■

As a special case of Corollary 1 we have:

Corollary 2. *There exists a computable function F from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$ which is universal for the class of all computable functions from $\mathbb{N}$ to $\mathbb{N}$.*

▶ We obtain Corollary 2 from Corollary 1 if we take both K and L to be one of the digital alphabets for writing numbers, for example, the alphabet $\{|\}$. ■

We shall say that two functions f and g from X to Y are everywhere different if there is no x in X for which the conditional equality $f(x) \simeq g(x)$ holds. This means that for every x at least one of the functions f or g is defined at x, and, if both functions are defined at x, then they have different values there.

Corollary 3 (from Corollary 2). *There exists a computable function d from $\mathbb{N}$ to $\mathbb{N}$ such that no computable function from $\mathbb{N}$ to $\mathbb{N}$ can be everywhere different from d.*

▶ Let F be the universal function in Corollary 2. We take d to be the function defined by the relation:

$$d(i) \simeq F(i, i).$$

Then $d(i) \simeq F_i(i)$, so that d and F_i cannot be everywhere different. But, since any computable function from $\mathbb{N}$ to $\mathbb{N}$ is F_i for some i, this means that no computable function from $\mathbb{N}$ to $\mathbb{N}$ can be everywhere different from d. ■

This corollary might at first seem paradoxical, since it would seem that, for example, the function $d_1(x) \simeq d(x) + 1$ is everywhere different from d. The explanation for this apparent paradox is that d is a function which is not defined everywhere, so that at values of x for which d (and hence d_1) is undefined we have the conditional equality $d_1(x) \simeq d(x)$. But what if, instead of d_1, we considered a function D_1 which extends d_1 and is everywhere defined (this means that D_1 is a function which is everywhere defined and which coincides with d_1 wherever d_1 is defined)? Now this D_1 is everywhere different from d: if $d(x)$ is defined, then $d_1(x)$ is also defined and is equal to $d(x) + 1$, in which case $D_1(x) = d(x) + 1 \not\simeq d(x)$; while if $d(x)$ is not defined, then we also have $D_1(x) \not\simeq d(x)$, because the left side is defined and the right side is not. Have we found a contradiction to Corollary 3? No, there is no contradiction here, we have merely proved that an everywhere defined extension of the function d_1 cannot be computable. This gives us:

Corollary 4 (from Corollary 3). *There exists a computable* *function from* $\mathbb{N}$ *to* $\mathbb{N}$ *which does not have a computable extension defined on all of* $\mathbb{N}$.

Suppose that q is a computable function as in Corollary 4, i.e., it does not have a computable extension to $\mathbb{N}$. Could the domain of definition of q be a decidable subset of $\mathbb{N}$? It is easy to see that the answer is no. Namely, if the domain of definition were a decidable subset of $\mathbb{N}$, then the function Q defined by setting

$$Q(x)=\begin{cases} q(x), & \text{if } x \text{ is in the domain of definition of } q,\\ 0, & \text{if } x \text{ is not in the domain of definition of } q,\end{cases}$$

would be a computable everywhere defined extension of q. Thus, the domain of definition of q is a nondecidable set. According to Corollary 2 of the protocol axiom, this set is enumerable. We have thereby proved

Corollary 5 (from Corollary 4). *There exists an enumerable nondecidable subset of the set of natural numbers.*

The fact that such a subset of $\mathbb{N}$ exists is one of the most important facts to come out of the theory of algorithms.

Since a subset of the natural numbers is decidable if and only if both it and its complement are enumerable (by Lemma 3), the preceding corollary can be restated as follows:

Corollary 6 (from Corollary 5). *There exists an enumerable subset of the set of natural numbers whose complement is not enumerable.*

5.3. The Third Axiom. If one ignores the (rather important) fact that computers can only work with functions defined on finite sets of natural numbers (since extremely large values of the argument simply will not fit in the computer), we may suppose that the functions which computers can compute are the computable numerical functions as defined above. It is well known that the basic operations which a computer can perform are addition, multiplication, and the logical operations. Experience working with computers leads one to the conviction that any computable function can be programmed using these operations. Consequently, one is led to believe that any enumerable set of natural numbers (since it is the set of values of a computable function) can be described in terms of addition, multiplication, and the logical operations. These considerations (for more details, see Appendix C) motivate the introduction of the following *arithmeticity axiom*:

every enumerable set of natural numbers is arithmetic.

Finally, the claim in the last section, which it was our goal to prove, is now an immediate consequence of this axiom:

there exists an arithmetic set which is not enumerable.

Namely, the complement of the set in Corollary 6 above is such an arithmetic set: it is a nonenumerable set with enumerable complement. Note that this set will be arithmetic because its complement is arithmetic (the first property of arithmetic sets).

We have thereby finished the proof of the incompleteness theorem. As we noted before, the existence of a nonenumerable arithmetic set implies the existence of a nonenumerable set such that membership in the set is expressible in arithmetic. This implies that there does not exist a deductive system for ⟨A, T⟩ which is complete and consistent relative to a certain enumerable subset V. Consequently, no consistent deductive system can be complete for ⟨A, T⟩.

Appendixes

A. The Syntactic and Semantic Formulations of the Incompleteness Theorem

A.1. Statement of the Problem. It is natural to call the version of Gödel's incompleteness theorem that we proved a "semantic" formulation, since it says something about the truth of statements of arithmetic. In general, the word "semantic" refers to the part of the study of a language (in our case the language of arithmetic) which is concerned with the meaning of expressions and their truth or falsity. This part of linguistic study is to be distinguished from syntax which investigates expressions in the language as combinations of symbols apart from their meaning. (Sometimes the word "syntactic" is used in a narrower sense, referring to the part of grammar which studies how words are combined in sentences of a natural language.) We would like to proceed to a syntactic formulation of the incompleteness theorem, i.e., our present purpose is to remove to whatever extent possible every reference to the truth of statements.

A completely satisfactory execution of this task would require us to make the notion of a proof much more concrete. This would take us beyond the scope of this short book. Nevertheless, in this appendix we shall take a few steps in this direction.

A.2. Syntactic Consistency and Syntactic Completeness. Suppose that $\langle \mathrm{P}, P, \delta \rangle$ is a deductive system over the alphabet A of the language of arithmetic. (For the remainder of this appendix we shall only be concerned with deductive systems over A.) We shall say that the deductive system is *syntactically consistent* if there does not exist a closed formula α for which both α and $\neg\alpha$ are provable in the deductive system. We shall say that the deductive system is *syntactically complete* if at least one of the closed formulas α or $\neg\alpha$ is provable in the deductive system for any closed formula α.

These definitions can be stated more briefly if one first defines the notion of a closed formula which is refutable in the deductive system: this is a closed formula α such that $\neg\alpha$ is provable in the deductive system. We can now restate the above definitions as follows: a deductive system is syntactically consistent if there is no closed formula which is both provable and refutable in it, and the system is syntactically complete if every closed formula is either provable or refutable.

The lemma that follows gives the connection between these notions and our earlier notions of a deductive system which is consistent or complete for $\langle A, T\rangle$. We recall that a deductive system is said to be consistent if all provable closed formulas are true, and it is said to be complete if all true closed formulas are provable.

Lemma A.1. (A) *A consistent deductive system is syntactically consistent.*

(B\ *A complete deductive system is syntactically complete.*

(C) *If a deductive system is consistent, then it is complete if and only if it is syntactically complete.*

▶ (A) If α and $\neg\alpha$ were both provable in a consistent deductive system, then α and $\neg\alpha$ would both be true, and this contradicts the definition of truth. (B) One of the closed formulas α or $\neg\alpha$ must be true, and hence must be provable if the deductive system is complete. (C) Suppose that the deductive system is consistent and syntactically complete. To show that it is complete, let α be a true closed formula. Then $\neg\alpha$ is false, and so $\neg\alpha$ cannot be provable (because the system is consistent). Then, since the system is syntactically complete, α must be provable. ■

Because of the lemma, it is natural to propose the following syntactic version of the incompleteness theorem:

there does not exist a syntactically consistent and syntactically complete deductive system for the language of arithmetic.

This version has the advantage that, in the first place, it implies the semantic version of the incompleteness theorem we proved above, and, in the second place, there is nothing in it which refers to the truth of a statement. However, this statement as it stands is false. For example, a deductive system in which a closed formula is provable if and only if the symbol $\neg$ occurs an even number of times (such a deductive system exists, by Theorem 1) is syntactically consistent and syntactically complete.

Upon reflection, we arrive at the conclusion that this failure is due to the absence in the above formulation of any connection with the usual meaning of the symbols of the alphabet A. In our example of a syntactically consistent and syntactically complete deductive system, both the formula

$(2 \cdot 2) = 4$ and the formula $(2 \cdot 2) = 5$ are provable. We can extricate ourselves from this situation if we impose the requirement on the deductive system that certain closed formulas must be provable in it. We now make this more precise.

Suppose that D_0 and D are deductive systems. We shall say that D is an *extension* of D_0 if every closed formula which is provable in D_0 is also provable in D. (In this case, obviously every closed formula which is refutable in D_0 is also refutable in D.) We shall say that a deductive system D_0 is *completable* if it has a *completion*, i.e., an extension which is a syntactically consistent and syntactically complete deductive system. The example above shows that the empty deductive system (in which no statement is provable) is completable.

Using the concept of completability, we can propose another syntactic version of Gödel's incompleteness theorem:

there exists an uncompletable deductive system.

But this version is meaningless since any syntactically inconsistent deductive system is uncompletable. Besides, we want the syntactic version of the incompleteness theorem to imply the semantic version proved above. This requirement will be satisfied if we choose the following version:

there exists an uncompletable consistent deductive system. (This assertion implies, by the way, that there cannot exist a complete and consistent deductive system, since such a deductive system would be a completion of any consistent system.) It is this syntactic version which we shall study.

But before proving this syntactic incompleteness theorem, we first explain why it is better than our original (semantic) version of the theorem. After all, it refers to the property of consistency, which is defined using the notion of truth. The crucial point is that it is possible to give an uncompletable consistent deductive system explicitly, and for this explicitly described deductive system the uncompletability property does not involve the concept of truth. (Of course, in our eyes the value of this property comes from our belief in the consistency of the deductive system.)

We now proceed to the proof of the above statement. We shall need some new concepts from the theory of algorithms.

A.3. Inseparable Sets. Suppose that K is an alphabet and A and B are disjoint subsets of K^∞. We shall say that the set C *separates* A *from* B if $A \subset C$ and $B \cap C \quad \varnothing$. If the set C separates A from B, then its complement (in K^∞)

separates B from A. We shall say that A and B are *separable* if there exists a decidable subset C of the set K^∞ which separates A from B. (In this case the complement of C is a decidable subset of K^∞ which separates B from A.)

Lemma A.2. *Two disjoint sets A and B are separable if and only if the function from K^∞ to $\mathbb{N}$ which is defined by setting*

$$f(x) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{if } x \in B, \\ \text{undefined}, & \text{if } x \notin A \cup B, \end{cases}$$

has an everywhere defined computable extension.

▶ If g is an extension of f which is computable and is defined everywhere, then the decidable set $\{x \mid g(x) = 1\}$ separates A from B. Conversely, if C is a decidable set which separates A from B, then the computable function g which equals 1 on elements of C and 0 elsewhere is an extension of f. ■

Lemma A.3. *There exist inseparable enumerable subsets of* $\mathbb{N}$.

▶ According to the preceding lemma, it suffices to show that there exists a computable function h from $\mathbb{N}$ to $\mathbb{N}$ which takes on only the two values 0 and 1 and which does not have an everywhere defined computable extension. In that case the sets $\{x \mid h(x) = 1\}$ and $\{x \mid h(x) = 0\}$ will be enumerable (by Lemma 6 and Corollary 1 of the protocol axiom) and inseparable. In order to construct a function h with the desired properties, we refer to the proof of Corollary 4 of the program axiom in Sec. 5 and consider the function d which has the property that no computable function can be everywhere different from it. We define the function h as follows:

$$h(x) = \begin{cases} 1, & \text{if } d(x) = 0, \\ 0, & \text{if } d(x) \text{ is defined and nonzero}, \\ \text{undefined}, & \text{if } d(x) \text{ is undefined}. \end{cases}$$

Any everywhere defined extension of h would be everywhere different from d; hence it could not be computable. ■

We now state a criterion for a deductive system to be uncompletable which uses the notion of inseparability.

Theorem A.1. *If the set of provable closed formulas and the set of refutable closed formulas in a given deductive system are inseparable, then this deductive system is uncompletable.*

▶ If the deductive system had a completion, then the set of provable closed formulas and the set of refutable closed formulas in the completion would be disjoint enumerable sets which together exhaust the set of all closed formulas. By Lemma 3, each of these two sets—in particular, the set S of provable closed formulas—is a decidable subset of the set of all closed formulas, and hence a decidable subset of the set A^{∞}. The set S separates the set of closed formulas provable in our original deductive system from the set of closed formulas which are refutable in that system. But this contradicts the hypothesis of the theorem. ■

A.4. Construction of an Uncompletable Deductive System. We shall use Theorem A.1 to construct an uncompletable deductive system. Let P and Q be inseparable enumerable subsets of $\mathbb{N}$ (such sets exist by Lemma A.3). P is an arithmetic set, by the arithmeticity axiom (see Sec. 5); let P be associated with the formula α. Let $[n \in P]$ denote the formula $S_{n}^{x_1}\alpha$ (where n is a number). The formula $[n \in P]$ is true if and only if $n \in P$. For each n in P we consider the (true) formula $[n \in P]$; for each n in Q we consider the (also true) formula $\neg\, [n \in P]$. These formulas form an enumerable set. According to Theorem 1, there exists a deductive system in which these formulas and only these formulas are provable. This deductive system is consistent. We now show that it is uncompletable. According to Theorem A.1, in order to do this it suffices to prove that the set of provable formulas in the deductive system and the set of refutable formulas are inseparable. We now show this. If $n \in P$, then the formula $[n \in P]$ is provable; if $n \in Q$, then the formula $[n \in P]$ is refutable. Thus, if S were a decidable set which separated the provable formulas from the refutable ones, then the decidable set $\{n \mid [n \in P] \in S\}$ would separate P from Q, and this is impossible. We have thus constructed an uncompletable deductive system.

B. Arithmetic Sets and Tarski's Theorem on the Nonarithmeticity of the Set of True Formulas of the Language of Arithmetic

As explained in Sec. 4, the closed formulas of the language of arithmetic are statements about properties of the set of natural numbers and the operations of addition and multiplication. These statements can be either true or false. However, the question "True or false?" has no meaning for formulas with parameters. If we replace the parameters in a formula by numbers, then we obtain a closed formula whose truth value in general depends upon which numbers we substituted in place of the variables. Thus, formulas with parameters can be interpreted as properties of natural numbers.

Example 1. The result of substituting n in place of x_1 in the formula $\exists x_2\ ((x_2 + x_2) = x_1)$ is a true statement if and only if n is even. Thus, we could say that this formula expresses the property "x_1 is even." It is also common to say (somewhat imprecisely) that this formula is true for even values of x_1 and false for odd values of x_1.

Example 2. The formula $\exists x_3\ ((x_1 + x_3) = x_2)$ expresses the property "$x_1 \leqslant x_2$."

Example 3. The formula $\exists x_2\ ((x_1 \cdot x_2) = x_3)$ expresses the property "x_1 divides x_3."

Example 4. Let [x_1 divides x_3] denote the formula in Example 3. Then the formula

$$\forall x_1\ ([x_1 \text{ divides } x_3] \rightarrow ((x_1 = 1) \vee (x_1 = x_3)))$$

expresses the property "x_3 is prime or equal to 1."

Example 5. Let [x_1 is even] denote the formula in Example 1. Then the formula

$$\forall x_1\ ([x_1 \text{ divides } x_3] \rightarrow ([x_1 \text{ is even}] \vee (x_1 = 1)))$$

expresses the property "every divisor of x_3 is either even or equal to 1," i.e., "x_3 is a power of 2."

Properties which are expressible by the formulas of the language of arithmetic are called arithmetic properties. The subset of $\mathbb{N}^k$ consisting of k-tuples of natural numbers having a certain arithmetic property is called an arithmetic subset of $\mathbb{N}^k$. The definition of an arithmetic subset of $\mathbb{N}$ that was given in Sec. 4 is a special case ($k = 1$) of this definition.

We now make these definitions precise. Suppose that α is a formula in the language of arithmetic, $w_1, \quad ., w_p$ are variables, and $c_1, \quad ., c_p$ are numbers. By the *result of substituting* $c_1, \quad ., c_p$ *in place of* $w_1, \quad ., w_p$ *in* α we mean the formula

$$S^{w_1 \ldots w_p}_{c_1 \ldots c_p}\alpha = S^{w_p}_{c_p} \ldots S^{w_2}_{c_2} S^{w_1}_{c_1}\alpha,$$

which is obtained from α by successively substituting c_1 in place of w_1, c_2 in place of w_2, . . ., c_p in place of w_p. (It is easy to see that these substitutions can be performed in any order; for example, the result would have been the same if we had defined $S^{w_1 \ldots w_p}_{c_1 \ldots c_p}\alpha$ as $S^{w_1}_{c_1} \ldots S^{w_p}_{c_p}\alpha$.)

Let α be a formula of arithmetic whose parameters are a subset of $x_1, \ldots, x_k$. We consider the subset of $\mathbb{N}^k$ consisting of k-tuples $\langle c_1, \ldots, c_k \rangle$ for which the closed formula $S^{x_1 \ldots x_k}_{c_1 \ldots c_k}\alpha$ is true. We shall say that this set is *associated with* the formula α. A set which is associated with a formula of arithmetic will be called an *arithmetic* set. When $k = 1$, this gives us the earlier definition (in Sec. 4) of an arithmetic subset of the set of natural numbers. We shall identify properties with the sets of objects having the property, and so shall speak of the arithmeticity of properties of natural numbers.

Example 6. The sets $\{\langle x_1, x_2 \rangle \mid x_1 = x_2\}$, $\{\langle x_1, x_2, x_3 \rangle \mid x_1 + x_2 = x_3\}$ and $\{\langle x_1, x_2, x_3 \rangle \mid x_1 \cdot x_2 = x_3\}$ are all arithmetic, since they are associated with the formulas $x_1 = x_2$, $(x_1 + x_2) = x_3$, and $(x_1 \cdot x_2) = x_3$, respectively.

Example 7. The set $\{\langle x_1, x_2 \rangle \mid x_1 \leqslant x_2\}$ is associated with the formula in Example 2, and so is an arithmetic set.

Example 8. The set $\{\langle x_1, x_2 \rangle \mid x_1 \text{ divides } x_2\}$ is arithmetic. To construct a formula with which it is associated, we must slightly modify the formula in Example 3 by interchanging x_2 and x_3.

Example 9. The set of prime numbers and the set of powers of 2 are arithmetic subsets of the set of natural numbers (see Examples 4 and 5).

The properties of arithmetic subsets of $\mathbb{N}$ that were given in Sec. 4 hold more generally for arithmetic subsets of $\mathbb{N}^k$. In particular, we have:

Lemma B.1. (a) *The complement (in $\mathbb{N}^k$) of an arithmetic subset of $\mathbb{N}^k$ is arithmetic;*

(b) *the union or intersection of two arithmetic subsets of $\mathbb{N}^k$ is arithmetic.*

The next lemma says that arithmeticity is preserved if one permutes the coordinates.

Lemma B.2. *Let σ be a permutation of the set $\{1, \ldots, k\}$ (i.e., a one-to-one correspondence from the set to itself), and let M be an arithmetic subset of $\mathbb{N}^k$. Then the set*

$$M^\sigma = \{\langle x_1, \ldots, x_k \rangle \mid \langle x_{\sigma(1)}, \ldots, x_{\sigma(k)} \rangle \in M\}$$

is arithmetic.

▶ If the set M is associated with the formula α, then the set M^σ is associated with the formula α^σ which is obtained from α by replacing each variable in the list $x_1, \quad ., x_k$ by the corresponding variable in the list $x_{\sigma(1)}, \quad . ., x_{\sigma(k)}$. ■

The next two lemmas give a connection between the classes of arithmetic subsets of $\mathbb{N}^k$ for different k.

Lemma B.3. *If M is an arithmetic subset of $\mathbb{N}^k$, then $M \times \mathbb{N}^h$ is an arithmetic subset of $\mathbb{N}^{k+h}$.*

▶ In fact, $M \times \mathbb{N}^h$ is associated with the same formula as M. ■

Lemma B.4. *If $M \subset \mathbb{N}^{k+h}$ is an arithmetic subset, then its projection onto the first k coordinates, i.e., the set*

$$M' = \{\langle x_1, \qquad x_k\rangle \mid \exists x_{k+1} \quad . \exists x_{k+h} (\langle x_1, \qquad x_{k+h}\rangle \in M)\},$$

is an arithmetic subset of $\mathbb{N}^k$.

▶ If M is associated with the formula α, then M' is associated with the formula $\exists x_{k+1} \ldots \exists x_{k+h}\alpha$.

Combining Lemmas B.2 and B.4, we can conclude that the projection of an arithmetic set onto any set of axes is arithmetic. ■

Suppose that $M \subset \mathbb{N}^2$ is an arithmetic set. For each $n \in \mathbb{N}$ we consider the "cross-section" M_n, which is the set of x for which $\langle n, x\rangle \in M$. Since it is a projection of the set $(\{n\} \times \mathbb{N}) \cap M$, it is arithmetic. We shall say that $M \subset \mathbb{N}^2$ is a *universal* arithmetic set if any arithmetic subset of $\mathbb{N}$ is a cross-section of M. It turns out that there is no such set.

Theorem B.1. *A universal arithmetic set does not exist. That is, for any arithmetic set $M \subset \mathbb{N}^2$ there exists an arithmetic set $Q \subset \mathbb{N}$ which is different from every cross-section of M.*

▶ The set $Q = \{x \mid \langle x, x\rangle \notin M\}$ is arithmetic, since it is a projection of the set $(\mathbb{N}^2 \setminus M) \cap \{\langle x, y\rangle \mid x = y\}$. But it cannot be a cross-section of M, since if Q were the same as M_n, then, by the definition of M_n, we would have $n \in Q \Leftrightarrow \langle n, n\rangle \in M$; but $n \in Q \Leftrightarrow \langle n, n\rangle \notin M$, by the definition of Q. (In other words, Q and M_n cannot coincide because they "look different" at n.) ■

A function f from $\mathbb{N}^k$ to $\mathbb{N}^h$ is said to be *arithmetic* if its graph is an arithmetic subset of $\mathbb{N}^{k+h}$.

Lemma B.5. *The image and preimage of an arithmetic set under an arithmetic function are arithmetic sets.*

▶ As a first case let us consider the image of an arithmetic set $A \subset \mathbb{N}$ under an arithmetic function f from $\mathbb{N}$ to $\mathbb{N}$. This image is a projection of the set (graph of f) $\cap$ $(A \times \mathbb{N})$, and so is an arithmetic set. In other words, if we let $[f(x_1) = x_2]$ denote the formula with which the graph of f is associated and let $[x_1 \in A]$ denote the formula with which the set A is associated, then the formula

$$\exists x_1([f(x_1) = x_2] \wedge [x_1 \in A])$$

is true for values of x_2 belonging to the image of A and for no other values. So in order to find a formula with which the image of A is associated, it suffices to permute the variables (interchange x_1 and x_2).

The preimage of an arithmetic set A under the function f is associated with the formula

$$\exists x_2([f(x_1) = x_2] \wedge [x_2 \in A]),$$

where $[x_2 \in A]$ denotes the formula obtained from $[x_1 \in A]$ by interchanging the variables x_1 and x_2. The general case of a function from $\mathbb{N}^k$ to $\mathbb{N}^h$ is proved in the same way. ■

We are interested in proving Tarski's theorem, which says:

the set of true formulas of arithmetic is nonarithmetic.

In order to make sense of this assertion, we have to explain what we mean by arithmeticity of a set of formulas, which is a subset of A^∞ This can be done as follows: choose a one-to-one correspondence between A^∞ and $\mathbb{N}$ (a *numbering* of A^∞), so that each word X in A^∞ is associated to some natural number (called the number of X in this numbering). We say that a set $M \subset \mathrm{A}^\infty$ is arithmetic (with respect to the chosen numbering) if the set of numbers of the words in M is an arithmetic subset of $\mathbb{N}$.

Of course, this definition depends on the choice of numbering of the words in the alphabet A. We say that two numberings are *arithmetically equivalent* if the function which goes from the number of a word in one numbering to its number in the other numbering is an arithmetic function.

Lemma B.6. *If a set $M \subset \mathrm{A}^\infty$ is arithmetic relative to a given numbering π_1, then it is arithmetic relative to any numbering π_2 which is arithmetically equivalent to π_1.*

▶ By assumption, $\pi_1(M)$ (which consists of all π_1-numbers of words in M) is an arithmetic set. Since the set $\pi_2(M)$ is the image of $\pi_1(M)$ under the function which goes from π_2-numbers to π_1-numbers, and since this function is arithmetic by assumption, it follows that $\pi_2(\dot{M})$ is also arithmetic. ■

Finally, we define an *arithmetic set* of words in the alphabet A to be a set which is arithmetic relative to some computable numbering of A^∞. (A numbering is said to be *computable* if the function which associates a number to every word is a computable function. In this case the inverse, which associates the word with number n to a natural number n, is also a computable function. The existence of computable numberings of A^∞ was established in Example 3 of Sec. 2.)

We now show that this definition can equally well be given in the form: an arithmetic set of words is a set which is arithmetic relative to any computable numbering of A^∞. If π_1 and π_2 are two computable numberings, then the function which goes from the π_1-number of a word to its π_2-number is a computable function from $\mathbb{N}$ to $\mathbb{N}$. (It can be computed by the following algorithm: given an argument x, run through all words in the alphabet A, compute their π_1-numbers, and wait for a word to appear whose π_1-number is x; once this word has been found, compute its π_2-number.) Thus, our claim follows once we prove the following lemma.

Lemma B.7. *Every computable function from* $\mathbb{N}^p$ to $\mathbb{N}^q$ *is arithmetic.*

▶ The graph of a computable function from $\mathbb{N}^p$ to $\mathbb{N}^q$ is an enumerable subset of $\mathbb{N}^{p+q}$ by Corollary 3 of the protocol axiom. Hence the lemma follows from the following strengthened form of the arithmeticity axiom:

every enumerable subset of $\mathbb{N}^k$ *is arithmetic.*

(In Sec. 5 we called the special case of this for $k = 1$ the arithmeticity axiom.) ■

Theorem B.2. *The set T of true formulas of arithmetic is not an arithmetic set.*

▶ We shall show that if T were arithmetic, then a universal arithmetic set would exist, in contradiction to Theorem B.1. Following Gödel, we shall call a formula a *class formula* if it has no parameters other than x_1. The set of all class formulas is a decidable subset of the enumerable set A^∞, and so is enumerable. We fix an enumeration $\alpha_0, \alpha_1, \alpha_2,$ of the set of class formulas. We consider the set

$M = \{\langle n, m\rangle \mid$ the result of substituting m in place of x_1 in α_n is a true statement$\}$.

Since the nth cross-section of this set is obviously associated with the formula α_n, it follows that the cross-sections of this set exhaust all arithmetic subsets of $\mathbb{N}$. It remains to show that if T were arithmetic, then M would also be arithmetic.

Recalling the definition of arithmeticity of a set of words, we fix an arbitrary computable numbering of A^∞ Let T' be the set of numbers of words in T under this numbering. Let S be the function which associates to the pair $\langle m, n\rangle$ the number of the word which is the result of substituting m in place of x_1 in α_n. S is a computable function, and so, by Lemma B.7, it is an arithmetic function. The set M is the preimage of T' under the function S. Thus, the arithmeticity of T' implies that M is arithmetic, by Lemma B.5. Theorem B.2 is proved. ■

A close examination of the proof of Theorem B.2 reveals that it is connected with the "liar paradox." We briefly explain this connection.

The liar paradox is the following. Someone announces: "What I am saying now is a lie." Is this statement true or false? Either answer to this question leads to a contradiction. If we say the statement is true, then, because of the very meaning of the statement, it must be false; and conversely. We nowgive a presentation of our proof of Theorem B.2 in a form which resembles this paradox.

▶ Suppose that the set of numbers of true statements of arithmetic is an arithmetic set. Let [word with number x_3 is true] denote the formula which has one parameter x_3 and which expresses the property "the word with number x_3 belongs to T, " i.e., the property "$x_3 \in T'$." The function S is arithmetic; we let

[x_3 is the number of the result of substituting x_2 in the x_1th class formula]

denote the formula with which the graph of S is associated. The formula

$\exists x_3$ ([word with number x_3 is true]

$\wedge$ [x_3 is the number of the result of substituting x_2 in the x_1th class formula])

has parameters x_1 and x_2. The set M is associated with this formula, which we shall henceforth denote by

[result of substituting x_2 in the x_1th class formula is true].

The rest of the argument follows the proof of the theorem that there is no universal arithmetic set. We consider the formula

$\neg\exists x_2\,((x_1 = x_2) \wedge$ [result of substituting x_2 in the x_1th class formula is true]),

which we shall denote by

[result of substituting x_1 in the x_1th class formula is false].

This last formula has one parameter x_1, and corresponds to the set Q in the proof of Theorem B.1, in the sense that the result of substituting n in place of x_1 in this formula is true if and only if the result of substituting n in the nth class formula is false. This formula is a class formula, and so has some number (which we denote n) in the enumeration of the class formulas. We now substitute n in place of x_1 in our formula, and denote the result of this substitution by

[result of substituting n in the nth class formula is false].

This is a closed formula which is true if and only if the result of substituting the number n in the nth class formula is false. But this closed formula is nothing other than that very result of substituting n in the nth class formula. Thus, the closed formula

[result of substituting n in the nth class formula is false]

is true if and only if it is false. We would have been completely justified in denoting the statement: [I am lying]. We have obtained a contradiction, which shows that the set of true formulas of arithmetic is not an arithmetic set. ∎

C. The Language of Address Programs, the Extended Language of Arithmetic, and the Arithmeticity Axiom

In this appendix we shall attempt to justify the arithmeticity axiom. Our plan of argument is as follows. First we shall describe a certain concrete class of algorithms—the class of address programs. It is natural to use the term "address-computable" to refer to functions which are computable using algorithms from this class. Next we shall prove that the range of values of any address-computable function is an arithmetic set. This will justify the arithmeticity axiom—provided that one believes that every computable function is address-computable.

As an aid in carrying out this program we shall introduce an extended language of arithmetic, which has some additional means of expression not in the language of arithmetic that was described in Sec. 4. We shall show that the set of values of any address-computable function can be described by a formula in this extended language of arithmetic. Then we shall show that the use of this extended language was not actually essential, i.e., for every formula in the extended language we can find a substitute in the usual language of arithmetic. This will imply that the set of values of any address-computable function can be described by a formula in the language of arithmetic, i.e., it is an arithmetic set.

We begin with the following simple observation: in order to justify the arithmeticity axiom (and even the strengthened version in Appendix B), it suffices for us to be able to prove that

the graph of any computable function from $\mathbb{N}$ *to* $\mathbb{N}$ *is an arithmetic subset of* $\mathbb{N}^2$.

(The definition of an arithmetic subset of $\mathbb{N}^2$ was given in Appendix B.) To see this, suppose that this assertion is true. Then every enumerable subset of $\mathbb{N}$ is arithmetic, since it is a projection of the graph of the computable function which enumerates it (see Lemma B.4). We now prove from this that the strengthened version of the arithmeticity axiom holds.

Let $M \subset \mathbb{N}^k$ be an enumerable set, and let g be the function from $\mathbb{N}$ to $\mathbb{N}^k$ which enumerates it. The value of g at the number n is a k-tuple $g(n) = \langle g_1(n), \ldots, g_k(n)\rangle$. The functions $g_1, \ldots, g_k$ are computable functions from $\mathbb{N}$ to $\mathbb{N}$, and so, by our assumption, their graphs are arithmetic. We let $[g_i(x_1) = x_2]$ denote the formulas with which these graphs are associated. The graph of g is an arithmetic subset of $\mathbb{N}^{k+1}$ because it is associated with the formula

$$[g_1(x_1) = x_2] \wedge ([g_2(x_1) = x_3] \wedge (\ldots \wedge [g_k(x_1) = x_{k+1}])\ldots),$$

where we have let $[g_i(x_1) = x_{i+1}]$ denote the formula obtained from $[g_i(x_1) = x_2]$ by the permutation of variables which interchanges x_2 and x_{i+1}. The set M is the projection of the graph of g onto the x_2, x_{k+1}-axes, and so is an arithmetic set.

Thus, our goal is now to prove that the graph of any computable function from $\mathbb{N}$ to $\mathbb{N}$ is an arithmetic subset of $\mathbb{N}^2$. This is not, however, a trivial task, as the reader will undoubtedly agree after trying to prove directly, for example, the arithmeticity of the exponential function to base 2, i.e., the arithmeticity of the set $\{\langle x, y\rangle | y = 2^x\}$.

C.1. The Language of Address Programs. We now describe a class of algorithms of a special type, which we shall call address programs. These programs are reminiscent of real "machine language" programs used with actual computers.

An *address program* is a sequence of commands listed in order. Each command has one of the following forms:

1° $R(a) \leftarrow b$ (assigning a value);
2° $R(a) \leftarrow R(b)$ (moving a value);
3° $R(a) \leftarrow R(b) + R(c)$ (addition);
4° $R(a) \leftarrow R(b) \cdot R(c)$ (multiplication);
5° **GO TO** n (unconditional transfer);
6° **IF** $R(a) = R(b)$ **GO TO** m **ELSE GO TO** n (conditional transfer);
7° **STOP.**

Here a, b and c are arbitrary natural numbers ("register numbers"), and m and n are natural numbers which denote the order of a command in the program. The last command in any program must be a command of the form 7° We have given names for the types of commands in parentheses.

Here is a simple example of an address program:

Example 1.

```
1 R(1) ← 1
2 R(2) ← 1
3 R(3) ← 1
4 R(2) ← R(2)·R(1)
5 R(1) ← R(1) + R(3)
6 IF R(1) = R(0) GO TO 7 ELSE GO TO 4
7 R(0) ← R(2)
8 STOP.
```

Address programs can be executed on "address machines" (which exist only in theory).

An address machine is assumed to have infinitely many locations for storing natural numbers (its memory). These locations are called registers. At a given instant exactly one number is stored in each register. The registers are numbered 0, 1, 2, and are denoted *R* (0), *R* (1), *R* (2), respectively.

An address machine executes the program in the order that the commands are numbered; this order is violated only when a conditional or unconditional transfer command is executed. Before giving more precise definitions, we shall describe how an address machine goes through the program in Example 1. Suppose that we start with the number 100 in the register *R* (0) and zero in each of the other registers. The first three commands assign the initial value 1 to the registers *R* (1)-*R* (3). The number in the register *R* (3) does not change during the rest of the execution of the program, the number in *R* (1) increases by 1 from time to time (at command 5; recall that *R* (3) always stores the number 1), and the number in *R* (2) is multiplied by the value in *R* (1) from time to time. The execution stops when the number in *R* (1) becomes equal to the number in *R* (0). The following table shows how the numbers in the registers change as the program is executed:

command number	*R* (0)	*R* (1)	*R* (2)	*R* (3)	*R* (4)
1	100	0	0	0	0
2	100	1	0	0	0
3	100	1	1	0	0
4	100	1	1	1	0
5	100	1	1	1	0
6	100	2	1	1	0
4	100	2	1	1	0
5	100	2	2	1	0
6	100	3	2	1	0
4	100	3	2	1	0
5	100	3	6	1	0
6	100	4	6	1	0
4	100	4	6	1	0
			•		
6	100	99	98!	1	0
4	100	99	98!	1	0
5	100	99	99!	1	0
6	100	100	99!	1	0
7	100	100	99!	1	0
8	99!	100	99!	1	0

As a result of the execution of this program the number $99!(1\cdot 2\cdot\ldots\cdot 99)$ is put into the register $R(0)$. If 200 rather than 100 were in R (0) at the beginning, then at the end we would have to fit the number $199!$ $(1\cdot 2\cdot\ldots\cdot 199)$ into our register R (0). And if all registers stored zero at the beginning, then the execution of the program would never stop.

We now give some precise definitions. The *state* of an address machine is an infinite sequence of natural numbers $s = (s_0, s_1, \ldots)$ almost all of which (i.e., all but finitely many) are zero. If $s_0 = 0$, then we call the state a *final* state (or *stop*); if $s_0 \geqslant 1$, then we call it a *working* state, and we call s_0 the *command number being executed*. We call s_{i+1} the *number in the ith register.*

Let p be an address program, and let $s = (s_0, s_1, \ldots)$ be a working state. We say that p is *applicable* to the state s if s_0 is the number of a command of p (there might not be a command with number s_0 if s_0 is too large). In this case we define a state s' which is called the *immediate result of applying the program p to the state s*. This state $s' = (s'_0, s'_1, \ldots)$ is determined as follows:

1° if command number s_0 has the form $R\ (a) \leftarrow b$, then $s'_0 = s_0 + 1$, $s'_{i+1} = s_{i+1}$ for $i \neq a$, and $s'_{a+1} = b$ (all registers except for the ath remain unchanged, the number in the ath register is replaced by b, and the machine moves on to the next command);

2° if command number s_0 has the form $R\ (a) \leftarrow R\ (b)$, then $s'_0 = s_0 + 1$, $s'_{i+1} = s_{i+1}$ for $i \neq a$, and $s'_{a+1} = s_{b+1}$ (all registers except for the ath remain unchanged, the number in the ath register is replaced by the number in the bth register, and the machine moves on to the next command);

3° if command number s_0 has the form $R\ (a) \leftarrow R\ (b) + R\ (c)$, then $s'_0 = s_0 + 1$, $s'_{i+1} = s_{i+1}$ for $i \neq a$, and $s'_{a+1} = s_{b+1} + s_{c+1}$ (all registers except for the ath remain unchanged, the number in the ath register is replaced by the sum of the numbers in the bth and cth registers, and the machine moves on to the next command);

4° if command number s_0 has the form $R\ (a) \leftarrow R\ (b) \cdot R\ (c)$, then $s'_0 = s_0 + 1$, $s'_{i+1} = s_{i+1}$ for $i \neq a$, and $s'_{a+1} = s_{b+1} \cdot s_{c+1}$ (same as 3° except with multiplication instead of addition);

5° if command number s_0 has the form **GO TO** n, then $s'_0 = n$, and $s'_{i+1} = s_{i+1}$ for all i (all registers remain unchanged, and the machine moves on to the nth command);

6° if command number s_0 has the form **IF** $R\ (a) = R\ (b)$

GO TO m **ELSE GO TO** n, then $s'_{i+1} = s_{i+1}$ for all i, and s'_0 is equal to m if $s_{a+1} = s_{b+1}$ and equal to n if $s_{a+1} \neq s_{b+1}$ (all registers remain unchanged, and the machine moves either to the mth or nth command, depending on whether or not the numbers in the ath and bth registers are equal);

7° if command number s_0 has the form **STOP**, then $s'_{i+1} = s_{i+1}$ for all i and $s'_0 = 0$ (the machine goes to the final state).

This completes the definition of the immediate result of applying a program to a state. We note that if p is applicable to the state s, then the immediate result of applying p to s is either a final state or else a state to which p is again applicable. We always assume that the numbers m and n in transfer commands are numbers of commands in the program, and that the last command is **STOP**.

By a *protocol* for an address program p we mean a sequence of states $s^0, s^1, \ldots, s^k$ such that each state after s^0 is the immediate result of applying p to the preceding state, and the last state is a final state. We call s^0 the *initial* state of the protocol. There exists at most one protocol for a given address program and a given initial state; there might be no protocol, if either p is not applicable to s^0 or there is no final state in the sequence of states obtained by successively applying p.

Suppose that p is an address program and k is a natural number. We consider the function f from $\mathbb{N}^k$ to $\mathbb{N}$ which is defined as follows: the value of f at the k-tuple $\langle a_1, \ldots, a_k \rangle$ is b if there exists a protocol for p whose initial state is $(1, a_1, a_2, \ldots, a_k, 0, 0, \ldots)$ and if b is the number in the 0th register at the final state of this protocol. In other words, the value of f at $\langle a_1, \ldots, a_k \rangle$ is the number in R (0) after the execution of the program if $a_1, \ldots, a_k$ were in R (0), $\ldots$, R $(k - 1)$ and zeros were in all the other registers at the beginning of the execution of the program, and the program begins with command number 1. We then call f a function which is *k-computable* by the program p (or simply *computable* by p if the value of k is clear from the context).

Example 2. Let p be the address program in Example 1. The following function f_1 is 1-computable by this program: f_1 (0) is not defined, $f_1(i) = (i - 1)!$ for $i \geqslant 1$. The function f_2 which is 2-computable by p is as follows: $f_2(i, j)$ is not defined if $i = 0$, $f_2(i, j) = (i - 1)!$ if $i \geqslant 1$.

Functions which are k-computable by address programs will be called *address-computable* functions of k variables. It is obvious that all address-computable functions are computable. And all known computable functions turn out to be address-computable. Thus, it is reasonable to conjecture that the class of computable functions from $\mathbb{N}^k$ to $\mathbb{N}$ and the class of address-computable functions are the same. Assuming this conjecture to be true, we shall now proceed to prove the arithmeticity axiom on that basis.

C.2. The Extended Language of Arithmetic. As an aid in proving arithmeticity of address-computable functions we shall develop an extended language of arithmetic. In order to define this language, we have to introduce some modifications to the definition of the language of arithmetic in Sec. 4.

We add two new symbols to the alphabet of arithmetic: v (for forming one-place functional variables) and w (for forming two-place functional variables). A word of the form (v^n) will be called a *one-place functional variable*, and a word of the form (w^n) will be called a *two-place functional variable*. (Here $n \geqslant 1$.) We shall use the notation v_n and w_n to abbreviate these one- and two-place functional variables. The interpretation we have in mind for one- and two-place functional variables is everywhere defined functions of one and two variables, where the variables and the functions take on natural number values. We shall call our old variables x_n *numerical* variables.

We define a *term* in the extended language of arithmetic as follows:

1° a numerical variable is a term;

2° if t and u are terms, then $(t + u)$ and $(t \cdot u)$ are terms;

3° if p is a one-place functional variable and t is a term, then $p\,(t)$ is a term;

4° if r is a two-place functional variable and t and u are terms, then $r\,(t,\, u)$ is a term.

Example 1. The words $(v_4\,(x_1) + x_2)$, $v_4\,(v_5\,(w_2\,(x_1,\, x_2)))$, and $w_2\,(w_2\,(x_1,\, x_4),\, x_7)$ are terms in the extended language of arithmetic.

As before, an *elementary formula* is two terms joined by an equals sign (except that now the terms are in the extended language). The *formulas* of the extended language of arithmetic are defined in the same way as in Sec. 4, except that in 4° the variable ξ can be either a numerical variable, a one-place functional variable, or a two-place functional variable.

Example 2. The words $\forall v_1\,(v_1\,(x_1) = v_1\,(x_2))$, $\forall x_1\,\forall x_2\,(v_1\,(x_1) = v_1\,(x_2))$, and $\forall w_1 \forall x_1 \forall x_2\,(w_1\,(x_1,\, x_1) = w_1\,(x_2,\, x_1))$ are formulas.

The *parameters* of terms and formulas are defined as in Sec. 4; parameters can include functional as well as numerical variables.

Example 3. The parameters in Example 1 are v_4, x_1 and x_2 for the first term; v_4, v_5, w_2, x_1 and x_2 for the second term; and w_2, x_1, x_4 and x_7 for the third term. The parameters in Example 2 are x_1 and x_2 for the first formula; v_1 for the second formula; and the third formula has no parameters.

Formulas which have no parameters are called *closed formulas* of the extended language of arithmetic.

We must now determine which are the true closed formulas of the extended language of arithmetic. We will have two different definitions of truth for the formulas of the extended language of arithmetic, i.e., two *interpretations* of this extended language. In one definition the functional variables can be any (everywhere defined) functions in one or two variables with the variables taking on natural number values, and in the other definition they can only be finite functions, i.e., functions which are nonzero only at finitely many values of the argument. In order to avoid having to give essentially the same definition twice, we shall refer to the class of *admissible* functions, by which we shall mean either the class of all functions or the class of all finite functions.

The definition of truth will be similar to that in Sec. 4. The new ingredient will be the case of a formula which begins with a quantifier with respect to a functional variable. Here we encounter the following problem: we would like to say, for example, that the formula $\forall v\alpha$ is true if, for all admissible values of v, the formula obtained from α by substituting the values in place of v is true. However, our language is not equipped with anything that can be substituted in place of a functional variable. This dilemma can be resolved as follows: we must introduce functional constants into our language, one for each admissible function.

We now give the precise definitions. We choose a set of symbols which is in one-to-one correspondence with the set of admissible functions. We shall call the symbols in this set *functional constants* which describe the corresponding functions. Each one is either a one-place or a two-place functional constant, depending on the number of variables in the corresponding function. We shall substitute functional constants in place of functional variables having the same number of arguments. By an *evaluated term* (or *evaluated formula*) we shall mean the result of substituting any functional constants in place of all of the functional parameters and any numbers in place of all of the numerical parameters in some term (or formula) of the extended language of arithmetic. This substitution is carried out in the same way as in Sec. 4, i.e., only occurrences of a variable which are outside of the range of action of quantifiers are replaced. A special case of an evaluated term is a constant term, i.e., one with no parameters (such a term is then also a term of the usual language of arithmetic). A special case of an evaluated formula is a closed formula of the extended language of arithmetic.

We can now define the *value* of an evaluated term or formula in a way which is completely analogous to the corresponding definitions for constant terms and closed formulas in the usual language of arithmetic. The values of evaluated terms are defined as follows:

1° the value of $(|^{n})$ is the number n;

2° the value of an evaluated term of the form $(t+u)$ is the sum of the values of the evaluated terms t and u, and the value of an evaluated term of the form $(t \cdot u)$ is the product of the values of the evaluated terms t and u;

3° the value of an evaluated term of the form $\gamma(t)$, where γ is a one-place functional constant and t is an evaluated term, is the value of the function described by γ at the number which is the value of the evaluated terms t;

4° the value of an evaluated term of the form $\delta(t, u)$, where δ is a two-place functional constant and t and u are evaluated terms, is the value of the function from $\mathbb{N}^2$ to $\mathbb{N}$ described by δ at the pair $\langle$value of t, value of $u\rangle$.

To define the value of an evaluated formula, we can now use the definition in Sec. 4 of the value of a closed formula in the language of arithmetic, replacing the words "closed formula" by "evaluated formula" and "constant term" by "evaluated term," specifying in 7° and 8° that ξ is a numerical variable, and inserting the following two paragraphs:

9° the evaluated formula $\exists\xi\alpha$, where ξ is a functional variable, is true if there exists a functional constant γ with the same number of arguments as ξ and having the property that the evaluated formula $S^{\xi}_{\gamma}\alpha$ is true; if no such functional constant exists, then the evaluated formula $\exists\xi\alpha$ is false;

10° the evaluated formula $\forall\xi\alpha$, where ξ is a functional variable, is true if, for every functional constant γ with the same number of arguments as ξ, the evaluated formula $S^{\xi}_{\gamma}\alpha$ is true; otherwise the evaluated formula $\forall\xi\alpha$ is false.

Now that we have defined the value of an evaluated formula, we know how to determine the value of a closed formula of the extended language of arithmetic as a special case. This special case will be especially important for us later on.

Example 4. The closed formula

$$\forall x_1 \forall x_2 \; (\forall v_1 \; (v_1 \; (x_1) = v_1 \; (x_2)) \rightarrow (x_1 = x_2)),$$

which says "if the values of all admissible functions at x_1 and x_2 are the same, then $x_1 = x_2$, "is true for either of the two interpretations of the set of admissible functions (all functions or only finite functions).

Example 5. The closed formula

$$\forall v_1 \exists x_1 \forall x_2 \; (v_1 \; ((x_1 + x_2)) = 0),$$

which says that "every admissible function is equal to 0 for all sufficiently large values of the argument," is true if admissible functions are the finite functions, and it is false if the class of admissible functions consists of all functions.

Example 6. The closed formula

$$\forall w_1 \exists v_1 \forall x_1 \; (v_1 \; (x_1) = w_1 \; (x_1, \; x_1))$$

is true for either interpretation of admissible functions.

Example 7. The following closed formula says that there exists an admissible one-to-one correspondence between $\mathbb{N}^2$ and $\mathbb{N}$:

$$\begin{aligned} \exists w_1 \; (\forall x_1 \exists x_2 \exists x_3 \; (w_1 \; (x_2, \; x_3) = x_1) \\ \wedge \forall x_2 \forall x_3 \forall x_4 \forall x_5 \; ((w_1 \; (x_2, \; x_3) \\ = w_1 \; (x_4, \; x_5)) \rightarrow ((x_2 = x_4) \wedge (x_3 = x_5)))). \end{aligned}$$

It is true if we take all functions to be admissible, and it is false if we take only the finite functions.

Example 8. The statement that "the inequality $2^{x_1} \geqslant x_1$ holds for all x_1" can be translated into the following formula in the extended language of arithmetic (with either of the two interpretations of admissible functions):

$$\begin{aligned} \forall x_1 \forall v_1 (((v_1 \; (0) \\ = 1) \wedge \forall x_2 \; ([x_2 \leqslant x_1] \rightarrow (v_1 \; ((x_2 + 1)) \\ = (2 \cdot v_1 \; (x_2))))) \rightarrow [x_1 \leqslant v_1 \; (x_1)]). \end{aligned}$$

Here $[x_2 \leqslant x_1]$ denotes the formula $\exists x_3((x_2 + x_3) = x_1)$, and $[x_1 \leqslant v_1(x_1)]$ denotes the formula $\exists x_3((x_1 + x_3) = v_1(x_1))$. This closed formula can be read as follows: "if v_1 is the sequence of natural numbers whose first term is 1 and each of whose successive terms through the $(x_1 + 1)$th term is twice the previous one, then $v_1(x_1) \geqslant x_1$." The stipulation "through the $(x_1 + 1)$th term" is necessary if only finite functions are admissible.

In Appendix B we interpreted formulas in the language of arithmetic which have parameters as expressing properties of the natural numbers. In the same way we may regard formulas in the extended language of arithmetic as expressing properties of natural numbers and functions.

Example 9. The formula

$$\exists x_3((v_1(x_2) + x_3) = v_1(x_1))$$

expresses the following property: the value of the admissible function v_1 at the number x_1 is not less than its value at the number x_2. This last statement is not phrased completely correctly, since v_1 is a functional variable and not a function, and x_1 and x_2 are numerical variables and not numbers. The phrasing is an abbreviated way of saying, "The result of substituting numbers n_1 and n_2 in place of x_1 and x_2 and substituting a functional constant describing an admissible function in place of v_1 is a true evaluated formula of the extended language of arithmetic (with either of the two interpretations of admissible functions) if and only if the value of this admissible function at n_1 is no less than its value at n_2."

Example 10. The formula

$$\forall x_1 \forall x_2 \exists x_3((v_1(x_1) + x_3) = v_1((x_1 + x_2)))$$

expresses the property that "the admissible function v_1 is a nondecreasing function."

Even if we only look at formulas in the extended language of arithmetic which have no functional parameters, we still have new possibilities that were absent in the earlier language of arithmetic.

Example 11. The formula

$$\exists v_1(((v_1(0) = 1) \wedge \forall x_3([x_3 \leqslant x_1] \\ \rightarrow (v_1((x_3 + 1)) = (2 \cdot v_1(x_3))))) \wedge (v_1(x_1) = x_2)),$$

where $[x_3 \leqslant x_1]$ is, of course, an abbreviation for $\exists x_2((x_3 + x_2) = x_1)$, expresses the property mentioned at the beginning of Appendix C: "$x_2 = 2^{x_1}$." This holds for either of the two interpretations of admissible functions; if all functions are admissible, then we do not need the stipulation $[x_3 \leqslant x_1]$.

The properties of natural numbers which are expressible by formulas in the extended language of arithmetic will be called *analytic* properties or *weakly analytic* properties, depending on whether we are taking the admissible functions to be all functions or only the finite functions. We shall identify properties with the sets of objects having the properties, and so shall also speak of analytic and weakly analytic sets.

More precisely, let α be a formula in the extended language of arithmetic not having any functional parameters and not having any numerical parameters other than $x_1, \ldots, x_k$. Let $n_1, \ldots, n_k$ be a set of k numbers. If we substitute these numbers in place of the respective variables $x_1, \ldots, x_k$, we obtain a closed formula in the extended language of arithmetic. We shall say that the set of k-tuples $\langle n_1, \ldots, n_k \rangle$ for which this closed formula is true is *associated with* the formula α (if we are taking all functions to be admissible) or *weakly associated with* α (if only the finite functions are admissible). A set which is associated (or weakly associated) with a formula in the extended language of arithmetic will be called an *analytic* set (respectively, a *weakly analytic* set).

Example 12. As Example 11 shows, the set $\{\langle x_1, x_2 \rangle \mid x_2 = 2^{x_1}\}$ is both analytic and weakly analytic.

Any arithmetic set is obviously both analytic and weakly analytic. We shall later prove that all weakly analytic sets are arithmetic. But not all analytic sets are arithmetic. It can be shown that the set of numbers of the true statements in the language of arithmetic (in any numbering of A) is analytic; but this set is not arithmetic, by Tarski's theorem (see Appendix B). We note in passing that an argument similar to the proof of Tarski's theorem can be used to show that the set of all closed formulas in the extended language of arithmetic which are true when one takes all functions to be admissible, is not an analytic set.

In the next subsection we prove that the graph of any address-computable function is a weakly analytic set. When combined with the result mentioned above about arithmeticity of any weakly analytic set, this will allow us to conclude that any address-computable function is arithmetic.

C.3. Expressibility of Address-Computable Functions in the Extended Language of Arithmetic. In this subsection we shall prove that the graph of any address-computable function is a weakly analytic set. Later (in Subsecs. C.4-C.6) we shall prove that any weakly analytic set is arithmetic; this will then complete the proof that address-computable functions are arithmetic, and so sets which are enumerated by such functions are also arithmetic.

Let p be an address program. We shall prove that certain properties connected with the program p are expressible in the extended language of arithmetic. In what follows, when we refer to the truth of evaluated formulas in the extended language of arithmetic, we shall always be interpreting admissible functions to be only the finite functions.

Recall that an address machine state is a sequence of natural numbers all but finitely many of which are zero. Such state is nothing more nor less than a finite function.

Lemma C.1. *The property "the state v_2 is the immediate result of applying the program p to the state v_1" is expressible in the extended language of arithmetic. (This means that there exists a formula α in the extended language of arithmetic with the one-place functional variables v_1 and v_2 as its parameters such that the result of substituting two functional constants describing finite functions in place of v_1 and v_2 is true evaluated formula if and only if the state described by the second functional constant is the immediate result of applying the program p to the state described by the first functional constant.)*

▶ Given a program p, we shall describe how to construct the required formula. (Of course, the formula itself will depend on p.) Our formula α will have the form $\alpha_1 \wedge \quad . \quad \wedge\alpha_n$ (where it makes no difference how the parentheses, which we have omitted, are inserted). Here n is the number of commands in the program, and the formula α_i corresponds to the ith command. Each α_i is one of seven types of formulas, depending on which of the seven types of commands for address machines is the ith command in the program. The formulas are constructed by following along the seven parts of the definition of the immediate result of applying an address program. We shall explain how this is done using two examples.

Example 1. Suppose that command 37 has the form

$$37 \ R\,(16) \leftarrow R\,(2) \cdot R\,(16).$$

In this case the formula α_{37} is as follows:

$$(v_1(0) = 37) \rightarrow (\forall x_1 (\neg(x_1 = 16) \rightarrow (v_2((x_1 + 1)) = v_1((x_1 + 1)))) \wedge (v_2(17) = (v_1(17) \cdot v_1(3)))).$$

Example 2. Suppose that command 81 has the form

81 **IF** $R(3) = R(4)$ **GO TO** 7 **ELSE GO TO** 23.

In this case the formula α_{81} is as follows:

$$(v_1(0) = 81) \rightarrow (\forall x_1 (v_2(x_1 + 1) = v_1(x_1 + 1)) \wedge (((v_1(4) = v_1(5)) \rightarrow (v_2(0) = 7)) \wedge (\neg v_1(4) = v_1(5)) \rightarrow (v_2(0) = 23)))).$$

This completes the proof of the lemma. ■

Our next step is to construct a formula which expresses the property of "being a protocol for applying an address program p of length $x_1 + 1$." A protocol is a sequence of states, i.e., a sequence of finite functions.

Our language does not have sequences of functions as such, but it does have objects which amount to the same thing. Namely, we shall identify a sequence of functions $s^0, s^1, \ldots$ of one variable with the function $S(n, m)$ of two variables, which is defined by setting $S(n, m) = s^n(m)$. Thus, we shall say that an everywhere defined function S from $\mathbb{N}^2$ to $\mathbb{N}$ *describes a protocol* for a program p of length $k + 1$ if the sequence $s^0, \ldots, s^k$ of functions of one variable which are defined by setting $s^i(x) = S(i, x)$ is a protocol for applying the address program p.

Lemma C.2. *There exists a formula β which has a two-place functional variable w_1, two one-place functional variables v_1 and v_2, and a single numerical variable x_1 as its parameters and which expresses the following property:*

"w_1 describes a protocol of length $x_1 + 1$,
v_1 is the initial state of this protocol, and
v_2 is the final state of this protocol."

▶ The desired formula has the following form:

$$(([v_1 = w_1^0] \wedge [v_2 = w_1^{x_1}]) \wedge (v_2(0) = 0))$$
$$\wedge \forall x_2 \forall v_3 \forall v_4 (([x_2 + 1 \leqslant x_1] \wedge ([v_3 = w_1^{x_2}] \wedge [v_4 = w_1^{x_2+1}]))$$
$$\rightarrow [v_4 \text{ is the immediate result of applying } p \text{ to } v_3]).$$

Here $[v_1 = w_1^0]$ is an abbreviation for the formula $\forall x_3 (v_1(x_3) = w_1(0, x_3))$; $[v_4 = w_1^{x_2+1}]$ is an abbreviation for $\forall x_3\ (v_4(x_3) = w_1\ ((x_2+1),\ x_3))$, and $[v_2 = w_1^{x_1}]$ and $[v_3 = w_1^{x_2}]$ are to be interpreted analogously; and, finally, $[v_4$ is the immediate result of applying p to $v_3]$ denotes the formula in Lemma C.1 with the variables v_1 and v_2 replaced by v_3 and v_4, respectively. ■

We are now ready to prove that the graph of an address-computable function is weakly analytic.

Theorem C.1. *The graph of an address-computable function from* $\mathbb{N}^k$ *to* $\mathbb{N}$ *is a weakly analytic set.*

▶ Let f be an address-computable function from $\mathbb{N}^k$ to $\mathbb{N}$ and let p be an address program which computes f. The graph of f consists of the $(k+1)$-tuples $\langle x_1, \ldots, x_k, x_{k+1}\rangle$ for which there exists a protocol w_1 of some length with initial state v_1 and final state v_2 such that

$$v_1(0) = 1,\ v_1(1) = x_1, \quad \ldots \quad v_1(k) = x_k,$$
$$v_1(x) = 0 \text{ for } x \geqslant k+1,\ v_2(1) = x_{k+1}.$$

If we write all this as a formula in the extended language of arithmetic, we obtain the desired formula—a formula with which the graph of f is weakly associated. The theorem is proved. ■

In the next subsections, C.4-C.6, we prove that any weakly analytic set—and hence the graph of any address-computable function—is an arithmetic set.

C.4. Reducing the Extended Language of Arithmetic to the Usual Language of Arithmetic. In this subsection we shall prove that any weakly analytic set is arithmetic, i.e., that adding to the language of arithmetic variables which run through all the *finite* functions of one or two natural variables does not increase the expressive possibilities of the language. As noted above, the condition that the functions be finite is essential: adding variables which run through *all* functions does result in a significantly more expressive language.

We first give a rough explanation of why the addition of variables denoting finite functions does not have any essential effect. The point is that these functions form a countable set, they can be labeled by the natural numbers (and this labeling turns out to be arithmetic in a sense that will be made precise below), and then we can speak of a function's label rather than the function itself. In this way we can limit ourselves to working with the natural numbers.

We now make this more precise. Suppose that ν is a map which associates an everywhere defined finite function of one variable to every element of $\mathbb{N}^k$, i.e., to every k-tuple of natural numbers. We call such a map a *labeling of the finite functions of one variable by means of elements of* $\mathbb{N}^k$ if each finite function corresponds to at least one (and perhaps more) elements of $\mathbb{N}^k$. If a k-tuple $\langle a_1, \ldots, a_k\rangle$ corresponds to a function s, then we call this k-tuple a *label* for s (in the particular labeling system). We say that a labeling is *arithmetic* if the set

$$\{\langle a_1, \ldots, a_k, x, y\rangle \mid \text{the value of the finite function with label } \langle a_1, \ldots, a_k\rangle \text{ at the number } x \text{ is equal to } y\}$$

is an arithmetic set.

The key step in our proof that weakly analytic sets are arithmetic is the following claim:

(∗) *for some k there exists an arithmetic labeling of the finite functions by means of elements of* $\mathbb{N}^k$.

In Subsecs. C.5-C.6 we shall give two different proofs of this claim. In the remainder of Subsec. C.4 we shall show how the claim implies arithmeticity of weakly analytic sets.

We could give a definition of an arithmetic labeling of everywhere defined finite functions of two variables by analogy with the above definition of an arithmetic labeling of finite functions of one variable. It turns out that the existence of such a labeling follows from the existence of an arithmetic labeling for finite functions of one variable. Namely, if we want to label a function f from $\mathbb{N}^2$ to $\mathbb{N}$, we first label its "cross-sections," i.e., the functions $f_n(x) = f(n, x)$, and we then label the sequence made up from the labels of the cross-sections. More precisely, we have the following easily proved lemma:

Lemma C.3. *If ν is an arithmetic labeling of the finite functions of one variable by means of elements of $\mathbb{N}^k$, where $\langle 0, 0, \ldots, 0\rangle$ is the label of the zero finite function, then the following function μ is an arithmetic labeling of the finite functions of two variables by means of elements of $\mathbb{N}^k$. μ associates the k^2-tuple $\langle a_1^1, \ldots, a_k^1, \ldots, a_1^k, \ldots, a_k^k\rangle$ to the function from $\mathbb{N}^2$ to $\mathbb{N}$ whose value at the pair $\langle p, q\rangle$ is*

$$\nu\,(\nu\,(a_1^1, \ldots, a_k^1)\,(p), \ldots, \nu\,(a_1^k, \ldots, a_k^k)\,(p))\,(q).$$

It is easy to see that the restriction that $\langle 0, 0, \ldots, 0\rangle$ be the label of the zero function is not essential. Any arithmetic labeling can easily be changed so as to have this property, by simply interchanging two labels; the resulting labeling is still arithmetic.

Thus, we suppose that we have some fixed arithmetic labeling ν of the finite functions of one variable by means of elements of $\mathbb{N}^k$, and also some fixed arithmetic labeling μ of the finite functions of two variables by means of elements of $\mathbb{N}^h$.

For each formula in the extended language of arithmetic, we shall construct its "translation," which is a formula in the language of arithmetic which says the same thing as the original formula, but about the labels of functions rather than the functions themselves. For convenience, we shall add to the language of arithmetic some new variables for numbers: k new variables V_i^1, ., V_i^k for each one-place functional variable v_i in the extended language of arithmetic, and h new variables W_i^1, ., W_i^h for each two-place functional variable w_i. Clearly, the class of arithmetic sets is not affected by this addition—it makes no difference what names one has for variables! We now give a precise definition of the translation of a formula.

Suppose that α is a formula in the extended language of arithmetic having numerical parameters x_{p_1}, x_{p_m}, one-place functional parameters v_{q_1}, ., v_{q_n}, and two-place functional parameters w_{r_1}, w_{r_s}. Let β be a formula of the (usual) language of arithmetic whose parameters are a subset of $\{x_{p_1}$, ., x_{p_m}, $V_{q_1}^1$, ., $V_{q_1}^k$, ., $V_{q_n}^1$, ., $V_{q_n}^k$, $W_{r_1}^1$, ., $W_{r_1}^h$, ., $W_{r_s}^1$, $W_{r_s}^h\}$. Then β is called a *translation* of α if, for any natural numbers x_{p_1}, ., x_{p_m}, $V_{q_1}^1$., $V_{q_n}^k$, $W_{r_1}^1$, ., $W_{r_s}^h$, the result of substituting these numbers in place of the corresponding variables in β is a true statement in the language of arithmetic if and only if the result of substituting x_{p_1}, x_{p_m}, $\nu(V_{q_1}^1$, $V_{q_1}^k)$, $\nu(V_{q_n}^1$, $V_{q_n}^k)$, $\mu(W_{r_1}^1$, $W_{r_1}^h)$, ., $\mu(W_{r_s}^1$, $W_{r_s}^h)$ (more precisely, numerical and functional constants which describe these numbers and functions) in place of x_{p_1}, ., x_{p_m}, v_{q_1}, ., v_{q_n}, w_{r_1}, w_{r_s} in the formula α is a true evaluated formula of the extended language of arithmetic.

Theorem C.2. *Every formula in the extended language of arithmetic has a translation.*

Before proving this theorem, we remark that a special case of the theorem—the case of formulas having no functional parameters—obviously implies our claim that weakly analytic sets are arithmetic: if a set is weakly associated with a formula in the extended language of arithmetic, then the same set is associated with the translation of this formula.

▶ Assuming for now that translations have been constructed for the elementary formulas in the extended language of arithmetic, we shall show how to build up translations of the other formulas.

Lemma C.4. 1° *If* β *is a translation of* α, *then* $\neg\beta$ *is a translation of* $\neg\alpha$;

2° *if* β_1 *and* β_2 *are translations of* α_1 *and* α_2, *then*

$$(\beta_1 \wedge \beta_2),\ (\beta_1 \vee \beta_2),\ (\beta_1 \rightarrow \beta_2),\ (\beta_1 \leftrightarrow \beta_2)$$

are translations, respectively, of the formulas

$$(\alpha_1 \wedge \alpha_2),\ (\alpha_1 \vee \alpha_2),\ (\alpha_1 \rightarrow \alpha_2),\ (\alpha_1 \leftrightarrow \alpha_2);$$

3° *if* β *is a translation of* α *and* Q *is one of the symbols* $\forall$ or $\exists$, *then*:

$Qx_i\beta$ *is a translation of* $Qx_i\alpha$,

QV^1_i. $QV^h_i\beta$ *is a translation of* $Qv_i\alpha$, *and*

QW^1_i. $QW^h_i\beta$ *is a translation of* $Qw_i\alpha$.

This lemma follows immediately from the definition of a translation and the definition of the truth value of formulas. Because of the lemma, our task is reduced to translating the elementary formulas, i.e., formulas of the form $(t = u)$, where t and u are terms in the extended language of arithmetic. If we replace this formula by $\exists\xi\,((t = \xi) \wedge (u = \xi))$, where ξ is a numerical variable which does not appear in either t or u, and use parts 2° and 3° of the lemma, we see that it suffices to translate formulas of the form $(t = \xi)$, where t is a term and ξ is a numerical variable. We prove that such a translation can be constructed using induction on the number of steps in the construction of the term t:

1° if t is a variable or number, then the formula is its own translation;

2° if t is $(u_1 + u_2)$, then we replace the formula $((u_1 + u_2) = \xi)$, by

$$\exists\eta_1\,\exists\eta_2\,(((u_1 = \eta_1) \wedge (u_2 = \eta_2)) \wedge (\xi = (\eta_1 + \eta_2))),$$

where η_1 and η_2 are numerical variables different from ξ and not occurring in t, and then use the induction assumption and Lemma C.4;

3° the case when t is of the form $(u_1 \cdot u_2)$ is analogous to 2°;

4° if t is $p(u)$, where u is a term and p is a one-place functional variable, then we first replace the formula $(p(u) = \xi)$ by $\exists\eta((u = \eta) \wedge (p(\eta) = \xi))$, where η is a numerical variable different from ξ and not occurring in u; hence, it suffices to be able to translate a formula of the form $(p(\eta) = \xi)$, and this is possible because of our assumption that an arithmetic labeling exists;

5° the case when t is of the form $r(u_1, u_2)$, where u_1 and u_2 are terms and r is a two-place functional variable, is analogous to 4°. ■

Example 1. A translation of the formula

$$\forall x_1 (v_1(x_1) = v_2(x_1))$$

is:

$\forall x_1 \exists x_2$ ([the value of the one-place function with label $\langle V_1^1, \ldots, V_1^k \rangle$ at the number x_1 is x_2] $\wedge$ [the value of the one-place function with label $\langle V_2^1, \ldots, V_2^k \rangle$ at the number x_1 is x_2]),

where the bracketed notation denotes the formulas in the language of arithmetic which express the properties described there. These formulas exist because there exists an arithmetic labeling.

Thus, in order to complete the proof that weakly analytic sets are arithmetic, it remains only to construct an arithmetic labeling of the finite functions of one variable.

C.5. First Method of Constructing an Arithmetic Labeling—Gödel's Method. We begin with the following observation: it is enough to prove that there exists an everywhere defined arithmetic function $\beta(x_1, \ldots, x_i, y)$ with the following property:

(∗) *for every finite sequence of natural numbers* $n_0, \ldots, n_k$ *there exist* $a_1, \ldots, a_i$ *such that* $\beta(a_1, \ldots, a_i, 0) = n_0$, $\beta(a_1, \ldots, a_i, 1) = n_1, \ldots, \beta(a_1, \ldots, a_i, k) = n_k$. (Here the values of $\beta(a_1, \ldots, a_i, y)$ for $y > k$ can be arbitrary.) Suppose that β has this property. Let ν be the map which to every $(i+1)$-tuple $\langle x_1, \ldots, x_i, h \rangle$ associates the finite function $s(y)$, which equals $\beta(x_1, \ldots, x_i, y)$ for $y \leqslant h$ and equals 0 for $y > h$. Then ν is an arithmetic labeling of the finite functions of one variable by means of elements of $\mathbb{N}^{i+1}$: it is a labeling because of (*), and it is arithmetic because β is an arithmetic function and the condition $y \leqslant h$ is an arithmetic property.

Thus, it suffices to construct a function of $i+1$ variables satisfying (*) for some natural number i. Following Gödel, we take our function to be

$$\beta(x_1, x_2, y) = (\text{the remainder when } x_1 \text{ is divided by } x_2(y+1)+1)$$

The arithmeticity of this function follows from the arithmeticity of the property "x_3 is the remainder when x_1 is divided by x_2," which is expressed by the formula $[x_3 < x_2] \wedge \exists x_4(x_1 = ((x_2 \cdot x_4) + x_3))$, where $[x_3 < x_2]$ denotes the formula $\exists x_5\,(((x_3 + x_5) + 1) = x_2)$. To prove (*), we must use some simple facts from number theory, whose proofs can be found in most introductory textbooks on the subject. For the rest of this subsection we shall say "number" when we mean "natural number."

A number a is called a divisor of a number b if $a = bc$ for some c. If a is a divisor of b and c, then it is a divisor of $b + c$ and of $b - c$. A prime number is a number $p > 1$ which has no divisors other than 1 and p. Every number can be written as a product of prime factors, and in essentially only one way, i.e., two factorizations can differ only in the order one writes the prime factors. If a product of several numbers is divisible by a prime p, then at least one of the factors is divisible by p. Two numbers a and b are said to be relatively prime if they have no divisors in common except for 1. The numbers a and b are relatively prime if and only if their factorizations into a product of primes have no prime factors in common. If $a_1, \ldots, a_n$ are pairwise relatively prime, and if b is divisible by each a_i, then b is divisible by $a_1 \cdot a_2 \cdot \ldots \cdot a_n$.

Suppose that $a_0, \ldots, a_k$ are pairwise relatively prime. We now consider the question: what $(k+1)$-tuples $\langle r_0, \ldots, r_k \rangle$ of remainders are possible when a number x is divided by $a_0, \ldots, a_k$? The remainder when x is divided by a_i is one of the numbers $0, 1, \ldots, a_i - 1$; thus, there are $a_0 \cdot a_1 \cdot \ldots \cdot a_k$ possible $(k+1)$-tuples of remainders. The next lemma says that all these possibilities actually occur.

Lemma C.5 (the "Chinese remainder theorem"). *Suppose that $a_0, \ldots, a_k$ are pairwise relatively prime, and $r_0, \ldots, r_k$*

satisfy $r_i < a_i$ for all i. Then there exists a number x having remainder r_i when divided by a_i, for each i.

▶ We shall call two numbers equivalent if they give the same remainders when divided by each of the a_i. If two numbers are equivalent, then their difference is divisible by each a_i, and hence by $a_0 \cdot \ldots \cdot a_k$ (here we use the relative primality of the a_i). Thus, no two of the numbers $0, 1, \ldots, a_0 \cdot \ldots \cdot a_k - 1$ are equivalent, i.e., each of these numbers has a different $(k + 1)$-tuple of remainders. But there are exactly as many of these numbers as there are possible $(k + 1)$-tuples of remainders. Thus, any $(k + 1)$-tuple $\langle r_0, \ldots, r_k \rangle$ for which $r_i < a_i$ for each i is a $(k + 1)$-tuple of remainders obtained by dividing some x by $a_0, \ldots, a_k$. ■

Lemma C.6. *For any n we can find a number b such that $b + 1, 2b + 1, \ldots, nb + 1$ are pairwise relatively prime. The number b can be chosen to be larger than any number specified in advance.*

▶ We first note that if p is a common prime divisor of $kb + 1$ and $hb + 1$, then p is a divisor of their difference $(k - h)\, b$. But p cannot divide b, since if it did we would obtain a remainder of 1 when $kb + 1$ or $hb + 1$ is divided by p. Hence $k - h$ is divisible by p. From this it follows that $b + 1, \ldots, nb + 1$ will be relatively prime if they have no common divisors less than n. This can be accomplished, for example, by taking b to be a multiple of $1 \cdot 2 \cdot \ldots \cdot n$; then the numbers $b + 1, \ldots, nb + 1$ will each give a remainder of 1 when divided by any number from 2 to n. This completes the proof of the lemma. ■

We can now easily prove the property (*) for our function β. Namely, suppose that $n_0, \ldots, n_k$ are arbitrary natural numbers. We have to find x_1 and x_2 such that the remainder when x_1 is divided by $x_2 (i + 1) + 1$ is n_i for $i \leqslant k$. According to Lemma C.6, we can find x_2 such that the numbers $x_2 + 1, \ldots, x_2 (k + 1) + 1$ are pairwise relatively prime and x_2 is greater than any of the numbers $n_0, \ldots n_k$. It finally remains to choose x_1 using Lemma C.5.

This completes the construction of an arithmetic labeling by Gödel's method. In the next subsection we shall look at another method of constructing an arithmetic labeling, without the use of number-theoretic considerations. This second method is due to R. M. Smullyan (see his book *Theory of Formal Systems*, Princeton, 1961).

C.6. Second Method of Constructing an Arithmetic Labeling —Smullyan's Method. We first introduce the notion of an arithmetic labeling of finite subsets of $\mathbb{N}$ by natural numbers, which is analogous to the notion of a labeling of finite functions by means of elements of $\mathbb{N}^1$. Namely, a function τ which associates a subset of $\mathbb{N}$ to every natural number is called a *labeling* if every finite subset of $\mathbb{N}$ is the value of τ at some natural number. If $\tau(y) = A$, we shall call y the *label* of the set A (relative to the labeling τ). A labeling is said to be *arithmetic* if the set

$$\{\langle x, y\rangle \mid x \in \tau(y)\}$$

is an arithmetic subset of $\mathbb{N}^2$.

Note that we do not require that all of the subsets corresponding to natural numbers be finite. (Here the analogy with the definition of a labeling of finite functions breaks down.)

We shall later show that arithmetic labelings of the finite subsets of $\mathbb{N}$ by natural numbers exist. But before proving this, we shall show how this fact implies the existence of an arithmetic labeling of the finite functions of one variable. Thus, suppose that we have an arithmetic labeling τ of the finite subsets of $\mathbb{N}$ by natural numbers.

Lemma C.7. *There exists an arithmetic function from $\mathbb{N}^2$ to $\mathbb{N}$ which is defined on all of $\mathbb{N}^2$ and takes distinct elements of $\mathbb{N}^2$ to distinct elements of $\mathbb{N}$.*

▶ We take the function which takes a pair $\langle n_1, n_2\rangle$ to the number which is the smallest label of the set $\{n_1, n_1 + n_2\}$. In other words, the value of this function at $\langle n_1, n_2\rangle$ is equal to k if and only if $n_1 \in \tau(k)$, $n_1 + n_2 \in \tau(k)$, any number belonging to $\tau(k)$ is equal to either n_1 or $n_1 + n_2$, and any number less than k does not have this property. All of this can be written as a formula in the language of arithmetic, and so our function is arithmetic. That this function takes distinct pairs to distinct numbers follows easily from the definition. ■

We now construct an arithmetic labeling of the finite subsets of $\mathbb{N}^2$ by natural numbers. Namely, we let the number k correspond to the subset of $\mathbb{N}^2$ consisting of pairs $\langle x, y\rangle$ for which $v(x, y)$ belongs to the set $\tau(k)$, where v is the function in Lemma C.7 and τ is an arithmetic labeling of the finite subsets of $\mathbb{N}$ by natural numbers. It is easy to see that every finite subset of $\mathbb{N}^2$ corresponds to some number, and that the set $\{\langle k, x, y\rangle \mid$ the pair $\langle x, y\rangle$ is in the subset of $\mathbb{N}^2$ corresponding to the number $k\}$

is an arithmetic subset of $\mathbb{N}^3$. (Of course, these two requirements are what we had in mind when we spoke of constructing an arithmetic labeling of the subsets of $\mathbb{N}^2$.)

We are now ready to construct an arithmetic labeling of the finite functions of one variable by means of elements of $\mathbb{N}^2$. We must describe the function s that corresponds to the pair of natural numbers $\langle k, h\rangle$. Let A be the subset of $\mathbb{N}^2$ which corresponds to k under our arithmetic labeling of the subsets of $\mathbb{N}^2$. Given a number $x \leqslant h$, if there exist y for which $\langle x, y\rangle \in A$, then $s(x)$ is equal to the smallest such y; if no such y exists, then $s(x) = 0$. Given a number $x > h$, we set $s(x) = 0$. The function corresponding to any pair $\langle k, h\rangle$ is a finite function, since it is zero for values of the argument greater than h. To find a label for a given finite function s, we first take h to be the greatest number at which s has a nonzero value, and then take k to be the label of the set $\{\langle x, y\rangle \mid x \leqslant h \text{ and } y = s(x)\}$. If we write the definition of the labeling we just constructed as a formula in the language of arithmetic, we see that the labeling is indeed arithmetic.

To complete the proof by Smullyan's method of the existence of arithmetic labelings of finite functions, it thus remains for us to construct an arithmetic labeling of the finite subsets of $\mathbb{N}$. In the construction we shall make use of the binary system for writing integers.

The binary representation of a natural number (except for 0) always begins with a 1. If we agree to drop that initial 1, then we obtain a one-to-one correspondence between the set of all positive integers and the set of all words in the alphabet $\{0, 1\}$. That is, the instruction "Take the number, add 1 to it, write the result in binary, and drop the initial 1" gives a one-to-one correspondence between the set of natural numbers and the set of words in the alphabet $\{0, 1\}$, as follows:

0	empty word
1	0
2	1
3	00
4	01
5	10
6	11
7	000
8	001

The words in the right column are in order of increasing length, with words of the same length arranged in alphabetical order. We shall call an integer in the left column the "number" of the word in the right column in the same row. In this numbering every set of (binary) words corresponds to a set of natural numbers. For example, the set of words consisting only of zeros corresponds to the set of numbers which are one less than a power of two. Using this numbering, we say that a set of words is arithmetic, meaning that the corresponding set of natural numbers is arithmetic. We shall also say that a property of words is arithmetic if the set of words satisfying the property is arithmetic. We similarly define the notion of an arithmetic subset of $(\{0, 1\}^\infty)^n$, the set consisting of all n-tuples of words, and speak of an arithmetic property of n-tuples of words.

We now show that certain concrete properties are arithmetic.

1° The word X comes before the word Y in the above ordering. In fact, this holds if and only if the number of X is less than the number of Y

2° The word X consists only of zeros. Here, by what was said above, it suffices to verify arithmeticity of the property of "being a power of two," and this follows, as explained in Appendix B, from the fact that a number x is a power of two if and only if every divisor of x is either 1 or an even number.

3° The word X consists only of ones. In fact, a word consists only of ones if and only if the succeeding word consists only of zeros.

4° The word Y consists only of zeros and has the same length as the word X. In fact, this is equivalent to requiring that the number of Y be the largest among all numbers of words consisting only of zeros which do not exceed the number of X.

5° The words X and Y have the same length. This is equivalent to the existence of a word Z consisting only of zeros and having the same length as X and the same length as Y; so arithmeticity follows using 4°.

6° The word X is the concatenation of the words Y and Z, i.e., it is obtained by writing Z after Y to the right, $X = YZ$. This is the most difficult in our list of arithmetic properties. Here the argument requires that we recall our particular method of numbering words. Roughly speaking, this property is arithmetic because the number x whose binary representation is obtained by joining the binary representations of y and z, is obtained by multiplying y by $2^{(\text{length of } z)}$ and adding z. Of course, we must take into account that our numbering system for words involves adding 1 and then dropping the first 1 in the binary representation. We now go through the argument in more detail.

Let x, y and z be the numbers of the words X, Y and Z. This means that $x + 1$ is written as $1X$ in binary, $y + 1$ as $1Y$, and $z + 1$ as $1Z$. Let u be the number of the word having the same length as Z but consisting only of zeros. Then $u + 1$ in binary is $100 \ldots 0$ (with length of Z zeros). If X is the concatenation of Y and Z, then multiplying $y + 1$ by $u + 1$ and adding $(z + 1) - (u + 1)$ gives $x + 1$. We can now conclude that our property is arithmetic, since a formula expressing the property in the language of arithmetic need only say: "There exists u, which is the number of a word of the same length as Z and consisting only of zeros, such that $(y + 1)\cdot(u + 1) + (z - u)$ is equal to $x + 1$." The clause about having the same length as Z and consisting only of zeros can be expressed in the language of arithmetic, by 4°.

The arithmeticity of the property of "being the concatenation" easily implies that several other properties are also arithmetic.

7° The word X is the beginning of the word Y, i.e., there exists a word Z such that Y is the concatenation of X and Z.

8° The word X is the end of the word Y, i.e., there exists a word Z such that Y is the concatenation of Z and X.

9° The word X is a subword of the word Y In fact, X is a subword of Y if and only if it is the beginning of an end of Y.

10° The word X is the concatenation of the words Y, Z and V. Here we note that X is the concatenation of Y, Z and V if and only if there exists a word W such that W is the concatenation of Y and Z and X is the concatenation of W and V.

We can similarly prove for any fixed n that the property "Y is the concatenation of $X_1, \ldots, X_n$" is arithmetic.

We are now ready to construct an arithmetic labeling of the finite sets of natural numbers. Suppose that x and y are natural numbers, and X and Y are the corresponding words. Let U be the longest word consisting only of zeros which is a subword of Y. We shall stipulate that the number x belongs to the set $\tau(y)$ if the word $1U1X1U1$ is a subword of Y We now prove that τ is a labeling of the finite sets of natural numbers.

Suppose that $\{x_1, \quad ., x_n\}$ is a finite set of natural numbers, and $\{X_1, \quad ., X_n\}$ is the set of corresponding words. Let U be a word consisting only of zeros which is longer than any of the words $X_1, \qquad X_n$. We let Y denote the word

$$1U1X_11U1X_21U1 \qquad 1U1X_n1U1.$$

Then the number of the word Y will be a label of the set $\{x_1, \ . ., x_n\}$. This labeling is easily shown to be arithmetic, using the above list of arithmetic properties of words. This completes the construction of an arithmetic labeling by Smullyan's method.

D. Languages Connected with Associative Calculi

In this appendix we shall look at examples of languages whose sets of true statements are relatively simple in structure. These examples are connected with the so-called "associative calculi."

An *associative calculus* in the alphabet L is an arbitrary finite set of rules which describe a type of transformation of words in L. These rules are called two-sided substitutions, or simply (since we shall not deal with one-sided substitutions) *substitutions* in the alphabet L. Each substitution in the alphabet L is written in the form

$$P \leftrightarrow Q,$$

where P and Q are words in L and the symbol $\leftrightarrow$ is not a letter of L. (For example, or $\leftrightarrow$ er is a substitution in the Latin alphabet.) A substitution rule $P \leftrightarrow Q$ means that whenever P occurs as part of another word it can be replaced by Q, and vice versa. We now state this more precisely in the form of some definitions.

Given an associative calculus (i.e., a list of substitutions), we introduce the notion of contiguity and equivalence of words. Two words A and B are said to be *contiguous*, denoted $A \perp B$, if there exist words P, Q, X and Y such that (1) $A = XPY$, (2) $B = XQY$, and (3) at least one of the substitutions $P \leftrightarrow Q$ or $Q \leftrightarrow P$ is a substitution in the given

associative calculus. A finite sequence of words $\langle C_1, .$
$. ., C_n \rangle$ in the alphabet L is said to be a contiguity chain if we have $C_i \perp C_{i+1}$ for each i. Finally, we say that two words A and B are *equivalent* if there exists a contiguity chain $\langle C_1, \quad C_n \rangle$ such that $C_1 = A$ and $C_n = B$.

Remark 1. If we form the quotient of the set L^∞ by this equivalence relation, we obtain an algebraic system with an associative operation (coming from the operation of writing words next to one another). This explains the terminology "associative calculus."

Suppose we have a fixed associative calculus in the alphabet L. There exists an algorithm which, given any two words A and B in L^∞, determines whether or not they are contiguous. For example, we could obtain such an algorithm by running through all 4-tuples of words P, Q, X, Y of length no greater than the maximum length of A and B, and then checking conditions (1)-(3). Thus, the set of all pairs of contiguous words is a decidable subset of $L^\infty \times L^\infty$ However, it is not at all obvious, except in the simplest cases, that there exists an algorithm which determines whether or not two words are equivalent.

Example 1. Suppose that $L = \{a, b, c\}$ and the associative calculus is given by the following substitutions:

$$ab \leftrightarrow ba, \quad ac \leftrightarrow ca, \quad bc \leftrightarrow cb.$$

Here it is obvious that A and B are equivalent if and only if the number of a's in the word A is equal to the number of a's in B, and likewise for b and c. This associative calculus is an example of what is called a *commutative* calculus.

In the general situation, it is not clear how to construct an algorithm which, given two arbitrary words, determines whether or not they are equivalent, i.e., whether or not there exists a contiguity chain connecting them. In fact, A. A. Markov and E. L. Post showed that it is possible to find an associative calculus whose equivalence recognition problem is undecidable (in other words, for which there is no algorithm which determines whether two words are equivalent). For a proof of the existence of such an associative calculus, see, for example, Kleene's book *Introduction to Metamathematics*. Here we shall give without proof an example due to G.S. Tseitin.

Example 2. Suppose that $L = \{a, b, c, d, e\}$, and the associative calculus is given by the following substitutions:

$$ac \leftrightarrow ca, \quad ad \leftrightarrow da, \quad bc \leftrightarrow cb, \quad bd \leftrightarrow db,$$
$$eca \leftrightarrow ce, \quad edb \leftrightarrow de, \quad cca \leftrightarrow ccae.$$

Tseitin showed that this associative calculus does not have an equivalence recognition algorithm.

An associative calculus will be said to be *decidable* if there exists an algorithm which recognizes equivalence in the calculus; otherwise, the calculus will be said to be *undecidable*. Clearly, decidability of an associative calculus is equivalent to decidability of the subset of $L^\infty \times L^\infty$ consisting of all pairs of equivalent words (or decidability of the subset of all pairs of inequivalent words).

Suppose we have a fixed associative calculus $\mathfrak{I}$ in the alphabet L. We let T^+ (respectively, T^-) denote the set of all words (in the alphabet L_*) of the form $A * B$, where $A \in L^\infty$, $B \in L^\infty$, and A is equivalent (respectively, inequivalent) to B. Thus, $T^+ \cup T^- = L^\infty \times L^\infty$ and the calculus $\mathfrak{I}$ is decidable if and only if T^+ (or equivalently T^-) is a decidable subset of $L^\infty \times L^\infty$.

Remark 2. Note that the set T^+ (or T^-) for the calculus in Example 2 is one example of an undecidable subset of $L^\infty \times L^\infty$. The characteristic function of this subset is then an example of a noncomputable function.

Given any associative calculus in an alphabet L, we can now speak of two languages connected with it: the *positive language*, whose statements are all possible assertions that two words in L are equivalent, and the *negative language*, whose statements are all possible assertions that two words in L are inequivalent. In both cases the statements can be regarded as elements of the set $L^\infty \times L^\infty$ In the positive language, the word $A * B$ will be interpreted as asserting that A is equivalent to B; hence T^+ is the set of true statements. In the negative language, the word $A * B$ will be interpreted as asserting that A is inequivalent to B; hence T^- is the set of true statements.

Recall that in Subsec. 1.1.3 we agreed to define a language by giving a fundamental pair. Thus, suppose we have a fixed alphabet L and an associative calculus $\mathfrak{I}$ in this alphabet. We shall call $\langle L_*, T^+\rangle$ the fundamental pair of the positive language connected with $\mathfrak{I}$, and $\langle L_*, T^-\rangle$ the fundamental pair of the negative language connected with $\mathfrak{I}$.

We shall be interested in whether there is a complete and consistent deductive system for $\langle L_*, T^+\rangle$ and for $\langle L_*, T^-\rangle$. We shall see that this question always has a positive answer in the first case, while in the second case the answer depends upon whether the calculus $\mathfrak{J}$ is decidable.

Lemma D.1. *The set E of all contiguity chains is a decidable subset of L_*^∞.*

▶ This lemma follows from the existence of an algorithm which determines whether an arbitrary pair of words in L^∞ satisfies the contiguity relation. ■

Theorem D.1. *In any associative calculus the set of all pairs of equivalent words is enumerable.*

▶ We define a function φ on L_*^∞ by setting

$$\varphi(C_1 * C_2 * \quad * C_n) = C_1 * C_n$$

for each word $C_1 * C_2 * . \quad . * C_n$, where each C_i is a word in the alphabet L. Then two words A and B are equivalent if and only if $A * B = \varphi(C)$ for some contiguity chain C. That is, $T^+ = \varphi(E)$, where E is the set of all contiguity chains. The set E is a decidable subset of L_*^∞, by Lemma D.1, and so is enumerable (by Lemma 2 of Sec. 2). But φ is obviously a computable function; hence, the set $\varphi(E) = T^+$ is enumerable, as was to be proved. ■

Remark 3. Thus, in an undecidable calculus the set T^+ is an example of an enumerable but undecidable subset of the enumerable set $L^\infty \times L^\infty$. By Lemma 3 of Sec. 2, such a subset is also an example of an enumerable set with non-enumerable complement. See also Remark 5 below.

Corollary of Theorem D.1. *There exists a complete and consistent deductive system for the fundamental pair of the positive language connected with an arbitrary associative calculus.*

Remark 4. To obtain the deductive system in the corollary, there is no need to refer to Theorem 1 of Sec. 2. We can simply choose the deductive system $\langle L_*, E, \varphi\rangle$, where E and φ are as in the proof of Theorem D.1; it will be a complete and consistent deductive system for the fundamental pair $\langle L_*, T^+\rangle$. This deductive system is quite natural from an intuitive point of view, since the best way to show that two words A and B are equivalent is to exhibit a contiguity chain connecting them.

We now ask about a deductive system for $\langle L_*, T^-\rangle$.

Theorem D.2. *Given an associative calculus, the set of all pairs of inequivalent words is enumerable if and only if the calculus is decidable.*

▶ We first recall that $L^\infty \times L^\infty$ is enumerable (see Example 6 in Sec. 2). Suppose that the calculus is decidable. Then T^- is a decidable subset of $L^\infty \times L^\infty$, and so T^- is enumerable, by Lemma 2 of Sec. 2. Now suppose that, conversely, T^- is enumerable. Since its complement T^+ in the enumerable set $L^\infty \times L^\infty$ is also enumerable, by Theorem D.1, it follows by Lemma 3 of Sec. 2 that T^- is a decidable subset of $L^\infty \times L^\infty$, and so our associative calculus is decidable. ■

Remark 5. Thus, if we have an undecidable associative calculus, T^+ will be an example of an enumerable set with nonenumerable complement (in an enumerable, larger set). By Lemma 3 of Sec. 2, any such example is also an example of an enumerable set which is undecidable (relative to the enumerable, larger set). Thus, the existence of an enumerable but undecidable set, which we proved in Sec. 5, can also be obtained as a corollary of the existence of undecidable associative calculi. But it should be mentioned that the usual proofs that some associative calculus is undecidable rely upon the existence of an enumerable undecidable set (this is true of Example 2, for instance). Thus, the latter fact should be proved without relying upon the existence of undecidable associative calculi.

Corollary of Theorem D.2. *There exists a complete and consistent deductive system for the fundamental pair of the negative language connected with an associative calculus if and only if the calculus is decidable.*

Given an arbitrary associative calculus $\mathfrak{J}$ in the alphabet L, we now introduce its *universal language*, whose statements consist both of assertions that two words are equivalent and assertions that two words are inequivalent. Here we need a way to distinguish between the two types of statements. For this purpose we add one more letter $\neg$ to the alphabet L, where we are supposing that $\neg$, like the symbols $\leftrightarrow$ and $*$, are not letters in L. We let L′ denote the alphabet $L \cup \{*, \neg\}$. Now we let $\neg T^-$ denote the set of all words of the form $\neg P$, where $P \in T^-$. We set $T^0 = T^+ \cup \neg T^-$, and form the fundamental pair $\langle L', T^0 \rangle$. It is natural to interpret an element t of T^0 as a true statement about the equivalence (if $t \in T^+$) or inequivalence (if $t \in \neg T^-$) of two words.

Theorem D.3. *Given any associative calculus $\mathfrak{J}$, the corresponding set T^0 is enumerable if and only if the calculus is decidable.*

▶ If $\mathfrak{J}$ is decidable, then T^- is enumerable, by Theorem D.2, and so $\neg T^-$ is also enumerable (see Example 5 in Sec. 2).

Then T^0 is enumerable, by Lemma 5 of Sec. 3. Now suppose that T^0 is enumerable. We form the set $\neg L'^{\infty}$ of all words in the alphabet L′ which begin with $\neg$; this set is enumerable (see Examples 2 and 5 in Sec. 2). By Lemma 5 of Sec. 3, the intersection $T^0 \cap \neg L'^{\infty}$ is enumerable. But $T^0 \cap \neg L'^{\infty} = \neg T^-$. Hence $\neg T^-$ is enumerable, and then so is T^- (see Example 5 in Sec. 2). But then the calculus $\mathfrak{I}$ is decidable, by Theorem D.2. ■

Corollary. *There exists a complete and consistent deductive system for the fundamental pair of the universal language connected with an associative calculus if and only if the calculus is decidable.*

E. Historical Remarks

One of the truly great mathematicians of the 20th century (and undoubtedly the greatest mathematical logician) was Kurt Gödel. He was born on 28 April, 1906 in Brno, in what was Austria-Hungary and is now Czechoslovakia. From the 1940s until his death on 14 January, 1978, Gödel worked at the Institute for Advanced Study in Princeton. The name Gödel is connected with the most important theorems in mathematical logic: the completeness theorem for predicate calculus (1930), the incompleteness theorem for arithmetic (1930), and the theorem on consistency of the axiom of choice with the continuum-hypothesis (1938).

The completeness theorem for predicate calculus says that it is possible to find a complete and consistent deductive system for the language of the logic of predicates. More precisely, Gödel proved that a particular concrete deductive system (that was known before) fulfills this role. Thus, in that deductive system one can prove all true statements of predicate logic, i.e., any formula which expresses a "law of logic"; and it is impossible to prove any other formula. Here by a "law of logic" we mean a formula whose truth is preserved regardless of the meaning ascribed to the various names in the formula.

On the other hand, the incompleteness theorem for arithmetic—to which the present book is devoted—says that we do not have this situation in arithmetic. Not only do all known deductive systems fail to be consistent or else fail to be complete, but it is inherently impossible to find a complete and consistent deductive system. As explained above in the main text, this means that there is no possible notion of a formal proof which would lead to all the truths of arithmetic and only the truths of arithmetic being provable. Below we shall state the incompleteness theorem in the form given by Gödel himself.

The theorem on consistency of the axiom of choice and the continuum-hypothesis says that set theory remains consistent if we add both the axiom of choice and an axiom expressing the continuum-hypothesis, provided that it was consistent before these two axioms were added. This theorem of Gödel's was the first fundamental result in the study of the consistency of the assertions of set theory. To a large extent it changed our way of thinking about the meaning of set-theoretic statements, and gave rise to a new line of investigation in mathematical logic.

Gödel's theorems are discussed in more detail in the books cited in Bibliography at the end of the book. Gödel was responsible for many other important concepts and results, such as the first definition (in 1934) of the notion of a recursive function (Herbrand-Gödel recursiveness). We shall not attempt to list all of Gödel's contributions here, rather we shall use the remainder of this appendix to describe Gödel's original formulation of the incompleteness theorem.

Gödel's famous paper "Über formal unentscheidbare Sätze der *Principia Mathematica* und verwandter Systeme I" ("On formally undecidable propositions of *Principia Mathematica** and related systems I") was published on p. 173-198 of the first issue of vol. 38 (1931) of the Leipzig journal *Monatshefte für Mathematik und Physik* (it was presented on 17 Nov., 1930). A preliminary synopsis of the results was published in the Viennese journal *Anzeiger der Akademia der Wissenschaften in Wien, Mathematischnaturwissenschaftliche Klasse*, no. 19 (1930) (report of meeting held on 23 Oct., 1930).

In this paper Gödel showed that for a large class of formal systems there must inevitably exist an undecidable statement, i.e., a statement such that neither it nor its negation can be derived from the axioms of the given system. The paper contained the following theorem (Theorem VI on p. 187):

> For every ω-consistent recursive class $\varkappa$ of *formulas* there exists a recursive *class formula* r such that neither

* The epic monograph on mathematical logic by A. Whitehead and B. Russell, *Principia Mathematica*, Cambridge, 1925.

v Gen r nor Neg (v Gen r) belongs to Flg ($\varkappa$) (where v is a *free variable in the formula* r).

Some words of explanation are needed for this formulation. The discussion below will suppose that the reader has a rudimentary acquaintance with some facts from mathematical logic.

Gödel's theorem here is speaking of the formulas in a certain formal system P, which is constructed on p. 176-178 of the paper. Rather than taking the time for a precise description of this formal system, we shall be content simply to give the following quotation from Gödel: "In essence, P is the system which is obtained if one supplements the Peano axioms with the logical structure of *Principia Mathematica* (with numbers as the individua and the 'follows after' relation as an undefined notion)" (p. 176).

The italics in Gödel's theorem have a special meaning. They indicate that we are not speaking directly about symbol combinations in the formal system (variables, formulas, etc.), but are rather referring to the numbers of these symbolic expressions in some fixed numbering system (now called the Gödel numbering). For example, a class formula is a formula with one free variable. Hence, a *class formula* is a nonnegative integer which is the number of a class formula.

The notation v Gen r stands for the number of the formula obtained by putting the universal quantifier and the variable with number v in front of the formula with number r. Neg (v Gen r) is the number of the negation of the formula with number v Gen r. Flg ($\varkappa$) denotes the class of numbers of the formulas which are deducible from the formulas whose numbers form the class $\varkappa$. (In the deductions the axioms can be used, so that here $\varkappa$ can actually be regarded as being combined with the axioms of the original system.)

We shall not define what "recursive classes" and "recursive formulas" are. These terms refer to the possibility of defining the classes and formulas under consideration by means of primitive recursive functions (which are called simply "recursive functions" in Gödel's paper).

The property that a class be ω-consistent is a stronger condition than simple consistency. (Later, Rosser strengthened Gödel's original formulation of the theorem by showing that ω-consistency can be replaced with a weaker consistency condition.) Whereas a class is consistent if it is impossible to derive both a formula and its negation, a class is ω-consistent if it is impossible to derive both a formula of the form “there exists x such that $\mathfrak{U}$ (x)” and also all formulas of the form “not $\mathfrak{U}$ (0),” “not $\mathfrak{U}$ (1),” “not $\mathfrak{U}$ (2),” and so on. In the notation of Gödel’s paper, a class $\varkappa$ of *formulas* (i.e., numbers of formulas) is said to be ω-consistent if there does not exist a *class formula* a for which: (1) Neg (v Gen a) $\in$ Flg $(\varkappa)$, and (2) Sb $(a^{v}_{Z(n)}) \in$ Flg $(\varkappa)$ for all n. Here Sb $(a^{v}_{Z(n)})$ denotes the number of the result of substituting the formula with number Z (n) into the formula with number a in place of the variable with number v; Z (n) is the number of n.

Thus, Theorem VI says that for any class of formulas which satisfies certain conditions there exists a formula having a rather simple form such that neither this formula nor its negation can be derived from the class. Since Peano’s axioms of arithmetic are the central ingredient in the formal system P upon which this theorem is based (recall that the theorem refers to the formulas of this system and deducibility using the rules of this system), Gödel’s theorem is often interpreted as a statement about the incompleteness of formal arithmetic. Here incompleteness should be understood from the syntactic point of view (see Appendix A).

Remark 1. If we regard the system P as being formal arithmetic, then the incompleteness of formal arithmetic is only a very special case of Theorem VI obtained by taking $\varkappa = \varnothing$. Here the theorem guarantees incompleteness provided that P itself is ω-consistent, i.e., its class of axioms is ω-consistent. In this special case Flg $(\varkappa)$ consists simply of the numbers of all formulas that are provable in P.

Remark 2. To be sure, the undecidable formula in Theorem VI—namely, the formula with number v Gen r—does not have an arithmetic character, i.e., it is not written in the simplest arithmetic language. However, in this connection Gödel’s paper contains some important further results. On p. 193 we find the following Theorem VIII:

> In any formal system as in Theorem VI, there exist undecidable arithmetic statements.

Here a formula is said to be “arithmetic” if it is constructed using variables indexed by natural numbers, the equality relation, and the addition and multiplication operations.

It should be noted in passing that the symbols $=$, $+$ and   are not in the original alphabet of P. Thus, an “arithmetic formula” can only really exist in a suitable extension of P. In P itself these symbols must be regarded as abbreviations. For example, on p. 177 of Gödel’s paper, the expression $x_1 = y_1$ is defined to be an abbreviation for the formula

"$x_2\Pi\ (x_2\ (x_1) \supset x_2\ (y_1))$," where $x_2\Pi$ is Gödel's notation for what we denoted $\forall x_2$. (Gödel's paper does not give the analogous definitions for $x + y$ and $x - y$.)

Remark 3. As Gödel himself noted (p. 190), his proof of Theorem VI not only applies to the concrete system P in his paper, but also for any system satisfying the following two fundamental properties:

(1) the system's axioms and rules of deduction can be defined recursively;

(2) any recursive relation can be defined within the system.

As Gödel notes, these properties hold for the axiom systems of Zermelo-Fraenkel and von Neumann for set theory, and also for axiomatic number theory based on the Peano postulates and recursive definitions. Thus, there are undecidable propositions in each of these systems, as we see by setting $\varkappa = \varnothing$ in the theorem (see Remark 1 above).

Of course, the incompleteness result for each of these systems requires that the system be ω-consistent. In all concrete cases, this consistency should be viewed as a working hypothesis, which comes from our belief in the reasonableness of the system, i.e., our confidence that it is a true reflection of a certain reality.

F. Exercises

This appendix contains exercises for various sections of the book. The starred exercises are the more difficult ones.

Exercises for Sec. 2

2.1. Show that in any set the union or intersection of two decidable subsets is a decidable subset.

2.2. Show that the union or intersection of two enumerable sets is enumerable.

2.3. Show that if the graph of a function is enumerable—then the function is computable. (The converse is proved in Sec. 5.)

2.4. Show that if $A \subset \mathbb{N}$ is decidable (or enumerable), then $\{x \mid 2x \in A\}$ is decidable (respectively, enumerable).

2.5. Show that if $A \subset \mathbb{N}$ is enumerable, then $B =$

$\{x \mid \exists\, k \in \mathbb{N}\ (kx \in A)\}$ is enumerable. (But if A is decidable, it does not necessarily follow that B is decidable; see Exercise 5.16).

2.6. Show that the set $\mathbb{N}^3$ of all triples of natural numbers is enumerable.

2.7. Prove that a set $A \subset \mathbb{N}$ is enumerable if and only if it is a projection of some decidable subset R of the set $\mathbb{N}^2$.

2.8. Show that if A is a decidable subset of B and C is a decidable subset of D, then $A \times C$ is a decidable subset of $B \times D$.

2.9. Prove that any infinite enumerable set P can be enumerated by a computable function without repetitions, i.e., there exists a computable function p defined on all of $\mathbb{N}$ such that $P = \{p(0), p(1), \ldots\}$ and $p(n) \neq p(m)$, when $n \neq m$.

2.10. Let A be some set of natural numbers. We define the *direct listing* of A to be the function which takes 0 to the smallest element in A, 1 to the next smallest element in A, and so on. (If A is a finite set, then the direct listing is not an everywhere defined function.) Prove that A is a decidable subset of $\mathbb{N}$ if and only if its direct listing is a computable function.

2.11. Show that an infinite enumerable set $P \subset \mathbb{N}$ always has an infinite subset which is a decidable subset of $\mathbb{N}$.

2.12. Let A and B be enumerable sets with nonempty intersection. Show that there exist enumerable sets A_1 and B_1 such that $A \cup B = A_1 \cup B_1$, $A_1 \cap B_1 \neq \varnothing$, $A_1 \subset A$, $B_1 \subset B$.

2.13. Let P be an enumerable subset of $\mathbb{N}^2$. Show that there exists a computable function f from $\mathbb{N}$ to $\mathbb{N}$ which is defined on all x such that $\langle x, y\rangle \in P$ for some y and whose value at such an x is one of those y's (i.e., $\langle x, f(x)\rangle \in P$ for all x on which f is defined).

2.14. Derive the fact in Exercise 2.12 as a consequence of Exercise 2.13.

2.15. A set A of natural numbers is said to be *computably infinite* if there exists an algorithm which, given n, finds a list of more than n distinct elements of A. Prove that the following properties are equivalent:

1° A is computably infinite;

2° A contains an infinite enumerable subset;

3° A contains an infinite subset which is a decidable subset of $\mathbb{N}$;

4° there exists a computable function from $\mathbb{N}$ to $\mathbb{N}$ which is defined on all natural numbers and has that property that $f(n) \in A$ and $f(n) \geqslant n$ for all n.

2.16*. Prove that there exists an infinite set which is not computably infinite.

2.17. Prove that the following function f is computable: $f(n) = 1$ if one can find at least n nines in a row in the decimal expansion of π; $f(n) = 0$ otherwise. (If we replace "at least" with "exactly" in the last sentence, then it is not known whether or not the resulting function f is computable.)

2.18*. Is the following a decidable subset of $\mathbb{N}^2$: take all pairs $\langle m, n\rangle$ of natural numbers such that $n \neq 0$ and $m/n < e$? (Here e is the base for natural logarithms.)

2.19*. Is the function which takes the natural number n to the nth digit in the decimal expansion of e a computable function?

2.20. Show that the following conditions on a real number x are equivalent:

1° there exists an algorithm which, given n, finds p and q such that $q \neq 0$ and $|p/q - x| < 1/n$;

2° the set $\{\langle p, q\rangle \mid q \neq 0 \text{ and } p/q < x\}$ is decidable;

3° the function which takes n to the nth digit in the decimal expansion of x is computable.

If the conditions in Exercise 2.20 hold, then x is called a *computable* real number.

2.21*. (Continuation of Exercise 2.20.) The sum, product, or quotient of two computable real numbers is computable; and any root of a polynomial with integer coefficients is computable. Prove these facts.

2.22. (Continuation of Exercise 2.20.) Prove that there exist real numbers which are not computable.

Exercises for Sec. 3

3.1. We say that an everywhere defined function f from K^∞ to L^∞ *reduces* the set $A \subset K^\infty$ to the set $B \subset L^\infty$ if the conditions $x \in A$ and $f(x) \in B$ are equivalent for all $x \in K^\infty$ (Thus, we "reduce" the problem of whether x belongs to A to the problem of whether $f(x)$ belongs to B.) Show that if B is decidable (or enumerable) and if there is a computable function which reduces A to B, then A is also decidable (respectively, enumerable).

3.2. (Continuation of Exercise 3.1.) Show that if A is undecidable (or nonenumerable) and if there is a computable function which reduces A to B, then B is also undecidable (respectively, nonenumerable).

3.3. (Continuation of Exercise 3.1.) Show that a set $X \subset \mathbb{N}$ is expressible by means of the fundamental pair $\langle \mathrm{L}, T\rangle$ if and only if there is a computable function which reduces X to T.

3.4. We say that two sets A and B of natural numbers are *almost everywhere equal* if their differences $A \setminus B$ and $B \setminus A$ are finite. Prove that a set which is almost everywhere equal to a decidable set is decidable, and that a set which is almost everywhere equal to an enumerable set is enumerable.

Exercises for Sec. 4

4.1. Add a new quantifier $\forall_e$ meaning "for all even" to the language of arithmetic. That is, $\forall_e \xi\alpha$ is true if the closed formula $S_n^\xi\ \alpha$ is true for all even n. Show that the class of arithmetic sets does not change as a result of doing this.

4.2. Let α, β and γ be any closed formulas. Show that the following closed formulas are true:

(a) $(\alpha \wedge (\alpha \to \beta)) \to \beta$;
(b) $(\alpha \wedge (\beta \vee \gamma)) \leftrightarrow ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$;
(c) $(\alpha \to (\beta \to \gamma)) \leftrightarrow ((\alpha \wedge \beta) \to \gamma)$;
(d) $((\alpha \to \gamma) \wedge (\beta \to \gamma)) \leftrightarrow ((\alpha \vee \beta) \to \gamma)$;
(e) $(((\alpha \to \beta) \to \alpha) \to \alpha)$;
(f) $\neg(\alpha \wedge \beta) \leftrightarrow (\neg\alpha \vee \neg\beta)$;
(g) $\neg(\alpha \vee \beta) \leftrightarrow (\neg\alpha \wedge \neg\beta)$;
(h) $((\alpha \to \beta) \wedge (\alpha \to \neg\beta)) \to \neg\alpha$;
(i) $(\alpha \wedge \neg\alpha) \to \beta$;
(j) $(\alpha \to \beta) \leftrightarrow (\neg\alpha \vee \beta)$;
(k) $(\alpha \leftrightarrow \beta) \leftrightarrow ((\alpha \to \beta) \wedge (\beta \to \alpha))$;
(l) $(\alpha \vee \beta) \leftrightarrow \neg(\neg\alpha \wedge \neg\beta)$.

4.3. Let α be a formula having no parameters other than ξ.

Show that the following closed formulas are true:

(a) $\forall\xi\alpha \to \exists\xi\alpha$;
(b) $\neg\forall\xi\alpha \leftrightarrow \exists\xi\neg\alpha$;
(c) $\neg\exists\xi\alpha \leftrightarrow \forall\xi\neg\alpha$.

4.4. Prove that the class of arithmetic sets remains the same if we eliminate the symbols $\vee$, $\to$, $\leftrightarrow$, and $\exists$ from the language of arithmetic (the symbols $\neg$, $\wedge$, and $\forall$ remain).

4.5. Show that the following sets are arithmetic:
$\{x \mid x$ is divisible by $3\}$,
$\{x \mid x$ is a power of $3\}$,
$\{x \mid$ the last decimal digit in x is $7\}$.

4.6*. Show that the set $\{x \mid x$ is a power of $10\}$ is arithmetic.

4.7. Prove that any arithmetic set is associated with some formula which does not contain any numbers.

4.8. Prove that there exists a set which is not arithmetic.

Exercises for Sec. 5

5.1. Show that the following properties are equivalent: (a) X is an enumerable set; (b) X is the domain of definition of some computable function; (c) X is the set of values of some computable function.

5.2. Prove that for every computable function f there exists a computable function g with the following property: $g(y)$ is defined if and only if y is in the set of values of f, and in that case $f(g(y)) = y$.

5.3. Prove that if f is a computable function from $\mathbb{N}$ to $\mathbb{N}$, then the set $\{x \mid f(x) = 1986\}$ is enumerable.

5.4. Show that there does not exist a function from $\mathbb{N}^2$ to $\mathbb{N}$ which is universal for the class of all everywhere defined functions from $\mathbb{N}$ to $\mathbb{N}$.

5.5. Show that there does not exist a function from $\mathbb{N}^2$ to $\mathbb{N}$ which is universal for the class of all functions from $\mathbb{N}$ to $\mathbb{N}$.

5.6. Let α be a function from which no computable function can be everywhere different. Show that the set $\{x \mid \alpha(x) = 1986\}$ is enumerable and undecidable.

5.7. Prove that there exists a nonenumerable set with nonenumerable complement, and that this set can be chosen to be arithmetic.

5.8. Let F be a computable function from $\mathbb{N}^2$ to $\mathbb{N}$. We shall call n the *number* of the function F_n relative to F. We shall say that the numbering given by a function F' *reduces* to the numbering given by F if there exists an everywhere defined computable function h from $\mathbb{N}$ to $\mathbb{N}$ which takes the F'-number of a function to the F-number of the same function, i.e., $F_{h(n)} = F'_n$. Show that there exists a function F from $\mathbb{N}^2$ to $\mathbb{N}$ such that the numbering given by

F' reduces to the numbering given by F for any function F' from $\mathbb{N}^2$ to $\mathbb{N}$. Show that a function with this property must be a universal function. Such functions are called *principal* universal functions.

5.9. Let F be a computable function from $\mathbb{N}^2$ to $\mathbb{N}$. Show that the sets $\{n \mid F_n$ is defined on a nonempty set$\}$ and $\{n \mid F_n$ takes the value $1986\}$ are enumerable.

5.10. Show that there is a computable function F from $\mathbb{N}^2$ to $\mathbb{N}$ such that the set $\{n \mid F_n$ is defined on a nonempty set$\}$ is an enumerable set with nonenumerable complement.

5.11. Show that there exists an enumerable set $P \subset \mathbb{N}^2$ whose set of lower points, i.e.,

$$\{\langle x, y\rangle \mid \langle x, y\rangle \in P \quad \text{and} \quad \forall y' < y\ (\langle x, y'\rangle \notin P)\},$$

is a nonenumerable set.

5.12. Let P be a subset of $\mathbb{N} \times \mathbb{N}$. Let P_n denote the set $\{x \in \mathbb{N} \mid \langle n, x\rangle \in P\}$. Prove that if P is enumerable, then all of the P_n are enumerable. Show that there exists an enumerable set P such that all enumerable subsets of $\mathbb{N}$ occur among the P_n. Such a set P is said to be *universal* for the class of enumerable subsets of $\mathbb{N}$.

5.13. (Continuation of Exercise 5.12.) Prove that, if P is an enumerable set which is universal for the class of enumerable subsets of $\mathbb{N}$, then the set $\{x \mid \langle x, x\rangle \notin P\}$ is not enumerable, and hence $\{x \mid \langle x, x\rangle \in P\}$ is an enumerable set with nonenumerable complement.

5.14. Prove that there does not exist a decidable subset R of $\mathbb{N}^2$ such that all decidable subsets of $\mathbb{N}$ occur among the cross-sections $R_n = \{x \in \mathbb{N} \mid \langle n, x\rangle \in R\}$.

5.15*. Show that there exists an enumerable set of natural numbers whose complement is infinite but does not contain an infinite enumerable subset (i.e., the complement is not computably infinite).

5.16. Show that there exists a decidable subset R of $\mathbb{N}$ such that the set $\{x \mid \exists\, k \in \mathbb{N}\ (kx \in R)\}$ is not decidable.

Exercises for Appendix A

A.1. Prove that if A and B are not separable, then neither A nor B is decidable.

A.2. Show that there exist three enumerable sets A, B and C such that any two of them are disjoint but inseparable.

A.3. Prove that if A and B are enumerable subsets of $\mathbb{N}$ and $A \cup B = \mathbb{N}$, then $A \setminus B$ and $B \setminus A$ are separable.

A.4. Show that there exist sets A, $B \subseteq \mathbb{N}$ such that there is no arithmetic set which separates A from B.

Exercises for Appendix B

B.1. Prove that the projection of an arithmetic set on any set of axes (in the sense explained in Sec. 2) is an arithmetic set.

B.2. Prove that the composition of arithmetic functions is an arithmetic function. Prove that if an arithmetic function has an inverse, then that inverse function is arithmetic.

B.3. Prove that any arithmetic function from $\mathbb{N}$ to $\mathbb{N}$ has an everywhere defined arithmetic extension.

B.4. Prove that the class of arithmetic sets is the smallest class which includes the sets in Example 6 of Appendix B and which satisfies Lemmas B.1-B.4.

B.5*. We define the *quantifier depth* of a formula as follows: the quantifier depth of an elementary formula is 0, the quantifier depth of a formula of the form $\neg\alpha$ is equal to the quantifier depth of α; the quantifier depth of a formula of the form $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \to \beta)$ or $(\alpha \leftrightarrow \beta)$ is equal to the maximum of the quantifier depths of α and β; and the quantifier depth of a formula of the form $\exists\xi\alpha$ or $\forall\xi\alpha$ is 1 greater than the quantifier depth of α. Prove that for any natural number k the set of true closed formulas of the language of arithmetic whose quantifier depth is no greater than k is an arithmetic set.

B.6*. Prove that for any numbering of the closed formulas of the language of arithmetic and any numbering of the class formulas the two sets $\{n \mid$ the nth closed formula is true$\}$ and $\{\langle m, n, k\rangle \mid k$ is the number of the closed formula obtained as the result of substituting n in place of x_1 in the mth class formula$\}$ cannot both be arithmetic sets.

Exercises for Appendix C

C.1. Prove that the following functions are address-computable:

(a) $f(x) = 2^x$;

(b) $f(x, y) = x^y$;

(c) $f(x) =$ the xth prime number;

(d) $f(x) =$ the sum of the decimal digits of x;

(e)* $f(x) =$ the xth decimal digit of e.

C.2*. Show that the class of address-computable functions from $\mathbb{N}$ to $\mathbb{N}$ would not change if we limited ourselves to some fixed number of registers, for example, if we did not allow any registers with numbers greater than 100 to be used in an address program.

C.3. Show that the class of address-computable functions would not be enlarged if we allowed commands with register numbers given indirectly, that is, commands of the form $R(a) \leftarrow R(R(b))$ and $R(R(b)) \leftarrow R(a)$. These commands say that the number in the register whose number is the number in register $R(b)$ is transferred to the ath register, and conversely.

C.4. Show that the following sets are weakly analytic:

(a) $\{\langle x, y, z\rangle \mid z = x^y\}$;

(b) $\{\langle x, y\rangle \mid x$ is the yth prime number$\}$.

C.5. Prove that every weakly analytic set is analytic. (Show this without using the theorem on arithmeticity of weakly analytic sets.)

C.6*. Prove that for any computable numbering of the words in A^∞ the numbers of the true statements of the language of arithmetic form an analytic set.

C.7*. Show that the class of analytic sets does not change if we eliminate the two-place functional variables from the extended language of arithmetic.

C.8*. Prove that if all functions are taken to be admissible, then the set of true closed formulas of the extended language of arithmetic is not analytic, but that if only the finite functions are taken to be admissible, then the set of true closed formulas is analytic.

C.9*. We say that a set $P \subset \mathbb{N}^k$ is *address-enumerable* if either it is empty or else there exist address-computable functions $g_1, \ldots, g_k$ from $\mathbb{N}$ to $\mathbb{N}$, defined on all of $\mathbb{N}$, such that $P = \{\langle g_1(n), \ldots, g_k(n)\rangle \mid n \in \mathbb{N}\}$. Prove (without using the protocol axiom) that the graph of an address-computable function is address-enumerable.

C.10*. Prove (without using the program axiom) that there exists an address-computable function from $\mathbb{N}^2$ to $\mathbb{N}$ which is universal for the class of address-computable functions from $\mathbb{N}$ to $\mathbb{N}$.

Exercises for Appendix D

D.1. Suppose that $L = \{a, b\}$, and our associative calculus is given by the substitutions $a \leftrightarrow aa$, $b \leftrightarrow bb$. Is this associative calculus decidable?

D.2. Prove that any associative calculus with a one-letter alphabet is decidable.

D.3. Associative calculi are examples of the more general concept of *calculi*. This concept is perhaps as fundamental as the notion of an algorithm. The main difference between a calculus and an algorithm is that a calculus *permits* certain actions to be performed, whereas an algorithm *prescribes* such actions. If we replace two-sided substitutions by one-sided substitutions (which allow us to replace the left side by the right side, but not vice versa), then formulate a rule telling us which substitution should be formed and in what place in the word, and finally specify when the processing of a word will be considered to be complete, we are going from a calculus to an algorithm. (It is in this way that the normal algorithms used by A. A. Markov to prove the existence of undecidable associative calculi are obtained). In this exercise we shall consider an algorithm which is somewhat similar to this type.

This algorithm is applicable to words in the alphabet $\{a, b\}$. It consists of the following instructions:

(1) If the word begins with a, i.e., if it has the form aP, where P is a word, then transform it to Pb.

(2) If the word has the form baP, then transform it to $Paba$.

(3) Repeat these transformations until a word of the form aaP is obtained. At this point the algorithm ends, and the result of the algorithm is the word P.

What happens if this algorithm is applied to the words *babaa*, *baaba*, and *abaab*?

D.4. Construct an algorithm which decides the associative calculus in the alphabet $\{a, b, c\}$ which is given by the following substitutions:

$$\begin{aligned} b &\leftrightarrow acc, \\ ca &\leftrightarrow accc, \\ aa &\leftrightarrow \;, \\ bb &\leftrightarrow \;, \\ cccc &\leftrightarrow \;, \end{aligned}$$

where the right side of the last three substitutions contains the empty word.

G. Answers and Hints for the Exercises

In this appendix we shall give answers or hints for some of the exercises in Appendix F.

2.2. See Sec. 3.

2.3. If the graph of f is $\{\langle p(n), q(n)\rangle \mid n \in \mathbb{N}\}$, then $f(x) = q$ (the least n for which $p(n) = x$).

2.4. See Lemma 6 in Sec. 3.

2.5. If $A = \{f(n) \mid n \in \mathbb{N}\}$ and $b \in B$, then B is the image of $\mathbb{N}^3$ under the function g defined by setting

$$g(k, h, n) = \begin{cases} h, & \text{if } kh = f(n), \\ b, & \text{otherwise.} \end{cases}$$

2.6. See Corollary 1 of Lemma 4.

2.7. If $A = \{f(n) \mid n \in \mathbb{N}\}$, then A is a projection of the set $R = \{\langle f(n), n\rangle \mid n \in \mathbb{N}\}$.

2.9. Suppose that the function f enumerates the set P. We can obtain the desired sequence $p(0)$, $p(1)$, . . . from the sequence $f(0)$, $f(1)$, . . . by crossing out repetitions, i.e., discarding those $f(n)$ for which $f(n) = f(k)$ for some $k < n$.

2.11. Use Exercise 2.9 to write P as $\{p(0), p(1), \ldots\}$, choose a computable monotonically increasing function (by discarding any terms which are less than earlier terms), and apply 2.10.

2.12. Fix enumerations a and b of the sets A and B: A $\{a(i) \mid i \in \mathbb{N}\}$, $B = \{b(i) \mid i \in \mathbb{N}\}$. Consider the sequence $a(0)$, $b(0)$, $a(1)$, $b(1)$, . . . Put the numbers which first appear in the even places in the set A_1. Put the numbers which first appear in the odd places in the set B_1.

2.13. Fix an enumeration of P. Then the value of f at x is the number y such that the pair $\langle x, y\rangle$ appears in the enumeration of P earlier than all other pairs $\langle x, z\rangle$, i.e., all other pairs whose first element is x.

2.14. In $\mathbb{N}^2$ consider the set $(A \times \{0\}) \cup (B \times \{1\})$, and apply 2.13.

2.15. Use 2.11 to prove $2° \Rightarrow 3°$.

2.16. We must construct a set I which does not contain a single infinite enumerable subset (see 2.15). The family of all infinite enumerable subsets is countable. Let W_0, W_1, . . . be all such sets. We shall construct I in steps, where in the ith step we arrange for W_i not to be a subset of I and for I to contain at least i elements. At the zeroth step we choose an element a_0 in W_0 and agree that a_0 will not be in I, thereby ensuring that $W_0 \not\subset I$. At the ith step we choose an element a_i in W_i which has not yet been included in I (this can be done, since W_i is infinite, and only finitely many elements have been put in I in the course of the first i steps). By agreeing not to put a_i in I, we ensure that $W_i \not\subset I$. In order to be sure that I contains at least i elements at the ith step, we proceed as follows: we choose any i numbers which we have not yet barred from membership in I (this can, of course, be done, since only $i + 1$ numbers have been barred from membership in I when we're at the ith step). We take I to be the union of the finite sets of i numbers chosen at the ith step.

2.17. The function f either is identically 1 or else is one of the functions g_i, where

$$g_i(x) = \begin{cases} 1 \text{ for } x \leqslant i, \\ 0 \text{ for } x > i. \end{cases}$$

All of these functions are computable.

2.18. Yes, it is. If we compute e with increasing precision (for example, by means of the well-known series of reciprocal factorials), we sooner or later find out whether $m/n < e$ or $m/n > e$. (Equality is impossible, since e is irrational.)

2.19. Yes; see Exercise 2.18.

2.20. It suffices to treat the case of irrational x, since all three conditions hold for rational x.

2.22. The set of all real numbers is uncountable, whereas the set of all computable real numbers is countable, since the number of computable real numbers is no greater than the number of algorithms which compute them.

3.1. See Lemma 6 in Sec. 3.

3.2. This obviously follows from Exercise 3.1.

4.1. The formula $\forall_e x_n \alpha$ can be replaced by the formula $\forall x_n$ ($[x_n$ even$] \rightarrow \alpha$).

4.2. Consider all eight possibilities for the truth or falsity of the closed formulas α, β, γ.

4.4. Use the relations in Exercises 4.2 (parts (l) (j), (k)) and 4.3 (part (b)).

4.5. See Examples 1, 3, and 5 in Appendix B.

4.6. See Appendix C.

4.7. Any formula of the form $(x = n)$ can be replaced by an equivalent formula with no numbers. For example, $(x = 0)$ is equivalent to $(x + x = x)$; $(x = 1)$ is equivalent to $(x \cdot x = x) \wedge \neg(x + x = x)$; $(x = 2)$ is equivalent to

$\exists y\,((y = 1) \wedge (x = y + y))$ (where $(y = 1)$ is replaced by the equivalent formula without the 1); and so on.

4.8. The family of all arithmetic sets is countable, whereas there are uncountably many subsets of $\mathbb{N}$.

5.1. See Corollary 1 of the protocol axiom. To prove that (a) $\to$ (b), note that the set of values of an everywhere defined function f coincides with the domain of definition of the function g defined as follows: $g(n) =$ (the least k for which $f(k) = n$).

5.2. See Exercise 2.13.

5.3. This set is a projection of the set:

$$(\text{graph of } f) \cap (\mathbb{N} \times \{1986\}).$$

5.4. If G is an everywhere defined function from $\mathbb{N}^2$ to $\mathbb{N}$, then the function g defined by setting $g(x) = G(x, x) + 1$ cannot be one of the functions G_n. Hence G is not universal.

5.6. If the set $M = \{x \mid \alpha(x) = 1986\}$ were decidable, then the function β defined by setting

$$\beta(x) = \begin{cases} 1986, & \text{if } x \notin M, \\ 0, & \text{if } x \in M, \end{cases}$$

would be a computable function which is everywhere different from α.

5.7. Let P be an enumerable undecidable subset of $\mathbb{N}$. Then $(P \times \{0\}) \cup ((\mathbb{N} \setminus P) \times \{1\})$ is the desired set.

5.8. Let G be a computable function from $\mathbb{N}^3$ to $\mathbb{N}$ which is universal for the class of all computable functions from $\mathbb{N}^2$ to $\mathbb{N}$ in the following sense: any computable function from $\mathbb{N}^2$ to $\mathbb{N}$ occurs among the functions G_n defined by the formula $G_n(x, y) = G(n, x, y)$. Consider the function F from $\mathbb{N}^2$ to $\mathbb{N}$ which is defined by setting

$$F(k, x) = G(\xi(k), \eta(k), x),$$

where ξ and η are the functions in the proof of Lemma 5 in Sec. 3. This F will be the desired function.

5.9. They are projections of the set (graph of f) and the set (graph of f) $\cap (\mathbb{N} \times \mathbb{N} \times \{1986\})$.

5.10. Let f be a computable function from $\mathbb{N}$ to $\mathbb{N}$ with undecidable domain of definition. Set

$$F(m, n) = \begin{cases} f(m), & \text{if } m = n, \\ \text{undefined}, & \text{if } m \neq n. \end{cases}$$

5.11. Let K be an enumerable undecidable set. Then set $P = (K \times \{0\}) \cup (\mathbb{N} \times \{1\})$.

5.12. Take P to be the domain of definition of a computable function which is universal for the class of computable functions from $\mathbb{N}$ to $\mathbb{N}$.

5.13. If $Q = \{x \mid \langle x, x\rangle \notin P\}$ were an enumerable set, then it would coincide with P_n for some n; but this is impossible, since $n \in Q \leftrightarrow n \notin P_n$.

5.14. If R is a decidable subset of $\mathbb{N}^2$, then the set S defined by setting $S = \{x \mid \langle x, x\rangle \notin R\}$ is a decidable subset of $\mathbb{N}$ which is different from all cross-sections of R.

5.15. The solution can be found in § 8.1 of Rogers' book (see Bibliography).

5.16. Let $p(0)$, $p(1)$, . . . be a listing of the elements in the enumerable undecidable set P. Then $R = \{(p(i)\text{th prime number})^i \mid i \in \mathbb{N}\}$ is a set with the desired property.

A.1. If A were a decidable set, then it would be a decidable set separating A from B.

A.2. Take A, B and C to be the sets $\{x \mid \alpha(x) = 0\}$, $\{x \mid \alpha(x) = 1\}$, and $\{x \mid \alpha(x) = 2\}$, where α is a computable function from which no computable function can be everywhere different.

A.3. Use Exercise 2.12.

A.4. A and B can be taken to be nonarithmetic sets which are the complements of one another in $\mathbb{N}$.

B.3. For an everywhere defined arithmetic extension of the arithmetic function f, we can take the function $\tilde{f}$ defined by setting

$$\tilde{f}(n) = \begin{cases} f(n), & \text{if } f(n) \text{ is defined,} \\ 0, & \text{if } f(n) \text{ is undefined.} \end{cases}$$

B.4. If a class of sets contains the sets in Example 6 of Appendix B and satisfies Lemmas B.1-B.4, then it contains the sets which are associated with formulas of the type $(t = s)$ and the sets which are associated with all formulas which are constructed in the various ways from the elementary formulas, i.e., it contains all arithmetic sets. (The proof uses induction on the length of the formula with which a set is associated.)

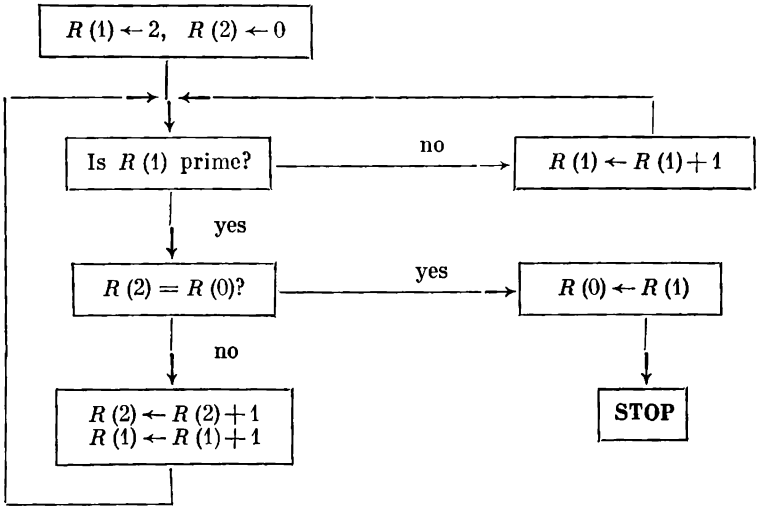
B.5. Prove this by induction on k. If $k = 0$, the assertion follows because our set is decidable. The induction step from k to $k + 1$ proceeds as follows. Let B_k be the set of true statements having quantifier depth at most k. Let A_{k+1} and E_{k+1} be the sets of true statements of quantifier depth at most $k+1$ which begin with the quantifiers $\forall$ and $\exists$, respectively. Use arithmeticity of B_k to prove that A_{k+1} and E_{k+1} are arithmetic. Then prove that B_{k+1} is arithmetic. Note that as k increases the quantifier depth of the formula with which B_k is associated also increases. It can be shown that this is unavoidable, i.e., the quantifier depth of any formula with which the B_k is associated must increase without bound as k increases.

B.6. Use the argument at the end of Appendix B.

C.1. (a) Here is the required program:

```
 1 $R(1) \leftarrow 0$
 2 $R(2) \leftarrow 1$
 3 $R(3) \leftarrow 2$
 4 $R(4) \leftarrow 1$
 5 IF $R(1) = R(0)$ GO TO 9 ELSE GO TO 6
 6 $R(1) \leftarrow R(1) + R(4)$
 7 $R(2) \leftarrow R(2) \cdot R(3)$
 8 GO TO 5
 9 $R(0) \leftarrow R(2)$
10 STOP
```

(c) The general plan for constructing the required address program is given by the following flow-chart:



One can check whether $R(1)$ is prime, for example, by running through all numbers less than $R(1)$ and determining whether $R(1)$ is divisible by any of them.

(e) Use the reciprocal factorial series for e.

C.2. A finite sequence of natural numbers can be represented by a single number. Thus, the operation of an address machine with infinitely many registers can be modelled on a machine with only 100 registers, if one of the registers stores the entire memory of the first machine (represented as a single number) and the other 99 registers are used for the various computations needed to go from one machine to the other (such as coding the memory of the first machine by a single number, decoding that number, etc.).

C.3. Using a technique similar to that in the hint for Exercise C.2, we can easily model indirect addressing.

C.4. Use Exercise C.1 and the arithmeticity of address-computable functions.

C.5. The formula (for all finite v) α can be replaced by

$$(\text{for all } v)\ ([v \text{ finite}] \rightarrow \alpha),$$

where [v finite] can be written as follows:

$$\exists\, x_1 \forall\, x_2\ (v\,(x_1 + x_2) = 0).$$

C.6. We have the equivalence:

(the closed formula with number n is true) $\leftrightarrow$ (there exists a function which assigns the value T or F to any closed formula of arithmetic, which has the properties indicated in the definition of truth and which assigns the value T to the nth closed formula).

C.7. Our computable function, which gives a one-to-one correspondence between $\mathbb{N}^2$ and $\mathbb{N}$, is arithmetic. We can use this function to reduce two-place functions to one-place functions.

C.8. Use an argument analogous to the proof of Tarski's theorem. The second part of the problem is similar to Exercise C.6.

C.9. Let all address programs be represented (coded) by natural numbers. Prove that, with a natural choice of such a coding, the function F from $\mathbb{N}^4$ to $\mathbb{N}$ which is given by setting $F(n, x, y, k)$ = (the number in the yth register after the kth step of the address program with code n when applied to input x) is address-computable. (This function is obviously computable.)

C.10. See the hint for Exercise C.9.

D.1. This calculus is decidable. Here is an algorithm which decides it. Suppose we are given words P and Q. In order to determine whether or not they are equivalent, replace all groups of repeating a's by a single a, and all groups of repeating b's by a single b. (For example, the word *aaababbaa* is transformed to *ababa*.) If the same word results for Q as for P, then they are equivalent; otherwise, they are not equivalent.

D.2. Suppose that the calculus has the form

$$a^{n_1} \leftrightarrow a^{m_1},$$
$$a^{n_2} \leftrightarrow a^{m_2},$$

$$a^{n_k} \leftrightarrow a^{m_k},$$

with $m_i \neq n_i$ for all i; let h_i denote $| m_i - n_i |$; and let p be the smallest number among all the m_i and n_i. Then no word of length less than p is equivalent to another word (since none of the substitutions can be applied to such a word), and the words of length m and n (where $m, n \geqslant p$) are equivalent to one another if and only if $m - n$ is divisible by g.c.d. $(h_1, \ldots, h_k)$.

D.3. The word *babaa* is transformed to *baaba* (the algorithm produces a result); the word *abaab* is processed to the word *bbabab*, and then the algorithm cannot be applied (the algorithm stops without producing a result); the word *baaba* is transformed as follows:

$$baaba \to abaaba \to baabab \to abababa \to bababab \to babababa \to$$

This process continues on indefinitely without stopping because the word $ba \ldots ba$ (with ba repeated n times) is transformed to the word $aba \ldots aba$ (n times), which is then transformed to $ba \ldots ba$ ($2n$ times).

D.4. Every word in this calculus is equivalent to one of the words a, ac, acc, $accc$, the empty word, c, cc, or ccc (we shall call these eight words reduced words). We can use the substitution $b \leftrightarrow acc$ to remove all b's, then use the substitution $ca \leftrightarrow accc$ to move all a's to the left of all c's. We can now use the substitution of the empty word for aa or $cccc$ to ensure that there are at most 1 occurrence of a and at most 3 occurrences of c.

We now prove that none of the reduced words are equivalent to one another. We do this by letting every word correspond to a certain transformation of the square (i.e., a certain permutation of the vertices of the square). Namely, we let the word *c* correspond to a 90° rotation, we let the word *a* correspond to reflection about the center line parallel to one of the sides, and we let the word *b* correspond to reflection about the center line parallel to the other pair of sides. We let a word obtained by joining two words *P* and *Q* correspond to the composition of the two transformations of the square corresponding to *P* and *Q* (in that order). It is not hard to verify that equivalent words correspond to the same transformation, and that the reduced words all correspond to different transformations. Hence, the reduced words are all inequivalent to each other.

The following algorithm decides this calculus. To determine whether or not two words *P* and *Q* are equivalent, first replace each by an equivalent reduced word, and then compare the two reduced words. If they are the same, then *P* and *Q* are equivalent; if not, then *P* and *Q* are inequivalent.

Bibliography

1. P. J. Cohen, *Set Theory and the Continuum Hypothesis*, W. A. Benjamin, New York-Amsterdam, 1966.
2. H. Freudenthal, *The Language of Logic*, Elsevier Pub. Co., Amsterdam-London-New York, 1966.
3. S. C. Kleene, *Introduction to Metamathematics*, North-Holland Pub. Co., Amsterdam, 1952.
4. S. C. Kleene, *Mathematical Logic*, Wiley, New York, 1967.
5. R. C. Lyndon, *Notes on Logic*, D. Van Nostrand Co., Princeton-Toronto-New York-London, 1966.
6. Yu. I. Manin, *A Course in Mathematical Logic*, Springer-Verlag, New York-Berlin-Heidelberg, 1977.
7. E. Mendelson, *Introduction to Mathematical Logic*, D. Van Nostrand Co., Princeton-Toronto-New York-London, 1964.
8. H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability*, McGraw-Hill Book Co., New York-Toronto-London-Sydney, 1967.

Few discoveries have had as much impact on our perception of human thought as Gödel's proof in 1930 that any logical system, such as the usual rules of arithmetic, must inevitably be incomplete, i.e., must contain statements which are true but can never be proved. Professor Uspensky's book makes both the precise statement and also a proof of Gödel's startling theorem understandable to someone without any advanced mathematical training, such as a college student or even an ambitious high school student. Also, Uspensky introduces a new method of proving the theorem, based on the theory of algorithms which is taking on increasing importance in modern mathematics because of its connection with computers. This book is recommended for students of mathematics, computer science, and philosophy, and for the scientific layman interested in logical problems of deductive thought.

Mir Publishers
Moscow